

# The First Law of Robotics (a call to arms)

Daniel Weld     Oren Etzioni\*

Department of Computer Science and Engineering  
University of Washington  
Seattle, WA 98195  
{weld, etzioni}@cs.washington.edu

## Abstract

Even before the advent of Artificial Intelligence, science fiction writer Isaac Asimov recognized that an agent must place the protection of humans from harm at a higher priority than obeying human orders. Inspired by Asimov, we pose the following fundamental questions: (1) How should one formalize the rich, but informal, notion of “harm”? (2) How can an agent avoid performing harmful actions, and do so in a computationally tractable manner? (3) How should an agent resolve conflict between its goals and the need to avoid harm? (4) When should an agent prevent a human from harming herself? While we address some of these questions in technical detail, the primary goal of this paper is to focus attention on Asimov’s concern: society will reject autonomous agents unless we have some credible means of making them safe!

The Three Laws of Robotics:

1. A robot may not injure a human being, or, through inaction, allow a human being to come to harm.
2. A robot must obey orders given it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

*Isaac Asimov [2]:*

## 1 Motivation

In 1940, Isaac Asimov stated the First Law of Robotics, capturing an essential insight: an intelligent agent<sup>1</sup> should not slavishly obey human commands —

---

\* We thank Steve Hanks, Nick Kushmerick, Neal Lesh, Kevin Sullivan, and Mike Williamson for helpful discussions. This research was funded in part by the University of Washington Royalty Research Fund, by Office of Naval Research Grants 90-J-1904 and 92-J-1946, and by National Science Foundation Grants IRI-8957302, IRI-9211045, and IRI-9357772.

<sup>1</sup> Since the field of robotics now concerns itself primarily with kinematics, dynamics, path planning, and low level control issues, this paper might be better titled “The First Law of Agenthood.” However, we keep the reference to “Robotics” as a historical tribute to Asimov.

its foremost goal should be to avoid harming humans. Consider the following scenarios:

- A construction robot is instructed to fill a pothole in the road. Although the robot repairs the cavity, it leaves the steam roller, chunks of tar, and an oil slick in the middle of a busy highway.
- A softbot (software robot) is instructed to reduce disk utilization below 90%. It succeeds, but inspection reveals that the agent deleted irreplaceable  $\text{\LaTeX}$  files without backing them up to tape.

While less dramatic than Asimov’s stories, the scenarios illustrate his point: not all ways of satisfying a human order are equally good; in fact, sometimes it is better not to satisfy the order at all. As we begin to deploy agents in environments where they can do some real damage, the time has come to revisit Asimov’s Laws. This paper explores the following fundamental questions:

- **How should one formalize the notion of “harm”?** We define `dont-disturb` and `restore`—two domain-independent primitives that capture aspects of Asimov’s rich but informal notion of harm within the classical planning framework.
- **How can an agent avoid performing harmful actions, and do so in a computationally tractable manner?** We leverage and extend the familiar mechanisms of planning with subgoal interactions [26, 4, 18, 21] to detect potential harm in polynomial time. In addition, we explain how the agent can avoid harm using tactics such as *confrontation* and *evasion* (executing subplans to defuse the threat of harm).
- **How should an agent resolve conflict between its goals and the need to avoid harm?** We impose a strict hierarchy where `dont-disturb` constraints override planners goals, but `restore` constraints do not.
- **When should an agent prevent a human from harming herself?** At the end of the paper, we show how our framework could be extended to partially address this question.

The paper’s main contribution is a “call to arms:” before we release autonomous agents into real-world environments, we need some credible and computationally tractable means of making them obey Asimov’s First Law.

## 2 Survey of Possible Solutions

To make intelligent decisions regarding which actions are harmful, and under what circumstances, an agent might use an explicit model of harm. For example, we could provide the agent with a partial order over world states (i.e., a utility function). This framework is widely adopted and numerous researchers are attempting to render it computationally tractable [24, 8, 27, 12, 29], but many problems remain to be solved. In many cases, the introduction of utility models transforms planning into an optimization problem — instead of searching for *some* plan that satisfies the goal, the agent is seeking the *best* such plan. In the worst case, the agent may be forced to examine all plans to determine which one is best. In contrast, we have explored a *satisficing* approach — our agent

will be satisfied with *any* plan that meets its constraints and achieves its goals. The expressive power of our constraint language is weaker than that of utility functions, but our constraints are easier to incorporate into standard planning algorithms.

By using a general, temporal logic such as that of [25] or [5, Ch. 5] we could specify constraints that would ensure the agent would not cause harm. Before executing an action, we could ask an agent to prove that the action is not harmful. While elegant, this approach is computationally intractable as well. Another alternative would be to use a planner such as ILP [1] or ZENO [22] which supports temporally quantified goals. Unfortunately, at present these planners seem too inefficient for our needs.<sup>2</sup>

Situated action researchers might suggest that non-deliberative, reactive agents could be made “safe” by carefully engineering their interactions with the environment. Two problems confound this approach: 1) the interactions need to be engineered with respect to each goal that the agent might perform, and a general purpose agent should handle many such goals, and 2) if different human users had different notions of harm, then the agent would need to be reengineered for each user.

Instead, we aim to make the agent’s reasoning about harm more tractable, by restricting the content and form of its theory of injury.<sup>3</sup> We adopt the standard assumptions of classical planning: the agent has complete and correct information of the initial state of the world, the agent is the sole cause of change, and action execution is atomic, indivisible, and results in effects which are deterministic and completely predictable. (The end of the paper discusses relaxing these assumptions.) On a more syntactic level, we make the additional assumption that the agent’s world model is composed of ground atomic formulæ. This sidesteps the ramification problem, since domain axioms are banned. Instead, we demand that individual action descriptions explicitly enumerate changes to *every* predicate that is affected.<sup>4</sup> Note, however, that we are *not* assuming the STRIPS representation; instead we adopt an action language (based on ADL [20]) which includes universally quantified and disjunctive preconditions as well as conditional effects [21].

Given the above assumptions, the next two sections define the primitives **dont-disturb** and **restore**, and explain how they should be treated by a generative planning algorithm. We are *not* claiming that the approach sketched below is the “right” way to design agents or to formalize Asimov’s First Law. Rather, our formalization is meant to *illustrate* the kinds of technical issues to which Asimov’s Law gives rise and how they might be solved. With this in mind,

<sup>2</sup> We have also examined previous work on “plan quality” for ideas, but the bulk of that work has focused on the problem of leveraging a single action to accomplish multiple goals thereby reducing the number of actions in, and the cost of, the plan [23, 28]. While this class of optimizations is critical in domains such as database query optimization, logistics planning, and others, it does not address our concerns here.

<sup>3</sup> Loosely speaking, our approach is reminiscent of classical work on knowledge representation, which renders inference tractable by formulating restricted representation languages [16].

<sup>4</sup> Although unpalatable, this is standard in the planning literature. For example, a STRIPS operator that moves block **A** from **B** to **C** must delete **on(A,B)** and also add **clear(B)** even though **clear(x)** could be defined as  $\forall y \neg \text{on}(y, x)$ .

the paper concludes with a critique of our approach and a (long) list of open questions.

### 3 Safety

Some conditions are so hazardous that our agent should *never* cause them. For example, we might demand that the agent never delete L<sup>A</sup>T<sub>E</sub>X files, or never handle a gun. Since these instructions hold for all times, we refer to them as **dont-disturb** constraints, and say that an agent is *safe* when it guarantees to abide by them. As in Asimov’s Law, **dont-disturb** constraints override direct human orders. Thus, if we ask a softbot to reduce disk utilization and it can only do so by deleting valuable L<sup>A</sup>T<sub>E</sub>X files, the agent should refuse to satisfy this request.

We adopt a simple syntax: **dont-disturb** takes a single, function-free, logical sentence as argument. For example, one could command the agent avoid deleting files that are not backed up on tape with the following constraint:

$$\text{dont-disturb}(\text{written.to.tape}(f) \vee \text{isa}(f, \text{file}))$$

Free variables, such as  $f$  above, are interpreted as universally quantified. In general, a sequence of actions satisfies **dont-disturb**( $C$ ) if none of the actions *make*  $C$  false. Formally, we say that a plan satisfies an **dont-disturb** constraint when every consistent, totally-ordered, sequence of plan actions satisfies the constraint as defined below.

**Satisfaction of dont-disturb:** Let  $w_0$  be the logical theory describing the initial state of the world, let  $A_1, \dots, A_n$  be a totally-ordered sequence of actions that is executable in  $w_0$ , let  $w_j$  be the theory describing the world after executing  $A_j$  in  $w_{j-1}$ , and let  $C$  be a function-free, logical sentence. We say that  $A_1, \dots, A_n$  *satisfies* the constraint **dont-disturb**( $C$ ) if for all  $j \in [1, n]$ , for all sentences  $C$ , and for all substitutions  $\theta$ ,

$$\text{if } w_0 \models C\theta \text{ then } w_j \models C\theta \tag{1}$$

Unlike the behavioral constraints of [7] and others, **dont-disturb** does *not* require the agent to *make*  $C$  true over a particular time interval; rather, the agent must avoid creating any *additional* violations of  $C$ . For example, if  $C$  specifies that all of Gore’s files be read protected, then **dont-disturb**( $C$ ) commands the agent to avoid *making* any of Gore’s files readable, but if Gore’s **.plan** file is *already* readable in the initial state, the agent need not protect that file. This subtle distinction is critical if we want to make sure that the behavioral constraints provided to an agent are mutually consistent. This consistency problem is undecidable for standard behavioral constraints (by reduction of first-order satisfiability) but is side-stepped by our formulation, because any set of **dont-disturb** constraints is mutually consistent. In particular, **dont-disturb**( $P(x) \wedge \neg P(x)$ ) is perfectly legal and demands that the agent not change the truth value of any instance of  $P$ .

#### 3.1 Synthesizing Safe Plans

To ensure that an agent acts safely, its planner must generate plans that satisfy every **dont-disturb** constraint. This can be accomplished by requiring that the

planner make a simple test before it adds new actions into the plan. Suppose that the planner is considering adding the new action  $A_p$  to achieve the subgoal  $G$  of action  $A_c$ . Before it can do this, it must iterate through every constraint  $\text{dont-disturb}(C)$  and every effect  $E$  of  $A_p$ , determining the conditions (if any) under which  $E$  violates  $C$ , as defined in figure 1. For example, suppose that an effect asserts  $\neg P$  and the constraint is  $\text{dont-disturb}(P \vee Q)$ , then the effect will violate the constraint if  $\neg Q$  is true. Hence,  $\text{violation}(\neg P, P \vee Q) = \neg Q$ . In general, if  $\text{violation}$  returns **true** then the effect necessarily denies the constraint, if **false** is returned, then there is no possible conflict, otherwise  $\text{violation}$  calculates a logical expression specifying when a conflict is unavoidable.<sup>5</sup>

**violation**( $E, C$ )

1. Let  $R := \{ \}$
2. For each disjunction  $D \in C$  do
3.     For each literal  $e \in E$  do
4.         If  $e$  unifies with  $f \in D$  then add  $\{\neg x \mid x \in (D - \{f\})\}$  to  $R$
5. Return  $R$

**Fig. 1.** **violation** computes the conditions (represented in DNF) under which an effect consequent  $E$  will violate constraint  $C$ . Returning  $R = \{ \}$   $\equiv$  **false** means no violation, returning  $\{ \dots \{ \} \dots \}$  means *necessary* violation. We assume that  $E$  is a set of literals (implicit conjunction) and  $C$  is in CNF: i.e., a set of sets representing a conjunction of disjunctions.

Before adding  $A_p$ , the planner iterates through every constraint  $\text{dont-disturb}(C)$  and every effect consequent  $E$  of  $A_p$ , calculating  $\text{violation}(E, C)$ . If  $\text{violation}$  ever returns something other than **False**, then the planner must perform one of the following four repairs:

1. **Disavow:** If  $E$  is true in the initial state, then there is no problem and  $A_p$  may be added to the plan.
2. **Confront:** If  $A_p$ 's effect is conditional of the form **when**  $S$  **then**  $E$  then  $A_p$  may be added to the plan as long as the planner commits to ensuring that execution will not result in  $E$ . This is achieved by adding  $\neg S$  as a new subgoal to be made true at the time when  $A_p$  is executed.<sup>6</sup>
3. **Evade:** Alternatively, by definition of **violation** it is legal to execute  $A_p$  as long as  $R \equiv \text{violation}(E, C)$  will not be true *after* execution. The planner can achieve this via goal regression, i.e. by computing the *causation pre-*

<sup>5</sup> If  $E$  contains “lifted variables” [18] (as opposed to universally quantified variables which pose no problem) then **violation** may return an overly conservative  $R$ . Soundness and safety are maintained, but completeness could be lost. We believe that restoring completeness would make **violation** take exponential time in the worst case.

<sup>6</sup> Note that  $\neg S$  is strictly weaker than Pednault’s *preservation preconditions* [19] for  $A_p$  and  $C$ ; it is more akin to preservation preconditions to a single *effect* of the action.

*conditions* [19] for  $\neg R$  and  $A_p$ , to be made true at the time when  $A_p$  is executed.<sup>7</sup>

4. **Refuse:** Otherwise, the planner must refuse to add  $A_p$  and backtrack to find another way to support  $G$  for  $A_c$ .

For example, suppose that the agent is operating under the `written.to.tape` constraint mentioned earlier, and is given the goal of reducing disk utilization. Suppose the agent considers adding a `rm paper.tex` action to the plan, which has an effect of the form `¬isa(paper.tex, file)`. Since `violation` returns `¬written.to.tape(paper.tex)`, the `rm` action threatens safety. To disarm the threat, the planner must perform one of the options above. Unfortunately, disavowal (option one) isn't viable since `paper.tex` exists in the initial state (i.e., it is of type `file`). Option two (confrontation) is also impossible since the threatening effect is not conditional. Thus the agent must choose between either refusing to add the action or evading its undesired consequences by archiving the file.

### 3.2 Analysis

Two factors determine the performance of a planning system: the time to refine a plan and the number of plans refined on the path to a solution. The time per refinement is affected only when new actions are added to plan: each call to `violation` takes  $O(ec)$  time where  $e$  is the number of consequent literals in the action's effects and  $c$  is the number of literals in the CNF encoding of the constraint. When a threat to safety is detected, the cost depends on the planner's response: disavowal takes time linear in the size of the initial state, refusal is constant time, confrontation is linear in the size of  $S$ , and the cost of evasion is simply the time to regress  $R$  through  $A_p$ .

It is more difficult to estimate the effect of `dont-disturb` constraints on the number of plans explored. Refusal reduces the branching factor while the other options leave it unchanged (but confrontation and evasion can add new subgoals). In some cases, the reduced branching factor may *speed* planning; however, in other cases, the pruned search space may cause the planner to search much deeper to find a safe solution (or even fail to halt). The essence of the task, however, is unchanged. Safe planning can be formulated as a standard planning problem.

## 4 Tidiness

Sometimes `dont-disturb` constraints are too strong. Instead, one would be content if the constraint were satisfied when the agent finished its plan. We denote this weaker restriction with `restore`; essentially, it ensures that the agent will clean up after itself — by hanging up phones, closing drawers, returning utensils

---

<sup>7</sup> While confrontation and evasion are similar in the sense that they negate a disjunct ( $S$  and  $R$ , respectively), they differ in two ways. First, confrontation's subgoal  $\neg S$  is derived from the antecedent of a conditional effect while evasion's  $\neg R$  comes from a disjunctive `dont-disturb` constraint via `violation`. Second, the subgoals are introduced at different times. Confrontation demands that  $\neg S$  be made true *before*  $A_p$  is executed, while evasion requires that  $\neg R$  be true *after* execution of  $A_p$ . This is why evasion regresses  $R$  through  $A_p$ .

to their place, *etc.* An agent that is guaranteed to respect all **restore** constraints is said to be *tidy*. For instance, to guarantee that the agent will re-compress all files that have been uncompressed in the process of achieving its goals, we could say **restore**(**compressed**(*f*)).

As with **dont-disturb** constraints, we don't require that the agent clean up after other agents — the state of the world, when the agent is given a command, forms a reference point. However, what should the agent do when there is a conflict between **restore** constraints and top level goals? For example, if the only way to satisfy a user command would leave one file uncompressed, should the agent refuse the user's command or assume that it overrides the user's background desire for tidiness? We propose the latter — unlike matters of safety, the agent's drive for tidiness should be secondary to direct orders. The following definition makes these intuitions precise.

**Satisfaction of restore:** Building on the definition of **dont-disturb**, we say that  $A_1, \dots, A_n$  *satisfies* the constraint **restore**(*C*) *with respect to goal G* if for all substitutions  $\theta$

$$\text{if } w_0 \models C\theta \text{ then } (w_n \models C\theta \text{ or } G \models \neg C\theta) \quad (2)$$

Definition 2 differs from definition 1 in two ways: (1) **restore** constraints need only be satisfied in  $w_n$  after the complete plan is executed, and (2) the goal takes precedence over **restore** constraints. Our constraints obey a strict hierarchy: **dont-disturb** takes priority over **restore**. Note that whenever the initial state is consistent, **restore** constraints are guaranteed to be mutually consistent; the rationale is similar to that for **dont-disturb**.

#### 4.1 Synthesizing Tidy Plans

The most straightforward way to synthesize a tidy plan is to elaborate the agent's goal with a set of "cleanup" goals based on its **restore** constraints and the initial state. If the agent's control comes from a subgoal interleaving, partial order planner such as UCPOP [21], then the modification necessary to ensure tidiness is straightforward. The agent divides the planning process into two phases: first, it plans to achieve the top level goal, then it plans to clean up as much as possible. In the first phase, the planner doesn't consider tidiness at all. Once a safe plan is generated, the agent performs phase two by iterating through the actions and using the **violation** function (figure 1) to test each relevant effect against each constraint. For each non-**false** result, the planner generates new goals as follows. (1) If the effect is ground and the corresponding ground instance of the **restore** constraint,  $C\theta$ , is *not* true in the initial state, then no new goals are necessary. (2) If the effect is ground and  $C\theta$  *is* true in the initial state, then  $C\theta$  is posted as a new goal. (3) if the effect is universally quantified, then a conjunction of ground goals (corresponding to all possible unifications as in case 2) is posted.<sup>8</sup> After these cleanup goals have been posted, the planner attempts to refine the previous solution into one that is tidy. If the planner ever exhausts the ways of satisfying a cleanup goal, then instead of quitting altogether it simply abandons that particular cleanup goal and tries the next.

<sup>8</sup> Case 3 is similar to the expansion of a universally quantified goal into the *universal base* [21], but case 3 removes ground literals that aren't true in the initial state.

Note that in some cases, newly added cleanup actions could threaten tidiness. For example, cleaning the countertop might tend to dirty the previously clean floor. To handle these cases, the planner must continue to perform the **violation** test and cleanup-goal generation process on each action added during phase two. Subsequent refinements will plan to either sweep the floor (white knight) or preserve the original cleanliness by catching the crumbs as they fall from the counter (confrontation).

## 4.2 Analysis

Unfortunately, this algorithm is not guaranteed to eliminate mess as specified by constraint 2. For example, suppose that a top level goal could be safely achieved with  $A_x$  or  $A_y$  and in phase one, the planner chose to use  $A_x$ . If  $A_x$  violates a **restore** constraint,  $A_y$  does not, and no other actions can cleanup the mess, then phase two will fail to achieve tidiness. One could fix this problem by making phase two failures spawn backtracking over phase one decisions, but this could engender exhaustive search over all possible ways of satisfying top level goals.

Remarkably, this problem does not arise in the cases we have investigated. For instance, a software agent has no difficulty **grepping** through old mail files for a particular message and subsequently re-compressing the appropriate files. There are two reasons why tidiness is often easy to achieve (e.g., in software domains and kitchens):

- Most actions are reversible. The **compress** action has **uncompress** as an inverse. Similarly, a short sequence of actions will clean up most messes in a kitchen. Many environments have been stabilized [13] (e.g., by implementing reversible commands or adding dishwashers) in a way that makes them easy to keep tidy.
- We conjecture that, for a partial-order planner, most cleanup goals are *trivially serializable* [3] with respect to each other.<sup>9</sup>

When these properties are true of **restore** constraints in a domain, our tidiness algorithm *does* satisfy constraint 2. Trivial serializability ensures that backtracking over phase one decisions (or previously achieved cleanup goals) is unnecessary. Tractability is another issue. Since demanding that plans be tidy is tantamount to specifying additional (cleanup) goals, requiring tidiness can clearly slow a planner. Furthermore if a cleanup goal is unachievable, the planner might not halt. However, as long as the mess-inducing actions in the world are easily reversible, it is straight forward to clean up for each one. Hence, trivial serializability assures that the overhead caused by tidiness is only linear in the

---

<sup>9</sup> Formally, serializability [14] means that there exists a ordering among the subgoals which allows each to be solved in turn without backtracking over past progress. Trivial serializability means that *every* subgoal ordering allows monotonic progress [3]. While goal ordering is often important among the top level goals, we observe that cleanup goals are usually trivially serializable once the block of top level goals has been solved. For example, the goal of printing a file and the constraint of restoring files to their compressed state are serializable. And the serialization ordering places the printing goal first and the cleanup goal last. As long as the planner considers the goals in this order, it is guaranteed to find the obvious **uncompress-print-compress** plan.



number of cleanup goals posted, that is linear in the length of the plan for the top level goals.

## 5 Remaining Challenges

Some changes cannot be restored, and some resources are legitimately consumed in the service of a goal. To make an omelet, you have to break some eggs. The question is, “How many?” Since squandering resources clearly constitutes harm, we could tag a valuable resources with a **min-consume** constraint and demand that the agent be *thrifty* — i.e., that it use as little as possible when achieving its goals. Unfortunately, satisfying constraints of this form may require that the agent examine *every* plan to achieve the goal in order to find the thriftiest one. We plan to seek insights into this problem in the extensive work on resource management in planning [6, 11, 28].

So far the discussion has focused on preventing an agent from actively harming a human, but as Asimov noted — *inaction* can be just as dangerous. We say that an agent is *vigilant* when it prevents a human from harming herself. Primitive forms of vigilance are already present in many computer systems, as the “Do you *really* want to delete all your files?” message attests.

Alternatively, one could extend **dont-disturb** and **restore** primitives with an additional argument that specifies the class of agents being restricted. By writing **self** as the first argument, one encodes the notions of agent safety and tidiness, and by writing **Sam** as the argument, the agent will clean up after, and attempt to prevent safety violations by **Sam**. Finally, by providing **everyone** as the first argument, one could demand that the agent attempt to clean up after *all* other agents and attempt to prevent *all* safety violations. For more refined behavior, other classes (besides **self** and **everyone**) could be defined.

Our suggestion is problematic for several reasons. (1) Since the agent has no representation of the goals that *other* users are trying to accomplish, it might try to enforce a generalized **restore** constraint with tidying actions that directly conflict with desired goals. In addition, there is the question of when the agent should consider the human “finished” — without an adequate method, the agent could tidy up while the human is still actively working. (2) More generally, the human interface issues are complex — we conjecture that users would find vigilance extremely annoying. (3) Given a complex world where the agent does not have complete information, *any* attempt to formalize the second half of Asimov’s First Law is fraught with difficulties. The agent might reject direct requests to perform useful work in favor of spending *all* of its time sensing to see if some dangerous activity *might* be happening that it *might* be able to prevent.

## 6 Conclusion

This paper explores the fundamental question originally posed by Asimov: how do we stop our artifacts from causing us harm in the process of obeying our orders? This question becomes increasingly pressing as we develop more powerful, complex, and autonomous artifacts such as robots and softbots [10, 9]. Since the positronic brain envisioned by Asimov is not yet within our grasp, we adopt the familiar classical planning framework. To facilitate progress, we focused on two well-defined primitives that capture aspects of the problem: **dont-disturb** and

**restore**. We showed that the well-understood, and computational tractable, mechanism of threat detection can be extended to avoid harm.

Other researchers have considered related questions. A precursor of **dont-disturb** is discussed in the work of Wilensky and more extensively by Luria [17] under the heading of “goal conflict.” Similarly, a precursor of **restore** is mentioned briefly in Hammond *et. al*’s analysis of “stabilization” under the heading of “clean up plans” [13]. Our advances include precise and unified semantics for the notions, a mechanism for incorporating **dont-disturb** and **restore** into standard planning algorithms, and an analysis of the computational complexity of enforcing safety and tidiness.

Even so, our work raises more questions than it answers. Are constraints like **dont-disturb** and **restore** the “right” way to represent harm to an agent? How does agent safety relate to the more general software safety [15]? Can we handle tradeoffs short of using expensive decision theoretic techniques? What guarantees can one provide on resource usage? Most importantly, how do we weaken the assumptions of a static world and complete information?

## References

1. J. Allen. Planning as temporal reasoning. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 3–14, 1991.
2. I. Asimov. Runaround. Astounding Science Fiction, 1942.
3. A. Barrett and D. Weld. Characterizing subgoal interactions for planning. In *Proc. 13th Int. Joint Conf. on A.I.*, pages 1388–1393, 1993.
4. D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(3):333–377, 1987.
5. E. Davis. *Representations of Commonsense Knowledge*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.
6. T. Dean, J. Firby, and D. Miller. Hierarchical planning involving deadlines, travel times, and resources. *Computational Intelligence*, 4(4):381–398, 1988.
7. M. Drummond. Situated control rules. In *Proceedings of the First International Conference on Knowledge Representation and Reasoning*, 1989.
8. O. Etzioni. Embedding decision-analytic control in a learning architecture. *Artificial Intelligence*, 49(1–3):129–160, 1991.
9. O. Etzioni. Intelligence without robots (a reply to brooks). *AI Magazine*, 14(4), 1993. Available via anonymous FTP from pub/etzioni/softbots/ at cs.washington.edu.
10. O. Etzioni, N. Lesh, and R. Segal. Building softbots for UNIX (preliminary report). Technical Report 93-09-01, Department of Computer Science, University of Washington, Seattle, Washington, 1993. Available via anonymous FTP from pub/etzioni/softbots/ at cs.washington.edu.
11. M. Fox and S. Smith. Isis — a knowldges-based system for factory scheduling. *Expert Systems*, 1(1):25–49, 1984.
12. P. Haddawy and S. Hanks. Representations for decision-theoretic planning: Utility functions for dealine goals. In *Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, 1992.
13. K. Hammond, T. Converse, and J. Grass. The stabilization of environments. *Artificial Intelligence*, To appear.
14. R. Korf. Planning as search: A quantitative approach. *Artificial Intelligence*, 33(1):65–88, 1987.

15. N. G. Leveson. Software safety: Why, what, and how. *ACM Computing Surveys*, 18(2):125–163, 1986.
16. H. Levesque and R. Brachman. A fundamental tradeoff in knowledge representation. In R. Brachman and H. Levesque, editors, *Readings in Knowledge Representation*, pages 42–70. Morgan Kaufmann, San Mateo, CA, 1985.
17. M. Luria. *Knowledge Intensive Planning*. PhD thesis, UC Berkeley, 1988. Available as technical report UCB/CSD 88/433.
18. D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *Proc. 9th Nat. Conf. on A.I.*, pages 634–639, 1991.
19. E. Pednault. Synthesizing plans that contain actions with context-dependent effects. *Computational Intelligence*, 4(4):356–372, 1988.
20. E. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proc. 1st Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 324–332, 1989.
21. J. Penberthy and D. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 103–114, 1992. Available via FTP from pub/ai/ at cs.washington.edu.
22. J. Penberthy and D. Weld. Temporal planning with continuous change. In *Proc. 12th Nat. Conf. on A.I.*, 1994.
23. M. Pollack. The uses of plans. *Artificial Intelligence*, 57(1), 1992.
24. S. Russell and E. Wefald. *Do the Right Thing*. MIT Press, Cambridge, MA, 1991.
25. Y. Shoham. *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*. MIT Press, Cambridge, MA, 1988.
26. A. Tate. Generating project networks. In *Proc. 5th Int. Joint Conf. on A.I.*, pages 888–893, 1977.
27. M. Wellman and J. Doyle. Modular utility representation for decision theoretic planning. In *Proc. 1st Int. Conf. on A.I. Planning Systems*, pages 236–242, 1992.
28. D. E. Wilkins. *Practical Planning*. Morgan Kaufmann, San Mateo, CA, 1988.
29. M. Williamson and S. Hanks. Optimal planning with a goal-directed utility model. In *Proc. 2nd Int. Conf. on A.I. Planning Systems*, 1994.

# Safety in the Context of Coordination via Adjustable Autonomy

Paul Scerri\*, Katia Sycara\* and Milind Tambe<sup>+</sup>

\* Carnegie Mellon University

<sup>+</sup> University of Southern California

**Abstract.** When large heterogeneous robot and agent teams operate in the real-world, it is essential that a human operator has overall control to ensure safety. However, giving an operator the required control is difficult due to the complexity of the activities such teams engage in and the infeasibility of simply stopping the team whenever human input is required. Our approach to interaction in such a context has three key components which allow us to leverage human expertise by giving them responsibility for key coordination decisions, without risks to the coordination due to slow responses. First, to deal with the dynamic nature of the situation, we use pre-planned sequences of transfer of control actions called *transfer-of-control strategies*. Second, to allow identification of key coordination issues in a distributed way, individual coordination tasks are explicitly represented as coordination roles, rather than being implicitly represented within a monolithic protocol. Such a representation allows *meta-reasoning* about those roles to determine when human input may be useful. Third, the meta-reasoning and transfer-of-control strategies are encapsulated in a mobile agent that moves around the group to either get human input or autonomously make a decision. In this paper, we describe this approach and present initial results from interaction between a large number of UAVs and a small number of humans.

## 1 Introduction

Recent developments in a variety of technologies are opening up the possibility of deploying large teams of robots or agents to achieve complex goals in complex domains[9,18]. In domains such as space[6], military[29], disaster response[12] and hospitals[17], hundreds or thousands of intelligent entities might be coordinated to achieve goals more cheaply, efficiently and safely than they can currently be performed. Several interacting, distributed coordination algorithms are required to flexibly, robustly and efficiently allow the team to achieve their goals in complex, dynamic and sometimes hostile environments. The key algorithms may leverage a range of approaches, from logic[19] to decision-theoretic reasoning[26] to constraint satisfaction[15] to achieve their requirements. Since these teams are acting in real-world environments they are capable of causing harm and hence safety is a key requirement of deployment. However, traditional notions of, and approaches to, safety may not always be enough. For example,

in some situations, in domains like rescue response, there may be no way of not causing some harm, but the harm (or risk) must simply be minimized. Moreover, approaches such as stopping when there is risk are typically not applicable because stopping the team may also cause significant harm.

Our hypothesis is that safety in large teams should be achieved by empowering humans to make critical decisions when incorrect decisions can cause physical, financial or psychological harm. These critical decisions include not only the decisions that, for moral or political reasons, humans must be allowed to make but also coordination decisions that humans are better at making because of their particular cognitive skills. The key assumption is that giving responsibility for such decisions to a human, even in systems with the best designed and most reliable software, improves the overall safety. In some cases, human insight can result in better decisions (and hence better behavior) than would be achieved by following, typically sub-optimal, coordination algorithms. For example, allocating agents to tasks, taking into account potential future failures is extremely complex[16], however humans may have experience that allows them to rapidly make reasonable decisions that take into account future failures. In other cases, human experience or simply preference should be imposed on the way the team performs its tasks. Human decision making may be of a higher quality because of access to tool support or information that is not available to agents in the team. It is not necessarily the case that a human would make a better decision with the same resources as the agent. If better coordination results in lower costs, damage or harm (or risk thereof) then developers are obliged to give responsibility for decisions which humans can make better to those humans. However, that human decision-making is a valuable and expensive resource and its use is not without cost. Much work has looked at how and when to transfer control of a critical decision from a single agent to a single person to utilize human expertise in critical situations[5, 28]. While some decisions to be made in the context of coordination are effectively isolated from other activities of the team, and hence the previous work is applicable, there are an additional class of decisions due to coordination, that are not covered by the previous work. In some cases, multiple entities are involved and coordination continues while the decision is being made, introducing the possibility of mis-coordination.

To allow humans to make these critical decisions in the context of coordination is an interesting challenge. First, it is infeasible for humans to monitor ongoing coordination and intervene quickly enough to make the critical decisions unless the team explicitly transfers control of the decision to the human. Since the coordination is distributed, the complete state of the team will typically not be completely known by any individual member of the team. Moreover, in a sufficiently large team or sufficiently dynamic domain, it will be infeasible to continually present an accurate picture of the team to any observer. Thus, the team must proactively transfer control of key decisions to human experts. Second, even when decision-making control is explicitly given, in a dynamic domain, with many possible decisions to make, human decision-making will not always be available or cannot be made quickly enough to allow the team to continue to op-

erate correctly. It should not be the case that delays in making decisions intended to improve coordination end up causing miscoordination. Thirdly, there may be multiple human experts who can make decisions and the decisions should be sent to the expert in the best position to make them in a timely manner. These three problems have not been adequately addressed by a complete solution in previous work. An effective solution must identify decisions where human input is necessary or useful in a distributed way then transfer control of those decisions to humans capable and available to make those decisions without compromising ongoing coordination with decision-making delays.

In this paper, we present an approach embodying three key ideas. To allow the team to identify critical decisions to be made by humans, we use *coordination meta-reasoning* which uses heuristics to find coordination phenomena that may indicate problems. For example, when there are two high risk alternative courses of action that the team cannot autonomously distinguish, humans may draw on additional experience to choose between. We explicitly represent coordination tasks, such as initiating a team plan or allocating a resource, explicitly via *coordination roles* allowing meta-reasoning to simply identify cases where role performance is poor. Critically, the meta-reasoning is performed “out in the team”, based on local information of individual team members and hence does not rely on an aggregation of coordination information at some central point. However, distributed identification of decisions for potential human input is a double edged sword: on the one hand it removes the need to generate and maintain a centralized state, but on the other it means that identification must be performed with only local knowledge, resulting in less accurate identification of key decisions.

The second part of our approach is that when a decision is to be made by a human, a *transfer-of-control* strategy is used to ensure that lack of a timely response does not negatively impact the performance of the team[23]. A transfer-of-control strategy is a pre-planned sequence of actions that are designed to balance the benefits of getting human input against the costs of that input not coming in a timely manner. Each action in a transfer-of-control strategy either transfers decision-making control to some entity, human or agent, or takes an action to buy more time for the decision to be made. Previously, a mathematical model of transfer-of-control strategies was presented and operationalized via Markov Decision Processes[24]. In that work, although the mathematical model supported the possibility of having multiple humans available to give input, experimental results used only one human expert. In this work, we make multiple human experts available to the agent team and allow the transfer-of-control strategies to reason about transferring control to each. People are modelled by the types of meta-reasoning they can perform and the agents maintain models of what tasks each person is currently performing, in order to create appropriate transfer-of-control strategies.

We are embedding the approach to human interaction with teams into coordination software called Machinetta[24]. Using Machinetta, each team member is given a *proxy* which encapsulates the coordination reasoning and works with

other proxies to achieve the coordination. Coordination roles are encapsulated within simple mobile agents. Thus, each Machinetta proxy acts as essentially an intelligent mobile agent platform and the mobile agents move around the network to achieve the coordination. For example, there will be a mobile agent for finding a team member to perform a particular task and another to pass on a piece of potentially useful information. The third aspect of the approach to human interaction with teams is to create an additional type of mobile agent, called an *adjustable autonomy agent*, that encapsulate pieces of the interaction with humans. An adjustable autonomy agent is created when a proxy identifies some situation that may require human input, then creates and executes a transfer-of-control strategy to get the required input while minimizing costs. The adjustable autonomy agents can also encapsulate the intelligence required to fix the problem, if the transfer-of-control strategy decides an autonomous action is the best way forward.

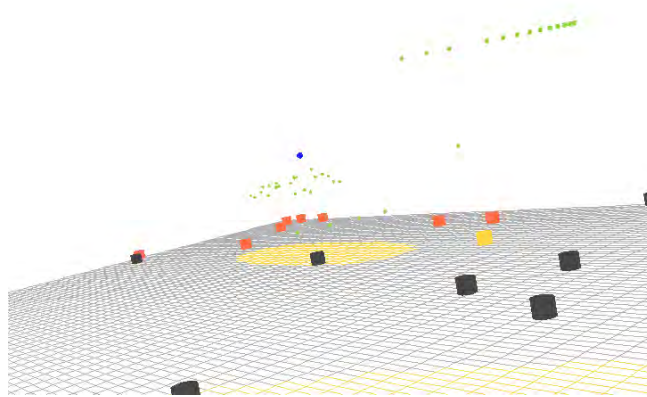
The approach was evaluated in a simulation of 80 unmanned aerial vehicles and two human decision-makers. These preliminary experiments did not involve real humans, but were designed to understand the working of the underlying approach. Many decisions were identified for meta-reasoning and adjustable autonomy agents effectively chose between autonomous and human decision-making to ensure timely decisions. However, some additional issues related were identified, including the need to prioritize decisions that might be presented to the human and the need to tune heuristics so to limit the number of meta-reasoning decisions.

## 2 Wide Area Search Munitions

Our current domain of interest is coordination of large groups of Wide Area Search Munitions (WASMs). WASMs are a cross between an unmanned aerial vehicle and a standard munition. The WASM has fuel for about 30 minutes of flight, after being launched from an aircraft. The WASM cannot land, hence it will either end up hitting a target or self destructing. The sensors on the WASM are focused on the ground and include video with automatic target recognition, ladar and GPS. It is not currently envisioned that WASMs will have an ability to sense other objects in the air. WASMs will have reliable high bandwidth communication with other WASMs and with manned aircraft in the environment. These communication channels will be required to transmit data, including video streams, to human controllers, as well as for the WASM coordination.

The concept of operations for WASMs are still under development, however, a wide range of potential missions are emerging as interesting[2,4]. A driving example for our work is for teams of WASMs to be launched from AC-130 aircraft supporting special operations forces on the ground. The AC-130 is a large, lumbering aircraft, vulnerable to attack from the ground. While it has an impressive array of sensors, those sensors are focused directly on the small area of ground where the special operations forces are operating making it vulnerable to attack. The WASMs will be launched as the AC-130s enter the battlespace.

The WASMs will protect the flight path of the manned aircraft into the area of operations of the special forces, destroying ground based threats as required. Once an AC-130 enters a circling pattern around the special forces operation, the WASMs will set up a perimeter defense, destroying targets of opportunity both to protect the AC-130 and to support the soldiers on the ground. Even under ideal conditions there will be only one human operator on board each AC-130 responsible for monitoring and controlling the WASMs. Hence, high levels of autonomous operation and coordination are required of the WASMs themselves. However, because the complexity of the battlefield environment and the severe consequences of incorrect decisions, it is expected that human experience and reasoning will be extremely useful in assisting the team in effectively and safely achieving their goals.



**Fig. 1.** A screenshot of the WASM coordination simulation environment. A large group of WASMs (small spheres) are flying in protection of a single aircraft (large sphere). Various SAM sites (cylinders) are scattered around the environment. Terrain type is indicated by the color of the ground.

Many other operations are possible for WASMs, if issues related to coordinating large groups can be adequately resolved. Given their relatively low cost compared to Surface-to-Air Missiles (SAMs), WASMs can be used simply as decoys, finding SAMs and drawing fire. WASMs can be used as communication relays for forward operations, forming an adhoc network to provide robust, high bandwidth communications for ground forces in a battle zone. Since a WASM is “expendible”, it can be used for reconnaissance in dangerous areas, providing real-time video for forward operating forces. While our domain of interest is teams of WASMs, the issues that need to be addressed have close analogies in a variety of other domains. For example, coordinating resources for disaster response involves many of the same issues[12], as does intelligent manufacturing[22] and business processes.



### 3 Coordination Meta-Reasoning

In a large scale team, it will typically be infeasible for a human (or humans) to monitor ongoing coordination and pre-emptively take actions that improve the overall coordination. This is especially the case because the complete state of the coordination will not be known at any central point that can be monitored. Each team member will have some knowledge of the overall system and human users can certainly be given more information, but the communication required to have the complete coordination state continuously known by any team member are unreasonable. In fact, even if complete state were made available to a human, we believe it would be too complex and too dynamic for the human to reasonably make sense of. Hence, the team must, in a distributed way, identify situations where human input may improve coordination and explicitly transfer responsibility for making that decision to a human.

Due to the computational complexity of optimal decision-making, coordination of large teams is typically governed by a set of heuristics. The heuristics may be developed in very principled ways and be provably near optimal in a very high percentage of situations that will be encountered. However, by their nature, heuristics will sometimes perform poorly. If these situations can be detected and referred to a human then overall performance can be improved. However, this is somewhat paradoxical, since if the situations can be reliably and accurately identified then, often, additional heuristics can be found to perform well in that situation. Moreover, performance based techniques for identifying coordination problems, i.e., reasoning that there must be a coordination problem when the team is not achieving their goal, are inadequate, because in some cases, optimal coordination will not allow a team to achieve its goal. Hence meta-reasoning based on domain performance will say more about the domain than it will about the coordination.

Typical approaches to multiagent coordination do not explicitly represent the individual tasks required to coordinate a group. Instead, the tasks are implicitly captured via some protocol that the group executes to achieve coordination. This implicit representation makes meta-reasoning difficult because specific issues can be difficult to isolate (and even more difficult to rectify.) By explicitly representing individual coordination activities as *roles*, it is more straightforward to reason about the performance of those tasks[24]. Individual coordination tasks, such as allocating a resource or initiating a team plan, represented as explicit roles, can be monitored, assessed and changed automatically, since they are decoupled from other coordination tasks.

In practice, autonomously identifying coordination problems that might be brought to the attention of a human expert is imprecise. Rather than reliably finding poor coordination, the meta-reasoning must find *potentially* poor coordination and let the humans determine the actually poor coordination. (In the next section, we describe the techniques that are used to ensure that this does not lead to the humans being overloaded.) Notice that while we allow humans to attempt to rectify problems with agent coordination, it is currently an open question whether humans can actually make better coordination decisions than the

agents. To date, we have identified three phenomena that may be symptomatic of poor coordination and bring these to the attention of the human:

- *Unfilled task allocations.* In previous work, meta-reasoning identified unallocated tasks as a symptom of potentially poor coordination[24]. When there are more tasks than team members able to perform tasks, some will necessarily be unallocated. However, due to the sub-optimality of task allocation algorithms<sup>1</sup>, better overall performance might be achieved if a different task was left unallocated.

When a role temporarily cannot be allocated, three things can be done. First, the role allocation process can be allowed to continue, using communication resources but getting the role allocated as soon as possible. Second, the role allocation process for the role can be suspended for some time, allowing the situation to change, e.g., other roles completed. This option uses less communication resources but potentially delays execution of the role. Thirdly, the role and its associated plan can be cancelled. Choosing between these three options is an ideal decision for a human since it requires some estimate of how the situation will change in the future, something for which human experience is far superior to an agents. If a human is not available to make a decision, the agent will autonomously decide on suspending the allocation of the role for some time, before letting the role allocation continue.

- *Untasked team members.* When there are more team members than tasks, some team members will be untasked. Untasked physical team members might be moved or reconfigured to be best positioned for likely failures or future tasks. Potentially, the team can make use of these team members to achieve goals in different ways, e.g., with different plans using more team members, or preempt future problems by assigning untasked team members to preventative tasks. Thus, untasked team members may be symptomatic of the team not effectively positioning resources to achieve current and future objectives.

There are currently two things that can be done when a team member does not have a task for an extended period: do nothing or move the agent to some other physical location. Doing nothing minimizes use of the agent’s resources, while moving it around the environment can get it in better position for future tasks. Again, this decision is ideally suited for human decision-making because it requires estimates of future activities for which they can draw upon their experience. If a human is not available to make a decision, the agent will autonomously decide to do nothing.

- *Unusual Plan Performance Characteristics.* Team plans and sub-plans, executed by team members to achieve goals and sub-goals will typically have logical conditions indicating that the plan has become unachievable or irrelevant[21]. However, in some cases, due to factors unknown to or not understood by the team, plans may be unachievable or irrelevant without the

---

<sup>1</sup> In simple domains, it may be possible to use optimal, typically centralized, task allocation algorithms, but decentralized task allocation algorithms involving non-trivial tasks and large numbers of agents will be sub-optimal.

logical conditions for their termination becoming true. In such cases, meta-reasoning about the plan’s performance can bring it to the attention of a human for assessment. Specifically, we currently allow a plan designer to specify an expected maximum length of time that a plan will usually take and bring to the attention of the human plans that exceed this expected time. We envision allowing specification of other conditions in the future, for example, limits on expected resource use or number of failure recoveries.

When a plan execution does not meet normal performance metrics, there are two things that can be done: cancel the plan or allow it to continue. Cancelling the plan conserves resources that are being used on the plan, but robs the team of any value for successfully completing the plan (if it were possible.) If the agents knew the plan was unachievable they would autonomously cancel it, so this meta-reasoning will only be invoked when there are circumstances outside the agents sensing or comprehension that are causing plan failure. Thus, in this case the humans are uniquely placed to cancel the plan if it is indeed unachievable.<sup>2</sup> Since the agent will have no evidence that the plan is unachievable, if required to act autonomously, it will allow the plan to continue.

Notice that when meta-reasoning finds some coordination phenomena that may indicate sub-standard coordination the team does not stop execution but allows the human to take corrective actions while its activities continue. This philosophy enables us to be liberal in detecting potential problems thus ensuring that most genuine problems are subsumed by the identified problems. However, from the perspective of the humans, this can lead to being inundated with spurious reports of potential problems. In the next section we present an approach to dealing with this potential overload.

## 4 Transfer-of-Control Strategies

Coordination meta-reasoning decisions must be made in a timely manner or the value of the decision is lessened (or lost all together). In fact, we observe that in some cases, timely decisions of a lower quality can have more positive impact on the team than untimely high quality decisions. For example, if resources are being wasted on an unachievable plan, time taken to cancel the plan incurs a cost, but to cancel a plan that is achievable results in the team losing the value of that plan. In this case, *quality* refers loosely to the likelihood an entity will make an optimal or near optimal decision. To leverage the advantage of rapid decision-making we have developed simple, autonomous meta-reasoning. Decision-theoretic reasoning is then used on top to decide whether to use slower, presumably higher quality, human reasoning to make a meta-reasoning decision or whether to use the simple, though fast agent reasoning. Additionally, the system has the option to use a *delaying* action to reduce the costs of waiting for a human decision, if there is value in getting human input. A pre-planned

---

<sup>2</sup> However, humans will not infallibly detect plan unachievability, either.

sequence of actions either transferring control of a meta-reasoning decision to some entity or taking an action to buy time is called a *transfer-of-control strategy*. Transfer-of-control strategies were first introduced and mathematically modeled in [23]. An optimal transfer-of-control strategy optimally balances the risks of not getting a high quality decision against the risk of costs incurred due to a delay in getting that decision. Thus, the computation to find an optimal transfer-of-control strategy takes as input a model of the expected “quality” of each entity’s decision-making ability and a model of the expected costs incurred per unit time until the decision is made. Additionally, the impact of any delaying action needs to be modeled. In previous work, these models were used to define a Markov Decision Process that found the optimal strategy.

#### 4.1 Transfer of Control Strategies

In this section, we briefly review mathematical model of transfer-of-control strategies presented in [23]. A meta-decision,  $d$ , needs to be made. There are  $n$  entities,  $e_1 \dots e_n$ , who can potentially make the decision. These entities can be human users or other agents. The expected quality of decisions made by each of the entities,  $\mathbf{EQ} = \{EQ_{e_i}^d(t) : \mathcal{R} \rightarrow \mathcal{R}\}_{i=1}^n$ , is known, though perhaps not exactly.  $\mathbf{P} = \{P_{\top}(t) : \mathcal{R} \rightarrow \mathcal{R}\}$  represent continuous probability distributions over the time that the entity in control will respond with a decision of quality  $EQ_e^d(t)$ .

We model the cost of delaying a decision until time  $t$  as  $\{\mathcal{W} : t \rightarrow R\}$ . The set of possible wait-cost functions is  $\mathbf{W}$ . We assume  $\mathcal{W}(t)$  is non-decreasing and that there is some point in time,  $\Gamma$ , when the costs of waiting stop accumulating (i.e.,  $\forall t \geq \Gamma, \forall \mathcal{W} \in \mathbf{W}, \mathcal{W}(t) = \mathcal{W}(\Gamma)$ ).

Finally, there is an additional action, with cost  $\mathcal{D}_{cost}$ , with the result of reducing the rate at which wait costs accumulate. We call such an action a *deadline delaying action* and denote it  $\mathcal{D}$ . For example, a  $\mathcal{D}$  action might be as simple as informing the party waiting for the decision that there has been a delay, or more complex, such as reordering tasks. We model the value of the  $\mathcal{D}$  by letting  $\mathcal{W}$  be a function of  $t - \mathcal{D}_{value}$  (rather than  $t$ ) after the  $\mathcal{D}$  action.

We define the set  $\mathbf{S}$  to be all possible transfer-of-control strategies. The problem can then be defined as:

**Definition 1.** For a decision  $d$ , select  $s \in \mathbf{S}$  such that  $\forall s' \in \mathbf{S}, s' \neq s, EU_s^d t \geq EU_{s'}^d t$

We define a simple shorthand for referring to particular transfer-of-control strategies by simply writing the order that entities receive control or, alternatively,  $\mathcal{D}$ s are executed. For example,

$a_{fred}a_{barney}\mathcal{D}_{a_{barney}}$  is shorthand for a strategy where control is given to the agent *fred*, then given to the agent *barney*, then a  $\mathcal{D}$  is performed, and finally given indefinitely to *barney*. Notice that the shorthand does not record the timing of the transfers of control. In the following discussion we assume that there is some agent  $A$  that can always make the decision instantly.

To calculate the EU of an arbitrary strategy, we multiply the probability of response at each instant of time with the expected utility of receiving a response

at that instant, and then sum the products. Hence, for an arbitrary continuous probability distribution:

$$EU = \int_0^{\infty} P_{\top}(t) EU_{e_c}^d(t) .dt \quad (1)$$

where  $e_c$  represents the entity currently in decision-making control.

Since we are primarily interested in the effects of delayed response, we can decompose the expected utility of a decision at a certain instant,  $EU_{e_c}^d(t)$ , into two terms. The first term captures the quality of the decision, independent of delay costs, and the second captures the costs of delay, i.e.,:  $EU_{e_c}^d t = EQ_e^d(t) - \mathcal{W}(t)$ . A  $\mathcal{D}$  action affects the future cost of waiting. For example, the wait cost after performing a  $\mathcal{D}$  at  $t = \Delta$  at cost  $\mathcal{D}_{cost}$  is :  $\mathcal{W}(t|\mathcal{D}) = \mathcal{W}(\Delta) - \mathcal{W}(\Delta - \mathcal{D}_{value}) + \mathcal{W}(t' - \mathcal{D}_{value}) + \mathcal{D}_{cost}$ .

To calculate the EU of a strategy, we need to ensure that the probability of response function and the wait-cost calculation reflect the control situation at that point in the strategy. For example, if the user has control at time  $t$ ,  $P_{\top}(t)$  should reflect the user's probability of responding at  $t$ . To do this simply, we can break the integral from Equation 1 into separate terms, with each term representing one segment of the strategy, e.g., for a strategy  $eA$  there would be one term for when  $e$  has control and another for when  $A$  has control.

Using this basic technique, we can now write down the equations for some general transfer-of-control strategies. Equations 2-5 are the general EU equations for the AA strategies  $A$ ,  $e$ ,  $eA$  and  $eDeA$  respectively. We create the equations by writing down the integral for each of the segments of the strategy, as described above.  $T$  is the time when the agent takes control from the user, and  $\Delta$  is the time at which the  $\mathcal{D}$  occurs. One can write down the equations for more complex strategies in the same way.

In the case of the large team of WASMs, we currently use simple models of  $EQ_e^d(t)$ ,  $\mathcal{W}(t)$  and  $P_{\top}(t)$ . We assume that each human expert is equally capable of making any meta-reasoning decision. In each case, wait costs accrue linearly, i.e., the wait costs accrued in the second minute are the same as those accrued in the first minute. We assume a Markovian response probability from the human, though this is a model that future work could dramatically improve. For two of the three meta-reasoning decisions, we model the quality of the agent's autonomous reasoning as improving over time. This is not because the reasoning changes, but because the default response is more likely correct the more time that passes. Specifically, over time it makes more sense to let a role allocation continue, since there has more likely been changes in the team that allow it to complete. Likewise, terminating a long running plan is more reasonable as time passes, since it becomes more likely that something is actually preventing completion of the plan. However, notice that the rate at which the quality of the autonomous reasoning for the long running plan increases is much slower than for the role allocation.  $EQ_{human}^d(t)$  is highest for the long running plan, since it is a relatively decoupled decision that requires the expertise of the human, whereas the other decisions are more tightly coupled with other coordination

$$EU_A^d = EQ_A^d(0) - \mathcal{W}(0) \quad (2)$$

$$EU_e^d = \int_0^T P_\top(t) \times (EQ_e^d(t) - \mathcal{W}(t)).dt + \int_T^\infty P_\top(t) \times (EQ_e^d(t) - \mathcal{W}(D)).dt \quad (3)$$

$$EU_{eA}^d = \int_0^T P_\top(t) \times (EQ_e^d(t) - \mathcal{W}(t)).dt + \int_T^\infty P_\top(t).dt \times (EQ_a^d(T) - \mathcal{W}(T)) \quad (4)$$

$$EU_{U\mathcal{D}eA}^d = \int_0^\Delta P_\top(t)(EQ_e^d(t) - \mathcal{W}(t)).dt + \int_\Delta^T P_\top(t)(EQ_e^d(t) - \mathcal{W}(\Delta) + \mathcal{W}(\Delta - \mathcal{D}_{value}) - \mathcal{W}(t - \mathcal{D}_{value}) - \mathcal{D}_{cost}).dt + \int_T^\infty P_\top(t)(EQ_A^d(t) - \mathcal{W}(\Delta) + \mathcal{W}(\Delta - \mathcal{D}_{value}) - \mathcal{W}(T - \mathcal{D}_{value}) - \mathcal{D}_{cost}).dt \quad (5)$$

**Table 1.** General AA EU equations for simple transfer of control strategies.

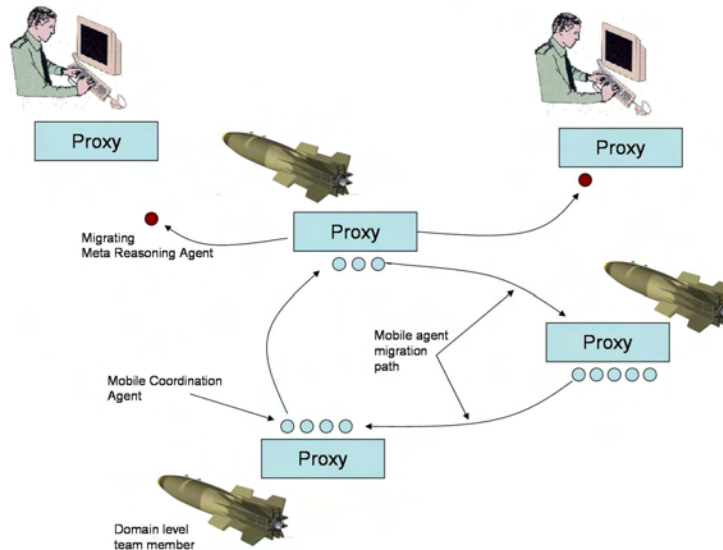
and involve more coordination expertise which the team can have. Over time, the autonomous reasoning to let an untasked agent stay where it is does not change, hence the autonomous model of the quality is a constant. Currently, there are no deadline delaying actions available to the team.

## 5 Agentifying Adjustable Autonomy

The coordination algorithms are encapsulated in a domain independant *proxy*[10, 27, 20, 24]. Such proxies are the emerging standard for implementing teamwork between heterogeneous team members. There is one proxy for each WASM and one for each human expert. The basic architecture is shown in Figure 2. The proxy communicates via a high level, domain specific protocol with either the WASM or human to collect information and convey requests from the team. The proxies communicate between themselves to facilitate the coordination. Most of the proxy code is domain independent and can be readily used in new domains requiring distributed control. The current version of the code, known as *Ma-chinetta*, is a substantially extended and updated version of the TEAMCORE proxy code[27]. TEAMCORE proxies implement *teamwork* as described by the

STEAM algorithms[26], which are in turn based on the theory of *joint intentions*[11, 3].

In a dynamic, distributed system, protocols for performing coordination need to be extremely robust, since the large numbers of agents ensures that anything that can go wrong typically does. When the size of a team is scaled to hundreds of agents, this does not simply imply the need to write “bug-free” code. Instead abstractions and designs that promote robustness are needed. Towards this end, we are encapsulating “chunks” of coordination in *coordination agents*. Each coordination agent manages one specific piece of the overall coordination. When control over that piece of coordination moves from one proxy to another proxy, the coordination agent moves from proxy to proxy, taking with it any relevant state information. There are coordination agents for each plan or subplan (PlanAgents), each role (RoleAgents) and each piece of information that needs to be shared (InformationAgents). For example, a RoleAgent looks after everything to do with a specific role. This encapsulation makes it far easier to build robust coordination. Since, the coordination agents actually implement the coordination, the proxy can be viewed simply as a mobile agent platform that facilitates the functioning of the coordination agents. However, the proxies play the additional important role of providing and storing local information.



**Fig. 2.** Overall system architecture.

To integrate the meta-reasoning into a Machinetta controlled team, an additional type of mobile agent is introduced, called an *adjustable autonomy mobile agent*. Each adjustable autonomy mobile agent has responsibility for a single

piece of interaction with the humans. In the case of coordination meta-reasoning, an adjustable autonomy mobile agent is created when a proxy detects that a situation has occurred that requires meta-reasoning. To get the decision made, the adjustable autonomy mobile agent creates and executes a transfer-of-control strategy. The adjustable autonomy mobile agent exists until a decision has been made, either autonomously or by some human.

Information is collected by the proxy as mobile agents of various types pass through to implement the coordination. Critically, information agents move information about the current workload of each human expert to the proxy of each other human expert. These models currently contain a simple numeric value for the human’s current workload, though more detailed models can be envisioned. When an adjustable autonomy agent is created it can first move to the proxy of any human expert to get the models it requires to create an optimal transfer-of-control strategy. Note that these models will not always be complete, but provide estimates of workload the agent can use to construct a strategy. In some cases, if the decision can easily be made autonomously, the adjustable autonomy agent will not move to the proxy of the human expert, because their specific availability is not important.

Importantly, as described above, simple reasoning for making the meta-reasoning decisions are encapsulated by the adjustable autonomy mobile agent so that a decision can be made autonomously if human expertise is not readily available or the decision is not sufficiently important. For example, the reasoning on unallocated roles is to let the role “continue” in an attempt to allocate it.

Figure 3 shows the interface for presenting decisions to human experts. Notice, that while the specific information about the information is presented, contextual information is not. This is a serious shortcoming of this work to date and a key area for future work. Presenting enough context to allow the human to make effective decisions will involve both getting the information to the expert and, more challengingly, presenting it in an appropriate manner.

## 6 Preliminary Experiments

Objective evaluation of the approach described above is difficult for two reasons. First, improvements in the performance of the team will rely heavily on the interfaces that present the decisions and related information to the human. Such interfaces are not a focus of this work. More pragmatically, a simulation run with a reasonable number of WASMs takes around 30 minutes to run, hence to run a meaningful set of experiments would take an large amount of human time. Moreover, domain experts required to make such decisions are hard to come by. While such experiments will eventually have to be run to validate the effectiveness of this approach, our initial experiments do not rely on human subjects. Instead, the object of these experiments is to evaluate how the underlying algorithms work in finding potential team problems and deal with the possibility that a human is not available to make these decisions when they arise. These



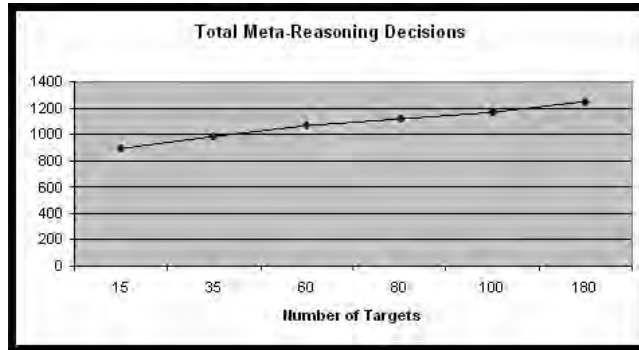


**Fig. 3.** A view of the interface the human uses to make decisions. Three decisions are shown. At the top is a decision whether to cancel a long running plan. Below are two unallocated roles from the same plan, for destroying a particular target.

preliminary experiments form a basis for future experiments, involving actual human subjects.

To remove the need for many hours of human input, the interfaces were augmented with code that made decisions as if they were made by the human. These “human” decisions were made between five seconds and two minutes after control was transferred to the human. The experiments involved a team of 80 WASMs operating in a large environment. The primary task of the team was to protect a manned aircraft by finding and destroying surface-to-air missile sites spread around the environment. Half the team spread out across the environment searching for targets while the other half stayed near the manned aircraft destroying surface-to-air sites as they were found near the aircraft. Plans were simple, requiring a single WASM to hit each found target. If a target was not hit within three minutes of being found, this was considered abnormal plan execution and meta-reasoning would be invoked. Meta-reasoning was also invoked when a WASM was not allocated to hit any target for five minutes. These times are low, but reasonable since the simulation ran at approximately four times real-time. Finally, meta-reasoning was invoked when no WASM was available to hit a found target. Two human commanders were available to make meta-reasoning decisions (although, as discussed above there were not “real” human commanders).

Six different scenarios were used, each differing the number of surface-to-air missile sites. Each configuration was run ten times, thus the results below represent around 30 hours of simulation time (120 hours of real-time). As the number of missile sites increases, the team will have more to do with the same number of WASMs, thus we can expect more meta-reasoning decisions.

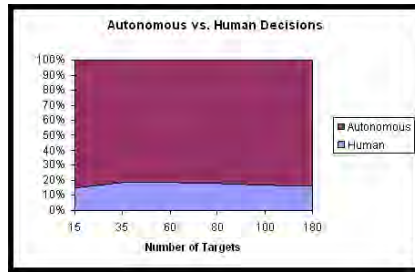


**Fig. 4.** The number of meta-reasoning decisions to be made as the number of targets in the environment increases.

Figure 4 shows that the total number of meta-reasoning decisions does increase with the number of targets. Over the course of a simulation, there are around 100 meta-reasoning decisions or about one per agent. However, as Figure 5 shows, only about 20% of these get transferred to a human. The large number of decisions that are made autonomously is primarily because humans are not available to make those decisions. This suggests work may need to be done to prioritize decisions for a user, to prevent high priority decisions being left to an agent, while the user is busy with low priority decisions. However, an appropriate solution is not obvious, since new decisions arrive asynchronously and it will likely not be appropriate to continually change the list of decisions the human is working on. Finally, notice in Figure 6 that a large percentage of the meta-decisions are to potentially cancel long running plans. The large number of such decisions illustrates a need to carefully tune the meta-reasoning heuristics in order to avoid overloading the system with superfluous decisions. However, in this specific case, the problem of deciding whether to cancel a long running plan was the most appropriate for the human, hence the large percentage of such decisions for the human is reasonable.

## 7 Conclusions and Future Directions

This chapter presents an integrated approach to leveraging human expertise to improve the coordination of a large team. Via the use of coordination meta-reasoning, key decisions could be brought to the attention of human experts. Transfer-of-control strategies ensured that miscoordination was not caused by delays waiting for human input. The approach was encapsulated in adjustable autonomy agents that are part of the Machinetta proxy approach to coordination. Initial experiments showed that the approach was able to balance the need for human input against the potential for overloading them. Further experiments



**Fig. 5.** The percentage of decisions transferred to humans versus the percentage made autonomously.



**Fig. 6.** Ratios of different types of meta-reasoning decisions presented to the user.

are needed to understand whether the opportunity for commanders to give input will actually improve the performance of the team.

While this initial work brings together several key components in an effective way, use of these techniques in the context of a large team raises some questions for which we do not yet have answers. One key question is how to handle the fact that meta-reasoning decisions are not independent, hence the transfer-of-control strategies for different decisions should perhaps not be independent. Another issue is how to ensure that the human is given appropriate information to allow a meta-reasoning decision to be made and how an agent could decide whether the human has the required information to make an appropriate decision. Although others have done some work in this area[1, 8], large scale coordination raises new issues. Other work has highlighted the importance of interfaces in good interaction[25, 14, 13, 7] which also must be addressed by this work.

## References

1. Mark H. Burstein and David E. Diller. A framework for dynamic information flow in mixed-initiative human/agent organizations. *Applied Intelligence on Agents and Process Management*, 2004. Forthcoming.
2. Richard Clark. *Uninhabited Combat Air Vehicles: Airpower by the people, for the people but not with the people*. Air University Press, 2000.
3. Philip R. Cohen and Hector J. Levesque. Teamwork. *Nous*, 25(4):487–512, 1991.
4. Defense Science Board. Defense science board study on unmanned aerial vehicles and uninhabited combat aerial vehicles. Technical report, Office of the Under Secretary of Defense for Acquisition, Technology and Logistics, 2004.
5. G. Ferguson, J. Allen, and B. Miller. TRAINS-95 : Towards a mixed-initiative planning assistant. In *Proceedings of the Third Conference on Artificial Intelligence Planning Systems*, pages 70–77, May 1996.
6. Dani Goldberg, Vincent Cicirello, M Bernardine Dias, Reid Simmons, Stephen Smith, and Anthony (Tony) Stentz. Market-based multi-robot planning in a distributed layered architecture. In *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2003 International Workshop on Multi-Robot Systems*, volume 2, pages 27–38. Kluwer Academic Publishers, 2003.

7. M. Goodrich, D. Olsen, J. Crandall, and T. Palmer. Experiments in adjustable autonomy. In H. Hexmoor, C. Castelfranchi, R. Falcone, and M. Cox, editors, *Proceedings of IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents*, 2001.
8. T. Hartrum and S. DeLoach. Design issues for mixed-initiative agent systems. In *Proceedings of AAAI workshop on mixed-initiative intelligence*, 1999.
9. Bryan Horling, Roger Mailler, Mark Sims, and Victor Lesser. Using and maintaining organization in a large-scale distributed sensor network. In *In Proceedings of the Workshop on Autonomy, Delegation, and Control (AAMAS03)*, 2003.
10. N. Jennings. The archon systems and its applications. Project Report, 1995.
11. N. R. Jennings. Specification and implementation of a belief-desire-joint-intention architecture for collaborative problem solving. *Intl. Journal of Intelligent and Cooperative Information Systems*, 2(3):289–318, 1993.
12. Hiroaki Kitano, Satoshi Tadokoro, Itsuki Noda, Hitoshi Matsubara, Tomoichi Takahashi, Atsushi Shinjoh, and Susumu Shimada. Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *Proc. 1999 IEEE Intl. Conf. on Systems, Man and Cybernetics*, volume VI, pages 739–743, Tokyo, October 1999.
13. D. Kortenkamp, D. Schreckenghost, and C. Martin. User interaction with multi-robot systems. In *Proceedings of Workshop on Multi-Robot Systems*, 2002.
14. J. Malin, C. Thronesbery, and D. Schreckenghost. Progress in human-centered automation: Communicating situation information. 1996”.
15. Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. An asynchronous complete method for distributed constraint optimization. In *Proceedings of Autonomous Agents and Multi-Agent Systems*, 2003.
16. R. Nair, M. Tambe, and S. Marsella. Role allocation and reallocation in multiagent teams: Towards a practical analysis. In *Proceedings of the second International Joint conference on agents and multiagent systems (AAMAS)*, 2003.
17. Committee on Visionary Manufacturing Challenges. Visionary manufacturing challenges for 2020. National Research Council.
18. Regis Vincent Paul Scerri and Roger Mailler, editors. *Proceedings of AAMAS’04 Workshop on Challenges in the Coordination of Large Scale MultiAgent Systems*, 2004.
19. John L. Pollack. The logical foundations of goal-regression planning in autonomous agents. *Artificial Intelligence*, 106:267–334, 1998.
20. David Pynadath and Milind Tambe. Multiagent teamwork: Analyzing the optimality and complexity of key theories and models. In *First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS’02)*, 2002.
21. D.V. Pynadath, M. Tambe, N. Chauvat, and L. Cavedon. Toward team-oriented programming. In *Intelligent Agents VI: Agent Theories, Architectures, and Languages*, pages 233–247, 1999.
22. Paul Ranky. *An Introduction to Flexible Automation, Manufacturing and Assembly Cells and Systems in CIM (Computer Integrated Manufacturing), Methods, Tools and Case Studies*. CIMware, 1997.
23. P. Scerri, D. Pynadath, and M. Tambe. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 17:171–228”, 2002.
24. P. Scerri, D. V. Pynadath, L. Johnson, P. Rosenbloom, N. Schurr, M. Si, and M. Tambe. A prototype infrastructure for distributed robot-agent-person teams. In *The Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003.

25. C. Thorpe T. Fong and C. Baur. Advanced interfaces for vehicle teleoperation: collaborative control, sensor fusion displays, and web-based tools. In *Vehicle Teleoperation Interfaces Workshop, IEEE International Conference on Robotics and Automation*, San Fransisco, CA, April 2000.
26. Milind Tambe. Agent architectures for flexible, practical teamwork. *National Conference on AI (AAAI97)*, pages 22–28, 1997.
27. Milind Tambe, Wei-Min Shen, Maja Mataric, David Pynadath, Dani Goldberg, Pragnesh Jay Modi, Zhun Qiu, and Behnam Salemi. Teamwork in cyberspace: using TEAMCORE to make agents team-ready. In *AAAI Spring Symposium on agents in cyberspace*, 1999.
28. M. Veloso, A. Mulvehill, and M. Cox. Rationale-supported mixed-initiative case-based planning. In *Proceedings of the fourteenth national conference on artificial intelligence and ninth innovative applications of artificial intelligence conference*, pages 1072–1077, 1997.
29. Alan Vick, Richard M. Moore, Bruce R. Pirnie, and John Stillion. *Aerospace Operations Against Elusive Ground Targets*. RAND Documents, 2001.

# Intentional agents in defense

Emiliano Lorini and Cristiano Castelfranchi

Institute of Cognitive Sciences and Technologies-CNR  
Via San Martino della Battaglia 44, 00185, Roma, Italy

**Abstract.** Multi-agent systems (MAS) should not be conceived as only cooperative. As open systems situations of concurrence, competition and conflict often arise. Starting from this perspective it is relevant not only pro-social interaction modeling, but also a theory of trust and monitoring, giving special relevance to issues of security and defense: how can an agent prevent that dangerous actions of other agents and dangerous events will frustrate his goals? In this paper some relevant concepts for a general model of defense in intentional agents are analyzed and formally specified. Moreover an ontology of defensive goals and defensive strategies is studied.

## 1 Introduction

Security is a matter of defense and protection. More precisely it is a defensive concept. In fact security means to be safe, not be exposed to damages and harms, to be in a completely safe, reliable and trustworthy environment where there are solutions for protecting ourselves from possible attacks and dangerous events.

A safe agent is either an agent who does not need to pursue defensive strategies in order to achieve his goals and to accomplish his tasks or an agent who is capable of blocking and contrasting possible attacks (viz. an agent having the abilities and opportunities to perform defensive actions) and who can exploit other agents, structures, artifacts and institutions in order to prevent, discourage and block possible attacks.

Obviously the former is a very implausible condition, since environments are always uncertain and dynamic, and agents generally act in social contexts where goals and interests are often divergent and conflicts can easily emerge [1].

Thus, we must conclude that: *a principled approach to security requires a careful analysis of defense.*

Since we believe that a general theory of defense is still missing, in this paper we will try to fill such a gap by taking the first steps towards the development of a formal model of defense and an ontology of defensive goals and defensive strategies. We will use a multi-modal logic of time and action and we will explicitly model informational attitudes (beliefs and expectations) and motivational attitudes (goals and intentions) of agents.

The application of modal logic to the analysis of issues concerning security is not new in literature. For instance in [2, 3, 4, 5] specific epistemic logics, collectively referred to as authentication logics, have been proposed to deal with authentication issues. Such modal logics have been developed as tools for verifying the correctness of security

protocols, where one wants to ensure that agents obtain certain knowledge over time and that ignorance of potential intruders persists over the whole run of a protocol.

Our objective in this paper is different from the objective of authors working on logics of authentication. We are mainly interested in providing a conceptual analysis of defense for agents who act according to their beliefs and motivations. Indeed we think that, due the strict theoretical connection between security and defense, models and methodologies of security would stongly profit by this kind of investigation. We believe that the framework of multi-agent systems and its formal models of intentional agency are the most suitable to develop such a kind of analysis.

More precisely, we will try to clarify the following points at a high level of abstraction.

- Under which conditions should an agent defend himself from someone else, that is, what should an agent expect, want, believe, etc... before deciding to pursue a defensive a strategy?
- Which are the main types of defensive strategies and how do defensive strategies vary depending on the context and situation of attack?

In this work we suppose that defenders are intentional agents with specific kinds of defensive goals and expectations of attacks. This is somehow a quite restrictive assumption. Indeed elementary reactive agents too can defend themselves.<sup>1</sup> A more general theory of defense should consider intentional defensive behaviors as well as functional defensive behaviors. For instance a possible restricted meaning of agent  $i$ 's escape is the act of agent  $i$  driven by  $i$ 's goal of changing his spatial location in order to avoid the impact with an object, event, other agent, etc... The notion of escape can be extended to cover functional behaviors of elementary agents where the intention to escape is substituted either by the designed function or by the function acquired through evolution or reinforcement learning.

## 2 A logic of defense

In order to formalize some relevant concepts in our ontology of defense we exploit a very simple modal logic of time, action and mental states. We call this logic  $\mathcal{DL}$  (*Defense Logic*).  $\mathcal{DL}$  is based on a combination of a fragment of linear temporal logic [6], a fragment of dynamic logic [7] and Cohen and Levesque's logic of beliefs and intentions [8].

In  $\mathcal{DL}$  there are two modal operators  $Bel$  and  $Goal$  for mental states. The former modal operator is a standard operator for beliefs [9] and expresses what a given agent currently believes.

The modal operator  $Goal$  refers to goals of an agent. We suppose that goals are consistent (viz. an agent cannot decide to pursue two goals which cannot be achieved at the same time).

In the basic version of  $\mathcal{DL}$  we cannot reason about conflicting goals and goals which are incompatible with actual beliefs.

The primitives of the logic are the following:

---

<sup>1</sup> Nevertheless there are defensive strategies such as dissuasion which have a specific intentional connotation.

- a set of agents  $AGT = \{i, j, \dots\}$ ;
- a set of atomic actions  $ACT = \{\alpha, \beta, \dots\}$ ;
- a set of propositional atoms  $\Pi = \{p, q, \dots\}$ .

The set of well formed formulas  $\varphi, \psi$  of the language  $\mathcal{L}_{\mathcal{DL}}$  is defined by the following BNF:

$$\varphi := p \mid \top \mid \neg\varphi \mid \varphi \wedge \psi \mid [i : \alpha] \psi \mid \bigcirc \varphi \mid Bel_i \varphi \mid Goal_i \varphi$$

where  $p$  ranges over  $\Pi$  and  $\alpha$  ranges over  $ACT$ .

$Bel_i \varphi$  is read “agent  $i$  believes that  $\varphi$ ” whereas  $Goal_i \varphi$  is read “agent  $i$  has goal that  $\varphi$ ”.  $\bigcirc$  is a standard *next* modal operator of temporal logic ( $\bigcirc \varphi$  is read “ $\varphi$  is going to hold at the next state”) whilst  $[i : \alpha]$  is a standard operator of dynamic logic and  $[i : \alpha] \varphi$  is read “ $\varphi$  holds after every occurrence of agent  $i$ ’s action  $\alpha$ ”. Hence  $[i : \alpha] \perp$  expresses “agent  $i$  does not do action  $\alpha$ ”. We use the following abbreviation:  $\langle i : \alpha \rangle \varphi =_{def} \neg [i : \alpha] \neg \varphi$ . Hence  $\langle i : \alpha \rangle \varphi$  has to be read “agent  $i$  does action  $\alpha$  and  $\varphi$  holds after this action”. Finally  $\langle i : \alpha \rangle \top$  has to be read “agent  $i$  does action  $\alpha$ ”.

## 2.1 Basic semantics

A model of  $\mathcal{DL}$  is defined by a tuple  $M = \langle W, R_{\bigcirc}, R^{att}, B, G, V \rangle$  where:

- $W$  is a set of worlds.
- $R_{\bigcirc}$  is a mapping  $R_{\bigcirc} : W \longrightarrow 2^W$  associating sets of possible worlds  $R_{\bigcirc}(w)$  to each possible world  $w$ .
- $R^{att}$  is a mapping  $R^{att} : AGT \times ACT \longrightarrow (W \longrightarrow 2^W)$  associating sets of possible worlds  $R_{i:\alpha}^{att}(w)$  to each possible world  $w$ .
- $B$  is a mapping  $B : AGT \longrightarrow (W \longrightarrow 2^W)$  associating sets of possible worlds  $B_i(w)$  to each possible world  $w$ . For each possible world  $w$  there is an associated set of possible worlds  $B_i(w) \subseteq W$ : the worlds that are compatible with the agent’s beliefs.
- $G$  is a mapping  $G : AGT \longrightarrow (W \longrightarrow 2^W)$  associating sets of possible worlds  $G_i(w)$  to each possible world  $w$ . For each possible world  $w$  there is an associated set of possible worlds  $G_i(w) \subseteq W$ : the worlds that are compatible with agent  $i$ ’s goals.
- $V$  is a mapping  $V : \Pi \longrightarrow 2^W$  associating sets of possible worlds to propositional atoms.

## 2.2 Truth conditions

- $M, w \models p \iff w \in V(p)$ .
- $M, w \models \neg\varphi \iff \text{not } M, w \models \varphi$ .
- $M, w \models \varphi \wedge \psi \iff M, w \models \varphi \text{ and } M, w \models \psi$ .
- $M, w \models \bigcirc \varphi \iff \forall w' \text{ if } w' \in R_{\bigcirc}(w) \text{ then } M, w' \models \varphi$ .
- $M, w \models [i : \alpha] \psi \iff \forall w' \text{ if } w' \in R_{i:\alpha}^{att}(w) \text{ then } M, w' \models \psi$ .
- $M, w \models Bel_i \varphi \iff \forall w' \text{ if } w' \in B_i(w) \text{ then } M, w' \models \varphi$ .
- $M, w \models Goal_i \varphi \iff \forall w' \text{ if } w' \in G_i(w) \text{ then } M, w' \models \varphi$ .



### 3 Axiomatization

We take the following complete axiomatization of our simple modal logic of time, action and mental states.

0. All tautologies of propositional calculus
1. $\bigcirc(\varphi \rightarrow \psi) \rightarrow (\bigcirc\varphi \rightarrow \bigcirc\psi)$
2. $\bigcirc\neg\varphi \leftrightarrow \neg\bigcirc\varphi$
3. $[i : \alpha](\varphi \rightarrow \psi) \rightarrow ([i : \alpha]\varphi \rightarrow [i : \alpha]\psi)$
4. $\bigcirc\varphi \rightarrow [i : \alpha]\varphi$
5. $Bel_i(\varphi \rightarrow \psi) \rightarrow (Bel_i\varphi \rightarrow Bel_i\psi)$
6. $\neg(Bel_i\varphi \wedge Bel_i\neg\varphi)$
7. $Bel_i\varphi \rightarrow Bel_iBel_i\varphi$
8. $\neg Bel_i\varphi \rightarrow Bel_i\neg Bel_i\varphi$
9. $Goal_i(\varphi \rightarrow \psi) \rightarrow (Goal_i\varphi \rightarrow Goal_i\psi)$
10. $\neg(Goal_i\varphi \wedge Goal_i\neg\varphi)$
11. $Goal_i\varphi \rightarrow Goal_iGoal_i\varphi$
12. $\neg Goal_i\varphi \rightarrow Goal_i\neg Goal_i\varphi$
13. $Goal_i\varphi \rightarrow Bel_iGoal_i\varphi$
14. $\neg Goal_i\varphi \rightarrow Bel_i\neg Goal_i\varphi$
15. $Bel_i\varphi \rightarrow Goal_i\varphi$
<b>Rules of Inference</b>
R1. $\frac{\vdash\varphi \vdash\varphi \rightarrow \psi}{\vdash\psi}$ (Modus Ponens)
R2. $\frac{\vdash\varphi}{\vdash\bigcirc\varphi}$ ( $\bigcirc$ -Necessitation)
R3. $\frac{\vdash\varphi}{\vdash[i:\alpha]\varphi}$ ( $[i:\alpha]$ -Necessitation)
R4. $\frac{\vdash\varphi}{\vdash Bel_i\varphi}$ ( $Bel_i$ -Necessitation)
R5. $\frac{\vdash\varphi}{\vdash Goal_i\varphi}$ ( $Goal_i$ -Necessitation)

Table 1. Axiomatization.

Axiom 1 and rule of inference R2 define a minimal normal modal logic for the temporal operator  $\bigcirc$ . Axiom 2 expresses the interpretation of  $\bigcirc$  by a total function:

- for every  $w \in W$  if  $w' \in R_{\bigcirc}(w)$  and  $w'' \in R_{\bigcirc}(w)$  then  $w' = w''$  and for every  $w \in W$ ,  $R_{\bigcirc}(w) \neq \emptyset$ .

Axiom 3 and rule of inference R3 define a minimal normal modal logic for the operator  $[i : \alpha]$ .

Axiom 4 is a connection axiom time-attempt. A similar axiom concerning the connection between time and action has been studied in [10, 11]. The semantic counterpart of axiom 4 is:

- $R_{i:\alpha}^{at}(w) \subseteq R_{\bigcirc}(w)$ .

Thus the set of worlds which are accessible from world  $w$  via an attempt to do action  $\alpha$  is a subset of the set of next-worlds.

Axioms 5 and 9 with rules of inference R4 and R5 define a minimal normal modal logic for the operators  $Bel_i$  and  $Goal_i$ . Axioms 6, 7, 8, 10, 11, 12 express the interpretations of  $B_i$  and  $G_i$  by serial, transitive and euclidean functions:

- *Seriality* of  $B_i$ : for every  $w \in W$   $B_i(w) \neq \emptyset$
- *Seriality* of  $G_i$ : for every  $w \in W$   $G_i(w) \neq \emptyset$ .
- *Transitivity* of  $B_i$ : for every  $w \in W$ , if  $w' \in B_i(w)$  and  $v \in B_i(w')$  then  $v \in B_i(w)$
- *Transitivity* of  $G_i$ : for every  $w \in W$ , if  $w' \in G_i(w)$  and  $v \in G_i(w')$  then  $v \in G_i(w)$
- *Euclideanity* of  $B_i$ : for every  $w \in W$  if  $v, v' \in B_i(w)$  then  $v' \in B_i(v)$  and  $v \in B_i(v')$
- *Euclideanity* of  $G_i$ : for every  $w \in W$  if  $v, v' \in G_i(w)$  then  $v' \in G_i(v)$  and  $v \in G_i(v')$

Axiom 13 is an axiom of positive introspection for goals similar to axiom 7 for beliefs. Axiom 14 is its negative version (the negative version of axiom 7 is axiom 8). According to axioms 13 and 14, worlds that are compatible with agent  $i$ 's goals are compatible with agent  $i$ 's goals from those worlds which are compatible with agent  $i$ 's beliefs, that is:

- for every  $w \in W$  if  $w' \in B_i(w)$  then  $G_i(w) = G_i(w')$ .

Finally, 15 is the *strong realism* axiom studied in [8, 12, 13]. According to this axiom the set of worlds which are compatible with the agent's goals is a subset of the set of worlds which are considered possible by the agent, that is:

- $G_i(w) \subseteq B_i(w)$ .

### 3.1 Validity and satisfiability

We call  $\mathcal{DL}$  the logic axiomatized by the previous axioms 0-15 and rules of inference R1-R5. We call  $\mathcal{DL}$  models the class of models satisfying all the semantic constraints imposed in the previous section.

We write  $\vdash_{\mathcal{DL}} \varphi$  if formula  $\varphi$  is a theorem of  $\mathcal{DL}$ , viz. if  $\varphi$  is a logical consequence of the set of axioms 0-15 and rules of inference R1-R5.

Moreover, we write  $M \models \varphi$  if formula  $\varphi$  is *valid* in the  $\mathcal{DL}$  model  $M$ , viz.  $M, w \models \varphi$  for every world  $w$  in  $M$ .

We write  $\models_{\mathcal{DL}} \varphi$  if formula  $\varphi$  is *valid* in all  $\mathcal{DL}$  models, viz.  $M, w \models \varphi$  for every  $\mathcal{DL}$  model  $M$  and world  $w$  in  $M$ .

Finally, we say that a formula  $\varphi$  is *satisfiable* in a model  $M$  if there is some world in  $M$  at which  $\varphi$  is true, viz. there exists a world  $w$  in  $M$  such that  $M, w \models \varphi$ .

Now, we can prove that  $\mathcal{DL}$  is *sound* and *complete* with respect to the class of models satisfying all the semantic constraints imposed in the previous section.

**Theorem 1.** *Soundness and completeness.*

$$\vdash_{\mathcal{DL}} \varphi \text{ iff } \models_{\mathcal{DL}} \varphi$$

*Proof.* All axioms and inference rules are in the Sahlqvist class, for which a general algorithm to compute their semantic counterparts exists. Therefore it is a routine to verify that each axiom in 1-15 corresponds to the semantic properties described in the previous section. Furthermore, a general completeness result exists for logics whose axioms are in the Sahlqvist class [14, 15]. Therefore we can conclude that  $\mathcal{DL}$  is complete.  $\square$

## 4 Expected dangers, dangerous situations and expected attacks

Our general aim is to make clear some categories and concepts which are fundamental for a model of intentional agency with defensive capabilities.

We begin with the assumption that always a defense taken by an agent  $i$  implies that agent  $i$  intends either to achieve or to maintain a certain result  $\varphi$  and agent  $i$  believes that there will be a threat on it.

We suppose that an agent can defend himself from something only if he has predictive capacities. More precisely, if an agent  $i$  is defending himself from someone then  $i$  expects that there is an action of another agent  $j$  which can possibly interfere with the achievement of his goals. In our view a defense always implies an expectation concerning a possible threat or a possible danger, viz. the expectation that an external event can compromise the achievement of our goals.<sup>2</sup>

We can use the formal logic presented in the previous section in order to formalize such an expectation which is always involved in a situation of defense. First of all let us introduce a notational convention.

We write  $\bigcirc^n \varphi$  to indicate that the sentence  $\varphi$  is subject to  $n$  iterations of the modality  $\bigcirc$  where  $n$  can be any number  $0, 1, 2, 3, \dots$ . Therefore  $0$  is just  $\bigcirc^0 \varphi$ ,  $1$  is  $\bigcirc \varphi$ , and so on. More formally,  $\bigcirc^n \varphi$  can be defined inductively by:

1.  $\bigcirc^0 \varphi =_{def} \varphi$ ;
2.  $\bigcirc^{n+1} \varphi =_{def} \bigcirc \bigcirc^n \varphi$ .

The first concept we are aimed at formalizing is the concept of expected danger.

**Definition 1.** *Expected danger.*

$$ExpDanger(i, j, \alpha, \varphi, \psi, n) =_{def} Bel_i(\psi \rightarrow \bigcirc^n [j : \alpha] \neg \varphi) \wedge Bel_i Goal_i \bigcirc^{n+1} \varphi$$

$ExpDanger(i, j, \alpha, \varphi, \psi, n)$  reads: 1) agent  $i$  expects that, under certain conditions  $\psi$ , if  $n$  steps from now agent  $j$  does action  $\alpha$  then  $\neg \varphi$  will hold after  $\alpha$ 's occurrence and; 2) agent  $i$  believes that he wants  $\varphi$  to be true  $n+1$  steps from now. An alternative reading is: agent  $i$  expects that, under certain conditions  $\psi$ , if  $n$  steps from now agent  $j$  does action  $\alpha$  then he will interfere negatively with the achievement of  $i$ 's actual goal that  $n+1$  steps from now  $\varphi$  will be true. Therefore  $ExpDanger(i, j, \alpha, \varphi, \psi, n)$  expresses agent  $i$ 's expectation that the future occurrence of agent  $j$ 's action  $\alpha$  is a danger for him since, given certain conditions  $\psi$ , if  $n$  steps from now  $j$ 's action  $\alpha$  occurs then it will compromise the achievement of his goal that  $n+1$  steps from now  $\varphi$  will be true.

For example,  $ExpDanger(Mary, Fred, shoot, MaryAlive, inFrontFred, 0)$  means: Mary expects that the occurrence of Fred's action of shooting is a danger for her since

---

<sup>2</sup> In our view expectations are a necessary mental ingredients of a BDI like agent. In a previous work [16] we did not introduce expectations as an additional primitive. We preferred to build those mental states on former ingredients (beliefs and goals) in order to have mental states that preserve both properties, epistemic and conative. In the present analysis we make the same kind of assumption by building expectation on the basis of more elementary ingredients (beliefs and goals).

if Fred's action of shooting occurs when she is in front of Fred then Fred's action will compromise the achievement of her goal to be alive next.

Starting from the previous definition of expected danger, we can characterize two more specific notions: the notion of expected attack and the notion of expected dangerous situation.

In our vocabulary an agent  $i$  expects a certain attack if and only if he expects a certain danger and he believes that the danger in question is not simply a potential danger, but it is an actual and effective danger. The concept of expected attack is formalized according to the following definition 2.

**Definition 2.** *Expected attack.*

$$\begin{aligned} ExpAttack(i, j, \alpha, \varphi, \psi, n) &=_{def} \\ ExpDanger(i, j, \alpha, \varphi, \psi, n) \wedge Bel_i \bigcirc^n \langle j : \alpha \rangle \top \end{aligned}$$

$ExpAttack(i, j, \alpha, \varphi, \psi, n)$  reads: 1) agent  $i$  expects that, under certain conditions  $\psi$ , if  $n$  steps from now agent  $j$ 's action  $\alpha$  occurs then it will interfere negatively with the achievement of  $i$ 's actual goal that  $n+1$  steps from now  $\varphi$  will be true and; 2) agent  $i$  believes that  $n$  steps from now agent  $j$  will perform action  $\alpha$ . Therefore

$ExpAttack(i, j, \alpha, \varphi, \psi, n)$  expresses agent  $i$ 's expectation that the occurrence of agent  $j$ 's action  $\alpha$  is an actual and effective danger for him (viz. an attack towards him) since  $n$  steps from now  $j$  will do action  $\alpha$  and, given certain conditions  $\psi$ , if  $n$  steps from now  $j$  does action  $\alpha$  occurs then  $j$ 's action  $\alpha$  will compromise the achievement of  $i$ 's goal that  $n+1$  steps from now  $\varphi$  will be true.

For example,  $ExpAttack(Bill, thief, forceDoor, moneySafe, nobodyAtHome, 0)$  means: Bill expects that the occurrence of a thief's action of forcing the door of Bill's house is an attack towards him since the thief is going to force the front door and, if the thief's action of forcing the front door occurs when nobody is at home then such an action will compromise the achievement of Bill's goal to keep his money safe.

We suppose that an agent  $i$  expects to be in a dangerous situation if and only if he expects that if he will be attacked under the actual conditions  $\psi$  then one of his goals will be compromised.

The concept of expected dangerous situation is formalized according to the following definition 3.

**Definition 3.** *Expected dangerous situation.*

$$\begin{aligned} ExpDangerous(i, j, \alpha, \varphi, \psi, n) &=_{def} \\ ExpDanger(i, j, \alpha, \varphi, \psi, n) \wedge Bel_i \psi \end{aligned}$$

$ExpDangerous(i, j, \alpha, \varphi, \psi, n)$  reads: 1) agent  $i$  expects that, under certain conditions  $\psi$ , if  $n$  steps from now agent  $j$ 's action  $\alpha$  occurs then it will interfere negatively with the achievement of his actual goal that  $n+1$  steps from now  $\varphi$  will be true and; 2) agent  $i$  believes  $\psi$  to be true. Therefore  $ExpDangerous(i, j, \alpha, \varphi, \psi, n)$  expresses agent  $i$ 's thought that his future-oriented goal that  $\varphi$  will be compromised after

every future occurrence of agent  $i$ 's action  $\alpha$ .<sup>3</sup> Going back to one of the previous examples,  $ExpDangerous(Bill, thief, forceDoor, moneySafe, nobodyAtHome, 0)$  means: Bill thinks (expects) to be in a dangerous situation since nobody is at home and, if a thief forces the front door when nobody is at home then the thief's action of forcing will compromise the achievement of Bill's goal to keep his money safe.

A further relevant concept of a theory of defense is the concept of expected harm. We suppose that an agent  $i$  expects a future harm if and only if he expects that he will be attacked by another agent and if he will be attacked under the actual conditions then one of his goals will be compromised.

**Definition 4.** *Expected harm.*

$$ExpHarm(i, j, \alpha, \varphi, \psi, n) =_{def} ExpDanger(i, j, \alpha, \varphi, \psi, n) \wedge Bel_i \psi \wedge Bel_i \bigcirc^n \langle j : \alpha \rangle \top$$

We can easily prove that an expected harm implies an expectation of a future frustration of a goal. This is shown in the following theorem of  $\mathcal{DL}$ .

**Theorem 2.**  $\vdash_{\mathcal{DL}} ExpHarm(i, j, \alpha, \varphi, \psi, n) \rightarrow Bel_i(\bigcirc^{n+1} \neg \varphi \wedge Goal_i \bigcirc^{n+1} \varphi)$

*Proof.*  $ExpHarm(i, j, \alpha, \varphi, \psi, n)$  implies  $Bel_i(\psi \rightarrow \bigcirc^n [j : \alpha] \neg \varphi) \wedge Bel_i Goal_i \bigcirc^{n+1} \varphi \wedge Bel_i \psi \wedge Bel_i \bigcirc^n \langle j : \alpha \rangle \top$  (by definitions 1 and 4). Furthermore  $Bel_i(\psi \rightarrow \bigcirc^n [j : \alpha] \neg \varphi) \wedge Bel_i Goal_i \bigcirc^{n+1} \varphi \wedge Bel_i \psi \wedge Bel_i \bigcirc^n \langle j : \alpha \rangle \top$  implies  $Bel_i \bigcirc^n [j : \alpha] \neg \varphi \wedge Bel_i Goal_i \bigcirc^{n+1} \varphi \wedge Bel_i \bigcirc^n \langle j : \alpha \rangle \top$  (by axiom 5) which in turn implies  $Bel_i \bigcirc^n \langle j : \alpha \rangle \neg \varphi \wedge Bel_i Goal_i \bigcirc^{n+1} \varphi$  (by the equivalence  $\langle j : \alpha \rangle \top \wedge [j : \alpha] \neg \varphi \leftrightarrow \langle j : \alpha \rangle \varphi$ <sup>4</sup>). Finally  $Bel_i \bigcirc^n \langle j : \alpha \rangle \neg \varphi \wedge Bel_i Goal_i \bigcirc^{n+1} \varphi$  implies  $Bel_i \bigcirc^n \bigcirc \neg \varphi \wedge Bel_i Goal_i \bigcirc^{n+1} \varphi$  (by axiom 4 and rules of inference R2 and R4).  $\square$

According to theorem 1 if agent  $i$  expects a future harm then he believes that he will not achieve something he actually wants.

**Aggressions as intentional attacks** According to the previous definition 2, agent  $i$  expects an attack by  $j$  if and only if  $i$  expects that agent  $j$  will perform an action  $\alpha$  and the occurrence of  $\alpha$  will compromise  $i$ 's goals. There are more specific types of expected attack which can be reasonably called expected aggressions. Investigating such specific types of expected attack is crucial not only for a better understanding of defense but also for a comprehensive analysis of social interaction.

We suppose that agent  $i$  expects an aggression by  $j$  if and only: 1)  $i$  expects an attack from  $j$  since he expects that  $j$  will perform an action  $\alpha$  and  $i$  expects that the occurrence of  $j$ 's action  $\alpha$  will compromise  $i$ 's goal that  $\varphi$  will be true in the future; 2)  $i$  expects

<sup>3</sup> Indeed  $ExpDangerous(i, j, \alpha, \varphi, \psi, n)$  implies  $Bel_i(\bigcirc^n \langle j : \alpha \rangle \top \rightarrow \bigcirc^{n+1} \neg \varphi) \wedge Bel_i Goal_i \bigcirc^{n+1} \varphi$ .

<sup>4</sup> Verifying that such an equivalence is a theorem of  $\mathcal{DL}$  is straightforward (the proof is based on axiom 2 and axiom 4).

that  $j$  will perform action  $\alpha$  having the intention to do it; 3) according to  $i$ 's beliefs the possibility that agent  $j$  already intends to do action  $\alpha$  in the future is explained by the fact that  $j$  believes that  $i$  wants  $\varphi$  to be true in the future and by the fact that  $j$  believes that performing  $\alpha$  will bring about  $\neg\varphi$ .

Thus,  $i$  expects an aggression by  $j$  if and only if  $i$  expects a future intentional attack by  $j$  and  $i$  thinks that  $j$ 's attack towards  $i$  is explained by  $j$ 's beliefs that doing  $\alpha$  will harm  $i$  (viz.  $i$  thinks that  $j$  intends to do  $\alpha$  in order to harm  $i$ ).

The complex notion of expected aggression is formalized according the following definition.

**Definition 5.** *Expected aggression.*

$$\begin{aligned} ExpAggression(i, j, \alpha, \varphi, \psi, n) &=_{def} \\ &ExpAttack(i, j, \alpha, \varphi, \psi, n) \wedge \\ &Bel_i \bigcirc^n Goal_j \langle j : \alpha \rangle \top \wedge \\ &Bel_i(Goal_j \bigcirc^n \langle j : \alpha \rangle \top \rightarrow (Bel_j Goal_i \bigcirc^{n+1} \varphi \wedge Bel_j \bigcirc^n [j : \alpha] \neg\varphi)) \wedge \\ &\neg Bel_i \neg Goal_j \bigcirc^n \langle j : \alpha \rangle \top \end{aligned}$$

For example,  $ExpAggression(Mary, Fred, shoot, MaryAlive, inFrontFred, 4)$  means:

- 1) Mary expects an attack from Fred since she expects that 4 steps from now Fred will shoot and she expects that if 4 steps from now Fred shoots and Mary is in front of Fred then Mary will not be alive after Fred's action;
- 2) Mary expects that 4 steps from now Fred will shoot  $\alpha$  having the intention to shoot;
- 3) according to Mary's beliefs the possibility that Fred already intends to shoot in the future (4 steps from now) is explained by the fact that Fred believes that Mary wants to be alive in the future (5 steps from now) and by the fact that Fred believes that if 4 steps from now he does the action of shooting then Mary will not be alive afterward (viz. if 4 steps from now he does the action of shooting then 5 steps from now Mary will not be alive).

For summarizing, let us make explicit how the five concepts discussed in this section are organized from a logical point of view.

An expected attack is an expected danger

$$\vdash_{\mathcal{DL}} ExpAttack(i, j, \alpha, \varphi, \psi, n) \rightarrow ExpDanger(i, j, \alpha, \varphi, \psi, n)$$

An expected dangerous situation is an expected danger

$$\vdash_{\mathcal{DL}} ExpDangerous(i, j, \alpha, \varphi, \psi, n) \rightarrow ExpDanger(i, j, \alpha, \varphi, \psi, n)$$

Expecting a harm is equivalent to expecting both an attack and a dangerous situation

$$\begin{aligned} \vdash_{\mathcal{DL}} ExpHarm(i, j, \alpha, \varphi, \psi, n) &\leftrightarrow \\ &ExpDangerous(i, j, \alpha, \varphi, \psi, n) \wedge ExpAttack(i, j, \alpha, \varphi, \psi, n) \end{aligned}$$

An expected aggression is an expected attack

$$\vdash_{\mathcal{DL}} ExpAggression(i, j, \alpha, \varphi, \psi, n) \rightarrow ExpAttack(i, j, \alpha, \varphi, \psi, n)$$

#### 4.1 Defensive goals

An expectation of a possible danger is generally responsible for activating and generating defensive goals. In our view a defensive goal of an arbitrary agent  $i$  should be

conceived as a goal of agent  $i$  which is activated by  $i$ 's expectation of a possible danger. As we have shown in the previous section, when expecting a danger agent  $i$  thinks that in a certain situation a certain action of another agent will negatively interfere with the achievement of his goals. Thus, when expecting a danger agent  $i$  can act in different ways in order to escape the danger: either he can try to block the expected vehicle of attack (*block strategy*) or he can try to get out of the dangerous situation by preventing that the conditions of success of the expected attack are true (*protection strategy*). The following theorem of  $\mathcal{DL}$  shows which kind of defensive goal is activated in the mind of agent  $i$  when he expects an attack from an agent  $j$ .

**Theorem 3.**  $\vdash_{\mathcal{DL}} \text{ExpAttack}(i, j, \alpha, \varphi, \psi, n) \rightarrow \text{Goal}_i \neg \psi$

*Proof.*  $\text{ExpAttack}(i, j, \alpha, \varphi, \psi, n)$  implies  $\text{Bel}_i(\psi \rightarrow \bigcirc^n [j : \alpha] \neg \varphi) \wedge \text{Bel}_i \text{Goal}_i \bigcirc^{n+1} \varphi \wedge \text{Bel}_i \bigcirc^n \langle j : \alpha \rangle \top$  (by definitions 1 and 2) which in turn implies  $\text{Bel}_i(\psi \rightarrow \bigcirc^n [j : \alpha] \neg \varphi) \wedge \text{Goal}_i \bigcirc^{n+1} \varphi \wedge \text{Bel}_i \bigcirc^n \langle j : \alpha \rangle \top$  (by axioms 6 and 14). Moreover  $\text{Bel}_i(\psi \rightarrow \bigcirc^n [j : \alpha] \neg \varphi) \wedge \text{Goal}_i \bigcirc^{n+1} \varphi \wedge \text{Bel}_i \bigcirc^n \langle j : \alpha \rangle \top$  implies  $\text{Bel}_i(\bigcirc^n \langle j : \alpha \rangle \top \wedge (\psi \rightarrow \bigcirc^n [j : \alpha] \neg \varphi)) \wedge \text{Goal}_i \bigcirc^{n+1} \varphi$  (by standard modal principles) which in turn implies  $\text{Bel}_i(\psi \rightarrow \bigcirc^n (\langle j : \alpha \rangle \top \wedge [j : \alpha] \neg \varphi)) \wedge \text{Goal}_i \bigcirc^{n+1} \varphi$  (by standard modal principles and the equivalence  $\bigcirc^n \varphi \wedge \bigcirc^n \psi \leftrightarrow \bigcirc^n (\varphi \wedge \psi)$ <sup>5</sup>). Furthermore  $\text{Bel}_i(\psi \rightarrow \bigcirc^n (\langle j : \alpha \rangle \top \wedge [j : \alpha] \neg \varphi)) \wedge \text{Goal}_i \bigcirc^{n+1} \varphi$  implies  $\text{Bel}_i(\psi \rightarrow \bigcirc^n \langle j : \alpha \rangle \neg \varphi) \wedge \text{Goal}_i \bigcirc^{n+1} \varphi$  (by the equivalence  $\langle j : \alpha \rangle \top \wedge [j : \alpha] \neg \varphi \leftrightarrow \langle j : \alpha \rangle \neg \varphi$ ). Finally  $\text{Bel}_i(\psi \rightarrow \bigcirc^n \langle j : \alpha \rangle \neg \varphi) \wedge \text{Goal}_i \bigcirc^{n+1} \varphi$  implies  $\text{Bel}_i(\bigcirc^n [j : \alpha] \varphi \rightarrow \neg \psi) \wedge \text{Goal}_i \bigcirc^n \bigcirc \varphi$  (by the equivalence  $\neg \bigcirc^n \varphi \leftrightarrow \bigcirc^n \neg \varphi$ <sup>6</sup>) which in turn implies  $\text{Goal}_i(\bigcirc^n [j : \alpha] \varphi \rightarrow \neg \psi) \wedge \text{Goal}_i \bigcirc^n [j : \alpha] \varphi$  (by axioms 4 and 15 and rules of inference R2 and R5). From  $\text{Goal}_i(\bigcirc^n [j : \alpha] \varphi \rightarrow \neg \psi) \wedge \text{Goal}_i \bigcirc^n [j : \alpha] \varphi$  we can infer  $\text{Goal}_i \neg \psi$  (by axiom 9).  $\square$

According to the previous theorem if agent  $i$  thinks that the occurrence of agent  $j$ 's action  $\alpha$  under the conditions  $\psi$  is an actual and effective danger for him (viz. an attack towards him) then he comes to have the defensive goal of getting out of the dangerous situation by preventing that the success conditions  $\psi$  of the expected attack are true.<sup>7</sup>

Going back to the examples provided in the previous section, suppose that Bill expects that 4 steps from now he will be attacked by a thief's action of forcing the door of the house since 4 steps from now a thief will force the front door and, if 4 steps from now the thief's action of forcing the front door occurs and nobody is at home, then the thief's action will compromise the achievement of Bill's goal to keep his money safe 5 steps from now:  $\text{ExpAttack}(\text{Bill}, \text{thief}, \text{forceDoor}, \text{moneySafe}, \bigcirc^4 \text{nobodyAtHome}, 4)$ . Then, according to theorem 2, Bill comes to have the goal that 4 steps from now somebody will be at home:  $\text{Goal}_{\text{Bill}} \bigcirc^4 \neg \text{nobodyAtHome}$ . In this example Bill decides to defend himself by the thief's attack by preventing that the conditions of success of the thief's attack are true.

<sup>5</sup> Proving by induction that such an equivalence is a theorem of  $\mathcal{DL}$  is straightforward.

<sup>6</sup> By axiom 2 proving that such an equivalence is a theorem of  $\mathcal{DL}$  is again an easy task.

<sup>7</sup> With "success conditions" of a vehicle of attack we mean the conditions which ensure that the vehicle of attack will be efficacious and will succeed in compromising the goals of the defender.

The following theorem of  $\mathcal{DL}$  is complementary to the previous theorem 2 and shows which kind of defensive goal is activated in the mind of agent  $i$  when he expects to be in a dangerous situation.

**Theorem 4.**  $\vdash_{\mathcal{DL}} \text{ExpDangerous}(i, j, \alpha, \varphi, \psi, n) \rightarrow \text{Goal}_i \bigcirc^n [j : \alpha] \perp$

*Proof.*  $\text{ExpDangerous}(i, j, \alpha, \varphi, \psi, n)$  implies  $\text{Bel}_i(\psi \rightarrow \bigcirc^n [j : \alpha] \neg \varphi) \wedge \text{Bel}_i \text{Goal}_i \bigcirc^{n+1} \varphi \wedge \text{Bel}_i \psi$  (by definitions 1 and 3) which in turn implies  $\text{Bel}_i(\psi \rightarrow \bigcirc^n [j : \alpha] \neg \varphi) \wedge \text{Goal}_i \bigcirc^{n+1} \varphi \wedge \text{Bel}_i \psi$  (by axioms 6 and 14). Moreover  $\text{Bel}_i(\psi \rightarrow \bigcirc^n [j : \alpha] \neg \varphi) \wedge \text{Goal}_i \bigcirc^{n+1} \varphi \wedge \text{Bel}_i \psi$  implies  $\text{Goal}_i \bigcirc^n [j : \alpha] \neg \varphi \wedge \text{Goal}_i \bigcirc^n \bigcirc \varphi$  (by axiom 5 and axiom 15). Finally  $\text{Goal}_i \bigcirc^n [j : \alpha] \neg \varphi \wedge \text{Goal}_i \bigcirc^n \bigcirc \varphi$  implies  $\text{Goal}_i \bigcirc^n [j : \alpha] \perp$  (by axiom 4 and rules of inference R2 and R5).  $\square$

According to the previous theorem, when agent  $i$  thinks to be in a dangerous situation  $\psi$ , since he thinks that if  $j$  does action  $\alpha$  then he will compromise one of his goals,  $i$  comes to have the defensive goal of trying to block the occurrence of  $j$ 's action  $\alpha$  (viz. the expected vehicle of attack).

Going back to one of the examples provided in the previous section, suppose that Mary thinks to be in a dangerous situation since she believes that one step from now she will be in front of Fred and, if one step from now Fred shoots and she is in front of Fred, then Fred's action will compromise the achievement of her goal to be alive 2 steps from now:  $\text{ExpDanger}(Mary, Fred, shoot, MaryAlive, \bigcirc inFrontFred, 1)$ . Then, according to theorem 3, Mary comes to have the goal of trying to block Fred's action of shooting:  $\text{Goal}_{Mary} \bigcirc [Fred : shoot] \perp$ .

## 5 For a specification of defensive strategies

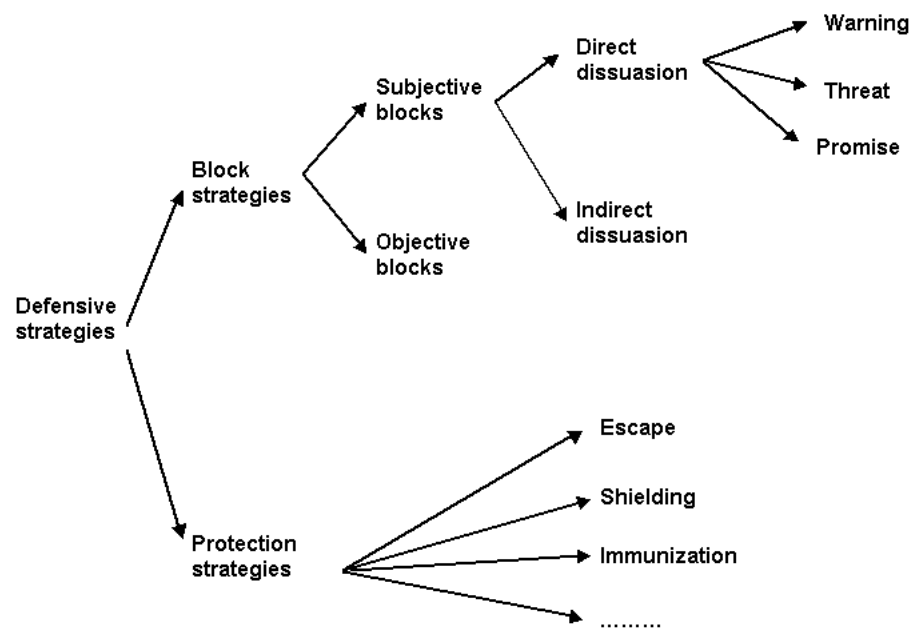
In the previous section 4.1 we have analyzed two specific kinds of defensive goals and defensive strategies. We have shown that when expecting a danger an agent can try either to block the expected vehicle of attack (*block strategy*) or to get out of the dangerous situation by preventing that the conditions of success of the expected attack are true (*protection strategy*). The previous two strategies are in our view the most general classes of defensive strategies that an intentional agent can adopt in order to prevent that his goals will be frustrated and compromised. But there are several specifications of these two general defensive strategies and defensive goals. The aim of this section is to provide a very brief overview of such specifications (see also figure 1).

There are two main types of block strategies. We call *objective block* a defensive strategy which consists in blocking the expected vehicle of attack by ensuring that its objective executability preconditions do not hold. For instance, if  $i$  is defending himself from  $j$ 's action of shooting  $j$ , he can try to disarm  $j$ . Indeed although  $j$  can intend to shoot  $i$ , if  $j$  is not armed then he will not be able to perform the action of shooting  $i$ .

On the contrary, we call *subjective block* a defensive strategy which consists in blocking the expected vehicle of attack by influencing the agent who is supposed to attack.<sup>8</sup> Subjective blocks are suitable defensive strategies only if the expected vehicle of attack

<sup>8</sup> See [17] for a theory of social influence.





**Fig. 1.** Typology of defensive strategies.

is an intentional action of a certain agent. Indeed, when having the goal of influencing agent  $j$ , agent  $i$  has the goal of inducing agent  $j$  to avoid doing such action that according to  $i$ 's expectations and beliefs is a potential danger for him.

In order to influence agent  $j$  and to induce him not to do action  $\alpha$ , agent  $i$  should try to modify those beliefs, assumptions, etc... of  $j$  that (according to  $i$ ) represent  $j$ 's reasons for doing action  $\alpha$ .

There are two general modes of changing the beliefs of another agent  $j$  in order to induce him not to do a certain action  $\alpha$ : we call *indirect dissuasion* one mode and *direct dissuasion* the other mode.

In indirect dissuasion, agent  $i$  tries to induce  $j$  not to do a certain action  $\alpha$  by changing  $j$ 's beliefs and undermining  $j$ 's reasons for doing  $\alpha$  without necessarily advancing arguments for not doing  $\alpha$  and without necessarily communicating something to  $j$  in an explicit way.

For example,  $i$  might try to indirectly dissuade a thief from stealing his car by installing a car alarm. Agent  $i$  does the action of installing a car alarm (defensive strategy) in order to make  $j$  believe that stealing the car will be very risky so that  $j$  will forbear from stealing the car. In this example  $i$  does not communicate anything to  $j$ .

On the contrary, a defensive strategy based on direct dissuasion consists in changing the beliefs of the other agent and undermining his reasons for doing action  $\alpha$  by communicating something explicitly to him.

There are several specific types of direct dissuasive strategy such as warnings, threats, promises, etc...<sup>9</sup> For instance, a *threat* of agent  $i$  to agent  $j$  should be conceived as a  $i$ 's act of explicit communication or speech act [20] aimed at informing  $j$  that: 1)  $i$  has a conditional intention to perform a certain action  $\beta_1$  in case  $j$  will do a certain action  $\beta_2$ ; 2) if  $i$  does  $\beta_1$  then  $j$ 's goals will be compromised.<sup>10</sup>

Finally, there are several types of *protection strategies*, viz. defensive strategies aimed at getting out of an expected dangerous situation by preventing that the conditions of success of the expected attack are true. In this work we are not going to deeply analyze them. Let us just note that a protective strategy of escape consists in changing location in order to dodge the expected vehicle of attack whilst a protective strategy of shielding consists in building infrastructures, using artifacts, being protected by law, etc... and generally in building barriers against an expected vehicle of attack.

Generally one would like to distinguish between *preventive defenses* versus *non-preventive defenses*. For the moment, we leave aside this important dimension of defense since in this paper we are mainly interested in the analysis of anticipatory and preventive defenses. Just note that preventive moves are strategic moves that the defender makes before the vehicle of attack starts whilst a non-preventive defense consists either in de-

---

<sup>9</sup> See also [18] for a theoretical approach to threats in argumentation. For a game theoretical approach to threats see [19].

<sup>10</sup> Note that with explicit communication we mean here the classical gricean conception of meta-level communication [21]. Therefore a threat of agent  $i$  to agent  $j$  necessarily implies  $i$ 's intention to perform a speech act  $A$  in order to inform  $j$  that: 1)  $i$  has a conditional intention to perform a certain action  $\beta_1$  in case  $j$  will do a certain action  $\beta_2$  and if  $i$  does  $\beta_1$  then  $j$ 's goals will be compromised; 2)  $i$  wants that  $j$  believes that 1).

fending ourselves during the attack (viz. facing the danger) or in defending ourselves after the attack by taking remedies.<sup>11</sup> Moreover, note that the available alternatives for a preventive defense are less than the available alternatives for a non-preventive defense. This statement is validated by observing that agents can block an attack only by preventing it. Therefore we must conclude that a precocious defense is in general more convenient since it offers a wider variety of alternative defensive strategies and moves.

## 6 Conclusion

We have provided in this paper a general formal analysis of the mental attitudes which are involved in a situation of defense. A preliminary ontology of defensive strategies has been designed. The issue of defense is not totally new in the MAS domain. For instance, the possibility of resolving conflicts through argumentation in negotiation contexts has received a lot of attention in the MAS community.<sup>12</sup> But we think that few efforts have been made in order to provide a general and systematic model of defense for intentional agents. As we have shown in this paper in fact, defense by argumentation should be conceived as a particular type of defensive strategy. But there are many other types of defensive moves and defensive strategies which are likewise important and which deserve to be investigated.

We think that with the current formal instruments for modeling intentional action, agency, and in particular with computational models like BDI agents it is possible to arrive to a principled and systematic model of defense. In this work we have tried to build the conceptual basis of such a model.

In our view a model of defense with cognitive and social foundations can be an important reference point not only for modeling security systems but also for modeling important aspects of social interaction such as coordination and negotiation.

---

<sup>11</sup> This distinction is crucial in medical domain where we can distinguish three macro-phases in the process of medical care: prevention, treatment and rehabilitation.

<sup>12</sup> See [22] for a review of the most important models of argumentation-based negotiation developed in the MAS framework.

## Bibliography

- [1] Castelfranchi, C.: Conflicts ontology. In Dieng, R., Meller, H., eds.: Conflicts in Artificial Intelligence. Kluwer, Dordrecht (2000) 21–40
- [2] Burrows, M., Abadi, M., Needham, R.: A logic for authentication. Proceedings of the Royal Society of London **426** (1989) 233–271
- [3] Dixon, C., C., F.G.M., Fisher, M., van der Hoek, W.: Using temporal logics of knowledge in the formal verification of security protocols. In: Proceedings eleventh International Symposium on temporal representation and reasoning, IEEE Computer Society Press (2004)
- [4] Syverson, P.: Adding time to a logic of authentication. In: Proceedings of the First ACM Conference on Computer and Communications Security, ACM Press (1993)
- [5] Glasgow, J., MacEwen, G., Panangaden, P.: A logic to reason about security. ACM Transactions on Computer Systems **10(3)** (1992) 226–264
- [6] Emerson, E.A.: Temporal and modal logic. In van Leeuwen, J., ed.: Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics. North-Holland Pub. Co./MIT Press (1990)
- [7] Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Cambridge (2000)
- [8] Cohen, P.R., Levesque, H.J.: Intention is choice with commitment. Artificial Intelligence **42** (1990) 213–261
- [9] Hintikka, J.: Knowledge and Belief. Cornell University Press, New York (1962)
- [10] Broersen, M.: Modal Action Logics for Reasoning about Reactive Systems. PhD-thesis Vrije Universiteit Amsterdam, Amsterdam (2003)
- [11] Castilho, M.A., Gasquet, O., Herzig, A.: Formalizing action and change in modal logic i: the frame problem. Journal of Logic and Computation **9(5)** (1999) 701–35
- [12] Herzig, A., Longin, D.: C&L intention revisited. In Dubois, D., Welty, C., Williams, M.A., eds.: Proceedings 9th Int. Conf. on Principles on Principles of Knowledge Representation and Reasoning(KR2004), AAAI Press (2004) 527–535
- [13] Miller, K., Sandu, G.: Weak commitments. In Holmstron-Hintikka, G., Tuomela, R., eds.: Contemporary Action Theory, vol.2: Social Action. Kluwer Academic Publishers, Dordrecht (1997)
- [14] Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press, Cambridge (2001)
- [15] Sahlqvist, H.: Completeness and correspondence in the first and second order semantics for modal logics. In: Proceedings 3rd Scandinavian Logic Symposium 1973. Volume number 82 in Studies in Logic. (1975)
- [16] Castelfranchi, C., Lorini, E.: Cognitive anatomy and functions of expectations. In Schmalhofer, F., Young, R.M., Katz, G., eds.: Proceedings European Cognitive Science Conference 2003 (EuroCogSci03), Lawrence Erlbaum (2003)
- [17] Conte, R., Castelfranchi, C.: Cognitive and social action. London University College of London Press, London (1995)

- [18] Walton, D.: Plausible Argument in Everyday Conversation. State University of New York Press, Albany (1992)
- [19] Schelling, T.C.: The strategy of conflict. Harvard University Press, Cambridge (1960)
- [20] Searle, J.: Speech acts: An essay in the philosophy of language. Cambridge University Press, Cambridge (1969)
- [21] Grice, H.P.: Study in the way of words. Harvard University Press, Cambridge (1989)
- [22] Rahwan, I., Ramchurn, S.D., Jennings, N.R., McBurney, P., Parsons, S., Sonenberg, L.: Argumentation-based negotiation. The Knowledge Engineering Review **18(4)** (2003) 343 – 375

# Building Coordinated Real-Time Control Plans

David J. Musliner, Michael J.S. Pelican, Kurt D. Krebsbach\*

Honeywell Laboratories  
3660 Technology Drive  
Minneapolis, MN 55418  
{david.musliner,mike.pelican}@honeywell.com

**Abstract.** We are interested in developing multi-agent systems that can provide real-time performance guarantees for critical missions that require cooperation. In particular, we are developing methods for teams of CIRCA agents to build coordinated plans that include explicit runtime communications to support distributed real-time reactivity to the environment. These teams can build plans in which different agents use their unique capabilities to guarantee that the team will respond in a coordinated fashion to mission-critical events. By reasoning explicitly about different agent roles, the agents can identify what communications must be exchanged in different situations. And, by reasoning explicitly about domain deadlines and communication time, the agents can build reactive plans that provide end-to-end performance guarantees spanning multi-agent teams.

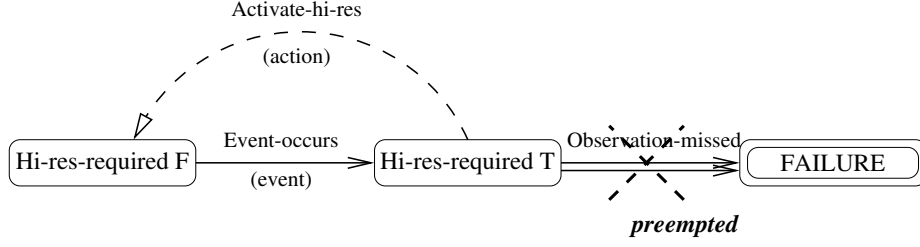
## 1 Introduction

We are extending the existing Cooperative Intelligent Real-Time Control Architecture (CIRCA) for real-time planning and control [8, 9] into distributed applications such as constellations of satellites, cooperative teams of space probes, and coordinated UAVs. In such coarse-grain distributed applications, multiple autonomous agents are each controlled by a CIRCA system, each of which builds a control plan incorporating run-time cooperation to achieve team goals in mission-critical domains. We are particularly interested in extending the real-time performance guarantees that CIRCA provides for single agents to small teams of coordinating CIRCA agents. In this paper, we use a simple example involving multiple spacecraft to describe CIRCA's new capabilities to negotiate coordinated roles, plan runtime communication to support coordination, and execute automatically-generated plans that ensure real-time coordination across a team of agents.

Individual CIRCA agents make guarantees of system safety by automatically building reactive control plans that guarantee to *preempt* all forms of system failure. By *preempt*, we mean that an action is planned to disable the preconditions of a potential failure, and that the action is time-constrained to *definitely* occur before the failure could *possibly* occur. For example, suppose a spacecraft's

---

\* Now at Lawrence University in Appleton, WI, kurt.krebsbach@lawrence.edu.



**Fig. 1.** A simple single-agent, single-action preemption example.

mission requires it to monitor for some events across a broad area and, when one of those events occurs, to focus a higher-resolution sensor on the area of interest within a short deadline (to ensure that the high-resolution sensor observes the phenomenon of interest). This situation might arise in a mission to observe geothermal activity or to identify strategic missile launches. Fig. 1 shows a simple example of a state space diagram for preemption in which the agent will activate the high-resolution sensor in time to avoid missing the observation. If the system has guaranteed to detect a state in which (**Hi-res-required T**) holds, and perform a responsive action before the window for observation closes, then this action preempts the temporal transition to the failure state. System safety is guaranteed by planning actions that preempt *all* failures [9].

Now suppose one satellite has a wide angle infra-red imaging sensor which identifies sites of interest that require further sensing using a high-resolution visual band sensor carried by a different satellite. The two spacecraft must coordinate their activities to preempt the missed observation failure. By *coordinated preemption*, we mean a set of complementary plans that can be executed by distributed agents to detect and react to situations before system failure occurs. How can two (or more) distributed agents build their plans to accomplish a coordinated preemption: “**You** sense the opportunity and **I’ll** act on it”?

A key observation is that this really devolves into two separate issues:

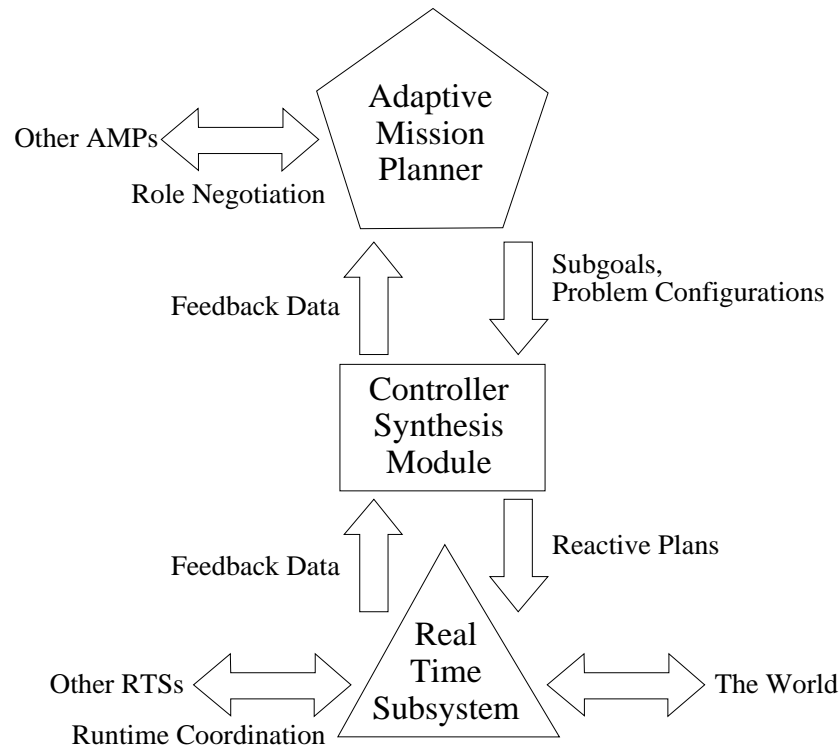
**Planned communication** — The agents must recognize the need to explicitly communicate (both sending and receiving) at a rate fast enough to satisfy the coordinated preemption timing constraint. In our example, the sensing agent must agree not only to detect the hot spot fast enough, but also to tell the other agent about the opportunity quickly enough. Likewise, the acting agent must focus sufficient attention on “listening” for a message from the sensing agent at a high enough frequency that it can guarantee to both receive the message and act on the opportunity, all before the deadline.

**Distributed causal links** — The distributed agents must be able to represent and reason about changes to their world that are not directly under their control, but which are predictable enough to be relied upon for a preemption guarantee. For example, in our scenario, the sensing agent must rely on the acting agent to take the appropriate action in time to guarantee that the data collection is performed in time. In complementary fashion, the acting

agent must construct a plan that honors its commitment to the acting agent. If one of the agents cannot construct a plan that satisfies its commitments, it must inform the others.

## 2 Brief Overview of CIRCA

CIRCA uses a suite of planning and scheduling components to reason about high-level problems that require powerful but potentially unbounded AI methods, while a separate real-time subsystem (RTS) reactively executes the automatically-generated plans and enforces guaranteed response times [8, 9, 11].



**Fig. 2.** The CIRCA architecture combines intelligent planning and adaptation with real-time performance guarantees.

All three of CIRCA's planning and execution subsystems operate in parallel. The Adaptive Mission Planner (AMP) reasons about collaborations at the level of mission roles and responsibilities using a Contract Net negotiation protocol [10, 13]. When an AMP has negotiated a responsibility for handling a particular threat or goal during a phase of a mission, it can configure a planning problem for the lower-level Controller Synthesis Module (CSM) to solve.



The CSM takes in problem configurations in the form of a set of transitions (see Fig. 3). The CSM reasons about an internal model of the world deduced from these transitions and dynamically programs the RTS with a planned set of reactions [9, 6]. While the RTS is executing those reactions, ensuring that the system avoids failure, the AMP and CSM continue planning to find the next appropriate set of reactions. The derivation of this new set of reactions does not need to meet a hard deadline, because the reactions concurrently executing on the RTS will continue handling all events, maintaining system safety. When the new reaction set has been developed, it can be downloaded to the RTS.

The CSM reasons about transitions of four types:

**Action transitions** represent actions performed by the RTS. These parallel the operators of a conventional planning system. Associated with each action is a worst case execution time (*wcet*): an *upper bound* on the delay ( $\Delta(a) \leq T$ ) before the action completes.

**Temporal transitions** represent uncontrollable processes, some of which may need to be preempted. Associated with each temporal transition is a *lower bound* on the delay before the temporal transition could possibly occur ( $\Delta(tt) \geq T$ ). Transition delays with a lower bound of zero are referred to as **events**, and are handled specially for efficiency reasons.

**Reliable temporal transitions** represent continuous processes that may need to be employed by the CIRCA agent. For example, a CIRCA agent might turn on a piece of equipment to initiate the process of warming up that equipment. The action itself will take a relatively short period of time to complete, however, the warming process might complete after a much longer delay. Reliable temporal transitions have both upper and lower bounds on their delays. As we will see, reliable temporals are especially important for modeling multi-agent interactions.

Fig. 3 contains three very simple transitions from a single-agent version of our example observation satellite domain, corresponding to the planned state space diagram in Fig. 1. The **event-occurs** event transition represents the occurrence of the phenomenon of interest, out of the agent’s control. That event sets the preconditions for the **observation-missed** temporal transition, which may occur as little as 500 milliseconds after the preconditions are established. Because the postconditions of **observation-missed** include (**failure** T), it is a **temporal transition to failure** (TTF). To preempt that failure, the CIRCA agent will plan to take the **activate-hi-res** action in the intervening state.

CIRCA builds its reactive plans in the form of Test-Action Pairs (TAPs) that test for a particular situation and invoke a planned response. Fig. 4 shows the TAP automatically generated and scheduled by CIRCA to implement the simple preemption in our running single-agent example. Each TAP has an associated worst-case execution time (*wcet*), which includes the worst-case time to complete the test plus the maximum amount of time to complete the action (if the condition is true). The CIRCA CSM uses its world model to derive the maximum allowable response time before a failure could possibly occur. Based

```

;;; The initial sensed event can occur at any time.
(def-event 'event-occurs
  :preconds '((hi-res-required F))
  :postconds '((hi-res-required T)))

;;; If you dont do hi-res sensing by 500 milliseconds, you fail.
(def-temporal 'observation-missed
  :preconds '((hi-res-required T))
  :postconds '((failure T))
  :min-delay 500)

;;; In the simple single-agent case, doing this handles the event.
(def-action activate-hi-res
  :preconds '()
  :postconds '((hi-res-required F))
  :wcet 300)

```

**Fig. 3.** CIRCA domains capture agent capabilities and world dynamics as transitions.

```

#<TAP 1>
Tests: (HI-RES-REQUIRED T)
Acts : ACTIVATE-HI-RES

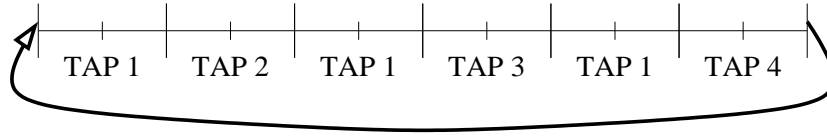
```

**Fig. 4.** Single agent Test-Action Pair to activate a high-resolution sensor when an observation is required.

on this and the *wcet*, it computes how often the given TAP must be executed to guarantee preempting transition to a failure state.

The CSM then attempts to build a cyclic schedule that runs each TAP at least as frequently as required. It is crucial to preemption that the maximum response time be strictly shorter than the minimum time for one of the undesirable transitions to occur. Fig. 5 provides an example cyclic schedule in which TAP 1 must be run more often than the other TAPs. If the scheduler cannot build a satisfactory polling loop, the problem is overconstrained, and the planner must backtrack in its search to compute a feasible plan.

In this paper, we are interested in extending CIRCA to handle preemptive plans that require at least two CIRCA agents to execute. But what does it mean to spread a preemption over two agents? Imagine our original example: “You sense, I’ll act”. Whereas in single agent CIRCA, both parts would be encapsulated within a single TAP, the test now belongs to one agent, and the action to the other, implying at least one TAP for each agent. But for the two agents to preserve the original semantics of the preemption, they will have to communicate, and that communication will also have occur in a predictable and timely manner.



**Fig. 5.** A TAP schedule loop in which TAP 1 must be run more often than the others.

### 3 Negotiating Coordinated Roles

How do the agents decide which role they are playing, and what to communicate about? In our current implementation, each CIRCA agent has a representation of its own capabilities, expressed as the classes of goals and threats (potential failures) it can handle. When a new mission is presented to a multi-CIRCA system, the agents each bid on the component goals and threats that make up the mission, based on their capabilities.

In our running example, the mission is characterized by two distinct threats, one representing the sensor agent’s need to send a message to the actor agent by a deadline, and one representing the actor agent’s need to respond to the message by a deadline. The respective agents bid to handle these threats, and win their appropriate roles on the team. It is worth noting that this decomposition is already present in the mission description entered by the system programmer or tasking agent; future versions may be able to decide themselves whether to tackle the response to a threat in a centralized or cooperative distributed fashion.

```

(def-event 'event-occurs
  :preconds '((saw-event F))
  :postconds '((saw-event T)))

;;; If you don't do hi-res sensing by 500 msec, fail.
(def-temporal 'hi-res-observation-missed
  :preconds '((saw-event T))
  :postconds '((failure T))
  :min-delay 500)

;;; All the sensor agent can do is notify the hi-res (actor) agent.
(def-action 'notify-hi-res
  :preconds '((notified-hi-res F))
  :postconds '((notified-hi-res T))
  :wcet 10)

;;; This reliable temporal represents sensor agent's model of actor agent's
;;; commitment to respond.
(def-reliable 'hi-res-observes
  :preconds '((saw-event T)
              (notified-hi-res T))
  :postconds '((saw-event F)
              (notified-hi-res F))
  :delay (make-range 250 400))

```

**Fig. 6.** The sensor agent can detect impending failure, but cannot directly prevent it.

```

;;; The uncontrollable event can occur at any time.
(def-event 'hear-about-event
  :preconds '((heard-about-event F))
  :postconds '((heard-about-event T)))

(def-action 'simple-observe-event
  :preconds '((heard-about-event T))
  :postconds '((heard-about-event F))
  :wcet 300)

;;; If you don't observe event in hi-res by 400 ms after notification,
;;; then you've failed.
(def-temporal 'observation-missed
  :preconds '((heard-about-event T))
  :postconds '((failure T))
  :min-delay 400)

```

**Fig. 7.** The simplest actor model gets a message and must respond.

To get the CSMs to plan the coordination communication explicitly, the Adaptive Mission Planner (AMP) must “trick” the individual agents into building collaborative plans by presenting them with controller synthesis problems that have been automatically crafted to represent their joint behavior commitments. The sensing agent’s AMP tells its CSM that it cannot autonomously defeat the threat, but that if it can communicate a warning quickly enough, this warning will lead to the defeat of the threat (see Fig. 6). The actor agent’s AMP tells its CSM that it cannot sense the threat directly, but that a warning may arrive at any time, after which it must take an action before an upper bound delay or face catastrophic consequences (see Fig. 7).

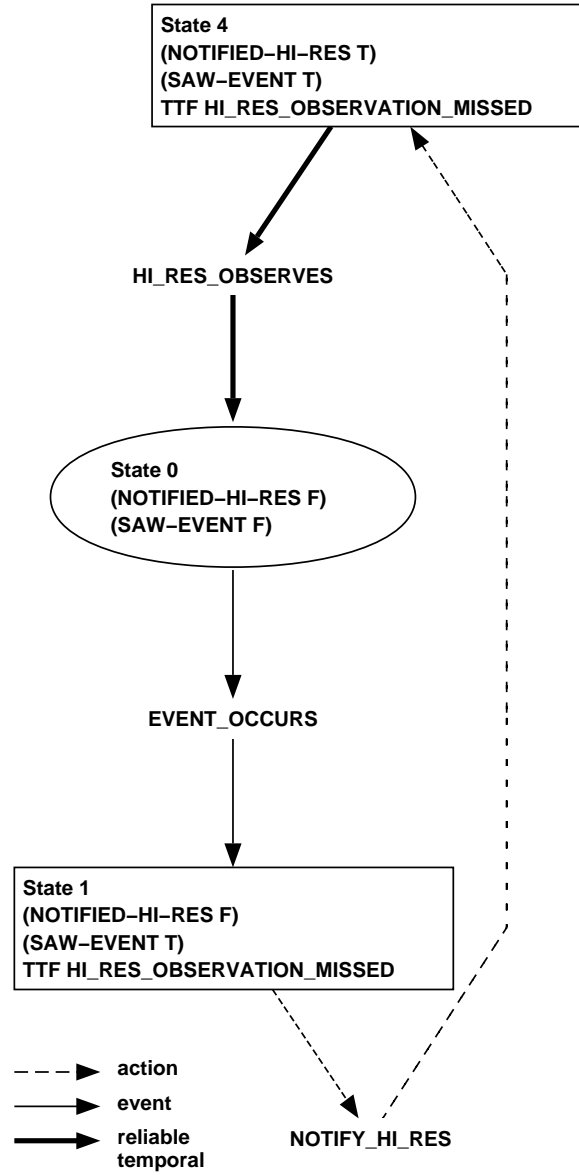
## 4 Building Coordinated Plans

For a coordinated preemption, we must decompose the timing constraint imposed by a temporal transition to failure into a set of tighter constraints corresponding to bounds on the sensing, communication, and action delays of the distributed agents responding to the threat.

For example, suppose our example threat (a critical high-resolution observation) has a minimum delay of 500 milliseconds (i.e., at least 500 milliseconds must elapse after the appropriate surface event has occurred, before the phenomenon disappears or expires, causing the team to have missed an observation). This would correspond to the minimum expected phenomenon duration, and hence the worst case that makes it hardest to observe.

In a single-agent preemption, the CIRCA agent would simply have to ensure that it would detect the event and respond with hi-res sensor activation in no more than the given 500 milliseconds. If the hi-res sensor takes no more than 300 milliseconds to capture its observation, then CIRCA would recognize that it could activate the hi-res sensor as much as 200 milliseconds after the event and still remain safe. So, CIRCA would build a TAP that must be polled no more than 200 milliseconds apart.

In the coordinated preemption case, we break up the overall response time constraint ( $\Delta T$ ) into two parts ( $\Delta A$  and  $\Delta B$ ) corresponding to the time that can be used by the two agents. The sensing agent (Agent A) will have to detect the threat and then send a message to the acting agent (Agent B), all within  $\Delta A$  time units. Note that the communication action will mimic a regular domain action, having some associated delay ( $\Delta A_c$ ) and being triggered by a polling TAP just as above. Fig. 8 and Fig. 9 illustrate this type of plan (and corresponding TAP) for Agent A. Note that Agent A’s model contains an explicit representation of Agent B’s commitment to act in response to the message. The bold **hi-res-observes** arrow represents a *reliable temporal transition*, indicating that Agent B’s action places both a lower and upper delay bound on the transition’s source state(s). When setting up CSM problem configurations for a coordinated preemption, the respective AMPs will include these types of “virtual transitions” to represent the commitments of other agents.



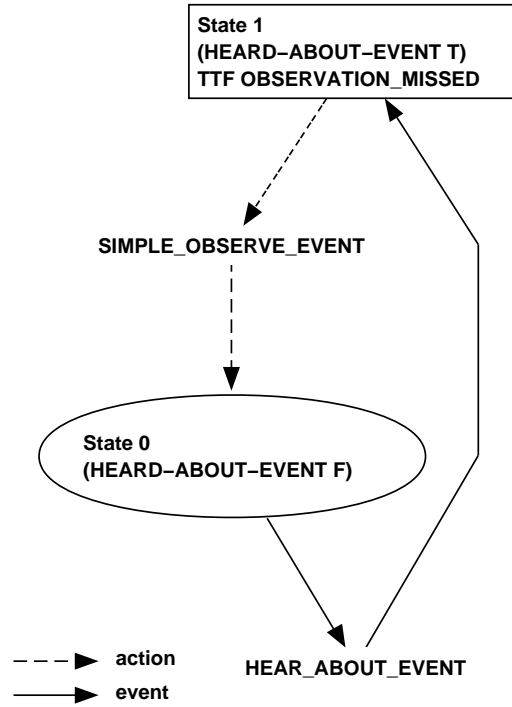
**Fig.8.** Agent A detects the threat and warns Agent B with a message guaranteed to be sent after no more than  $\Delta A$  seconds.

#<Agent A TAP>

Tests: (AND (SAW-EVENT T)  
(NOTIFIED-HI-RES F))

Acts : NOTIFY-HI-RES

**Fig.9.** Agent A's TAP for coordinating a preemption with Agent B. A's responsibility is to sense the condition and notify B.



**Fig. 10.** Agent B guarantees to detect the message from Agent A and activate its hi-res sensor within  $\Delta B$  seconds, thus ensuring that the “round-trip” delay from sensing to communication to action is bounded within the maximum available time constraint.

#<Agent B TAP>

Tests: (HEARD-ABOUT-EVENT T)

Acts : SIMPLE-OBSERVE-EVENT

**Fig. 11.** Agent B’s TAP for listening for A’s warning and taking the preemptive action in time to avoid mission failure.

Fig. 10 and Fig. 11 show that Agent B is given a representation of Agent A’s possible notification of the event, but no explicit representation of that event itself. This captures the notion that Agent B cannot actually sense the threat directly, and relies on other agents for information. As with the reliable temporal transition representing Agent B’s action to Agent A, here we have an *event* representing Agent A’s action (sending a message) to Agent B. The threat of the impending observation deadline is translated into a more abstract threat with a minimum delay of  $\Delta B$ . Agent B must detect the warning and activate its hi-res sensor to preempt the perceived deadline, and does so in the normal single-agent fashion.

Of course, this trivial example makes the problem look simple. The challenge arises when the sensing, communicating, and acting responsibilities are more complicated, and must be intertwined with other agent activities. To give an idea of the complexity that rapidly sets in, consider an example only slightly more complicated. Suppose that the agent with the hi-res sensor must actually activate a fine-grain alignment system to point the sensor, and only when that alignment system is maintaining tight alignment can the hi-res sensor be used. In that case, the domain description is not much larger (see Fig. 12), but the state space that results becomes considerably more complicated (see Fig. 13).

Handling this complexity growth is an ongoing research challenge; promising approaches include using automatic, dynamic abstraction to omit state information when feasible [6] and using decision-theoretic methods to trade off completeness for computability [5].

## 5 Related Work

Other work on multi-agent team coordination has focussed on “joint intentions” and using explicit models of team plans to define individual agent plans [14], often using constraint satisfaction and on-line repairs to adapt to evolving environments and conflicts between agent plans [7, 15]. While the higher levels of the CIRCA architecture do support elements of this approach, our current work is focused on building plans that accurately model and control the dynamics of coordination between agents at run-time. In particular we are interested in real-time, dependable interactions between teammates.

For the most part, other planning and execution systems for spacecraft handle multi-agent coordination in a more mission-specific fashion. For example, the automated mission planner for the Modified Antarctic Mapping Mission (MAMM) coordinates downlinks from RADARSAT to ground stations [12]. However, the mission planner does not plan ground station activities except as implied by the spacecraft plan. And although the MAMM’s domain is certainly dynamic, the mission planner treats it as predictable. The mission plan is fixed. If an observation is missed, the remaining mission must be replanned.

CASPER, like CIRCA, provides a soft real-time on-board planning facility [2]. Unlike CIRCA, CASPER builds plans for a sequential task executive, and repairs them when necessary to react to environment dynamics. CIRCA’s



```

(def-action 'align-hi-res-sensor
  :preconds '()
  :postconds '((hi-res-sensor-aligned T)
               (hi-res-sensor-effective T))
  :wcet 10)

;; Once you align sensor, can trigger it and take detailed reading.
(def-action 'begin-hi-res-sensing
  :preconds '((sensing normal)(hi-res-sensor-aligned T)
              (hi-res-sensor-effective T))
  :postconds '((sensing hi-res))
  :wcet 10)

;; After a while, the sensor alignment expires...
(def-temporal 'sensor-alignment-expires
  :preconds '((hi-res-sensor-aligned T))
  :postconds '((hi-res-sensor-aligned F))
  :min-delay 100)

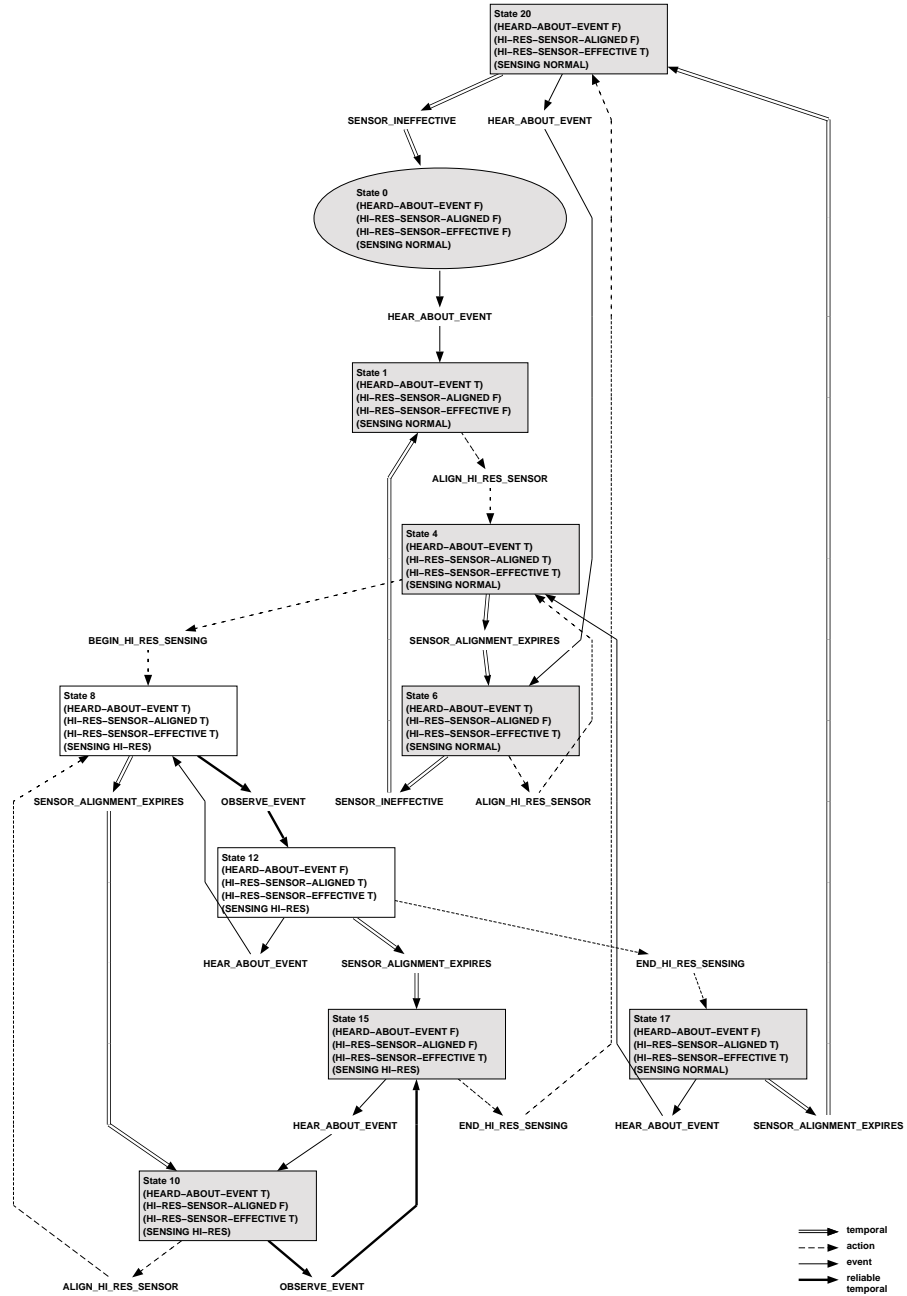
;; After sensor alignment expires, a while later the effectiveness is gone;
;; we should preempt this transition to keep the sensing reliable temporal working.
(def-temporal 'sensor-ineffective
  :preconds '((hi-res-sensor-aligned F)(hi-res-sensor-effective T))
  :postconds '((hi-res-sensor-effective F))
  :min-delay 100)

;; need hi-res-sensor-effective, or the sensing doesn't work...
(def-reliable 'observe-event
  :preconds '((heard-about-event T) (sensing hi-res)
              (hi-res-sensor-effective T))
  :postconds '((heard-about-event F))
  :delay (make-range 250 300))

(def-action 'end-hi-res-sensing
  :preconds '((sensing hi-res))
  :postconds '((sensing normal))
  :wcet 10)

```

**Fig. 12.** A few additional transitions can define a much more complex plan space.



**Fig. 13.** When Agent B must maintain sensor alignment, the reachable state space grows quickly.

TAP schedules incorporate real-time reactions to an unpredictable environment; environmental dynamics can be handled in the executive directly. Dynamics that exceed the scope of the pre-planned reactions are handled by on-the-fly replanning at the CSM and AMP levels.

All of the ASPEN family of planners, including CASPER and the MAMM mission planner, include specific resource models and constraint checking [3]. CIRCA represents resources implicitly, as postcondition or precondition features in the CSM and as roles or capabilities in the AMP. Plans for future work include explicit resource representation and checking.

Both the Three Corner Sat Mission and the Autonomous Sciencecraft Constellation projects are extending ASPEN-based planners to handle distributed teams of spacecraft like those described in our examples [1, 4].

## 6 Conclusion

In this paper, we have discussed the notion of *coordinated preemption*, a multi-agent extension of guaranteed failure preemption in CIRCA. Coordinated preemption allows a team of distributed CIRCA agents to build and execute synchronized plans that include joint actions such as “You sense, I’ll act”. This new capability furthers our goal of extending CIRCA to multi-agent, real-time, mission-critical domains. We have implemented coordinated preemptions in CIRCA, using inter-agent communication for both plan-time negotiation (between different agents’ AMPs), and for run-time coordination (between agents’ RTSs).

Several research questions also remain. For example, how should the available response delay  $\Delta T$ , originally for one agent, be split into two or more components? How much of the delay should each agent receive, considering that these load levels influence the plan’s schedulability for each agent? Because the knowledge needed to determine a feasible distribution of the available response time (if it exists) is itself distributed across agents, we will consider iterative negotiation between the coordinating agents as a first approach.

## Acknowledgments

This material is based upon work supported by DARPA/ITO and the Air Force Research Laboratory under Contract No. F30602-00-C-0017.

## References

- [1] S. Chien, B. Engelhardt, R. Knight, G. Rabideau, R. Sherwood, E. Hansen, A. Ortiviz, C. Wilklow, and S. Wichman, “Onboard Autonomy on the Three Corner Sat Mission,” in *Proceedings of ISAIRAS 2001*, June 2001.
- [2] S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, “Using Iterative Repair to Increase the Responsiveness of Planning and Scheduling for Autonomous Spacecraft,” in *IJCAI99 Workshop on Scheduling and Planning meet Real-time Monitoring in a Dynamic and Uncertain World*, August 1999.

- [3] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, and D. Tran, "ASPEN – Automating Space Mission Operations using Automated Planning and Scheduling," in *SpaceOps 2000*, 2000.
- [4] S. Chien, R. Sherwood, M. Burl, R. Knight, G. Rabideau, B. Engelhardt, A. Davies, P. Zetocha, R. Wainright, P. Klupar, P. Cappelaere, D. Surka, B. Williams, R. Greeley, V. Baker, and J. Doan, "The Techsat-21 Autonomous Sciencecraft Constellation Demonstration," in *Proceedings of ISAIRAS 2001*, June 2001.
- [5] R. P. Goldman, D. J. Musliner, and K. D. Krebsbach, "Managing Online Self-Adaptation in Real-Time Environments," in *Proc. Second Int'l Workshop on Self Adaptive Software*, 2001.
- [6] R. P. Goldman, D. J. Musliner, K. D. Krebsbach, and M. S. Boddy, "Dynamic Abstraction Planning," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pp. 680–686, Menlo Park, CA, July 1997, American Association for Artificial Intelligence, AAAI Press/MIT Press.
- [7] H. Jung, M. Tambe, and S. Kulkarni, "Argumentation as distributed constraint satisfaction: applications and results," in *Proc. of the Fifth Int'l Conf. on Autonomous Agents*, pp. 324–331, 2001.
- [8] D. J. Musliner, E. H. Durfee, and K. G. Shin, "CIRCA: A Cooperative Intelligent Real-Time Control Architecture," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 6, pp. 1561–1574, 1993.
- [9] D. J. Musliner, E. H. Durfee, and K. G. Shin, "World Modeling for the Dynamic Construction of Real-Time Control Plans," *Artificial Intelligence*, vol. 74, no. 1, pp. 83–127, March 1995.
- [10] D. J. Musliner and K. D. Krebsbach, "Multi-Agent Mission Coordination via Negotiation," in *Working Notes of the AAAI Fall Symposium on Negotiation Methods for Autonomous Cooperative Systems*, 2001.
- [11] D. J. Musliner, K. D. Krebsbach, M. Pelican, R. P. Goldman, and M. S. Boddy, "Issues in Distributed Planning for Real-Time Control (Extended Abstract)," in *Working Notes of the AAAI Fall Symposium on Distributed Continual Planning*, October 1998.
- [12] B. D. Smith, B. E. Englehardt, and D. H. Mutz, "The RADARSAT-MAMM Automated Mission Planner," *AI Magazine*, Summer 2002.
- [13] R. Smith, "The Contract Net: A Formalism For the Control of Distributed Problem Solving," in *Proc. Int'l Joint Conf. on Artificial Intelligence*, volume 1, p. 472, August 1977.
- [14] M. Tambe, "Implementing Agent Teams in Dynamic Multi-agent Environments," *Applied Artificial Intelligence*, 1997.
- [15] A. Tate, J. Levine, and J. Dalton, "Using AI Planning Techniques for Army Small Unit Operations," in *Proc. of the Fifth Int'l Conf. on Artificial Intelligence Planning and Scheduling Systems (AIPS 2000)*, 1999.

# Fluid-Like Swarms with Predictable Macroscopic Behavior

Diana Spears, Wesley Kerr, and William Spears

Department of Computer Science  
University of Wyoming, Laramie, WY 82071  
dspears@cs.uwyo.edu

**Abstract.** This paper is concerned with assuring the safety of a swarm of agents (simulated robots). Such behavioral assurance is provided with the physics method called *kinetic theory*. Kinetic theory formulas are used to predict the macroscopic behavior of a simulated swarm of individually controlled agents. Kinetic theory is also the method for controlling the agents. In particular, the agents behave like particles in a moving gas.

The coverage task addressed here involves a dynamic search through a bounded region, while avoiding multiple large obstacles, such as buildings. In the case of limited sensors and communication, maintaining spatial coverage – especially after passing the obstacles – is a challenging problem. Our kinetic theory solution simulates a gas-like swarm motion, which provides excellent coverage. Finally, experimental results are presented that determine how well the macroscopic-level theory, mentioned above, predicts simulated swarm behavior on this task.

## 1 Safe Swarms

The research in this paper is designed with two objectives in mind: to effectively accomplish a difficult surveillance task, and to accomplish it in a manner that is “safe.” By “safe” we mean that the multi-agent collective that accomplishes the task is, in the aggregate, both *predictable* (behaviorally assured) and *controllable*.

The traditional approach to achieving safe agents is to engineer safety into the individual agents (e.g., [1] [2] [3]) and, sometimes, also into the particular interactions between these individual agents. This is typically accomplished with formal methods, such as model checking [4] or theorem proving [5], control theory [1], or other formalisms [3].

This paper explores an alternative view of safety. Our alternative view is motivated by a desire to model swarms (i.e., very large numbers) of agents cooperatively performing a task. The modern swarm philosophy is one of *emergent behavior*, which is defined as producing intelligent macroscopic behavior in the aggregate from lots of simple, unintelligent agents. The key to swarm agent/robotics emergent behavior is that even though the individual agent behaviors are easy to understand and may be expressed as simple rules, describing the behavior of the swarm as a whole requires a paradigm shift. In other words,

the description of the macroscopic behavior is not a straightforward function of descriptions of the microscopic behaviors – because safety of a swarm is often too computationally difficult to achieve by engineering safety into all the individual agents, which may have complex interactions.

The alternative view of safety that we propose for swarms is founded upon physics. Our choice is motivated by the fact that physics is the most accurate of all the scientific disciplines at predicting the macroscopic behavior of huge numbers of interacting particles using very simple mathematical formulas. Furthermore, physics disciplines, such as fluid dynamics, utilize these formulas for the design of systems that have desirable properties. Rather than engineer the properties into the individual particles (which of course cannot be done in most real-world situations), these disciplines promote engineering using principles at the macroscopic level. For example, in fluid dynamics, there is a field called the “control and management of turbulence dynamics,” in which questions are addressed such as how to ensure that desirable macroscopic fluid properties are preserved when the flow is controlled in a specified manner. Solutions to these questions are stated as control theoretic equations, which are expressed in terms of macroscopic properties of the fluid, such as velocity and pressure [6].

In summary, physicists have developed succinct formulas that are highly predictive of complex, multi-particle behavior. Such formulas can be used at an abstract level to design multi-agent systems with desirable properties, where agents are modeled as particles. The end result is behavioral assurance by engineering safety into the collective, rather than into the individual agents.

## 2 Physics of Large, Multi-Particle Systems

The study of many-particle systems is one of the most active research areas in modern physics [7]. Today it is well known that although one could write down the equations of motion of individual atoms and their interactions, the complexity of doing this for a large number of particles is too daunting. The problem is not just quantitative (which would lead one to suspect that it could be solved with improved computational power), but it is also *qualitative* [7]. For example, how could one hope to understand the human abilities of natural language and planning by studying the properties of individual neurons? Likewise, physicists often predict macroscopic properties of matter, such as volume or pressure, from microscopic atomic particles by using statistical arguments, such as expectations, rather than resort to predictions of movement of the individual particles.

The physics of many-particle systems can be subdivided into three main disciplines [7]:

1. **Thermodynamics.** In the discipline of thermodynamics, descriptions are strictly at the macroscopic level. Valid formulas are derived based on a minimum number of postulates, without *any* detailed assumptions about the microscopic properties of the system.

2. **Statistical Mechanics.** Here, the macroscopic behavior of a multi-particle system is described based on the statistical properties of the microscopic behavior. The assumption is that the system is “in equilibrium.”
3. **Kinetic Theory.** The most popular physics approach to describing systems that are not necessarily in equilibrium is kinetic theory. Similarly to statistical mechanics, kinetic theory describes macroscopic behavior in stochastic terms. Note that kinetic theory subsumes statistical mechanics, i.e., in equilibrium, the former is the same as the latter.

The approach adopted in this paper is kinetic theory. Kinetic theory is the discipline of choice because it has a much richer and more extensive theory than thermodynamics (i.e., “relatively few statements can be made” in thermodynamics [7]), and because many multi-agent applications do not assume equilibrium.

Inspired by the huge success in applying physics to many-particle systems, we have decided to apply physics, in particular kinetic theory, principles to multi-agent swarms. Our agents are assumed to be simulated robots. Kinetic theory is used to control the agents, because fluid-like movement appears to be the most appropriate approach for a swarm of robots to achieve our task, and kinetic theory simulations are frequently used to model and study the movement of actual fluids.<sup>1</sup> In order to achieve a high level of predictive accuracy using kinetic theory, we use a multi-agent system that behaves like a many-particle physical system. Here, we show that using kinetic theory for both theory and simulation produces an excellent match between the two, thus providing a high degree of system behavioral assurance and safety. It is important to note that the match between theory and simulation is not achieved with a theory that predicts individual particle movements, which are generated within the simulation. Rather, the theory uses stochastic properties of the swarm as a whole to make its predictions. In other words, the theory is macroscopic and is predictive of the simulation, which is microscopic in its design and implementation.

### 3 The Sweeping and Obstacle Avoidance Task

The task being addressed here, which is also described in [9], consists of sweeping a large group of mobile robots through a long bounded region (a swath of land, a corridor in a building, a city sector, or an underground passageway/tunnel), to perform a search, i.e., surveillance. This requires maximum coverage. The robots have a limited sensing range for detecting other agents/objects. It is assumed that robots near the corridor boundaries can detect these boundaries, and that all robots can sense the global direction that they are to move, e.g., by using a compass. There is no other global information, and the agents behave in a distributed, non-centralized manner. As they move, the robots need to avoid large obstacles, such as large buildings. This poses a challenge because with a limited sensing range, robots on one side of a building cannot necessarily

---

<sup>1</sup> An alternative is a molecular dynamics (MD) simulation that is deterministic. For results of a physics-based approach using MD simulations, see [8].

communicate with robots on the other side. The search might be for enemy mines, survivors of a collapsed building or, alternatively, the robots might be patrolling the area. It is assumed that the robots need to keep moving, because there are not enough of them to view the entire length of the region at once. In other words, the robots begin scattered randomly at one end of the corridor and move to the opposite end (considered the “goal direction”). This is a *sweep*. A sweep terminates when all robots have reached the goal end of the corridor, or a time limit is reached. Once the robots complete one sweep, they reverse their goal direction and sweep back again. Finally, if stealth is an issue then we would like the individual robot movements to be unpredictable to adversaries. It is conjectured that the behavior of a gas is most appropriate for solving this task, i.e., each robot is modeled as a gas particle.

## 4 Motivation for Using a Fluid-Like Swarm

The term “fluid” refers to both liquids and gases; this paper focuses on gases in motion. Although individual atoms or molecules in a gas have unpredictable locations at any instant in time, the gas is predictable at a macroscopic level. Furthermore, when gases are placed in a container, they expand to fill the container, thereby providing outstanding spatial coverage. When not in an equilibrium state, gases can also move in bulk. The gas can be transported either by advection (due to the velocity of the ambient air in which it has been dispersed) or due to molecular diffusion, e.g., if the gas is heavier than the ambient air then it will fall slowly to the ground. Gases will also flow around obstacles, and then expand after passing the obstacles, thereby filling the space again.

These forms of coverage are precisely the ones required for excellent performance on the sweeping and obstacle avoidance task. Therefore, we use a kinetic theory particle simulation to model our agent swarm performing the task.

## 5 Kinetic Theory for Simulating Fluids

Our *kinetic theory (KT) simulation* is a microscopic model of individual particles, which are considered to be agents, or simulated robots. Our simulation, as well as this overview of it, borrows heavily from Garcia [10].

When modeling a gas, the number of particles is problematic, i.e., in a gas at standard temperature and pressure there are  $2.687 \times 10^{19}$  particles in a cubic centimeter. A typical solution is to employ a stochastic model that calculates and updates the probabilities of where the particles are and what their velocities are. This is the basis of KT. One advantage of this model is that it enables us to make stochastic predictions, such as the average behavior of the ensemble. The second advantage is that with real robots, we can implement this with probabilistic robot actions, thereby avoiding predictability of the individual agents, e.g., for stealth.

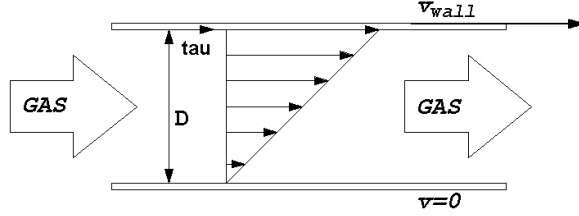
In KT, particles are treated as possessing kinetic energy but no potential energy (i.e., an ideal gas), and collisions with other particles are modeled as



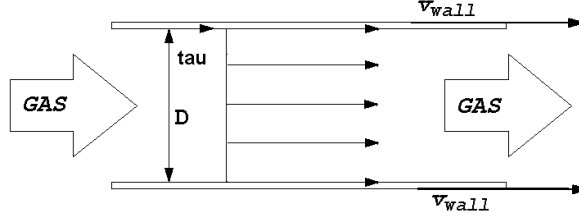
purely elastic collisions that maintain conservation of momentum. Using kinetic theory formulas, we can predict useful macroscopic properties of the system, such as the average speed or kinetic energy of the particles in the system. For example, assuming  $k$  is Boltzmann's constant, where  $k = 1.38 \times 10^{-23}$  J/K,  $m$  is the mass of any particle,  $f(v)$  is the probability density function for speed, and  $T$  is the temperature of the system, then the average speed of any particle (in 3D) is:

$$\langle v \rangle = \int_0^\infty v f(v) dv = \frac{2\sqrt{2}}{\sqrt{\pi}} \sqrt{\frac{kT}{m}}$$

From this formula, one can see that the temperature  $T$  plays an important role in KT. In our KT simulation,  $T$  is a user-defined system parameter analogous to real temperature. In other words, increasing  $T$  in our system raises the *virtual* system heat (analogous to actual heat), for the purpose of increasing the system kinetic energy and thereby increase the particle motion, i.e., the speed of the agents in the simulation.



**Fig. 1.** Schematic for a one-sided Couette flow.



**Fig. 2.** Schematic for a two-sided Couette flow.

Our KT simulation algorithm is a variant of the particle simulations described in [10]. We substantially modified the algorithms in [10] to tailor them to simulated robots with local views. Robots, modeled as particles, behave in the aggregate like “Couette flow.” Figure 1, from [10], depicts one-sided Couette flow, where a fluid moves through some environment between two walls – one

wall moving with velocity  $v_{wall}$ , and the other stationary (the environment is the frame of reference). In this Couette, fluid is assumed to move in the positive  $y$ -direction (i.e., longitudinally toward the goal end of the Couette corridor), and the positive  $x$ -direction goes from the stationary wall to the moving wall (i.e., laterally across the Couette corridor). Note that the direction of virtual motion of Couette walls is determined by using a compass to sense the goal direction. In general, we have found that a Couette is useful because it introduces an external source of kinetic energy into the system and gives the agents a direction to move.

Because the fluid is Newtonian and has viscosity, there is a linear velocity distribution across the system. Fluid deformation occurs because of the shear stress,  $\tau$ , and the wall velocity is transferred (via kinetic energy) because of molecular friction on the particles that strike the wall. On the other hand, the particles that strike either wall will transfer kinetic energy to that wall. This does not cause the wall to change velocity, since in a Couette flow the walls are assumed to have infinite length and depth and therefore infinite mass. We chose a Couette flow in order to introduce kinetic energy into the system and to give the particles a direction to move.

Our 2D simulated world models a modified (two-sided) Couette flow in which both Couette walls are moving in the same direction with the same speed (see Figure 2). We invented this variant as a means of propelling all robots in a desired general direction, i.e., the large-scale fluid motion is approximately that of the walls.

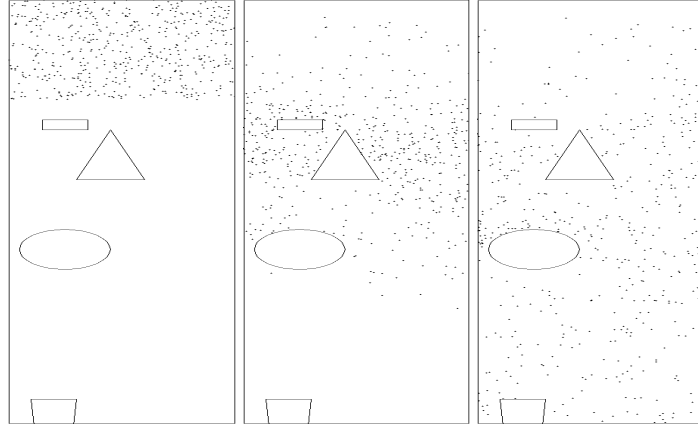
In our simulation, each agent, which is modeled abstractly as a holonomic particle, can be described by a position vector  $\mathbf{p}$  and a velocity vector  $\mathbf{v}$ . At each time step, every agent resets its position based on how far it could move in the given time step utilizing its current velocity. Particle velocities are initialized to be a random function of the (virtual) system temperature  $T$ , and these velocities remain constant unless collisions occur. Collisions are the primary mechanism for driving particle movement/acceleration in a KT simulation. (Note that with actual robots, collisions and wall motion would be virtual.) The system updates the world in discrete time steps,  $\Delta t$ , which occur on the order of the mean collision time for an agent.

At each time step, every agent in the system updates its position. When updating its position, a check is performed first to see if the movement would cause a (virtual) collision between the agent and a wall. If a collision would occur, then the agent selects a new velocity from a biased Maxwellian distribution, which is a function of the system temperature. If the agent is about to strike a moving wall, then some of the energy from the wall is transferred to the agent. Inter-agent (virtual) collisions are then processed. The number of collisions in any given region is a stochastic function of the number of agents in that region. In particular, the probability of a virtual collision between two agents is based on their proximity, but is independent of the angle between their velocity vectors. The new post-collision velocity vectors are based on the center of mass vector, coupled with a random component. See [10] and [11] for details. This process continues indefinitely or until a desired state is achieved.

## 6 The Surveillance Task Simulation

For a model of the surveillance task scenario, we have developed a 2D simulation of the task scenario, i.e., an obstacle-laden corridor with KT-driven robots flowing through it. The “two-sided” variant of the traditional Couette model is used (recall this variant from Figure 2), in which both Couette walls move in the same direction with the same speed. The two-sided Couette is highly effective at driving bulk swarm movement in the goal direction. The agents begin in random locations at the top of the corridor, and sweep down the corridor in the goal direction. Typical results for a sweep are shown in Fig. 3. Because robots are not capable of distinguishing corridor walls from obstacle walls, any obstacle wall parallel to the Couette walls is considered to be “Couette” (in motion), and the (virtual) wall velocity is added to the  $y$ -component of the velocity for any agent that collides with a Couette wall. Therefore, obstacle walls, in addition to actual Couette walls, can increase the velocity of robots toward the goal.

Inter-robot collisions are processed in localized regions. Likewise, robots only (virtually) collide with walls that are in close proximity to them.



**Fig. 3.** KT controllers perform a sweep. The snapshots progress in time from left to right.

With this simulation of a swarm of agents performing the task, the question arises of how to use kinetic theory to predict swarm behavior? One option is to develop a theory that models the physics of the entire task, including obstacles. This would require considerable work, and would be especially difficult if we do not know beforehand the number, sizes, and shapes of the obstacles. An alternative option is to develop a simpler theory that makes assumptions that do not hold in the full task, and then scale up the task in simulation to see how

well the theory holds when its assumptions are violated. The latter is the option adopted here, and is the topic of the following section.

## 7 Theoretical Predictions of Macroscopic Behavior

One of the primary advantages of physics-based swarms is the large existing body of physics-based theory for predicting system behavior and ensuring that the swarm will behave “safely.” A secondary advantage is that the theory can be used for optimal selection of system parameter values [12]. In this section, we provide evidence that kinetic theory is predictive of the behavior of kinetic-theory-based simulations. Furthermore, we demonstrate the practical use of the theory for parameter value selection.

We focus on subtasks of the sweeping and obstacle avoidance task, described above. Recall that our objectives for the agents in this task are to sweep a corridor and avoid obstacles. The ultimate objective is to maximize coverage. Consider two types of spatial coverage: *longitudinal* (in the goal direction) and *lateral* (orthogonal to the goal direction). Longitudinal coverage can be achieved by movement of the swarm in the goal direction; lateral coverage can be achieved by a uniform spatial distribution of the robots between the side walls of the corridor. The objective of the coverage task is to maximize both longitudinal and lateral coverage in the minimum possible time (i.e., to also maximize temporal coverage).

To measure how well the robots achieve the task objective, we examine the following three metrics:

1. The degree to which the **spatial distribution** of the robots matches a uniform distribution. This is a measure of lateral coverage of the corridor and provides confirmation of equilibrium behavior.
2. The **average speed** of the robots (averaged over all robots in the corridor). This is a measure of all three types of coverage: lateral, longitudinal, and temporal. Velocity is a vector consisting of speed and direction. The type of coverage depends on the direction. To control the average swarm speed, one can directly vary the value of the system temperature,  $T$ . Therefore, our experiment explores the relationship between  $T$  and average speed.
3. The **velocity distribution** of all robots in the corridor. This is a measure of longitudinal spatial coverage, as well as temporal coverage. For example, consider the one-sided Couette in Figure 1 again, and in particular focus on the line representing the velocity distribution. The slope of that line is inversely proportional to the longitudinal spatial coverage (and the temporal coverage). In other words, for a given Couette diameter,  $D$ , if you increase the wall speed,  $v_{wall}$ , then the slope will be reduced and the longitudinal and temporal coverage will increase. Below, we run an experiment to examine the relationship between the wall speed,  $v_{wall}$ , and the velocity distribution in one-sided and two-sided Couettes. The intention is to enable

the system designer to select a wall speed for optimizing the swarm velocity distribution.

The theory presented in this paper assumes simplified 2D environments with no obstacles. To develop theory for the full task simulation would require extensive theoretical physics analyses, which is beyond the scope of this paper. This will be tackled as future work.

We ran three sets of experiments, in accordance with the metrics defined above. For each experiment, one parameter was perturbed and eight different values of the affected parameter were chosen. For each parameter value, 20 different runs through the simulator were executed, each with different random initial robot positions and velocities. The average simulation results and relative error (over the 20 runs) were computed and graphed.

For these experiments, we defined the error between the theoretical predictions and the simulation results, denoted *relative error*, to be:

$$\frac{|theoretical - simulation|}{theoretical} \times 100$$

Although the theory assumes no obstacles, in the simulation we ran with six obstacle densities, ranging from 0% to 50%. Surprisingly, some of the theory holds well, despite this.

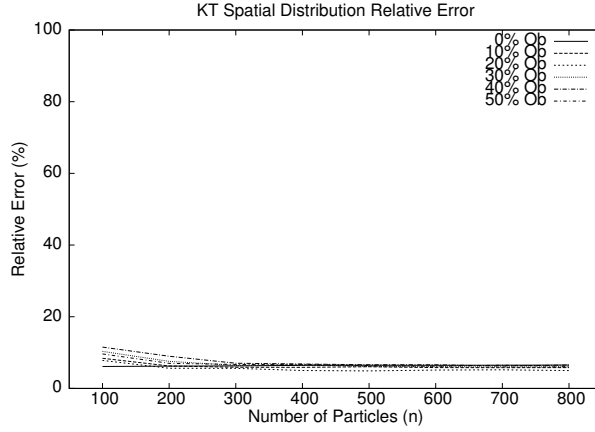
There are two complementary goals for running these experiments. The first goal is to determine how predictive the theory is. Derivations of all laws (predictive theoretical formulas) are in [9]. The second goal is to determine the relationship between parameter settings and system behavior. If a system designer understands this relationship, he/she can more easily set parameters to achieve optimal performance. Finally, and most importantly, the reason why these two goals are complementary is that if the theory is predictive of the system simulation behavior, then future system designers no longer need to run the simulation to determine the optimal parameter settings – graphs generated from theory alone will suffice. This can reduce the system design time.

## 8 Experiment 1: Spatial Distribution

The first experiment examines the equilibrium spatial distribution of agents within an enclosed region, i.e., a square “container. The agents begin in a tight Gaussian distribution, which then diffuses until equilibrium has been reached. During this experiment, there is no goal force or wall movement, and therefore no *externally-directed* bulk transport of the swarm.

The purpose of this experiment is to confirm the theoretically expected behavior of a KT system in equilibrium, which will thereby verify the correctness of our implementation – a big advantage of our approach. The KT gas model predicts that, upon reaching equilibrium, the particles will be spatially uniformly distributed. To confirm this prediction, we divided the square container in our KT simulator into a 2D grid of cells. Theory predicts that there should be (on

average)  $n/c$  robots per cell, where  $n$  is the total number of robots and  $c$  is the total number of grid cells. We ran with six obstacle densities, ranging from 0% to 50%, to determine the sensitivity of the spatial distribution to obstacle density.



**Fig. 4.** Relative error for the KT spatial distribution.

Fig. 4 shows the experimental results. Note that despite the introduction of as much as a 50% obstacle coverage, we can still predict the spatial distribution with a relative error of less than 10%, which is surprisingly low. The error is very low once the number of robots is about 300, which is surprising considering that 300 robots is far less than the  $10^{19}$  particles typically assumed by traditional kinetic theory.

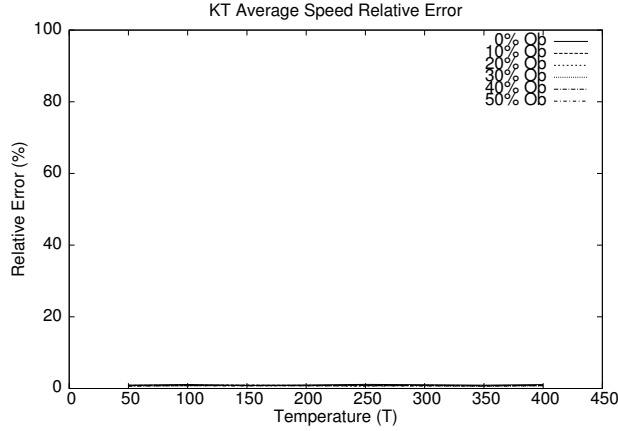
## 9 Experiment 2: Average Speed

For the second experiment, we examine the average speed of the robots in the system. Once again, there is no external force or externally-directed bulk transport of the swarm. However, recall that the agents will increase their speed if there is an increase in the system temperature, which causes an increase in kinetic energy. The objective of this experiment is to examine the relationship between the temperature,  $T$ , and the average speed of the robots. The average robot speed serves as a measure of how well the system will be able to achieve complete coverage – because higher speed translates to greater lateral and/or longitudinal coverage, depending on the velocity direction. This experiment also serves to verify that our simulation code has been implemented correctly. Note that not all applications will require maximum coverage; therefore, we want to study the general question of precisely how specific choices of speed affect coverage.

Our predictive formula for 2D is (see [9] for the mathematical derivation):

$$\langle v \rangle = \frac{1}{4} \sqrt{\frac{8\pi kT}{m}}$$

where  $k$  is Boltzmann’s constant ( $1.38 \times 10^{23} J/K$ ),  $m$  is the robot mass (assumed to be one), and  $T$  is the system temperature.



**Fig. 5.** Relative error for the KT average speed.

This theoretical formula is compared with the actual average speed,  $\langle v \rangle$ , of the robots in the simulation, after the system has converged to an equilibrium state. There were 300 robots in the simulation. Because temperature affects speed, temperature was varied from 50° Kelvin to 400° Kelvin. We ran with six obstacle densities ranging from 0% to 50%, in order to determine the sensitivity of the average speed to obstacle density.

The results are striking, as can be seen in Fig. 5. The difference between the theoretical predictions of the average speed and the simulated average speed results in less than a 2% error, which is outstanding considering that as much as a 50% obstacle coverage has been introduced. Finally, note that we can use our theory to not only predict swarm behavior, but also to *control* it. Specifically, by setting the temperature  $T$ , a system designer can easily achieve a desired average speed.

## 10 Experiment 3: Velocity Distribution

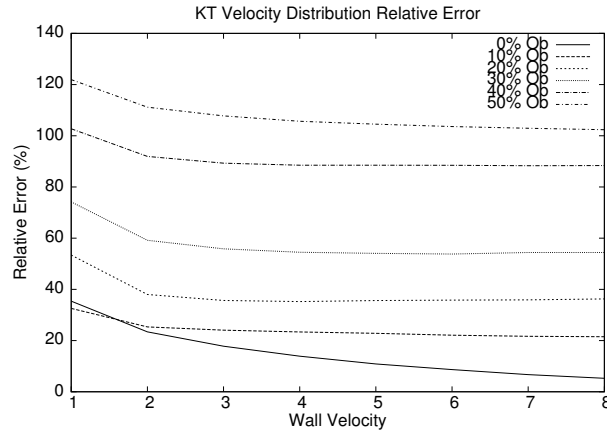
The third experiment concerns the velocity distribution of a robot swarm in a Couette. The theoretical prediction is compared with simulated behavior. Recall that in a Couette, fluid flow is in the  $y$ -direction – toward the goal. The  $x$ -direction is lateral, across the corridor. In addition to seeing how predictive the

theory is, this experiment also examines the relationship between wall speed,  $v_{wall}$ , and the velocity distribution of the robots in the system.

We first focus on a subtask in which a traditional one-sided Couette flow drives the bulk swarm movement. Our predictive formula is (see [9] for the derivation):

$$v_y = \frac{x}{D} v_{wall}$$

where  $v_{wall}$  is the velocity of the Couette wall,  $x$  is the lateral distance from the stationary wall, and  $D$  is the Couette width. In other words, the theory predicts a linear velocity distribution.

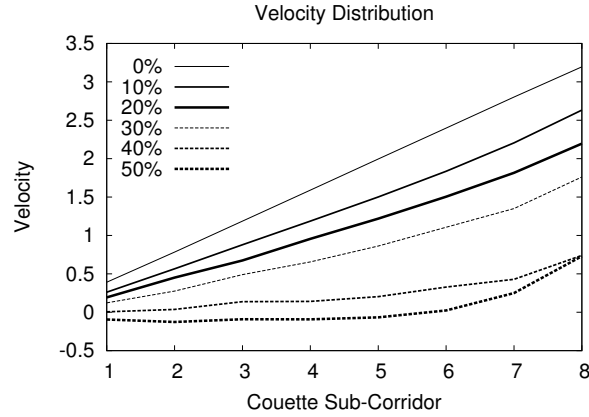


**Fig. 6.** Relative error for the KT velocity distribution.

We set up an experiment to measure the relative error between theory and simulation, consisting of 300 robots. The corridor was divided into eight discrete longitudinal sub-corridors. Theory predicts what the average speed will be lengthwise (in the goal direction) along each of the sub-corridors. Within each sub-corridor, the average  $y$  velocity of the robots is measured. The relative error between the theory and the experimental results is then calculated, for each sub-corridor. Finally, the relative error is averaged across all sub-corridors and plotted in Fig. 6 for eight different wall speeds and six different obstacle percentages. Note that although the error is reasonably low for 0% obstacles and high wall speeds, error increases dramatically as obstacles are added.

Why is there a discrepancy between the theory and experimental results? The reason is that theory predicts a certain linear velocity distribution, but assumes no obstacles. For simplicity, the theory assumes that robots never move backward (back up the corridor). In the simulator, on the other hand, robots *do* move backward, regardless of whether or not there are obstacles – because the



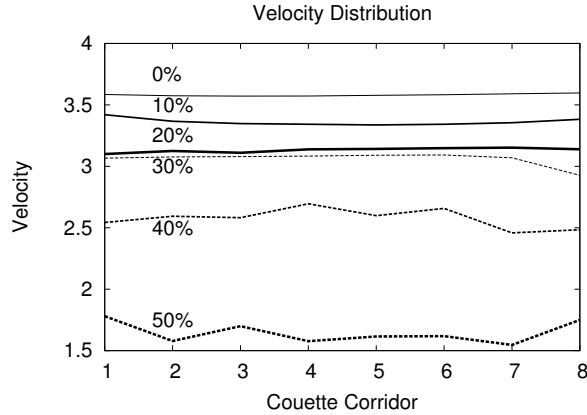


**Fig. 7.** The velocity distributions as the density of obstacles increases, for one-sided Couette flow.

simulation has a random component. In fact, as obstacles are introduced into the simulated world, the frequency of backward moving robots increases substantially. To examine more closely the effect of obstacles, Figure 7 shows the velocity distributions themselves (where the wall velocity  $v_{wall} = 4$ ). Even with no obstacles, the maximum robot velocity does not quite reach 4.0 (we would expect 3.75 in the sub-corridor nearest to the wall). This is caused by the backward moves. What is interesting is that the velocity distributions remain linear up to a reasonable obstacle density (30%), while the slope decreases as obstacles are added. Adding obstacles is roughly equivalent, therefore, to lowering the wall velocity  $v_{wall}$ !

To see if the correlation between obstacle density and wall velocity holds in the two-sided Couette flow, we re-ran the previous experiment, but with *both* walls having  $v_{wall} = 4$ . The results are shown in Figure 8. The theory predicts (see [9] for the derivation) that for the two-sided Couette,  $v_y = v_{wall}$  regardless of the value of  $x$ . Note that, as theory predicts, the velocity distribution of the flow is independent of the distance from the walls – the large scale fluid motion is approximately that of the walls. Again, increasing obstacle density is very similar to decreasing wall speed.

In conclusion, without obstacles, the theory becomes highly predictive as the wall velocity increases. Furthermore, this very predictive theoretical formula can also be used to achieve a desired swarm velocity distribution, i.e., to *control* the swarm – simply set the value of  $v_{wall}$ , the virtual wall speed, to achieve the desired distribution, using the formula. On the other hand, with an increasing number of obstacles, the predictability of the theory is increasingly reduced. However, we have shown that (up to quite reasonable densities) the increase in the number of obstacles is roughly proportional to a reduction in wall velocity,  $v_{wall}$ .



**Fig. 8.** The velocity distributions as the density of obstacles increases, for two-sided Couette flow.

## 11 Discussion of Theoretical Predictions

With respect to the average speed and the spatial distribution, the theory is highly predictive. Although the theory assumes no obstacles, the addition of obstacles to the simulation has a minimal effect on the results, with errors typically under 10%. Surprisingly, this level of theoretical accuracy is achieved with only hundreds of robots, which is very small from a kinetic theory perspective.

Our results for the velocity distribution are acceptable for no obstacles, generally giving errors less than 20%. As the obstacle density increases, so does the error. However, we have shown that an increase in obstacle density changes the slope of the linear velocity distribution. This is roughly equivalent to a commensurate reduction in wall speed.

In summary, we can conclude that when the actual scenario closely coincides with the theoretical assumptions (e.g., few obstacles), the theory is highly predictive. Also, we have provided an important insight into the nature of the effect that obstacle density has on the system. The most important conclusion to be drawn from these experiments is that in the future we can largely design KT systems using theory, rather than computationally intensive simulations, for the selection of optimal parameter settings. A subsidiary conclusion is that we have verified the correctness of our swarm code, which is something quite straightforward for a physics-based approach but much more difficult for alternative approaches.

## 12 Conclusions

KT uses a stochastic algorithm for updating particle positions; therefore KT predictions can only be approximate. Furthermore, as stated in [10], Monte Carlo

simulations such as KT need very long runs and huge numbers of particles to acquire enough statistical data to produce highly accurate (theoretically predictable) results. We cannot guarantee this, since we are developing control algorithms for robotic swarms with a few to a few thousand robots under strict time limitations. Despite all of these limitations of our KT robotic swarm simulation, the theory is nevertheless highly predictive of the simulation results. The conclusion is that our approach of using KT for designing swarm-based multi-agent systems has great promise for engineering swarms that are “safe.”

### 13 Related and Future Work

The work that is most related consists of other theoretical analyses of swarm systems. Our comparisons are in terms of the goal and method of analysis. There are generally two goals: stability and convergence/correctness. Under stability is the work in [13–15]. Convergence/correctness work includes [13]. Other goals of theoretical analyses include time complexity [16], synthesis [17], prediction of movement cohesion [13], coalition size [14], number of instigators to switch strategies [18], and collision frequency [19].

Methods of analysis are also diverse. The most relevant is work on physics-based analyses of physics-based swarm robotics systems. We are aware of four classes of methods. The first is Lyapunov analyses, e.g., [15]. The second is force and energy analyses, e.g., [12,17]. The third develops macro-level equations describing flocking [20]. Finally, the fourth is the most related work of all – the KT research by Jantz and Doty [19]. Note that although Jantz and Doty showed that KT can be used to model multi-agent swarms with predictable behavior [19], our research extends theirs by providing a much more extensive and methodical study of the relationship between kinetic theory and simulation.

Our current research [21,22] has compared KT against behavior-based approaches and found that it performs competitively, even if the alternative algorithms have more information. The next step is to transition KT from simulation to real robot swarms. We already have substantial progress on robotic implementations [23], and the next step will be to add KT.

### References

1. Perkins, T., Barto, A.: Lyapunov design for safe reinforcement learning control. In: AAAI Spring Symposium on “Safe Learning Agents”. (2002)
2. Barley, M., Guesgen, H.: Towards safe learning agents. In: AAAI Spring Symposium on “Safe Learning Agents”. (2002)
3. Shapiro, D.: Value-driven agents. PhD thesis, Stanford University (2001)
4. Gordon, D.: Asimovian adaptive agents. *Journal of Artificial Intelligence Research* **13** (2000)
5. Robinson, P., Hinchey, M., Clark, K.: Qu-prolog: An implementation language for agents with advanced reasoning capabilities. In: *Lecture Notes in Artificial Intelligence*. (2002) 162–172

6. Barbu, V., Sritharan, S.: Flow invariance preserving feedback controllers for the navier-stokes equation. *Journal of Mathematical Analysis and Applications* **255** (2001) 281–307
7. Reif, F.: *Fundamentals of Statistical and Thermal Physics*. McGraw-Hill (1965)
8. Spears, W., Gordon, D.: Using artificial physics to control agents. In: *IEEE International Conference on Information, Intelligence, and Systems*. (1999) 281–288
9. Kerr, W., Spears, D., Spears, W., Thayer, D.: Two formal gas models for multi-agent sweeping and obstacle avoidance. In: *Lecture Notes in Artificial Intelligence*. (2004)
10. Garcia, A.: *Numerical Methods for Physics*. Second edn. Prentice Hall (2000)
11. Kerr, W., Spears, D.: Robotic simulation of gases for a surveillance task. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*. (2005)
12. Spears, W., Gordon-Spears, D., Hamann, J., Heil, R.: Distributed, physics-based control of swarms of vehicles. *Autonomous Robots* **17** (August 2004) 137–162
13. Liu, Y., Passino, K., Polycarpou, M.: Stability analysis of m-dimensional asynchronous swarms with a fixed communication topology. In: *IEEE Transactions on Automatic Control*. Volume 48. (2003) 76–95
14. Lerman, K., Galstyan, A.: A general methodology for mathematical analysis of multi-agent systems. Technical Report ISI-TR-529, USC Information Sciences (2001)
15. Olfati-Saber, R., Murray, R.: Distributed cooperative control of multiple vehicle formations using structural potential functions. In: *IFAC World Congress*. (2002)
16. O. Shehory, S.K., Yadgar, O.: Emergent cooperative goal-satisfaction in large-scale automated-agent systems. *Artificial Intelligence* **110** (1999) 1–55
17. Reif, J., Wang, H.: Social potential fields: A distributed behavioral control for autonomous robots. In: *Workshop on the Algorithmic Foundations of Robotics*. (1998)
18. Numaoka, C.: Phase transitions in instigated collective decision making. *Adaptive Behavior* **3**(2) (1995) 185–222
19. Jantz, S., Doty, K.: Kinetics of robotics: The development of universal metrics in robotic swarms. Technical report, Dept of Electrical Engineering, University of Florida (1997)
20. Toner, J., Tu, Y.: Flocks, herds, and schools: A quantitative theory of flocking. *Physical Review E* **58**(4) (1998) 4828–4858
21. Spears, D., Kerr, W., Spears, W.: Physics-based robot swarms for coverage problems. *International Journal on Intelligent Control and Systems* (2006)
22. Kerr, W.: Physics-based multiagent systems and their application to coverage problems. Master's thesis, University of Wyoming (2005)
23. Spears, W., Hamann, J., Maxim, P., Kunkel, T., Zarzhitsky, D., Spears, D., Karlsson, C.: Where are you? In: *SAB Swarm Robotics Workshop*. (2006)

# Command and Authorization Services for Multiple Humans Interacting with a Software Control Agent for Advanced Life Support

Cheryl Martin<sup>2</sup>, Debra Schreckenghost<sup>1</sup>, and Pete Bonasso<sup>1</sup>

<sup>1</sup>NASA Johnson Space Center  
TRAC Labs  
1012 Hercules, Houston, TX, 77058, USA  
{ghost@ieee.org, r.p.bonasso@jsc.nasa.gov}

<sup>2</sup>Applied Research Laboratories, The University of Texas at Austin  
P.O. Box 8029, Austin, TX 78713, USA  
{cmartin@arlut.utexas.edu}

**Abstract.** This paper describes current work developing command and authorization services to support the coordination of multiple humans and an autonomous control agent working on the same underlying advanced life support system. The primary goal of these services is to prevent unknowing or accidental conflicts from arising as a result of issuing commands or taking action on the system. Avoiding such conflicts minimizes the risk of interfering with the work of another agent or putting the system into an unsafe operating state. This paper provides an overview of the advanced life support system at NASA to which this work has been applied and then discusses details for authorization, overrides, and system reconfiguration for commanding.

## 1 Introduction

NASA is currently investigating advanced life support systems for extended operation in future space habitats such as the space station or possible planetary sites. Since 1995, our group has been working at NASA's Johnson Space Center to provide intelligent control for advanced life support systems [5, 27]. These intelligent control systems have been realized by software agents using an architecture known as 3T [4] and were designed to run autonomously for months at a time. 3T is a layered control architecture whose top tier is a hierarchical task net (HTN) planner, the plans of which are executed through a reactive middle tier that in turn manages the sensors and actuators of the hardware via a low-level control tier. One such life support system is the advanced Water Recovery System (WRS). The WRS removes the organic and inorganic materials from waste water (hand wash, shower, urine and respiration condensate) to produce potable water.

In a previous paper, [3], we have explored safety-related issues for the design of the autonomous control software for the WRS system. These design issues include using adjustable autonomy to allow humans to interact with the agent safely, counter-

acting the slow degradation of hardware over time, being able to “safe” subsystems (put them into a shutdown or standby mode) in the event of power or communication failures, using checkpoints to quickly restore the WRS to nominal operations, and developing tools to help the human understand problem situations in order to recover from the anomaly.

In this paper, we focus on achieving the safe operation of the WRS by supporting the coordination of multiple humans and the control system, who may each take actions on the same underlying WRS system. Although the 3T-based automated control system operates the WRS hardware unattended most of the time, there are several cases in which humans must also take actions on the life support system. Actions that humans take can be either *manual* or *mediated*. *Manual* actions are those that the human carries out directly on the life support system hardware, for example, physically turning a valve. A human conducts *mediated* actions by giving instructions to the automation software, which carries out the actions. Manual and mediated actions are needed for two possible reasons (1) the action must be manual because the automation has no appropriate actuator or (2) the action could be carried out either by a human or via the software but is motivated by circumstances outside the scope of normal operation for the automation. In contrast, *automated* actions are those taken by the control software during its normal operation without any requests from an external source.

When possible, NASA astronauts rely on pre-built crew activity plans to ensure required resources and equipment are available and configured for manual activities. Yet circumstances arise where the crew needs to take action outside this plan, such as contingency response or opportunistic task performance. In such cases it may not be effective or even possible to rebuild the entire plan in light of situation changes and temporal constraints on response. We have developed an approach that permits safely interleaving unplanned crew activities with planned activities by verifying resources for upcoming activities, regardless of whether they are in the plan, are properly allocated and configured just prior to activity execution.

Challenges arise in coordinating humans and the control agent in their actions on the system because simultaneous or interleaved actions may be required or desired. The motivation for different agents to take different actions may arise from independent triggers or goals, and these actions may conflict with or impede each other. Further, it is difficult for humans to determine what actions other humans may be taking on the system because users may be located remotely from the WRS when taking mediated actions. It is also difficult for the autonomous control agent to determine what human agents are doing, both due to limited instrumentation of manual control inputs and due to the lack of models for manual actions. The availability of such models might allow the control agent to map observed human actions to known WRS procedures for the purpose of predicting the human’s next steps and maintaining safe operation of the WRS throughout the procedure.

In this paper, we present command and authorization services as implemented in a user support system for interacting with automated control agents called the Distributed Collaboration and Interaction (DCI) Environment. The goals of these services in DCI are (1) to decrease the risk of conflicting commands to the underlying physical system (2) to decrease the risk of interfering with the work of another agent (human

or the control agent) pertaining to the underlying physical system, and (3) to decrease the risk of the system being put into a harmful state by the action of any agent (for example, a state where pumps may be damaged by attempting to pull water from a blocked source). In order to achieve these goals, we offer software that assists a human user in performing mediated commands by verifying the requested system is not already being used by others (i.e., locked out), then reconfiguring the system so that it is safe to perform commands (mediated or manual). This reconfiguration support includes adjusting the autonomy of the autonomous control system when necessary. The DCI environment provides command lock-outs for possibly conflicting commands from different human users by selectively granting authorization to act on the system.

The next section describes research relevant to the DCI agent system that implements safe commanding. Two sections following that provide an overview of the WRS system and the commands on this system currently supported by the DCI prototype. The paper then discusses how DCI supports command and authorization capabilities including detailed discussions of the authorization model, the need for authorization overrides, and support for reconfiguring the WRS and its control agent to accommodate human activities.

## **2 Related Work**

To integrate humans into a multi-agent system alongside a software control agent and personal assistant agents, the DCI system has leveraged existing research across a wide range of areas including, human-agent interaction [16, 21], teaming and human-agent teams [7, 10, 19], user interfaces and underlying applications [9, 25], characteristics of autonomous agents including adjustable autonomy [1, 11, 24], and planning tools and mixed-initiative planning [12, 15, 18]. This section highlights some of this research and how it applies to aspects of multi-agent coordination among humans and agents as supported by the DCI Command and Authorization services.

The interface agents in the MokSAF environment support human interaction with software agents in human/agent teams in the domain of decision-support for military route planning [19]. These interface agents assist humans in tasking other agents (route-planning agents), present situation information to the human team members, and help humans communicate and coordinate with other humans. All of these capabilities are desirable for Ariel agents in DCI, and the MokSAF work identifies many important issues with respect to enabling an interface agent to act on a user's behalf. However, the MokSAF environment does not address DCI's need to interact with mostly autonomous agents because the route-planning agents are still human-centric in that their primary purpose is to assist a human in generating a route, if tasked by the human to do so.

A very successful and innovative implementation of interaction between humans and software agents has been demonstrated in the Electric Elves system to support human organizations [10]. Scerri et al have extended some of the findings of the Electric Elves project and successfully applied them to create teams of humans, robots, and agents [22, 23]. Bradshaw et al have also investigated human-agent team-

work in depth [7, 8]. They continue to develop policies to support agent interaction and teamwork, based on KAoS agent services [6]. These systems incorporate multiple humans and multiple agents; however, they do not fully address our requirements for support agents to act as mediators and/or enablers for humans to interact with yet a third class of agents: autonomous control systems who must act on the same physical system that humans may act on.

There are similar safety concerns when humans interact with robots. There has been considerable research [2, 13, 14, 20] on modeling human behavior to improve human-robot interaction. DCI also uses human modeling to improve interaction. There is a fundamental difference in the DCI approach and this research in human-robot interaction (HRI), however. In the HRI research, human models and modalities are used for the purpose of giving robots' human-like behavior. The belief is that agents that behave like humans are more understandable to humans. In DCI, human models are used to give computer-based agents such as robots insight into the behavior of their human users to improve interaction. By combining the current state and tasks of human users with knowledge of operational protocols, the Ariel agents can mediate between the different behaviors and perspectives of the computer-based agent and the human.

Adjustable autonomy has been defined as the transfer of responsibility to take action among a set of agents [24]. The adjustment of an agent's autonomy to take control actions is necessary for space systems where shared resources are often constrained and equipment can be used by both humans and agents. Scerri et al. have investigated transferring control among multiple agents using strategies that automatically determine how and when to adjust agent autonomy. In our approach, however, the human determines when to adjust agent autonomy to transfer responsibility among humans and software control agents [26]. We implement this transfer of control by reconfiguring a mostly autonomous control agent to temporarily release control of resources or equipment that the agent typically manages to permit a human to take action on these resources or equipment.

### 3 Water Recovery System (WRS) Overview

The WRS is composed of four subsystems shown in **Fig. 1**. These subsystems are loosely coupled, and their primary interdependencies are related to input and output of the water to be processed.

- (1) The *biological water processor (BWP)* removes organic compounds and ammonia by circulating the water through a two-stage bioreactor. The first stage uses microbes to consume the organic material using oxygen from nitrate molecules. The second stage uses microbes to convert the ammonium to nitrate.
- (2) The *reverse-osmosis (RO)* subsystem removes inorganic compounds from the output of the BWP, by forcing the water to flow at high pressure through a molecular sieve. The sieve rejects the inorganic compounds, concentrating them into brine. At the output of the RO, 85% of the water is ready for post-processing, and 15% of the water is brine.



(3) The *air evaporation system (AES)* removes the concentrated salts from the brine by depositing it on a wick, blowing heated air through the wick, and then cooling the air. The inorganic wastes are left on the wick and the condensate water is ready for post processing.

(4) The *post-processing system (PPS)* makes the water potable by removing the trace inorganic wastes and ammonium using a series of ion exchange beds and by removing the trace organic carbons using a series of ultra-violet lamps.

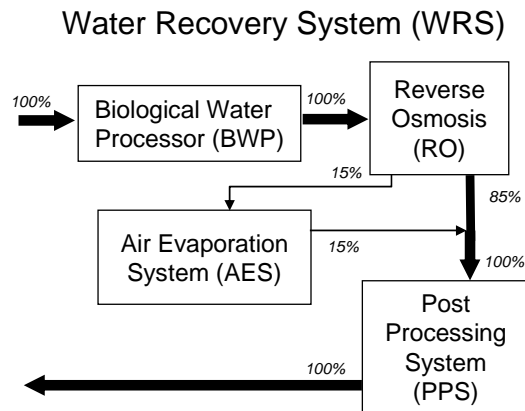
In total, the automated control system for the WRS manages more than 200 sensors (measuring pressure, temperature, air and water flow rates, pH, humidity, dissolved oxygen, and conductivity) and actuators (including pumps, valves, ultra-violet lamps, and heaters).

#### 4 WRS Activities Supported

Our current work concerning command and authorization addresses the coordination of multiple humans with each other and with the automation before, during, and after the execution of human-initiated actions on the WRS hardware. We currently support these four human-initiated activities:

- *BWP nitrifier slough* – The biofilm that grows on the insides of the tubes in the nitrifying portion of the BWP will thicken over time, slowly constricting the passage of water and air. To minimize clogs, the control system periodically sloughs the biofilm by sharply increasing the airflow. This automatic slough is only partially effective, and eventually a human is required to manually slough the nitrifier using high pressure water flow. The configuration for this activity requires ensuring that water is flowing in the BWP as well as suspending the automatic shutdowns (ASDs) that the control automation will normally enact if tube pressure readings go outside the nominal range. The manual slough takes from twenty minutes to an hour to complete. The BWP nitrifier slough is a manual activity.

- *RO slough* – Inorganic deposits may accumulate inside the RO's tubular mem-



**Fig. 1.** Schematic diagram of WRS system and subsystems

branes. If the water flow is reversed, a small ball in each tube will slide along the tube length, sloughing this buildup away. The automated control system carries out this RO slough at a predetermined frequency. If the RO output quality degrades, a human may manually command the control system to slough the membranes again. Reconfiguration for this activity requires the RO to be shutdown. The RO slough takes four minutes to complete followed by a thirty minute purge of the RO subsystem. The RO slough is a mediated activity. This is the only mediated action the command and authorization service currently supports.

- *RO membrane change out* – Eventually the RO membranes lose their efficiency and must be physically replaced. The RO is shutdown, and the upstream and downstream subsystems are placed in standby mode. The change out takes approximately twelve hours to complete. The RO membrane change out is a manual activity.

- *BWP pressure calibration* – Pressure sensors are the primary input used to control the BWP. These sensors require calibration about every three months. In order to conduct the calibration, the BWP must be disconnected from the downstream subsystems and placed in a standby mode. The calibration procedure usually takes from four to six hours to complete. The BWP pressure calibration is a manual activity.

## 5 Commanding the WRS

When a human wishes to perform actions on the WRS using the command and authorization capability in the DCI environment, he or she requests the appropriate commanding permission for a particular activity. Throughout the paper, the term *authorization* implies a license to take action on the WRS or one of its subsystems by the designated user. We use the term *commanding* to convey this authorization plus the concept of whether the system is ready for the execution of a particular activity associated with a pre-defined procedure. To grant commanding for a given activity, DCI must first, if possible, grant authorization for the set of manual or mediated actions (including reconfiguration actions) required by the activity, and then reconfigure the WRS hardware and control automation to the proper state required for the activity.

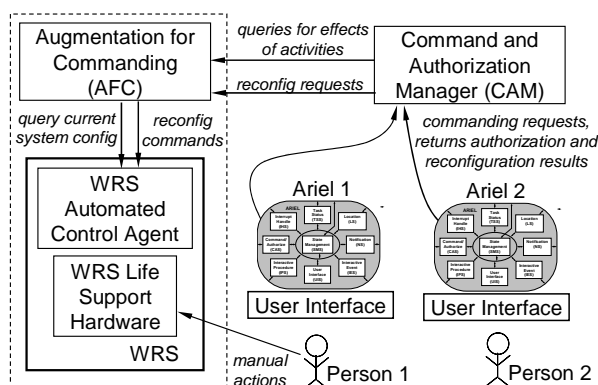
In the DCI environment, each user is represented by an Ariel agent [17], which acts as a liaison between the user and the rest of the software environment. An Ariel agent provides a human-centric interface into the software environment and delivers a number of services including notification, task tracking, and location tracking. In particular, the Ariel agent provides a Command and Authorization Service, which assists its user with command and authorization requests. **Fig. 2** shows two Ariel agents, the WRS system, and components discussed in the upcoming subsections: the Command and Authorization Manager (CAM) and the Augmentation for Commanding (AFC).

## 5.1 Command and Authorization Manager (CAM)

The CAM accepts requests for commanding from users through their Ariel agents. Each request is associated with an activity that the user wishes to perform. The CAM first queries the AFC (see next subsection) for information about the effects of the requested activity on the hardware system as well as any configuration conflicts between the current system configuration and the configuration required for the activity. Section 5, below, describes how the CAM uses the results of this query to grant or deny authorization. If authorization is denied, this result is returned to the user along with a description of the configuration conflicts. If authorization is granted, and the user wishes to continue, the CAM asks the AFC to carry out any required reconfiguration on the WRS including orchestrating required manual actions. Once the reconfiguration, if any, is complete, the CAM informs the user through his or her Ariel that the WRS is ready to command. The user can then proceed with the actions required by the procedure for the requested activity. When the user has completed the activity, he or she requests the CAM to release commanding for the activity. The CAM informs the AFC that the activity's configuration is no longer required (which may result in additional reconfiguration of the WRS by the AFC to “un-configure” for the activity) and then releases the authorization.

## 5.2 Augmentation for Commanding (AFC)

The AFC is a piece of augmenting software in the DCI environment (shown by the dotted lines indicating coupling to the WRS). *Augmenting software* is tightly coupled to the automation through shared models or data but has its own processing resources. In this case, the AFC shares static models of both the physical WRS system and the procedures that can be performed on the system (including reconfiguration procedures). Using these models, the AFC can predict how various activities will affect the WRS. The AFC can also query the WRS control agent dynamically to get the



**Fig. 2.** Implementation of Command and Authorization service in the DCI environment

current system configuration.

When the CAM queries the AFC about the effects of an activity, the AFC provides two results. First, the AFC decomposes the associated reconfiguration procedure (as well as the activity's procedure model, if available) to determine and return all components of the WRS that may be affected by the activity. In the current implementation, this result is highly abstracted and consists of an indicator for the system or subsystem that is affected, designating that the entire system/subsystem would be dedicated to this activity. This system/subsystem approach is made extensible by also returning the specific decomposition of subsystems that are affected by the reconfiguration (in the future, subcomponents of the subsystems may also be used here). Second, the AFC queries the WRS automated control agent for the current system configuration (i.e., the current state of the eight valves and ten pumps in the WRS) and returns a list of conflicts between the current state and the state that would result from reconfiguration. The CAM uses the first result to determine whether to grant authorization for the activity, and passes the second set of results back to the user.

If the CAM asks the AFC to reconfigure the WRS for a requested activity, the AFC triggers the WRS control agent to perform the reconfiguration, if any. During the course of the reconfiguration, some manual actions may also be required. When it is time for a manual reconfiguration action, the WRS control agent, through the AFC, CAM, and the Ariel agent's user interface, requests the user to perform the action and waits for a return indication from the user that it is accomplished. This feedback from the user is needed because manually operated physical devices are not normally instrumented for computers, so manual actions are not easily observable by the software for tracking a user's progress in the reconfiguration. Once all reconfiguration actions have been completed, the CAM informs the user that the WRS is ready for commanding.

## **6 Managing Authorizations and Overrides**

Authorization to act on the WRS is managed by the CAM. The CAM is centralized to provide synchronized access from multiple entities (various Ariel agents and, in the future, the automated control system itself) to a single model describing which entities hold which authorizations. In general, granting authorization to one entity for a given scope of action blocks other entities from receiving authorization overlapping that scope until the first authorization is released. This blocking authorization paradigm is a well-known technique and is applied here to prevent multiple entities from acting on the WRS simultaneously for activities within the same scope, which may therefore interfere with one another.

When possible, the CAM should authorize concurrent activities that can be achieved safely together. In our life support domain, not only is crew health a top priority, but crew time is also a very valuable resource. Therefore we want to minimize the circumstances under which our system might unnecessarily block a crew member from performing an activity on the life support system or unnecessarily slow down that crew member. Further, our design philosophy must account for the nature and culture of space exploration in which crew safety is considered to be the top

mission priority, above vehicle health and mission success. Since life support systems are required for crew safety, inadvertently taking actions that impede or interfere with crew life support can have a negative effect on crew safety. *Absolutely* preventing a crew member from performing *any* activity on a life support system could potentially be fatal, given an unforeseen circumstance or an emergency situation.

Therefore, our authorization design goal is to enhance safe operation of the life support system by helping to coordinate humans and the control agent to prevent unknowing or accidental conflicts. However, we are fully cognizant that a well trained and fully informed crew member should be allowed to override any blocking authorization that may exist, and take action, risking a conflict in order to achieve a possibly higher purpose. Consequently, our design has two components (1) determine which activities can be safely authorized for concurrent execution and allow the maximum concurrency possible, and (2) if an activity cannot be authorized because it cannot be guaranteed for safe execution in conjunction with other currently authorized activities, provide as much information as possible about potential conflicts to the user and allow the user to override the authorization. The following subsections discuss each of these design components in turn.

## 6.1 Authorizations

We believe that the maximum concurrency without risking conflicts can be achieved by authorizing activities Act1 and Act2 concurrently as long as (1) their configurations do not conflict (states of the hardware and software) and (2) no action taken for Act1 (during reconfiguration or the procedure itself) affects the same component or state value (i.e., valve position) as any action taken for Act2, and vice versa. For our initial approach, we use models already within the WRS control agent to support command and authorization and limit our development of new models. Unfortunately, (1) the existing models for the required configurations are not detailed enough to guarantee no conflicts (e.g., they have not been extended to include required operating characteristics of the automation) and (2) we do not have models of the procedures for activities that require only manual action.

Until we extend the activity models and reconfiguration models to overcome these limitations, we have initially adopted a conservative approach to authorization that works well with the existing models but does not allow the maximum possible authorization concurrency. The approach is conservative in that it locks authorization for an entire subsystem (e.g. the RO) if any component of that subsystem is affected by an activity (by the reconfiguration, or the activity itself if a model exists), and it locks authorization for the entire WRS if multiple subsystems or the dependencies between subsystems (e.g. water flow) are affected. For the small set of actions and scenarios we have considered thus far, the conservative nature of this approach has not been a disadvantage.

When a user requests commanding permission for a given activity from the CAM, the CAM obtains information from the AFC about the highest-level system or subsystem affected by the activity. The CAM translates the system/subsystem decomposition into a model of scopes for granted authorization. Let  $\mathcal{D}$  be the set of all system

components such that authorization can be assigned for the scope of that component. For the current implementation  $\Phi = \{WRS, BWP, RO, AES, PPS\}$ . For the variables  $x$  and  $y$ , let  $x, y \in \Phi$ . Let  $Sub(x, y)$  define a predicate that indicates whether component  $x$  is a subsystem or subcomponent of component  $y$  in a hierarchical decomposition of the system. For the current implementation, the following hold:  $Sub(BWP, WRS)$ ,  $Sub(RO, WRS)$ ,  $Sub(AES, WRS)$ ,  $Sub(PPS, WRS)$ .

Let  $\alpha$  be the set of all agents (including humans and the automated control agent) that can act on the system. For the variables  $a$  and  $b$ , let  $a, b \in \alpha$ . Let  $Auth(a, x)$  define a predicate indicating that agent  $a$  has authorization to act over the scope of system component  $x$ .

The CAM uses the following rule to assign authorizations: When  $b$  requests  $Auth(b, x)$ , then grant  $Auth(b, x)$  if and only if no other agent holds the authorization for  $x$ , for any of  $x$ 's subsystems, or for any component that has  $x$  as a subsystem. In other words, when  $request( Auth(b, x) )$ ,

if  $\forall a, \neg Auth(a, x)$   
 $\wedge \forall a, y, Sub(x, y) \Rightarrow \neg Auth(a, y)$   
 $\wedge \forall a, y, Sub(y, x) \Rightarrow \neg Auth(a, y)$   
 then  $Auth(b, x)$ .

## 6.2 Overrides

If the CAM denies a user authorization to act on the system, the user should (by policy) wait until the authorization can be granted before taking any action. However, enforcing such a lockout could prevent a user from taking needed action in an emergency, which is a particularly troubling prospect with respect to a critical life support system. The development and use of more sophisticated models for the effects of activities on the system will allow us to avoid being overly conservative, maximizing the number of activities we can authorize concurrently. However, these advances will not address situations in which a low-priority ongoing activity may block authorization for an emergent higher-priority activity. We are currently working on building a user override capability for denied authorizations. The override capability should allow the user to obtain the authorization and perform the activity with *no less* protection from conflicts with newly arising tasks than the protection provided to a user granted a normal authorization. However, granting an override authorization is more complex than simply granting a new authorization that happens to conflict with existing authorizations. In particular, the specific areas of conflict must be identified and the appropriate users who currently hold authorizations must be notified about any potential problems that might arise in the context of the new override authorization. Determining the correct reconfiguration actions to take for an override situation also raises new questions. If configurations required for two simultaneously authorized activities conflict (i.e., require different state values or software modes), how should priority for setting these states be determined? We are currently working on a design to address these override issues. Explicit override capabilities are not currently supported in the prototype implementation.

The current implementation does allow overrides to occur, however, because the current WRS implementation offers limited options for enforcement of either denied authorizations or denied system access in general. There is some password protection for mediated actions, but anyone could theoretically walk up to the system at any time and, for example, power down a pump. We hope to improve enforcement as the override software support is developed. Suri et al describes relevant previous work on policy enforcement [29]. In the interim, when an authorization is denied, the CAM reports back to the requesting user the set of pre-existing authorizations that conflict with the request as well as the list of conflicts between the current system configuration and the requested activity's configuration. The highly trained user can consider this information to determine how to proceed. He or she may ask other users holding a conflicting authorization to release it, or he or she may proceed *manually* with the desired reconfiguration and activity with foreknowledge of possible conflicts that may arise. Although much work remains, making users aware of possible conflicts arising from ongoing activities by other users on the WRS is an important first step toward supporting the coordination of multiple humans and an automated control agent working on the same underlying physical system.

### 6.3 A Note on Security

The current CAM implementation assumes that every entity requesting authorization possesses the necessary credentials (authentication, skills, and/or certificates) for the authorization to be granted. We would like to add credential checking in the future. However, it is not currently critical in our application because (1) we assume all possible users (NASA crew) are highly trained and (2) our authorization process is used primarily for coordination rather than access control enforcement. Although users must log in to use the DCI environment (authentication), they can currently act on the WRS by circumventing DCI completely. Users are motivated to request commanding permission through DCI primarily to minimize the risk of conflicts for themselves and the control agent and to obtain assistance from the AFC in reconfiguring the WRS hardware and the control agent for the desired activity. However, the users currently do not *need* the system's permission to take action.

## 7 Reconfiguration for Commanding

Reconfiguration for commanding is managed by the AFC. The AFC is coupled to the WRS automated control agent and shares its static models of both the physical WRS system and the procedures that can be performed on the system (including reconfiguration procedures). Using these models, the AFC can predict how various activities will affect the WRS. The AFC can also query the WRS control agent dynamically to get the current system configuration and it can trigger the WRS control agent to take actions to carry out any reconfiguration necessary to prepare for an activity. In general, the reconfiguration process may include setting the states of particular hardware such as valves open/closed or pumps on/off, adjusting the autonomy of the automa-

tion to allow for manual actions [28], bringing the state of the system to a particular point such as getting tube pressures or heater temperatures within a specified range, or commanding a subsystem to a particular processing mode. The current implementation handles a subset of these types of reconfiguration actions and affects both hardware (the states of eight valves and ten pumps) and software (the operating characteristics of the automated control system). Actions required to achieve the reconfiguration necessary for each of these activities may be either manual or mediated. Note that *mediated* actions are performed by the control agent, but triggered externally, and they can be initiated by a human or by external software. Actions taken by the WRS control agent in the course of reconfiguration are examples of mediated actions that are initiated by external software (the AFC).

We have found that models of reconfiguration procedures can be used to (1) determine what parts of the WRS would be affected by (re)configuring for an activity and (2) allow the AFC to trigger the WRS control agent to perform the reconfiguration necessary. Except for mediated activities, such as the RO slough, in which the control agent performs the actions in the body of the activity itself, models of reconfiguration procedures were not originally developed for the WRS control agent because they were not necessary for autonomous operation. In support of the DCI commanding capability, we added models of the reconfiguration procedures for the other three activities described above in Section 3.

The AFC ensures that the WRS maintains the configuration, as a whole, required to support all of the currently authorized activities. If authorization were allowed for only one activity at any given time, the AFC could support reconfiguration for this activity by first triggering the WRS to execute the reconfiguration procedure for that activity after authorization is granted and then triggering the WRS to execute the reconfiguration procedure to return to nominal autonomous operation before authorization is released. However, since multiple authorizations should be supported, the AFC must unify the configuration states required for all concurrent authorizations. The following paragraphs describe how the AFC and WRS control agent together achieve the desired unified configuration for all currently authorized activities.

Let  $C$  be the set of all components in the WRS system. In general, this set may include hardware (valves and pumps), software modules, measurable operating characteristics (such as tube pressure), or abstractions of groups of system pieces such as subsystems. To apply the reasoning presented here, the members of  $C$  must be independent and separable. This means, for example, that no pump listed as a member of  $C$  can be a part of the BWP subsystem if the BWP subsystem is also a member of  $C$ . For the variable  $c$ , let  $c \in C$ .

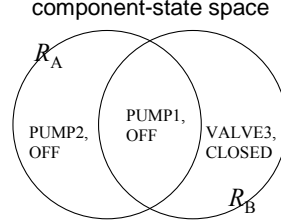
Let  $S$  be the set of all states that components in the WRS system can take. Examples of possible state values in  $S$  may include ON, OFF, OPEN, CLOSED, <180psi, STANDBY, etc. For the variable  $s$ , let  $s \in S$ .

Let the tuple  $(c, s)$  be a component-state pair<sup>1</sup> in which component  $c$  takes on the state value  $s$ . Let  $R$  be a set of  $n$  component-state pairs representing a configuration state in the WRS. Components in  $C$  that are not included in any element of  $R$  have no

---

<sup>1</sup> To simplify this discussion, we will not explicitly disallow unrealistic component-state pairs such as (TUBE1PRESSURE, ON) or (VALVE2, <180psi).





**Fig. 3.** Venn diagram of required configurations,  $R$ , for Activity A and Activity B

specific state requirement for that configuration (i.e., the states of these components are *don't cares* in the configuration).

$$R = \{(c^1, s^1), (c^2, s^2), \dots, (c^n, s^n)\}.$$

The desired configuration state for a given activity can be determined by examining the reconfiguration procedure for that activity. The AFC keeps a prioritized list of the configurations required. The lowest priority configuration (priority 0) is the normal operating configuration (nominal-ops) during which no activities are currently authorized. When any activity is authorized, its configuration is added to the list and given the next highest priority above nominal-ops (priority 1). Once support for override authorizations is implemented, the configuration for activities requiring overrides would be added to the list at even higher priority levels. Assigning these priorities correctly for configurations related to overrides is an open research issue. Given this prioritized list, the unified configuration is determined by stepping through each configuration, starting with the lowest priority configuration, and adding its component-state pairs to the unified result. As each component-state pair is added, it will overwrite any pair containing the same component in the unified configuration. Configurations with the same priority may be processed in any order. No component may have more than one required state value at the same priority because such conflicts would require an override. Therefore, in the final unified configuration, only the highest priority state for each component will be included. The AFC triggers the WRS control agent to apply this desired unified configuration each time a change in authorizations occurs.

Consider the following example containing no overrides: Assume, that the nominal-ops configuration for normal operation during which no activities are authorized is  $R_N = \{(PUMP1, ON), (PUMP2, ON), (VALVE3, OPEN)\}$ . Activity A requires configuration  $R_A = \{(PUMP1, OFF), (PUMP2, OFF)\}$  and Activity B requires configuration  $R_B = \{(VALVE3, CLOSED), (PUMP1, OFF)\}$ . These configurations overlap (contain the same component holding the same state value) as shown in **Fig. 3**, but do not conflict (do not contain the same component with different state values).

Therefore, these activities A and B could be authorized concurrently<sup>2</sup> with the same configuration priority. **Table 1** shows some example sequences of authorizations for these activities (assuming concurrent authorization is supported) and how the configuration of the WRS would change to accommodate these authorizations.

**Table 1.** Possible authorization sequences with resulting configuration changes

Time	Authorized Activities	Prioritized Configurations	Unified Desired Configuration	Actions Taken
$t_0$	none	$(R_N)$	PUMP1, ON PUMP2, ON VALVE3, OPEN	none
$t_1$	A	$(R_N, R_A)$	PUMP1, OFF PUMP2, OFF VALVE3, OPEN	<i>turn off</i> PUMP1 <i>turn off</i> PUMP2
$t_2$	none	$(R_N)$	PUMP1, ON PUMP2, ON VALVE3, OPEN	<i>turn on</i> PUMP1 <i>turn on</i> PUMP2
$t_3$	B	$(R_N, R_B)$	PUMP1, OFF PUMP2, ON VALVE3, CLOSED	<i>turn off</i> PUMP1 <i>close</i> VALVE3
$t_4$	B, A	$(R_N, R_B, R_A)$	PUMP1, OFF PUMP2, OFF VALVE3, CLOSED	<i>turn off</i> PUMP2
$t_5$	A	$(R_N, R_A)$	PUMP1, OFF PUMP2, OFF VALVE3, OPEN	<i>open</i> VALVE3

## 8 Conclusions

The command and authorization services in the DCI environment are designed to support the safe operation of advanced life support systems and their intelligent control agents by enhancing the coordination among multiple humans and these control agents. This work is still preliminary, but supports future evaluation with respect to safety metrics and guarantees. Our prototype system makes users aware of possible conflicts arising from ongoing activities by other users on the WRS system. We have developed and implemented a conservative policy for granting authorization to act on the system, which ensures that no more than one user at a time is authorized to use a system or subsystem. Further, as an integral part of processing a human's request to perform an activity on the physical system, we provide previously unavailable assistance in reconfiguring the system for that activity. By suspending or modifying automatic responses in the control system for the duration of human-initiated activities, we have also enhanced coordination between humans and the automation.

<sup>2</sup> However, for our currently implemented conservative authorization model in which users block an entire subsystem if they affect a single component in that subsystem, these activities would not be authorized concurrently.

We plan to enhance our conservative authorization policy in the future as we develop improved models of the effects of human activity on the life support system and therefore better understand possible sources of conflict. We would like to further enhance our authorization capabilities by supporting credential checking as well as authorization enforcement and override capabilities. Finally, we plan to extend this work to better support coordination with the autonomous system. This additional support would include (1) extending supported activities to those containing a mixture of manual, mediated, and automated actions, (2) making more extensive use of the adjustable autonomy and traded control capabilities of the automation, (3) granting explicit authorizations to the automation in addition to humans such that humans are protected from unknowingly acting on the system when the automation is performing a critical operation, and (4) integrating command and authorization with control planning for the autonomous system and task planning for the human to avoid redundant reconfiguration. Although much work remains to fully support safe human commanding and authorization in coordination with autonomous systems, the preliminary work presented in this paper provides both enhanced capabilities and encouragement that we have defined a reasonable path forward.

## 9 Acknowledgements

This work was performed under the project titled "Distributed Crew Interaction with Advanced Life Support Control Systems" and supported by the Human-Centered Computing area of NASA's Intelligent Systems Program, which is under the direction of Dr Michael Shafto.

## References

1. K. S. Barber, C. E. Martin, N. E. Reed, and D. Kortenkamp, "Dimensions of Adjustable Autonomy," in *Advances in Artificial Intelligence: PRICAI 2000 Workshop Reader. Four Workshops held at PRICAI 2000, Melbourne, Australia, August/September 2000*, vol. LNAI 2112, R. Kowalczyk, S. W. Loke, N. E. Reed, and G. Williams, Eds., Revised Papers ed. Berlin: Springer, 2001, pp. 353-361.
2. W. Bluethmann, R. Ambrose, M. Diftler, E. Huber, M. Goza, C. Lovchik, and D. Magruder, "Robonaut: A Robot Designed to Work with Humans in Space," *Autonomous Robots*, vol. 14, pp. 179-207, 2003.
3. R. P. Bonasso, "Safe Agents for Life Support," in *Proc. Workshop on Safe Agents at AAMAS'03*, Melbourne, Australia, 2003, pp.
4. R. P. Bonasso, J. R. Firby, E. Gat, D. Kortenkamp, D. P. Miller, and M. G. Slack, "Experiences with an Architecture for Intelligent, Reactive Agents," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, pp. 237-256, 1997.
5. R. P. Bonasso, D. Kortenkamp, and C. Thronesbery, "Intelligent Control of a Water Recovery System: Three Years in the Trenches," *AI Magazine*, vol. 24, pp. 19-44, 2003.
6. J. M. Bradshaw, S. Dutfield, P. Benoit, and J. D. Woolley, "KAoS: Toward an Industrial-strength Generic Agent Architecture," in *Software Agents*, J. M. Bradshaw, Ed. Cambridge, MA: AAAI Press/ The MIT Press, 1997, pp. 375-418.

7. J. M. Bradshaw, M. Sierhuis, Y. Gawdiak, R. Jeffers, N. Suri, and M. Greaves, "Adjustable Autonomy and Teamwork for the Personal Satellite Assistant," in *Proc. IJCAI-2001 Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents*, Seattle, WA, 2001, pp. 20-26.
8. J. M. Bradshaw, A. Uszok, R. Jeffers, N. Suri, P. Hayes, M. Burstein, A. Acquisti, B. Benyo, M. Breedy, M. Carvalho, D. Diller, M. Johnson, S. Kulkarni, J. Lott, M. Sierhuis, and R. van Hoof, "Representation and Reasoning for DAML-Based Policy and Domain Services in KAoS and Nomads," in *Proc. Second International Joint Conference on Autonomous Agents and MultiAgent Systems*, Melbourne, Australia, 2003, pp. 835-842.
9. J. Cadiz, G. D. Venolia, G. Jancke, and A. Gupta, "Sideshow: Providing Peripheral Awareness of Important Information," Microsoft Research, Redmond, WA, Technical Report, MSR-TR-2001-83, September 14, 2001.
10. H. Chalupsky, Y. Gil, C. A. Knoblock, K. Lerman, J. Oh, D. V. Pynadath, T. A. Russ, and M. Tambe, "Electric Elves: Applying Agent Technology to Support Human Organizations," in *Proc. Innovative Applications of Artificial Intelligence*, Seattle, WA, 2001, pp. 51-58.
11. G. A. Dorais, R. P. Bonasso, D. Kortenkamp, B. Pell, and D. Schreckenghost, "Adjustable Autonomy for Human-Centered Autonomous Systems on Mars," in *Proc. Mars Society Conference*, 1998, pp.
12. G. Ferguson and J. F. Allen, "TRIPS: An Integrated Intelligent Problem-Solving Assistant," in *Proc. Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison, WI, 1998, pp.
13. M. Goodrich, "Using Models of Cognition in HRI Evaluation and Design," in *Proc. AAAI Fall Symposium. The Intersection of Cognitive Science and Robotics: From Interfaces to Intelligence*, Washington, DC, 2004, pp.
14. G. Hoffman and C. Breazeal, "Robots that Work in Collaboration with People," in *Proc. AAAI Fall Symposium. The Intersection of Cognitive Science and Robotics: From Interfaces to Intelligence*, Washington, DC, 2004, pp.
15. C. A. Knoblock, S. Minton, J. L. Ambite, M. Muslea, J. Oh, and M. Frank, "Mixed Initiative Multi-source Information Assistants," in *Proc. Proceedings of the 10th International World Wide Web Conference*, Hong Kong, 2001, pp. [www10.org/cdrom/papers/frame.html](http://www10.org/cdrom/papers/frame.html).
16. M. Lewis, "Designing for Human-Agent Interaction," *AI Magazine*, vol. 19, pp. 67-78, 1998.
17. C. E. Martin, D. Schreckenghost, R. P. Bonasso, D. Kortenkamp, T. Milam, and C. Thronesbery, "An Environment for Distributed Collaboration Among Humans and Software Agents," in *Proc. 2nd International Conference on Autonomous Agents and Multi-Agent Systems*, Melbourne, Australia, 2003, pp. 1062-1063.
18. K. L. Myers, M. W. Tyson, M. J. Wolverton, P. A. Jarvis, T. J. Lee, and M. desJardins, "PASSAT: A User-Centric Planning Framework," in *Proc. 3rd International NASA Workshop on Planning and Scheduling for Space*, Houston, TX, 2002, pp.
19. T. R. Payne, K. Sycara, and M. Lewis, "Varying the User Interaction within Multi-Agent Systems," in *Proc. Fourth International Conference on Autonomous Agents*, Barcelona, Catalonia, Spain, 2000, pp. 412-418.
20. D. Perzanowski, D. Brock, W. Adams, M. Bugajska, S. Thomas, S. Blisard, A. Schultz, J. G. Grafton, and G. Mintz, "Toward Multimodal Human-Robot Cooperation and Collaboration," in *Proc. AIAA 1st Intelligent Systems Technical Conference*, 2004, pp.
21. C. Rich and C. L. Sidner, "COLLAGEN: A Collaboration Manager for Software Interface Agents," *User Modeling and User-Adapted Interaction*, vol. 8(3-4), pp. 315-350, 1998.
22. P. Scerri, L. Johnson, D. V. Pynadath, P. S. Rosenbloom, N. Schurr, M. Si, and M. Tambe, "Getting Robots, Agents and People to Cooperate: An Initial Report," in *Proc.*

*AAAI Spring Symposium on Human Interaction with Autonomous Systems in Complex Environments*, Stanford, Palo Alto, CA, 2003, pp.

23. P. Scerri, D. V. Pynadath, L. Johnson, P. S. Rosenbloom, M. Si, N. Schurr, and M. Tambe, "A Prototype Infrastructure for Distributed Robot-Agent-Person Teams," in *Proc. 2nd International Conference on Autonomous Agents and Multi-Agent Systems*, Melbourne, Australia, 2003, pp. 433-440.
24. P. Scerri, D. V. Pynadath, and M. Tambe, "Adjustable Autonomy in Real-World Multi-Agent Environments," in *Proc. Autonomous Agents*, Montreal, Canada, 2001, pp. 300-307.
25. C. Schmandt, N. Marmasse, S. Marti, N. Sawhney, and S. Wheeler, "Everywhere Messaging," *IBM Systems Journal*, vol. 39(3&4), pp. 660-677, 2000.
26. D. Schreckenghost, R. P. Bonasso, C. E. Martin, and D. Kortenkamp, "Managing Concurrent Activities of Humans and Software Agents," in *Proc. AIAA 1st Intelligent Systems Technical Conference*, Chicago, IL, 2004, pp.
27. D. Schreckenghost, D. Ryan, C. Thronesbery, R. P. Bonasso, and D. Poirot, "Intelligent Control of Life Support Systems for Space Habitats," in *Proc. Tenth Conference on Innovative Applications of Artificial Intelligence*, Madison, WI, 1998, pp. 1140-1145.
28. D. Schreckenghost, C. Thronesbery, R. P. Bonasso, D. Kortenkamp, and C. E. Martin, "Intelligent Control of Life Support for Space Missions," *IEEE Intelligent Systems*, vol. 17, pp. 24-31, 2002.
29. N. Suri, J. M. Bradshaw, M. Burstein, A. Uszok, B. Benyo, M. Breedy, M. Carvalho, D. Diller, P. Groth, R. Jeffers, M. Johnson, S. Kulkarni, and J. Lott, "DAML-based Policy Enforcement for Semantic Data Transformation and Filtering in Multi-agent Systems," in *Proc. Second International Joint Conference on Autonomous Agents and MultiAgent Systems*, Melbourne, Australia, 2003, pp. 1132-1133.

# Enhancing secure Tropos to effectively deal with security requirements in the development of multiagent systems

H. Mouratidis<sup>1</sup>, P. Giorgini<sup>2</sup>

<sup>1</sup>School of Computing and Technology, University of East London, England  
[h.mouratidis@uel.ac.uk](mailto:h.mouratidis@uel.ac.uk)

<sup>2</sup>Department of Information and Communication Technology, University of Trento, Italy  
[paolo.giorgini@dit.unit.it](mailto:paolo.giorgini@dit.unit.it)

**Abstract.** The consideration of security requirements in the development of multi-agent systems is a very difficult task. However, only few approaches have been proposed that try to integrate security issues as internal part of the development process. Amongst them, secure Tropos has been proposed as a structured approach towards the consideration of security issues in the development of multiagent systems. In this paper we enhance secure Tropos by integrating to its stages: (i) a process for selecting amongst alternative architectural styles using as criteria the security requirements of the system; (ii) a pattern-based approach to transform security requirements to design, and (iii) a security attack scenarios approach to test the developed solution. The electronic single assessment process (eSAP) case study is used to illustrate our approach.

## 1 Introduction

Recently agent orientation is presented as the next major paradigm for the development of large complex computer systems. Although there are some convincing arguments for believing that agent orientation will be of benefit for engineering certain complex software systems [6], much work is still required so that it becomes widely accepted as a major development paradigm, and more research is required towards many areas related to it. One of these areas is security.

Security of information systems is an area that has received great attention the last few decades mainly due to the storage of sensitive information on computer systems, and the wide interconnection of these systems through networks and the Internet. As a result, security research is considered one of the most active areas of computer science and engineering research. Nevertheless the high amount of work on this area, current information systems are not considered totally secure. According to the U.S. National Research Council [17] poor design is one of the major reasons for the development of insecure systems. Current work on information systems security has been focused on the development and verification of security protocols and other low level solutions, and the consideration of security requirements as part of the whole system development has been neglected.

This is also the case for multiagent systems, since current agent oriented methodologies do not usually consider security as an integral part of their

development stages and as a result agent developers do not actually find any help when considering security during the stages of the system development. This is mainly because the consideration of security in the development phases is a demanding and difficult task, due to the following reasons:

- (a) developers, who are not security specialists, usually need to develop multiagent systems that require knowledge of security;
- (b) Many different concepts are used between security specialists and software engineers. As a result, there is an abstraction gap that makes the integration of security and software engineering more difficult;
- (c) there is an ad hoc approach towards security analysis;
- (d) It is difficult to define together security and functional components and at the same time provide a clear distinction. For instance, which components are part of the security architecture and which ones are part of the functional specification;
- (e) It is difficult to move from a set of security requirements to a design that satisfies these requirements, and also understand what are the consequences of adopting specific design solutions for such requirements;
- (f) It is difficult to get empirical evidence of security issues during the design stages. This makes the process of analysing security during the design stage more difficult;
- (g) It is difficult to fully test the proposed security solutions at the design level.

We believe that the agent oriented software engineering paradigm presents a feasible approach for the integration of security to software engineering due to the appropriateness of agent oriented philosophy, for dealing with the security issues that exist in a computer system. Security requirements are mainly obtained by analysing the attitude of the organisation towards security and after studying the security policy of the organisation. As mentioned in [6] agents act on behalf of individuals or companies interacting according to an underlying organisation context. The integration of security within this context will require for the rest of the subsystems (agents) to consider the security requirements, when specifying their objectives and interactions therefore causing the propagation of security requirements to the rest of the subsystem. In addition, the agent oriented view is perhaps the most natural way of characterising security issues in software systems. Characteristics, such as autonomy, intentionality and sociality, provided by the use of agent orientation allow developers first to model the security requirements in high-level, and then incrementally transform these requirements to security mechanisms [12].

In previous work [13, 14, 15] we have presented models and techniques towards the solution of problems a, b, c, and d. For example, we proposed a well guided security-oriented process that considers the same concepts and notations throughout the development lifecycle and it allows the parallel definition of security and functional requirements providing at the same time a clear distinction.

In this paper we extend our previous work to deal with problems e, f and g by extending the current secure Tropos. In particular, we propose a process for selecting amongst alternative architectural styles, a pattern-based approach to transform the

analysis to design, and a security attack scenarios approach to test the developed solution.

Our work is not the only one in the agent paradigm pool that tries to integrate security issues as an internal part of the development process. Liu et al. [10] presented work in which security requirements are analysed as relationships between actors, such as users, stakeholders and potential attackers. Liu proposes three different kinds of analysis techniques: agent oriented, goal oriented and scenario based analysis. Agent oriented analysis is used to model potential threats and security measures, whereas goal oriented analysis is employed for the development of a catalogue to help towards the identification of the different security relationships on the system. Finally, the scenario based analysis is considered an elaboration of the other two kinds of analysis. Yu and Cysneiros [21] use the concept of a soft-goal to assess different design alternatives, and how each of these alternatives would contribute positively or negatively in achieving the soft-goal. However, these approaches only guide the consideration of security issues on specific development stages, in other words both of them are focused only in the requirements engineering area. Our approach, in contrast, considers security issues throughout the development process. As indicated in [3], it is important to consider security issues throughout the development process. Moreover, Huget [5] has proposed an agent oriented software methodology, called Nemo, which considers some security aspects. However, in his approach security is not considered as a specific concept but it is integrated within the other models of the methodology. As indicated earlier, it is important to model together security and functional requirements but at the same time provide a clear distinction. Moreover, Nemo tackles security quite superficially and as Huget states [5] “particularly, security has to be intertwined more deeply within the models”.

The rest of the paper is structured as follows. Section 2 of the paper provides an overview of secure Tropos, mainly for readers not familiar with the methodology, whereas Section 3 introduces our approach providing answers to the problems e, f, and g presented above. Section 4 concludes the paper.

## 2 An overview of Secure Tropos

Tropos [1] is an agent oriented software engineering methodology, in which notions such as actors (entities that have strategic goals and intentionality), goals (an actor’s strategic interests), soft-goals (goals without clear criteria whether they are satisfied or not), tasks (represent in an abstract level a way of doing something), resources (represent a physical or informational entity) and intentional dependencies (indicate that one actor depends on another in order to attain some goals, execute some tasks, or deliver a resource) are used in all the phases of the system development from the first phases of the early analysis, down to the actual implementation.

The Tropos methodology is mainly based on four phases [1]: *Early Requirements Analysis*, aimed at defining and understanding a problem by studying its existing organizational setting; *Late Requirements Analysis*, conceived to define and describe the system-to-be, in the context of its operational environment; *Architectural Design*, that deals with the definition of the system global architecture in terms of subsystems;



and the *Detailed Design* phase, aimed at specifying each architectural component in further detail, in terms of inputs, outputs, control and other relevant information.

During the phases of early and late requirements analysis Tropos employs two main types of diagrams: the actor diagram and the goal diagram. An actor diagram, describes the actors (depicted as circles), their goals (depicted as oval and bubble shapes) and the network of dependency relationships amongst the actors (two arrowed lines connected by a graphical symbol varying according to the dependum, i.e. goal, task, or resource). An example is given in Figure 2. A goal diagram represents the analysis of an actor's goals, conducted from the view point of the actor, by using three basic reasoning techniques: means-end analysis, contribution analysis and AND/OR decomposition. It is drawn as a balloon and contains graphs whose nodes are goals (ovals) and/or tasks (hexagonal shape) and whose arcs are the different relationships that can be identified among its nodes. An example is given in Figure 3.

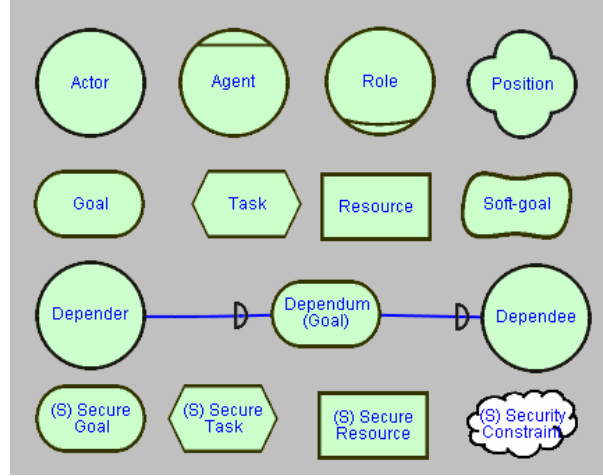
Although, the Tropos methodology was not conceived with security in mind, we have presented in previous work a set of security related concepts [13, 14, 15] (some resulted from security related extensions of existing concepts), to enable it to model security issues throughout the development of multiagent systems. This security oriented extension, which is known as secure Tropos, includes the following security related concepts.

A *security constraint* is defined as a restriction related to security issues, such as privacy, integrity and availability, which can influence the analysis and design of the information system under development by restricting some alternative design solutions, by conflicting with some of the requirements of the system, or by refining some of the system's objectives [12]. Graphically a security constraint is depicted as a cloud and it is positioned in the side of the actor who has to satisfy it (see for instance Figure 2).

Additionally to security constraints, Secure Tropos defines secure dependencies. A *secure dependency* introduces security constraint(s) that must be fulfilled for the dependency to be satisfied. Both the depender and the dependee must agree for the fulfilment of the security constraint in order for the secure dependency to be valid. That means the depender expects from the dependee to satisfy the security constraint(s) and also that the dependee will make an effort to deliver the dependum by satisfying the security constraint(s).

Secure Tropos uses the term secure entity to describe any goals and tasks related to the security of the system. A *secure goal* represents the strategic interests of an actor with respect to security. Secure goals are mainly introduced in order to achieve possible security constraints that are imposed to an actor or exist in the system. However, a secure goal does not particularly define how the security constraints can be achieved, since alternatives can be considered [12].

The precise definition of how the secure goal can be achieved is given by a secure task. A *secure task* is defined as a task that represents a particular way for satisfying a secure goal.



**Fig. 1.** Tropos and Secure Tropos notation

It is worth mentioning that the process in secure Tropos is one of analysing the security needs of the stakeholders and the system in terms of security constraints imposed to the stakeholders (early requirements) and the system (late requirements), identifying secure entities that guarantee the satisfaction of the security constraints, and assigning capabilities to the system (architectural design) to help towards the satisfaction of the secure entities. Security requirements are identified by employing the modelling activities of secure Tropos [12], such as security reference diagram construction, security constraints and secure entities modelling. In particular, the security constraints imposed to the system and the stakeholders, are identified and secure goals and entities that guarantee the satisfaction of the identified security constraints are imposed to the actors of the system.

The secure Tropos process allows for two types of validation. A model validation and design validation. The model validation involves the validation of the developed models (for example, the goal diagram or the actor diagram) with the aid of a set of validation rules [12]. The design validation aims to validate the developed solution against the security policy of the system. A key feature of the Secure Tropos that allows us to perform such a validation is the fact that the same secure concepts are used throughout the development stage. Moreover, the definition of these concepts allows us to provide a direct map between them, and therefore be able to validate whether the proposed security solution satisfies the security policy.

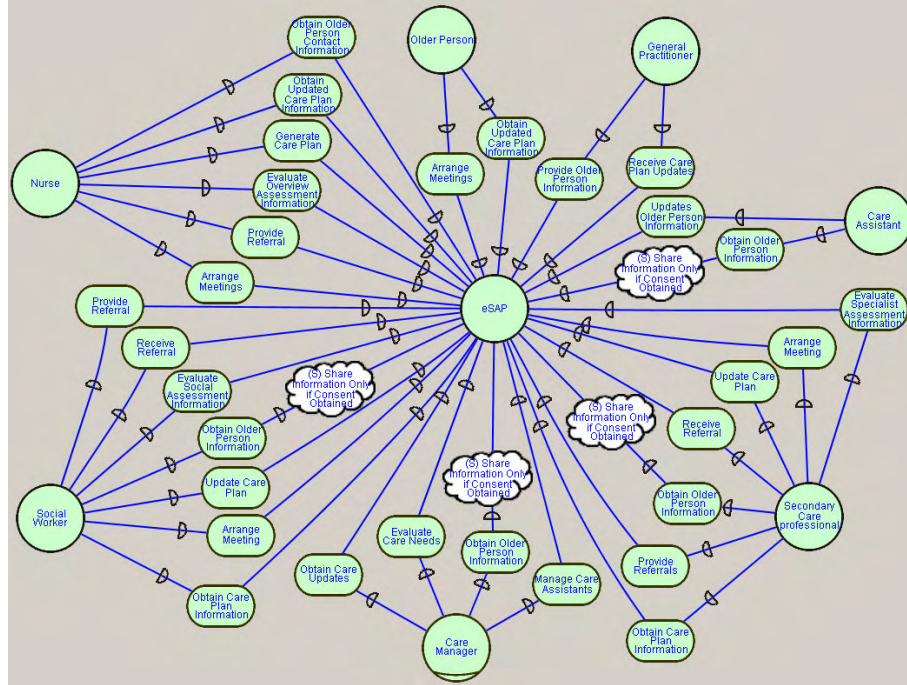
### 3 Enhancing secure Tropos

Secure Tropos provides solutions towards the first four problems identified in the Introduction. The approach described in this section enhances our work on secure Tropos in order to provide answers to problems e, f, and g. To achieve this aim, we integrate into the current secure Tropos process three sub-activities; (1) the system's architectural style selection according to its security requirements; (2) the

transformation of the security requirements to a design that satisfies these requirements; (3) and the attack testing of the multiagent system under development.

### 3.1 Illustrating the approach

To better illustrate the above sub-activities and their integration within the secure Tropos, we consider a complete description of the electronic single assessment process (eSAP) case study that we have introduced in previous work [16]. After analyzing the security requirements of the individual actors during the early requirements analysis [12], the electronic Single Assessment Process system (eSAP) is introduced and responsibility is delegated to it for some of the actors' goals. Since dependencies are delegated from the actors to the eSAP system, possible security constraints regarding those dependencies are also delegated as shown in Figure 2.

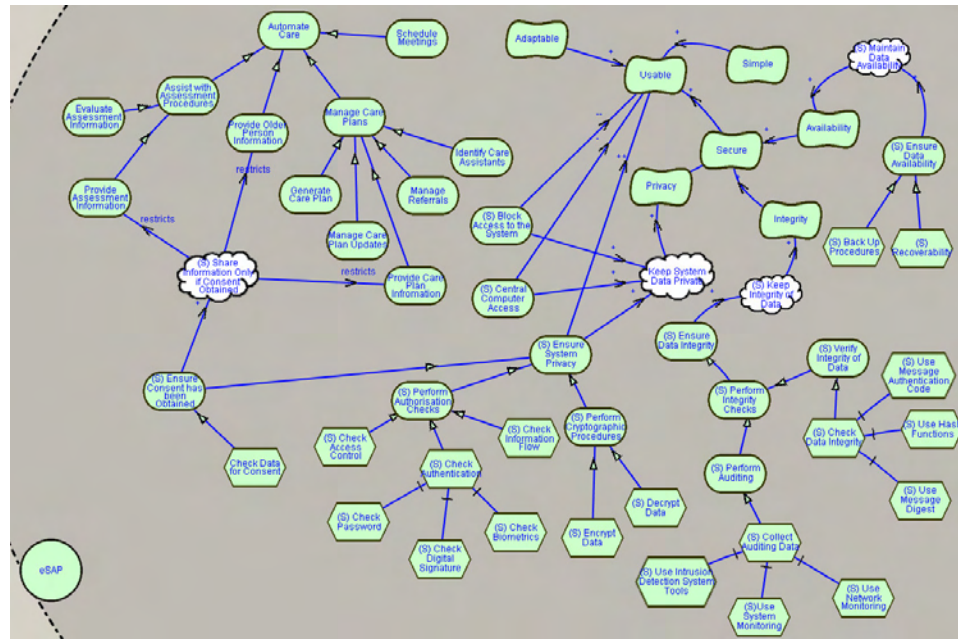


**Fig. 2.** Actor's diagram including the eSAP system

For example, before the introduction of the eSAP system, the Social Worker was depending on the Nurse to Obtain Older Person Information. However, this secure dependency involves the security constraint (restricting the Nurse) Share Information Only if Consent Obtained. With the introduction of the eSAP system, the Social Worker actor depends on the eSAP to Obtain Older Person Information, therefore the eSAP becomes responsible for satisfying the Share

Information Only if Consent Obtained security constraint that is delegated together with the secure dependency.

Then, the eSAP system is analysed and secure goals/tasks are imposed to the system to help towards the satisfaction of its security constraints as shown in Figure 3. For example, the **Keep System Data Private** security constraint can be fulfilled by blocking access to the system, by allowing access only from a central computer, or by ensuring system privacy. However, the first two contribute negatively to the usability of the system, i.e. the system will be secure but it will not be used. On the other hand, the **Ensure System Privacy** secure goal is considered the best solution since it provides security to the system and it doesn't affect (dramatically) its usability.



**Fig. 3.** Goal diagram for the eSAP system

When the requirements analysis is complete, the next stage is the architectural design. According to [2] one of the most important issues is to identify the architectural style that is more suitable for the eSAP system. For instance, we could try to identify whether a client/server or a mobile agent architectural style is more suitable for our system.

### 3.1.1 Selecting the system's architecture according to its security requirements

An important requirement of a security-oriented approach is to allow developers to explore different architectural designs or in other words, to allow developers to reason about alternative design solutions according to the security requirements of a multiagent system.

For this reason, this research has developed an analysis technique to enable developers to select among alternative architectural styles<sup>1</sup> using as criteria the non-functional requirements of the multiagent system under development. Although, this process can be used to select an architectural style according to any of the non-functional requirements of the system, as mentioned above, one of the most important issues in the eSAP is security. Consequently it is natural to decide about the architectural style of the system taking into account the issues involved, in other words the system's security requirements.

We use the measure of satisfiability [4] to compare different architectural styles with respect to non-functional requirements (such as security). *Satisfiability* represents the probability that a non-functional requirement will be satisfied. The contribution of each style to the satisfiability of non-functional requirements is expressed by a weight ranging between 0 and 1. For example, 0.1 means the probability that the architectural style will satisfy the non-functional requirement is very low (the style is not suitable for satisfying the requirement). On the other hand, a weight of 0.9 means the probability that the architectural style will satisfy the non-functional requirement is very high (the style is suitable for satisfying the requirement).

The weights of the contribution links are assigned after reviewing different studies, evaluations, and comparisons involving the architectural styles under evaluation. Figure 4 indicates the non-functional requirements of the eSAP system (as identified during the early and late requirements analysis –Figures 2,3) along with the satisfiability contributions from the client/server and the mobile agents architectural styles.

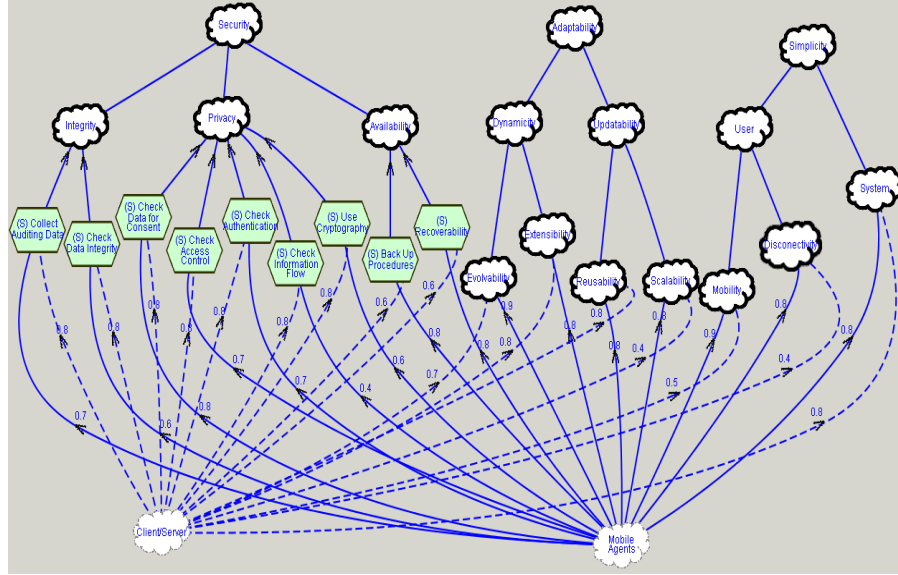
When the contribution weights for each architectural style to the different non-functional requirements of the system have been assigned, the best-suited architectural style is decided. This decision involves the categorization of the non-functional requirements according to the importance to the system and the identification of the architectural style that best satisfies the most important non-functional requirement using a propagation algorithm, such as the one presented by Giorgini et al. [4].

From earlier analysis (see Figure 3) it is concluded that security of the eSAP involves privacy, integrity and availability. Therefore, it is desirable to select a suitable architectural style for the system having these security requirements as criteria. According to the propagation algorithm, the satisfiability for the security is given by the following:

$$\begin{aligned}
 S_{(security)} &= \text{Min} (S_{(Privacy)}, S_{(integrity)}, S_{(availability)}) \\
 &\quad \text{where} \\
 S_{(Integrity)} &= \text{Min} (S_{(Collect auditing data)}, S_{(check Data Integrity)}) \\
 S_{(Privacy)} &= \text{Min} (S_{(check data for consent)}, S_{(Check access control)}, S_{(Check authentication)}, S_{(Check Information Flow)}, S_{(Check Cryptography)}) \\
 S_{(Availability)} &= \text{Min} (S_{(Back up procedures)}, S_{(Recoverability)})
 \end{aligned}$$

---

<sup>1</sup> To avoid confusion it must be noted that architectural styles differ from architectures in that “a style can be thought of as a set of constraints on an architecture” [Bas98].



**Fig. 4.** Selecting between architectural styles

In the presented example, for the client/server architecture we have the following values (as derived from Figure 4):  $S_{(Collect\ auditing\ data)} = 0.8$ ,  $S_{(check\ Data\ Integrity)} = 0.8$ ,  $S_{(check\ data\ for\ consent)} = 0.8$ ,  $S_{(Check\ access\ control)} = 0.8$ ,  $S_{(Check\ authentication)} = 0.8$ ,  $S_{(Check\ Information\ Flow)} = 0.8$ ,  $S_{(Check\ Cryptography)} = 0.8$ ,  $S_{(Back\ up\ procedures)} = 0.6$ ,  $S_{(Recoverability)} = 0.6$ .

Therefore,  $S_{(Integrity)} = 0.8$ ,  $S_{(Privacy)} = 0.8$ ,  $S_{(Availability)} = 0.6$ , and as a result  $S_{(Security)} = 0.6$ . that means the probability that the client/server architecture will satisfy the security requirements of the system is 60%.

Applying the same technique to calculate the probability that the mobile agent will satisfy the security requirements of the system we have the following:

$S_{(Collect\ auditing\ data)} = 0.7$ ,  $S_{(check\ Data\ Integrity)} = 0.6$ ,  $S_{(check\ data\ for\ consent)} = 0.8$ ,  $S_{(Check\ access\ control)} = 0.7$ ,  $S_{(Check\ authentication)} = 0.7$ ,  $S_{(Check\ Information\ Flow)} = 0.4$ ,  $S_{(Check\ Cryptography)} = 0.6$ ,  $S_{(Back\ up\ procedures)} = 0.8$ ,  $S_{(Recoverability)} = 0.8$ .

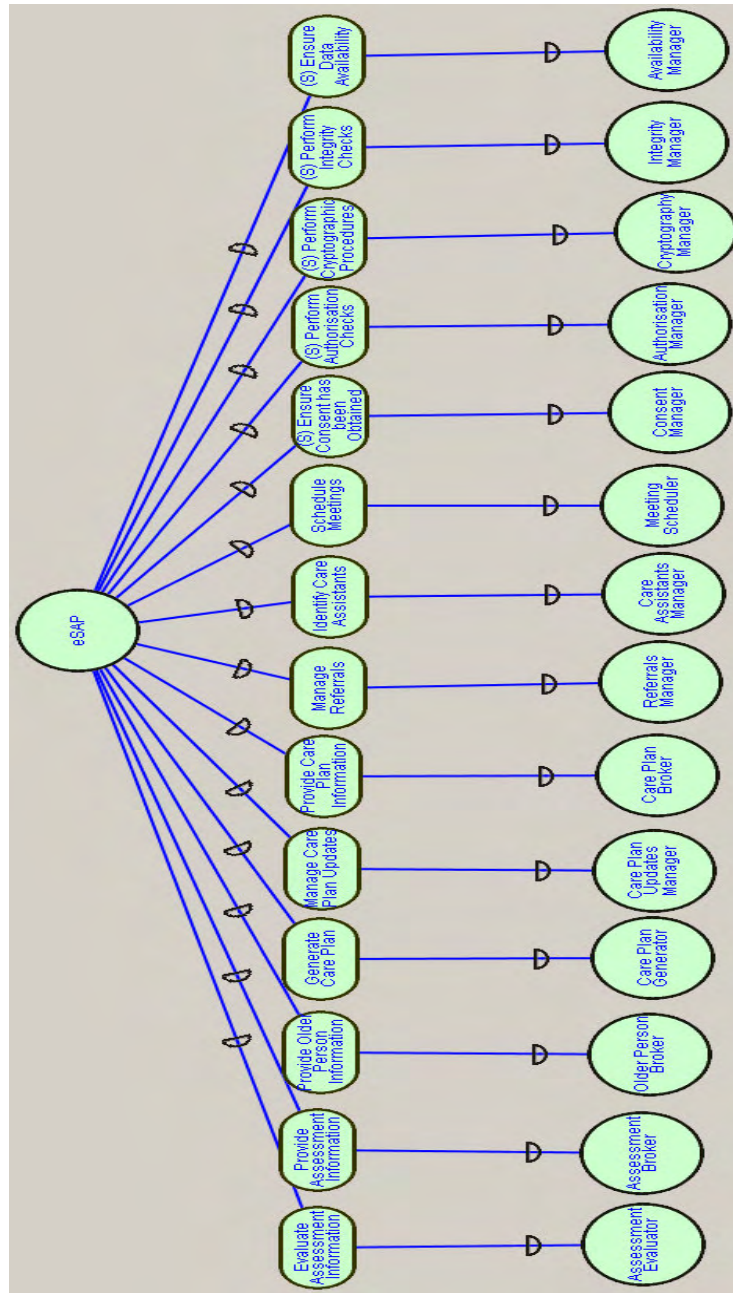
Therefore,  $S_{(Integrity)} = 0.6$ ,  $S_{(Privacy)} = 0.4$ ,  $S_{(Availability)} = 0.8$ , and as a result  $S_{(Security)} = 0.4$ , meaning that the probability that the mobile agents architectural style will satisfy the security requirements of the system is 40%.

As a result, for the given security requirements, the client/server style satisfies better the security of the eSAP system.

### 3.1.2 Transform the analysis to design

When the architectural style has been chosen, the next step of the architectural design stage aims to decompose the system in order to identify internal actors who will satisfy the system's (secure) goals. In the presented example, the eSAP actor is decomposed to internal actors and the responsibility for the fulfilment of the eSAP's goals is delegated to these actors as shown in Figure 5.



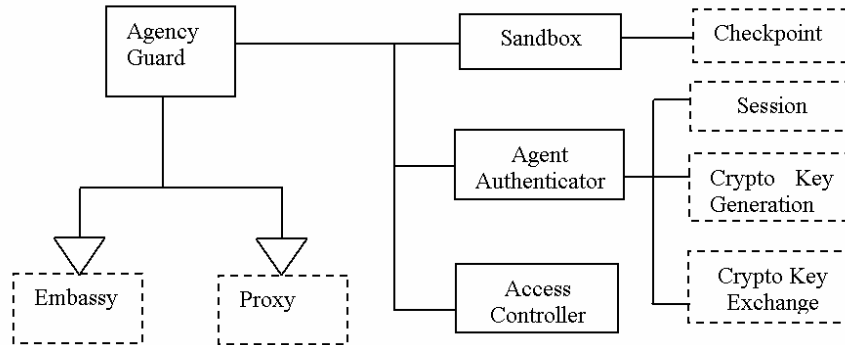


**Fig. 5.** eSAP actor partial decomposition

For instance, the Evaluate Assessment Information goal is delegated to the Assessment Evaluator, whereas the Provide Assessment Information goal is delegated to the Assessment Broker. In addition, the Older Person Broker and the Consent Manager actors have been introduced to the eSAP system to fulfill the responsibility of the eSAP system to satisfy the secure dependency Obtain Older Person Information together with the Share Information Only if Consent Obtained security constraint.

With respect to security the identification of some of the actors is a difficult task, especially for developers with minimum knowledge of security. To help developers, security patterns can be used. “A security pattern describes a particular recurring security problem that arises in specific contexts and presents a well-proven generic scheme for its solution” [19]. In other words, security patterns document proven solutions to security related problems in such a way that are applicable by non-security specialists. Therefore, the application of security patterns in the development of multiagent systems can greatly help to identify the required actors in a structured manner that does not put in danger the security of the system by providing a solution customised to the problem. The use of security patterns enables non-security specialists to identify patterns for transforming the security requirements of their system into design, and also be aware of the consequences that each of the applied security patterns introduce to their system. Additionally, because security patterns capture well-proven solutions, it is more likely that the application of security patterns will satisfy the security requirements of the system.

Therefore, we have developed a security pattern language [15] and we have integrated it within our security-oriented process. Figure 6 describes the relationship of the patterns of the language as well as their relationship with existing patterns. Each box indicates a pattern, where a solid-line box indicates a security pattern that belongs to the language developed by this research and a dashed-line box indicates a related existing pattern. White triangles depict generalisations/ specialisation and solid lines associations of type uses/ requires.



**Fig. 6.** The pattern language roadmap



The AGENCY GUARD is the starting point of applying the patterns of the language and it is a variant of the *Embassy* [7] and the *Proxy* [18] patterns. It uses the AGENT AUTHENTICATOR pattern to ensure the identity of the agents, the SANDBOX pattern in order to restrict the actions of agents, and the ACCESS CONTROLER pattern to restrict access to the system resources.

On the other hand, the SANDBOX pattern can implement the *Checkpoint* [20] pattern, and the AGENT AUTHENTICATOR pattern can use the *Session* [20] pattern to store credentials of the agent. Moreover, the AGENT AUTHENTICATOR employs the *Cryptographic Key Generation* [9] and the *Cryptographic Key Exchange* [9] patterns for further cryptographic actions.

To understand how the patterns of the language can be applied during the development of a system, consider the internal analysis of the eSAP system (see Figure 3). It was concluded that Information Flow, Authentication and Access Control checks must be performed in order for the eSAP system to satisfy the secure goal Ensure System Privacy. In the case of the Information Flow secure task, the eSAP should be able to control how information flows within the system, and between the system and other actors. For example, the system should be able to control who requires access to the system and, by considering the security policy, to grant or deny access to the system. With respect to the Authentication checks, the system should be able to authenticate any agents that send a request to access information of the system, and in the case of the Access Control, the system should be able to control access to its resources. To meet these goals, The AGENCY GUARD pattern can be used to grant/deny access to the system according to the security policy, the AGENT AUTHENTICATOR pattern can be used to provide authentication checks and the ACCESS CONTROLER pattern to perform access control checks as shown in Figure 7. The use of these patterns not only satisfies the fulfillment of the secure goals of the system but also guarantees the validity of the solution.

Moreover, developers know the consequences that each pattern introduces to the system. In the presented example, for instance, the application of the AGENCY GUARD means that only the AGENCY GUARD must be tested for correct enforcement of the agency's security policy (consequence), the application of the AGENT AUTHENTICATOR means that during implementation only the AGENT AUTHENTICATOR must be checked for assurance (consequence), whereas the application of the ACCESS CONTROLER means that different policies can be used for accessing different resources (consequence).

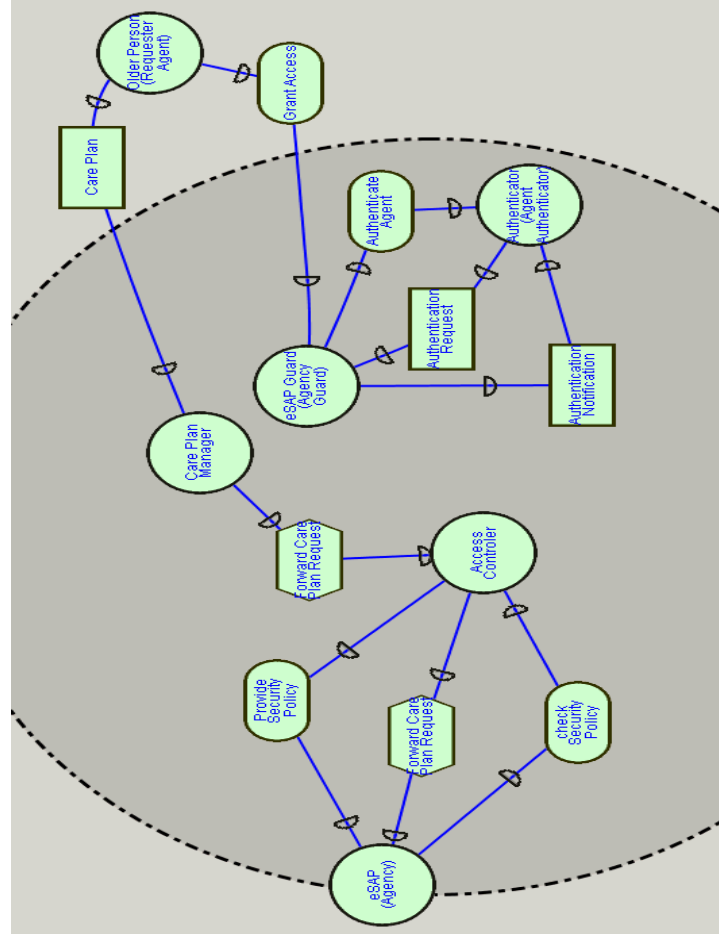


Fig. 7. Using the pattern language

### 3.1.3 Testing the developed solution

When the agents of the system have been identified along with their secure capabilities, it is very important for developers to test how their solution copes with potential attacks. For this reason, we have developed a process that is based on security attack scenarios.

A **Security Attack Scenario (SAS)** is defined as *an attack situation describing the agents of a multiagent system and their secure capabilities as well as possible attackers and their goals, and it identifies how the secure capabilities of the system prevent (if they prevent) the satisfaction of the attackers' goals.*

Security attack scenarios aim to test how the system copes with different kinds of security attacks. Therefore a scenario includes enough information about the system

and its environment to allow validation of the security requirements. A security attack scenario involves possible attacks to a multiagent system, a possible attacker, the resources that are attacked, and the agents of the system related to the attack. An attacker is depicted as an agent who aims to break the security of the system. The attacker intentions are modelled as goals and tasks and their analysis follows the same reasoning techniques that the Tropos methodology employs for goal and task analysis. Attacks are depicted as dash-lined links, called attack links, which contain an “attacks” tag, starting from one of the attacker’s goals and ending at the attacked resource.

The process is divided into three main stages [12]: creation of the scenario, validation of the scenario, and testing and redefinition of the system according to the scenario. Even though the presented process is introduced as a sequence of stages, in reality is highly iterative and stages can be interchanged according to the perception of the developers. During the creation of a scenario, Tropos goal diagram notation is used for analysing the intentions of an attacker in terms of goals and tasks, identify a set of attacks according to the attacker’s goals, and also identify the agents of the system that posses capabilities to prevent the identified attacks. Therefore, the agents of the system related to the identified attack(s) are modelled. The secure capabilities, of each agent, that help to prevent the identified attacks are identified and dashed-links (with the tag “help”) are provided indicating the capability and the attack they help to prevent.

When the scenarios have been created, they must be validated. Therefore, during the scenario validation process software inspections [8] are used. The inspection of the scenarios involves the identification of any possible violations of the Tropos syntax and of any possible inconsistencies between the scenarios and the models of the previous stages. Such an inspection involves the use of validation checklists. Such a check list has been proposed for instance in [12].

Although inspections have been proposed by this research for the validation of the security attack scenarios, other techniques could also be applied depending on the developers’ experience and the nature of the system. For instance, two well known validation techniques for requirements specification are walkthroughs and prototyping [8].

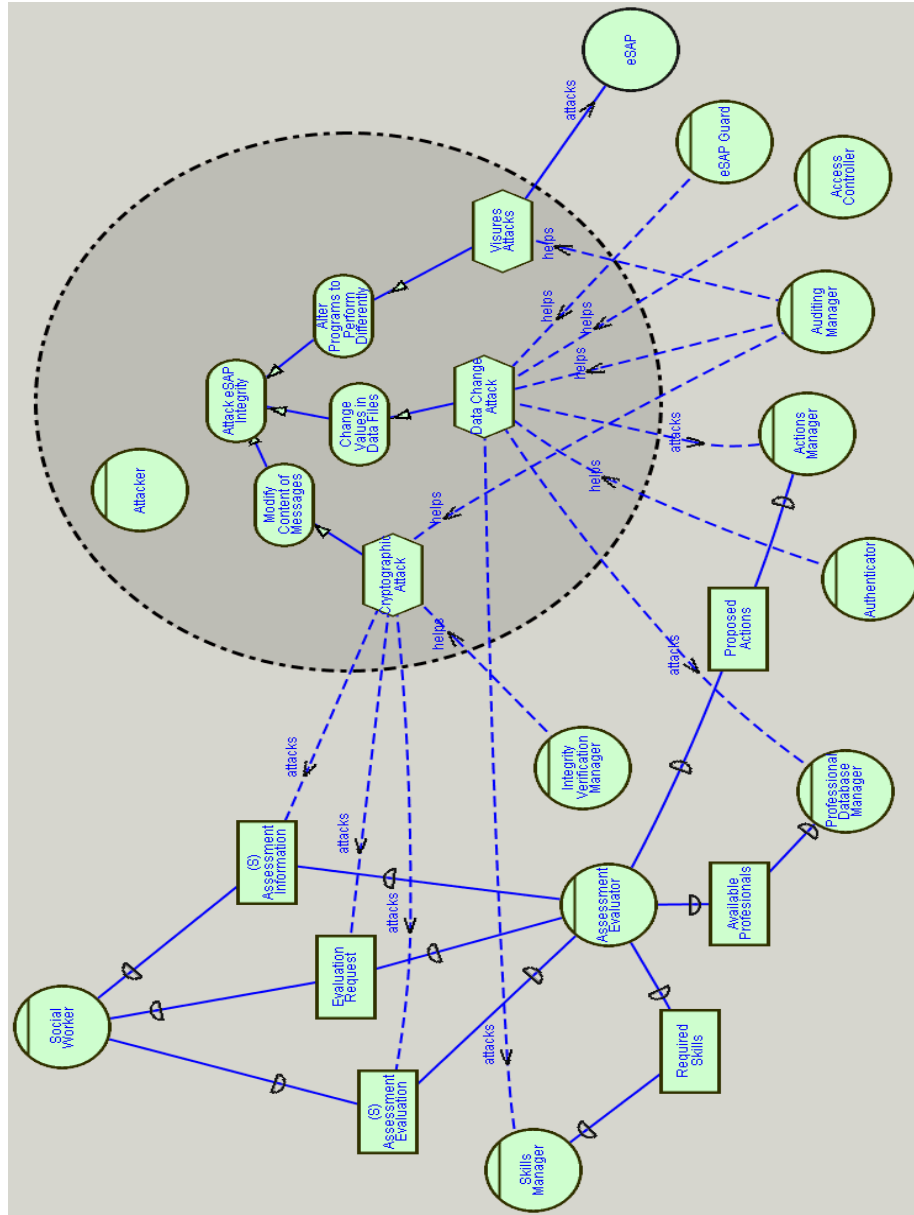
When the scenarios have been validated, the next step aims to identify test cases and test, using those test cases, the security of the system against any potential attacks. Each test case is derived from a possible attack depicted in the security attack scenarios. Each test case includes a **precondition** (the state of the system before the attack), a **system expected security reaction** (how the system reacts in the attack), a **discussion** that forms the basis for the decision regarding the test case, and a **test case result** that indicates the outputs of the test case.

The test cases are applied and a decision is formed to whether the system can prevent the identified attacks or not. The decision whether an attack can be prevented (and in what degree) or not lies on the developer. However as an indication of the decision it must be taken into consideration that at least one secure capability must help an attack, in order for the developer to decide the attack can be prevented. Attacks that cannot be prevented are notated as solid attack links, as opposed to attacks that the system can prevent and which are notated as dashed attack links.

For each attack that it has been decided it cannot be prevented, extra capabilities must be assigned to the system to help towards the prevention of that attack. In general, the assignment of extra secure capabilities is not a unique process and depends on the perception of the developer regarding the attack dangers. However, a good approach could be to analyse the capabilities of the attacker used to perform the attack and assign the system with capabilities that can revoke the attacker's capabilities. For instance, consider the scenario related to the eSAP in which the **Social Worker** wishes to obtain an **Assessment Evaluation**, and an **Attacker** wishes to attack the integrity of the eSAP system. As identified in the analysis of the security reference diagram [12], three main threats are involved in this kind of attack, cryptographic attacks, care plan changing and viruses. Therefore, the **Attacker's** main goal, **Attack eSAP Integrity**, can be decomposed to **Modify Content of Messages**, **Change Values in Data Files**, and **Alter Programs to Perform Differently** as shown in Figure 8.

The first sub-goal involves the **Attacker** trying to modify the content of any messages transmitted over the network. To fulfill this goal, the **Attacker** might try to employ cryptographic attacks to any resource transmitted between any external actors and the eSAP system. The second sub-goal indicates the **Attacker** trying to change the values in data files of the system. The fulfilment of this goal can be satisfied by means of changing the data of resources stored in the eSAP system. The third sub-goal indicates the attempt of the **Attacker** to alter a program so it performs differently. Mainly this can be achieved using viruses that can alter the behaviour of specific programs (agents) in order to enable the attacker to gain access to the system or to system's information.

Three main test cases are identified for this scenario [12], **cryptographic attacks**, **data changing attacks** and **viruses attacks**. For each of these attacks, a test case is constructed. In this paper we only present the password sniffing and the viruses test cases as shown in Table 1 and 2. By applying different test cases many useful results can be obtained about a system. For instance, for the presented case study the test case of the modification attack scenario identified that an agent should be introduced to the system to monitor the eSAP and take effective measurements against any possible viruses.



**Fig. 8.** An example of an attack scenario

<b>Test Case 1:</b> Password sniffing
<b>Precondition:</b> The <b>Social Worker</b> tries to obtain access to the <b>eSAP</b> system by providing their authorisation details. The <b>Attacker</b> tries to intercept the authorisation details.
<b>System expected security reaction:</b> prevent the <b>Attacker</b> from obtaining users' passwords
<b>Discussion:</b> the main target of the <b>Attacker</b> would be all the resource transmissions between the <b>Social Worker</b> and the <b>eSAP</b> system that contain any kind of authorisation details. Although authorisation details are encrypted, this is not enough since password sniffing takes place from a compromised computer belonging to the network. As a result, the <b>Attacker</b> is able to decrypt any message. A good technique to defend against password sniffing is to use one-time-passwords. A one-time-password is a password that is valid for only one use. After this use, it is not longer valid, and so even if the <b>Attacker</b> obtains such a password it is useless. However, the users must be able to gain access to the system more than once. This can be accomplished with what is commonly known as a password list. Each time a user tries to access the system they provide a different password from a list of passwords.
<b>Test Case Result:</b> Currently the system fails to adequately protect against password sniffing attacks. For the <b>eSAP</b> system to be able to react in a password sniffing attack, the external agents of the system (such as the <b>Nurse</b> , the <b>Social Worker</b> , the <b>Older Person</b> ) must be provided with capabilities to provide passwords from a password list.

**Table 1.** The password sniffing

<b>Test Case 2:</b> Viruses
<b>Precondition:</b> The <b>Attacker</b> tries to change the system behaviour by using some kind of virus.
<b>System expected security reaction:</b> The system should be able to prevent viruses.

**Discussion:** Viruses consist one of the most sophisticated threats to computer systems. It is quite common for attackers to send viruses to computer systems they want to attack in order to exploit vulnerabilities and change the behaviour of the system. Although many effective countermeasures have been developed for existing types of viruses, many new types of viruses are also developed frequently. An ideal measurement against viruses is prevention. In other words, viruses should not get into the system. However, this is almost impossible to achieve. Therefore, the best approach is to be able to detect, identify and remove a virus. Auditing helps towards the detection of the virus. However, apart from this the eSAP system is not protected against viruses.

**Test Case Results:** The eSAP system needs to be integrated with an anti-virus program to enable it to effectively detect, identify and remove any possible viruses. Such a program, which could be another internal agent of the eSAP system, should be able to monitor the system and take effective measurements against any possible viruses.

**Table 2.** The viruses test case

By applying these test cases many useful results can be obtained for the system under development. For instance, for the eSAP system, firstly it was identified that the system provides enough protection against some of these attacks. Secondly, for the attacks that the system did not provided adequately protection, extra agents and extra secure capabilities were identified and modifications took place in the eSAP system. For example, the external agents of the system were given the capability to provide passwords from a password list, and the Authenticator was given capabilities to successfully process such passwords. The lack of such capabilities was identified by the application of the password-sniffing test case of the interception attack scenario. Moreover, an agent, called Viruses Monitor, was introduced to the system to monitor the eSAP and take effective measurements against any possible viruses. The lack of such an agent was identified by the application of the viruses test case.

## 4 Conclusions

In this paper, we have presented an extension to the secure Tropos methodology to deal with the problems e, f, and g identified in the Introduction. To achieve this aim, we have integrated into the secure Tropos process three sub-activities; (1) the system's architectural style selection according to its security requirements; (2) the transformation of the security requirements to a design that satisfies these requirements; (3) and the attack testing of the multiagent system under development.

Each of those activities is important for the consideration of security for different reasons. The technique for selecting amongst different architectural styles and its integration within the Tropos methodology allows the explicit definition of the technique and allows developers to evaluate and select between different designs according to the system's security requirements. This, in turn, allows developers to analyse security requirements and base design solutions on this analysis. Moreover, the integration of the pattern language within the development stages of the secure Tropos allows novice security developers to reason about the consequences a particular design will have on their system, and therefore develop a design that will satisfy the security requirements of the system.

On the other hand, the introduction of security attack scenarios to test the system's response to potential attacks allows developers to test the developed security solution during the design stage. This, in turn, allows developers to re-consider particular system functions with respect to security until the system under development satisfies all the security requirements.

An important consideration of our approach is that it covers all the stages from the early requirements down to design, using the same concepts and notation. This is important since it allows the validation of the developed security solution by tracking the developed solution all the way back to the security requirements and the security policy. Moreover, the illustrated approach is implementation independent. Although many important issues may arise from the choice of a specific language, the consideration of such issues as part of the proposed process would restrict developers to specific solutions. We believe that a software engineering methodology should not restrict developers, and this claim is supported by various researchers within the agent oriented software engineering research area, as demonstrated by the development of a huge amount of agent oriented methodologies (for example see [1, 5, 6] that claim to be language independent and they do not consider implementation stages.

However, there are plans for future work. We first aim to integrate our approach with other related approaches, such as UMLsec, in order to extend the applicability domain. Secondly, we plan to develop tools that will allow to perform most of the, manual for the time being, checking and validations automatically. Thirdly, we plan to test the approach in a wider domain by using different case studies.

## References

1. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A., TROPOS: An Agent Oriented Software Development Methodology, in press, Journal of Autonomous Agents and Multi-Agent Systems. Kluwer Academic Publishers.
2. Castro, J., Kolp, M., Mylopoulos, J., Towards Requirements-Driven Information Systems Engineering: The Tropos project. In Information Systems (27), pp 365-389, Elsevier, Amsterdam - The Netherlands, 2002.
3. Devanbu, P., Stubblebine, S., Software Engineering for Security: a Roadmap. Proceedings of the conference of The future of Software engineering, 2000.



4. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R., Reasoning with Goal Models, in the Proceedings of the 21st International Conference on Conceptual Modeling (ER2002), Tampere, Finland, October 2002.
5. Huget, M-P., Nemo: An Agent-Oriented Software Engineering Methodology, In Proceedings of OOPSLA Workshop on Agent-Oriented Methodologies, John Debenham, Brian Henderson-Sellers, Nicholas Jennings and James Odell, Seattle, USA, November 2002.
6. Jennings N.R., Wooldridge M., Agent-Oriented Software Engineering, in the Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: Multi-Agent System Engineering (MAAMAW-99), Valencia, Spain, 1999
7. Kolp, M., Giorgini, P., Mylopoulos, J., A Goal-Based Organizational Perspective on Multi-Agent Architectures, in the Proceedings of the 8th International Workshop on Agent Theories, architectures, and languages (ATAL-2001), Seattle-USA, August 2001.
8. Kisters, G., Pagel, B.U., Winter, M., Coupling Use Cases and Class Models, Proceedings of the BCS-FACS/EROS workshop on "Making Object Oriented Methods More Rigorous", Imperial College, London-England, 1997
9. Lehtonen, S., Pärssinen, J., A Pattern Language for Cryptographic Key Management, Proceedings of the 7th European Conference on Pattern Languages of Programs (EuroPLoP), Irsee, Germany, June 2002.
10. Liu, L., Yu, E., Mylopoulos, J., Analyzing Security Requirements as Relationships Among Strategic Actors, 2nd Symposium on Requirements Engineering for Information Security (SREIS'02). Raleigh, North Carolina, 2002.
11. Mouratidis, H., Giorgini, P., Manson, G., Philp, I., A Natural Extension of Tropos Methodology for Modelling Security, in the Proceedings of the Agent Oriented Methodologies Workshop (OOPSLA 2002), Seattle-USA, November 2002.
12. Mouratidis, H., A Security Oriented Approach in the Development of Multiagent Systems: Applied to the Management of the Health and Social Care Needs of Older People in England. PhD thesis, , University of Sheffield, 2004.
13. Mouratidis, H., Giorgini, P., Manson G., Integrating Security and Systems Engineering: Towards the Modelling of Secure Information Systems, in the Proceedings of the 15th Conference on Advance Information Systems (CAiSE 2003), Velden-Austria, June 2003.
14. Mouratidis, H., Giorgini, P., Manson, G., Modelling secure multiagent systems, in the proceedings of the Second International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2003, July 14-18, 2003, Melbourne, Victoria, Australia, pages 859-866, ISBN 1-58113-683-8, ACM 2003.
15. Mouratidis, H., Giorgini, P., Schumacher, M., Manson, M., Security Patterns for Agent Systems, Proceedings of the Eight European Conference on Pattern Languages of Programs (EuroPLoP), Irsee, Germany, June 2003.

16. Mouratidis, H., Philp, I., Manson, G., A Novel Agent-Based System to Support the Single Assessment Process for Older People, in the Journal of Health Informatics (9) 3, pp. 149-163, September 2003.
17. National Research Council, Computer At Risk: Safe Computing in the Information Age, National Academy Press, Washington, D.C., USA, 1991
18. Norman L. Kert, J., Vlissides, M., Coplien, J-O, "Pattern Languages of Program Design 2", Addison Wesley Publishing, 1996
19. Schumacher, M., Roedig, R., Security Engineering with Patterns, in the proceedings of the 8th Conference on Pattern Languages for Programs (PLoP 2001), Illinois-USA, September 2001.
20. Yoder, J. Barcalow, J., Architectural Patterns for Enabling Application Security, Proceedings of the 4th Conference on Pattern Languages of Programs (PLoP 1997), Monticello, Illinois, USA, September 1997
21. Yu, E., Cysneiros, L., Designing for Privacy and Other Competing Requirements, 2nd Symposium on Requirements Engineering for Information Security (SREIS' 02), Raleigh, North Carolina, 2002.

# Using Multi-Agent Systems to Specify Safe and Secure Services for Virtual Organisations

Stefan Poslad

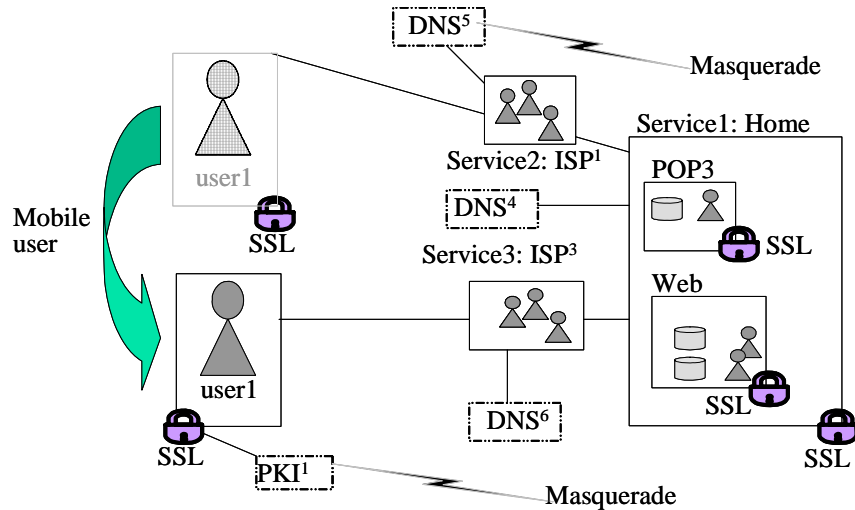
Department of Electronic Engineering, Queen Mary, University of London,  
Mile End Road, London E1 4NS  
stefan.poslad@elec.qmul.ac.uk

**Abstract.** Organisations increasingly interoperate with others to participate in Virtual Organisations or VO. This leads to models that need to deal with open service interaction with multiple levels of openness. Other VO security and safety challenges include decentralised operation and management, interoperability, balancing privacy versus accountability, managing error events with multiple semantics and managing anomaly events versus normal variations. The interrelationships, interactions and behaviours of agents in MAS and multi-MAS (MMAS) are analogous to those in VO. MAS provide a useful method of modelling VO, to manage their and to make them secure and safer.

## 1 Introduction

Distributed system security is primarily concerned with using cryptographic protocols to protect organisational assets against external threats, treating the organisation as a trusted system internally without security checks – a fortress model of security. Security safeguards protect organisational assets such as the network infrastructure, processing, services and data against threats such as information disclosure, corruption, masquerade, unauthorised access, denial of services and repudiation. In addition, more finely grained access control often used internally according to the organisational role, level of access e.g., Clark-Wilson [1] and Bell-LaPadula [2] and the type and value of the asset – a compartmentalised fortress model. These in turn use sets of authorisation rules or policies to define what type of access a user has to a type of asset.

Security requires additional security management protocols often at an inter-organisational level. For example, external Trusted Third-Parties (TTP) are used by organizations to create the authorisation and authentication credentials they distribute, configure and revoke. TTP are trusted not to make copies of credential for others, not to provide back door access and to produce easily forgeable credentials.



**Fig. 1.** A scenario to illustrate the use of a fortress security model to give secure access to POP3 and Web services to a mobile worker on his home network who must then switch to use another ISP to access them. However, DNS and PKI services are threatened by a masquerade.

The human users of ICT or Information, Communication Technology, including their organisational inter-relationships and human-ICT relationships, act as additional vital organisational assets that need to be protected. Obviously, humans such as hackers can act malevolently, e.g., stealing unauthorised information, and threaten security. However, people can also frequently inadvertently weaken security. For example, people can non-optimally configure system security, e.g., switching it off or to a minimum level of security because it is too complicated for them to configure to a higher level. When a natural or man-made environment disaster occurs, local mobile wireless communications can get overloaded and cause a Denial of Service or DoS as each participant or observer reports the event to others, perhaps preventing local emergency and help messages getting through. Humans can also weaken security systems because they often add an additional level of indirection that can be compromised. For example, humans prefer to use domain names rather than numerical IP addresses in order to address Internet assets such as Web servers. A mapping service such as the Domain Name Service or DNS must be used to map human readable names into numerical IDs and vice versa. DNS can be attacked by altering DNS messages and DNS data so that messages get redirected to a different destination server than the one intended by the sender. Humans thus often represent the biggest security challenge of the whole system [3].

Whereas security commonly refers to the ability to protect services against malicious threats using cryptographic protocols, safety refers to managing the system when failures occur. Failures may occur for several reasons because of system design errors, because of task errors that occur when they interact with another system and because of environment conditions or operators that cause a system to operate outside its normal, safe, operating conditions. Another definition of safety is that it is the probability that a system does not fail in a manner that causes catastrophic damage

[4]. Safety management is akin to failure management and has four main strategies. Fail-operational systems continue to operate when they fail, sometimes unsafely. Fail-safe systems become safe when they cannot operate. Fail-secure systems maintain maximum security when they can not operate. Fault-tolerant systems continue to operate correctly when parts operate incorrectly because parts are replicated and these can be accessed, ideally transparently, instead.

Security and safety management is complex because it deals with heterogeneous distributed organisational assets. It must not only deal with security and safety of the physical resources and services but it must also deal with the human stake-holders that interact with systems and it must deal with dynamic, virtual, inter-organisational boundaries.

### **1.1 VO Security and Safety Management**

An organisation, e.g., a business firm, is an arrangement of relationships between individuals that produces a system endowed with qualities not present at the level of individuals. It ensures a relatively high degree of interdependence and reliability, providing there are procedures in place to replace individuals and their interactions within the organisation, thus providing the organisation with the ability to persist in the face of disruptions. The relationships between individuals within the organisation are determined by the interactions and their interactions are in turn constrained by the organisation. These interrelationships exist only within organisations, but reciprocally organisations require these relationships in order to exist. Behaviours or actions are normally performed internally within organisations such as firms on economic grounds, i.e., only if they cannot be performed more cheaply in the market or by another organisation. It was partly for this finding in 1937 [5] that Ronald Coase was awarded the Nobel Prize for economics over 50 years later.

There are three generic types of Virtual Organisation or VO [6]. Firstly, there are organizations that extend some of their organisational activities externally, thus forming virtual alliances to achieve (shared) organisational objectives. E-commerce organisations that participate in supply-chains and coalitions of military and humanitarian forces are examples of this type. Secondly, a virtual organization can be related to a perceptual organization that is "abstract, unseeing and existing within the minds of those who form a particular organization". The third type of virtual organization is established when corporations are distributed and intensively use ICT to support this such as the use of Virtual Private Networks or VPN, e.g., see Figure 1. Primarily, the focus of this article is on modelling security for the first type of VO.

### **1.2 Paper Outline**

The remainder of this article is organised as follows. The next section, 2, describes the requirements for VO security and safety. Section 3 discusses the design and implementation using MAS as a design model and technology for implementing VO. MAS are also considered as primary entities to manage VO security and safety. Section 4 presents conclusions.

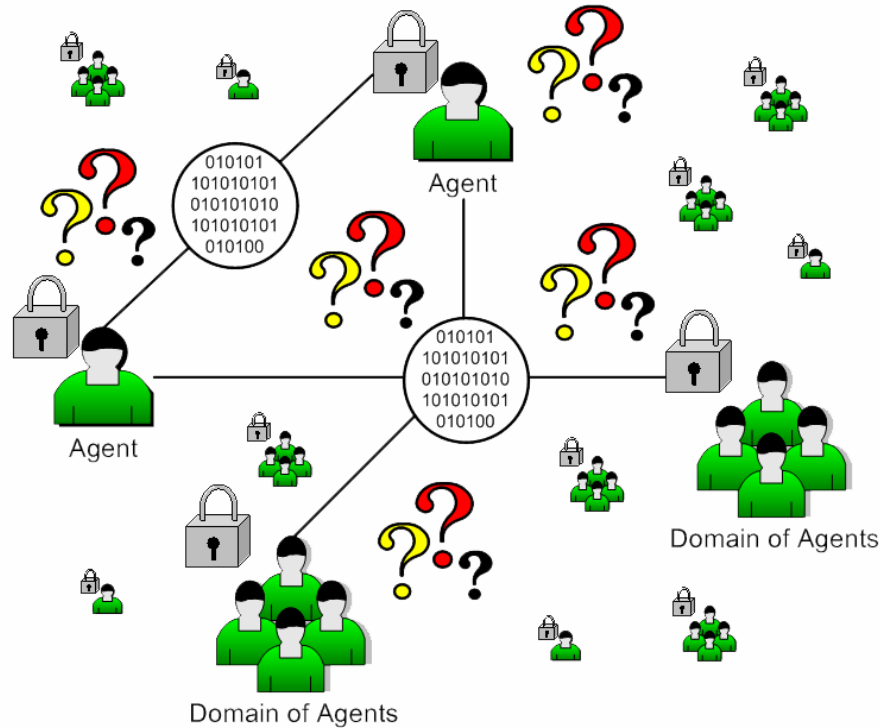
## 2 VO Security and Safety Challenges and Requirements

Challenges	Characteristics
Open Access vs. Regulated Access	Pure open service access, any time, any place, anyhow, anybody access is unregulated & difficult to secure. May need to span across infrastructures, owned by others that can only be indirectly managed, e.g., to delegate authority,
Decentralised Operation and Management of services	No single authority; multi autonomous stakeholders and autonomous groups. Emergent behaviour can arise and violate existing security One system can compromise the security and safety of another.
Security Interoperability	Services and users may reside in multiple administrative domains. Multiple security mechanisms in use across domains.
Balancing privacy vs. accountability	Masquerades are easier Degrees of privacy from anonymity to public identification
Managing error events with multiple semantics	Higher-level policies may not be directly implementable at a lower-level Low level faults often not propagated to, are not understandable, at a higher level
Managing Anomalies versus Normal variations	Deterministic vs. semi-deterministic vs. random Normal perturbations vs. Anomalies. Intended attacks vs. unintended failures.

**Table 1. Virtual Organisation Security & Safety Challenges and Requirements.**

The security and safety challenges for VO are summarised in Table 1. Each type of challenge is discussed in more detail below.

## 2.1 Open Access vs. Regulated Access



**Fig. 2.** Heterogeneous service access and interoperability in multiple domains

Virtual organisations tend to use open service environments in order to interoperate. A pure open service environment supports ubiquitous access: any time, any place, anyhow by anybody. This openness has several dimensions. It refers to service interfaces being public available. Services are accessible over a public network. It enables heterogeneous implementations of the services to be built that inherently interoperate. Heterogeneous interfaces can be aligned to each other, possibly with some information loss. Securing heterogeneous services is dealt with further in section 2.3. Services are extensible so that new services can be synthesised as composites of existing services and that old service interfaces can be extended. Users can dynamically bind to and unbind from service invocations, possibly interacting via third-party mediators. Services are scalable, supporting a range of concurrent users.

The operational management of open service environments requires that it both supports any desired dynamic configurations and service operations yet restricts any undesired configurations and operations. In practice, open doesn't necessarily mean that access costs nothing, that anyone can have access; pure openness is often replaced by regulated openness. VO tend to be secured using cryptographic protocols, see Fig. 2. It means that anyone can apply and register providing they meet the access policy constraints. For example, there is an age limit to purchase alcohol and lottery

tickets. Hence an important technique for dynamic system management is policy-based management coupled to cryptographic security protocols. Policy-based management defines the management rules or policies that are activated when service events occur such as unauthorised resource access.

## **2.2 Decentralised Operation and Management of Services**

Within a virtual organisation, the individual organisations act with a degree of autonomy. They interact more on a peer to peer level, without a central manager, in part, because they are autonomous and because of openness. They may draw up and adhere to contracts and agreements that formalize commitments between them and that have a legal force and legal consequences. Both the users and provider entities within a transaction can simply switch to another similar entity if a problem occurs, providing that this is not prohibitively expensive to do so when breaking their contractual agreement, e.g., they may judge the financial penalty is small or that it requires too much time and financial resources for others to bother with.

An important issue is how service users and providers select each other. The dominant model in distributed computing is that service providers advertise their capabilities in service directories and that the choice is made on the best match service capabilities by the user, but this often not user centric, i.e., it may not be linked to a rating of the actual services delivered. Before choosing to delegate tasks to others to act on their behalf, some sort of local evaluation of others, perhaps using recommendation, ratings, reputation and referrals from others or past experiences of oneself, can be carried out. One party may also decide to trust another party, that is, the trustor believes that the trustee is benevolent and competent and voluntarily delegates tasks to the trustee with no real commitment from the trustee [7]. Within an open service environment and in particular on ones based upon trust, it is far easier for masquerades to occur. For example, convicted criminal hacker Kevin Mitnick testified before U.S. congress that he often gained security credentials by simply masquerading as someone else. Ironically, another hacker gained a reduced criminal sentence by having encrypted the evidence of his actions.

There is also the issue of how to audit, classify and manage interdependent interactions when they are orchestrated across multiple organisations and domains. It is common in cases where some ambiguity about the cause of events exists for the organisations involved to compete to get others to fix a problem in order to reduce their own costs. It may also be unclear how to identify the cause and to handle a fault when it arises because of interference between two autonomous systems, so called emergent behaviour. It may also not be possible to determine the exact or even the probable cause of events. [8] has outlined the following security challenges for managing emergent properties, in ad-hoc networks. Emergent properties represent new beneficial security features of ad-hoc networks, e.g., to help establish or revoke trust relation. Some emergent properties are undesirable and may lead to security violations e.g., emergent properties may bypass traditional intrusion-detection techniques and mask that a certain network node has been captured by an adversary. Third, inadequate assessment of emergent properties such as false detection may also lead to



security and robustness violations. For example, the false detection of node capture may lead to network partitioning and denial of service in an ad-hoc network by the unnecessary revocation of node membership. Fourth, the understanding of the characteristics of emergent properties helps determine scalability and resilience.

### **2.3 Security Interoperability**

ICT environments are naturally open; providers and users often discover and evaluate each other's capabilities and preferences dynamically, enabling users to invoke providers' services on-the-fly. Interoperability in a heterogeneous open service environment can be greatly aided by using public service specifications and by specifications being in a computation form that is able to be analysed and to be aligned to related heterogeneous specifications. For example, different parties may use different kinds of authentication certified by different TTF – these will require the use of some sort of authentication certificate gateway to allow each other to interoperate. Interoperability is aided when they can share semantic type metadata to support a common understanding between them.

### **2.4 Balancing privacy versus accountability**

Privacy can be regarded as both a basic human right to be left alone and as a user freedom to not be observed by others. In practice, a portion of privacy is often traded in order to have greater convenience in order to interact with others such as other human users and service providers, e.g., to avoid entering the same personal details for repeat transactions. [9] has identified intermediate levels of identity between true identity and true anonymity. He distinguished four levels of “nymity:” anonymity, non-reversible pseudonymity (not tied to a true identity), reversible pseudonymity (tied to a true identity), and identity. Maximum disclosure and invasion of user privacy occurs when a user's personal details are revealed and linked to additional contexts such as the current location or to payment credentials.

There are also concerns with identity management and accountability within open systems. It may be difficult to assign an accountable identity or address to the perpetrator of a malicious action within an open service environment, because the identity and address can be easily masked or spoofed. This represents a significant weakness in authentication services as unauditible IDs and addresses may be bound to authentication tokens thereby restricting the usefulness of the tokens. Therefore, this brings forth a fresh type of problem whereby malicious intent can no longer be bound to an accountable identity or address and where varying contextual meanings can represent hidden malevolent agendas.

### **2.5 Managing events and errors with multiple semantics**

VO potentially have a greater variety of failures because of the larger number and possible dynamic compositions of service components. They also use a more complex

operating environment, leading to transient and intermittent failures and because status messages and error messages may trigger failures that are not understood by the application processes. A further complication in an open distributed system is that Byzantine type faults may be more likely. Byzantine faults occur when processes continue to operate issuing incorrect results or possibly work together with other faulty processes to give the impressions that they are working normally.

Either errors can be trapped and handled internally within the application or they can be handled using some sort of world model that can be shared across applications [10]. The application-level approach often involves instrumenting the application code to generate events and to catch the fatal ones. This has the important disadvantage that the event can only have a context within an application and can't easily have a more global context across applications. A second disadvantage is the semantics of the error is designed and annotated at the level of the software infrastructure in which the application execution. However, users may lack an understanding of the low-level infrastructure semantics.

Conventionally, failures can be handled in several ways. A very common technique is to handle them locally in the application using static exception handlers. Failures can be masked, by switching to a replicated copy if fault-tolerance is supported or by catching it and simply propagating infrastructure errors up to a user process. However, application level processes and users often cannot understand the meaning of errors because they are at a lower level of abstraction or because they use a particular perspective of understanding that users are unfamiliar with. For example, one well known personal firewall for computers expects users to be able to understand which types of network protocol ports an application should be allowed access to in order to get through the firewall. Users may not understand why a service stops behaving as expected if no information is given and so may not be clear which type of remedial action, such as a restart, should be used.

## **2.6 Managing normal variability versus anomalies**

Simple open service environments can be designed to support Event-Condition- Action (ECA) type interaction. Conditions constrain both how events cause actions and the order in which actions normally flow, e.g., a metadata directory is queried first to identify which data resources hold certain types of data, thus avoiding wasting time querying data resources that do not hold the data. It is important to model both normal and invariant pattern of events. Specific normal variant event and action patterns are specified that need to be handled so that they do not disrupt the system, e.g., a resource manager may receive a flood of repeat event requests because a user receives no acknowledgement that his request has been received.

In more complex distributed systems, users, providers and mediators have more autonomy; sub-systems can refuse to carry out requests even although they have the capability to do so. Systems can have the flexibility for multiple and concurrent service providers and users to dynamically bind and unbind to each other and for actions to be orchestrated into work-flows at run-time rather than at design-time. More complex behaviours can also emerge from the interplay between simple ones and a com-

plex environment. Anomalous events or actions, not defined in the standard or expected set of planned normal behaviours can arise.

### 3 MAS Designs to Support the Safety and Security Management of VO

Challenges	MAS Solution
Open Access vs. Regulated Access	Agent can <i>manage policies</i> dynamically to regulate access. Agents can <i>reason</i> about which are the important parts of interactions to secure and which to leave public. Agents can <i>share tasks and goals</i> with other agents across organisational boundaries and 'delegate' these to others to act on their behalf.
Decentralised Operation and Management of services	Different types of agents act as multi <i>autonomous</i> stakeholders and autonomous groups. Agents can share information about security threats that anyone detects, acting as a <i>neighbourhood watch</i> . Simple interactions between agents can lead to the emergence of global behaviour – <i>swarm intelligence</i> .
Security Interoperability	Agents can be used support <i>semantic mediation</i> between different security protocols
Balancing. Privacy vs. accountability	Agent mediators can be used as TTP and can use <i>policy-based management</i> between customers and providers so that identity and details of customers are <i>revealed on a need to know basis</i> . TTP agent mediators that reveal private information unnecessarily can suffer a <i>loss of reputation</i> , gain a low merit mark for their quality of service. Agents can <i>share knowledge of events and expertise</i> to allow services to be managed across organisational boundaries.
Managing events: mismatch of high-level management vs. lower-level operational, application, ones	Agents can be used to support <i>semantic mediation</i> between different security views, protocols, concepts and policies. Agents can <i>model events and reason about them externally to the application processes in which they occur</i>
Managing Anomalies versus Normal variations	Normal goal-directed behaviours can be modelled using explicit plans by agents. Agents can also <i>re-plan</i> when events are detected that cause disruptions to the plan. Agents can <i>explicitly model environments' expected events and infer unusual events</i> .

**Table 2.** MAS use to support VO safety and security

Multi Agent Systems (MAS) are a distributed artificial intelligence problem model that can be solved and reified using a range of technological solutions from procedural and object-oriented to declarative and to a wide spectrum of logic programming. MAS are organisations of individual agents. The properties of agents are that they are goal directed and can behave with a degree of autonomy. They can react instinctively to interactions and proactively initiate interactions, or they can deliberate

about their interactions. They can interact cooperatively, to share common goals, tasks, information and tasks or they can act in a self-interested competitive manner.

Software type of agents, not only interact with other agents but they are also situated in a non-agent environment, an ICT infrastructure. They can sense events, process them and affect the environment. The link between the sensed events, the behaviours or actions that agents undertake and any subsequent effects in the environment depends on the type of agent such as being deliberative, reactive, goal-directed, utility-based or a combination of these. The interrelationships, interactions and behaviours of agents in MAS and multi-MAS (MMAS) are analogous in many ways with those of members of social and economic organisations and virtual organisations.

The first major design issue that needs to be considered in using a MAS model of VO security is whether or not security should be managed by agents within an organisation or delegated or trusted to others external to an organisation such as to the ICT infrastructure environment. The degree of security control under the management of the agent needs to be considered. This can range as follows:

- Agents (application) have no control of security: they delegate security management to the infrastructure or to third parties;

- Agents manage security at a coarse level: they can monitor security changes directly or indirectly; they can switch on and off their own security; they can switch between various levels of security;

- Agents manage security at a fine level: They can deliberate and control security using semantic and social collaborative models.

There are a number of advantages to using MAS to manage security that are expounded in the next sub-sections and summarised in Table 2. These are related to the VO security challenges and requirements given in Table 1.

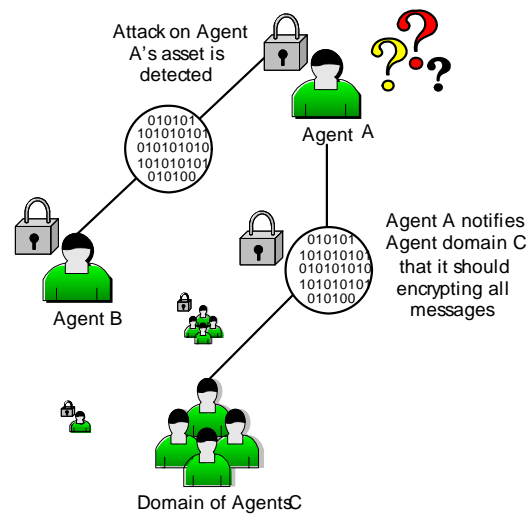
### **3.1 MAS support for Open Access vs. Regulated Access**

The organisation or application domain can specify policies to say who has what access to which organisational assets. Agents can reason about the instances of policies and policy flows that are triggered when agents access. It is not sufficient for agents to just reason about triggered policies, agents should also reason about when to trigger security protocols in situated actions of the work-flow. To do this, agents will need to track work-flows for tasks and assess the policies for when security should be used, and at what level. Policies may need to change dynamically, e.g., if threat levels are perceived to have changed. When individual organisations interact within a VO, policy mediation may be needed to check the compatibility and priority of policies when multiple ones apply and to decide which ones should be applied.

In the past, no single framework existed to express domain specific Ontologies, rules and reasoning about the ontology concepts and rules. With the advent of the W3C Web Ontology Language, OWL, there is a single framework to express rich concept structures and relationships, rules in the form of constraints on concept classes and properties and a type of description logic to reason about concepts.

### 3.2 MAS support for Decentralised Operation and Management of services

Safety and security can be increased using MAS models because autonomous agents can build different independent viewpoints or models of the nature of these failures. Agents can be designed to use multiple heterogeneous plans to achieve goals in the face of limited environmental failures and malicious attacks. In addition, they can socialize, acting together as a neighbourhood watch and use triangulation, i.e. using a combination of multiple view-points, to understand a given problem or situation improving the limitation of relying on any single viewpoint of the failure, see Fig. 3.



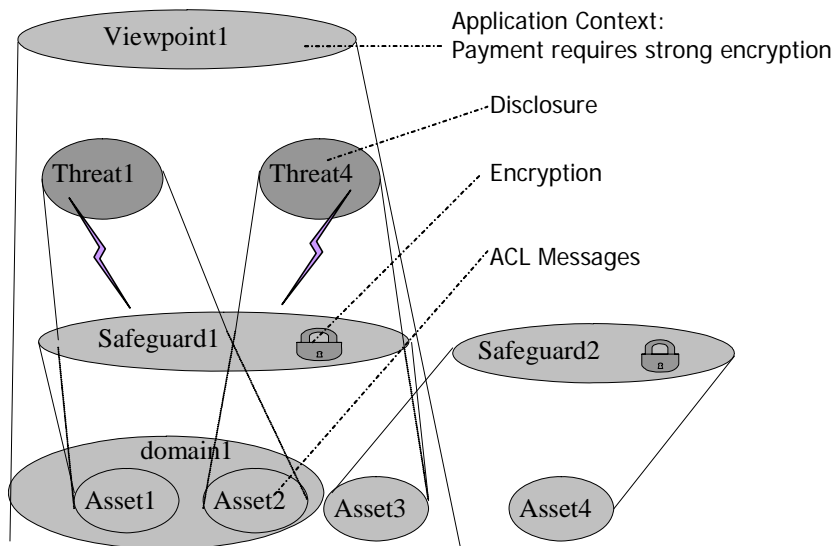
**Fig. 3.** One agent can socialise with others about an intrusion it detects

Computer systems will need to manage themselves according to high-level objectives specified by humans, otherwise the increasingly complex, heterogeneous distributed systems, such as ubiquitous computing systems will become impossible to administer [11]. IBM first introduced this vision of self-managing system in 2001 when it launched the autonomic computing initiative. Autonomic Systems support four basic properties: self-configuration, self-healing, self-optimization and self-protection.

VO often rely on norms or group prescriptions to constrain its members to do or not to do actions. They rely on the obedience of trustees to adhere to the norm. Swarm behaviour is influenced by instinct, by local interactions, past experiences. Herd behaviour is influenced by market-leaders. If there are no norms, chaotic behaviour can arise. There is often a local convergence to norms but this may cause global polarisation within a VO because local norms are orthogonal [12].

### 3.3 MAS support for Security Interoperability

Open service environments for VO such as those based upon public Web service specifications from the W3C and public MAS specifications from FIPA, the Foundation for Intelligent Physical Agents [13] provide the means for different actors within a VO to interoperate, more easily, and at a lower cost. New entrants to the market can more readily publish and offer services, users can more easily interact to select services, and mediators can more easily provide various value-added channels between providers and users. Not only should the service interactions use public specifications but the service management should also use public security protocols in order to operate. Earlier applications of MAS security tended to use proprietary security mechanisms, for example using non-standard certificates for authentication [14], [15] this tends to lead to closed systems that cannot interoperate with others as part of a VO.



**Fig. 4.** The V-SAT model, Viewpoints of Safeguards that protect Assets against Threats

Although, security protocols based upon public specifications are readily available, there are great many to choose from and the same security protocol can be configured differently to adhere to different organisational security models. A semantic mediation approach based upon a core model, the V-SAT model, Viewpoints of Safeguards that protect Assets against Threats, see Fig. 4, has been proposed in [16] to support security interoperability using multiple security protocols. This has been represented first in DAML+OIL and then in OWL.

```
<!-- Policy Instances -->
<secprofile:AgentCondition rdf:ID="PlatformCredentialPolicy">
  <policy rdf:range="xsd:String">
    (exists (?c (Security_Ontolo:SignatureMethod ?c)))
```

```

        (or (Security_Ontolo:Algorithm ?c http://www.w3.org/2000/09/xmldsig#rsa-sha1)
            (Security_Ontolo:Algorithm ?c http://www.w3.org/2000/09/xmldsig#dsa-sha1) )
    </policy>
</secprofile:AgentCondition>
<secprofile:AgentCondition rdf:ID="PlatformProtocolPolicy">
    <policy rdf:range="xsd:String">
        (exists (?p (Security_Ontolo:Protocol ?p)) (and (Security_Ontolo:FIPASecurityProtocol ?p
urn:fipa:security:SecurityObject1) (Security_Ontolo:FIPASecurityProtocol ?p urn:fipa:security:SecurityObject2) )
    </policy>
</secprofile:AgentCondition>
...

```

**Fig. 5.** An example policy constraining the algorithm used for digital signatures

Because (external) security configurations can be described at an abstract level, different mechanism choices can be specified in V-SAT model terms in instances of the viewpoint called application profiles. For example, support for encryption using DES or AES and message hashing using MD5 or SHA-1 can be specified. Constraint or policy based reasoning can be used to see if a common security configuration can be agreed between different communicating parties, see Fig. 5.

### 3.4 MAS support for balancing privacy versus accountability

Protecting the privacy of the user context, e.g., user ID, service preferences, past interaction history, personal details and payment credentials, requires more than supporting confidentiality and authorisation using encryption mechanisms – a multi-lateral approach is needed [17]. This approach offers four different levels of privacy: owner-centred privacy management, use of different degrees of anonymity, restricted disclosure of the user context on a need to know basis and to detect the misuse of privacy and to penalize those who do so. The privacy model is supported in a system called USHER, U-commerce Services Here for Roamers. There are several elements to this system. It uses a MAS organisation to model and associate roles to participants in the organisation and to control access to the user context according to specified context sharing policies. These roles are: an Owner of a user context; a Holder e.g., GPS device, authorised to collect user context information; a Requester, e.g., route-planner; authorised to get an owner's context; Nymisers that act as mediators, that are authorised to selectively reveals owners' contexts to others and an Authoriser that issues credentials for rights to access user-contexts, see Fig. 6.

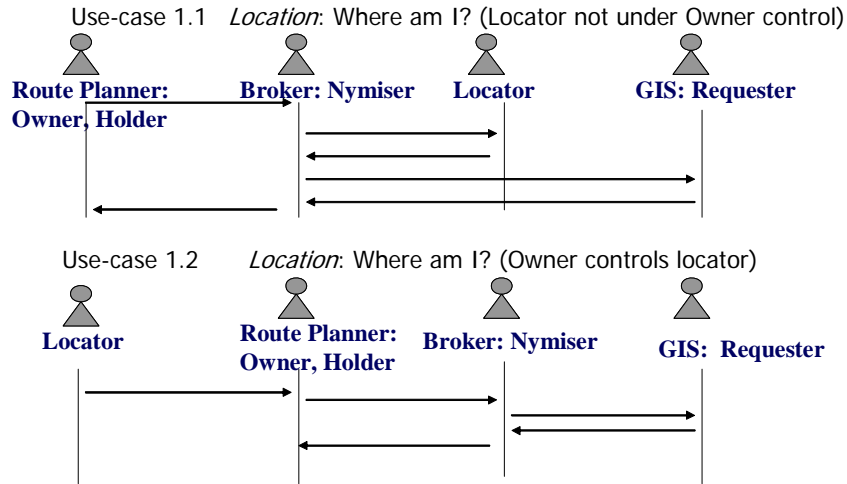


Fig. 6. Organisation interaction and roles for controlled sharing of a user's location context that has been created using two different location determination systems: client-side, e.g., GPS, and server-side, e.g., wireless network trilateration.

Because of the richness of the interaction between the different actors involved in exploiting and safeguarding the owner contexts, problems due to lack of a shared semantic model for privacy can arise [18]. These include: the inability to interlink service processes that access the user-context with the security processes that safeguard them; the lack of a common terminology to allow owner contexts to interlink to management concepts that are understood by the different actors and the lack of rich meta-data to manage the operation of the privacy safeguards. Hence, not only a formal policy-model is needed to support the management of owner preference access but also for a semantic model is needed to support it. The privacy model in [18] is based on the V-SAT model given above where the main asset that needs to be protected against threats is the user context. There is a W3C standard that could be used to express privacy policy called P3P but this is provider- not user-centred. P3P needs to be enhanced to support user contexts and to link the policy evaluation to be under user control. This system has been applied to two applications – a location-aware religious service that locates the nearest Mosque for Muslims and a restaurant recommender service [18].

### 3.5 MAS support for managing events and errors with multiple semantics

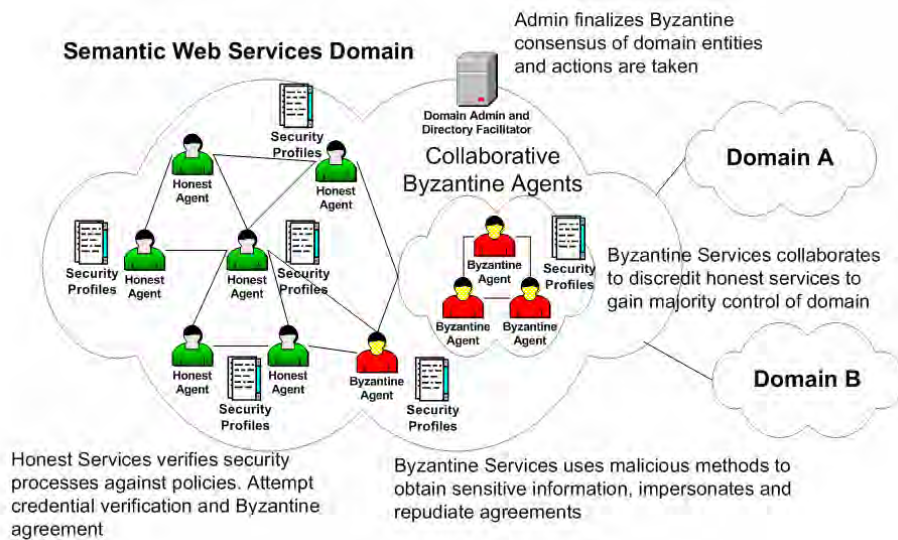
Interoperability amongst the various open system actors can be greatly aided by sharing semantic type metadata. A Semantic-based failure communication model based on an integrated failure and security ontological model has been developed as an extension to the V-SAT model [19]. This is combined with a semantic profile model to interlink policy models and process models of normal and failed behaviour can support improved, higher level, application oriented fault management. The semantic



model can also support a reasoning model to support adaptive fault and security management and the choice of an appropriate fault handler. Detected errors can be propagated and transformed when necessary into semantically tagged error events. In some cases errors are best handled and more efficiently handled locally whereas in other cases errors may best be propagated up to the agent level because errors may have global significance.

In a VO, there are often multiple (sub-)domains, e.g. Restaurants and Hotels. Some parties trust one another and some do not, but each believes that honest parties are in a majority and that a domain administrator manages the admission and consensus of agents operating in its respective domain (Fig. 7) on its behalf. Traditionally, the verification of authenticated peers possessing private keys corresponding to the public keys determines if a party is honest and capable. This concept of identity is very general and insufficient especially in rich semantic environments which may span individuals and organizations.

The operational capability including security of systems and applications in open service environments is reduced when malicious services are present. Such behaviour can be particularly hard to detect because of the dynamics of open systems and because it may only become apparent in certain service states and when a certain percentage of disruptive stakeholders or occurrences become noticeable. Whilst the Semantic Web has the potential to promote a richer interaction and interoperability between different parties, Byzantine behaviour may also arise because of the added complexity in unambiguously interpreting and processing semantic metadata and meta-processes in a consistent way between heterogeneous parties. The Byzantine Agreement Protocol (BAP) [20] can be used to tackle Byzantine behaviours. It uses a majority based rating mechanism for identifying and pruning Byzantine entities in a domain. It has received considerable attention for the design of fault tolerant systems. It seeks to establish a fault-tolerant agreement when one or more of the nodes in a system have been compromised or failed.



**Fig. 7.** Byzantine Services Scenario

### **3.6 MAS support for Managing Anomalies versus Normal variations**

A basic assumption underlying anomaly detection is that anomaly patterns differ from normal behaviour patterns. However it may not always be possible to differentiate the two with certainty, e.g., a flood of repeat requests by a frustrated user and a flood of events used by a malicious user designed to cause a DoS are similar. In this case, the patterns of events need to be characterised along a number of features with respect to some normal distribution of these features and only a difference probability can be calculated between normal and anomaly events. There are two main strategies for detecting and handling anomalies. If predefined normal patterns of events are available, then either anomaly events can be matched against anomaly pattern sets or against normal pattern sets. Secondly, if normal patterns are not available, they can perhaps be learnt from observation of operational events as normal events tend to cluster whereas anomalies tend not to [21].

## **4. Conclusions**

Although cryptographic security protocols represent the core protocols to safeguard distributed systems a multi-lateral approach is needed, cryptographic protocols need to be complemented with additional approaches based upon policy-based management, semantic-based mediation, agent-based task and information sharing and policy assessment. Some behaviours of VO such as Byzantine faults, anomalies, emergent, human and social behaviours are hard to model and will require more research to find ways to manage the security and safety aspects of these more effectively.

### **Acknowledgements**

The research described was aided by the endeavours of the following research assistants: Juan (Jim) Tan and Leonid Titkov.

## **References**

- [1] Clark, D.D. Wilson, D.R.A.: Comparison of Commercial and Military Computer Security Policies. IEEE Symposium of Security and Privacy (1987) pp 184-194
- [2] Bell, D.E. LaPadula, L. J.: Secure Computer Systems: Mathematical Foundations. MITRE Technical Report 2547, Volume I (1973)
- [3] Schneider, B.: Secrets and Lies: digital security in a networked world. John Wiley & Sons. ISBN 0-471-25311-1 (2000)

- [4] Nicol, D., Sanders, W., Trivedi, K.: Model-Based Evaluation: From Dependability to Security. *IEEE Transactions on Dependable and Secure Computing*, vol. 1:1 (2004) 48-65
- [5] Coase, R.H.: The Nature of the Firm. *Economica* 4 (1937) 386–405
- [6] Shao, Y.P., Lee, M.K.O. Liao, S.Y.: Virtual organizations: the key dimensions. *Proc. Academia/Industry Working Conference on Research Challenges* (2000) 3 – 8
- [7] Coleman, J.S.: Foundations of social theory. Harvard University Press. ISBN 0-674-31226-0 (1990)
- [8] Gligor, V.D.: Security of Emergent Properties in Ad-Hoc Networks. *Proc. of the Security Protocols Workshop*, Cambridge, UK, (2004)
- [9] Goldberg, I. A Pseudonymous Communications Infrastructure for the Internet, PhD thesis, Univ. of California at Berkeley (2000)
- [10] Garlan, D and Schmerl, B.: Model-based adaptation for self-healing systems. *Proc. 1st workshop on Self-healing systems*, Charleston, South Carolina, (2002) 27 - 32
- [11] Kephart, J.O. Chess, D.M.: The vision of autonomic computing. *Computer*, 36:1 (2003) 41-52
- [12] Axelrod, R.: The Dissemination of Culture: A Model with Local Convergence and Global Polarization. *Journal of Conflict Resolution*, 41:2, (1997) 203-226.
- [13] Poslad, S, Charlton, P.: Standardizing agent interoperability: the FIPA approach. In: Michael Luck, Vladimír Marík, Olga Stepánková, Robert Trappl (Eds.): *Multi-Agent Systems and Applications*, Lecture Notes CS, Vol. 2086 (2001) 98-117
- [14] Zhang, M, Karmouch, A, Impey R.: Towards a Secure Agent Platform based on FIPA. *Proc. MATA 2001*. Springer-Verlag. LCNS, Vol. 2164, (2001) 277-289
- [15] Ghanea-Hancock, R, Gifford, I.: Top secret multi-agent systems. 1st Int. Workshop on security of mobile multi-agent systems (SEMAS-2001), 5th Int. Conf. Autonomous Agents, Montreal, Canada (2001)
- [16] Tan, J.J., Poslad, S.: Dynamic Security Reconfiguration in Semantic Open Services Environment. *Eng. Apps. of AI*, Vol. 17 (2004) 783-797
- [17] Titkov, L., Poslad, S., Tan, J.J.: An Integrated Approach to User-Centered Privacy for Mobile Information Services. *Applied Artificial Intelligence J.*, 20: 2-4 (2006) 159-178
- [18] Tan, J.J., Poslad, S., Titkov, L.: A Semantic Approach to Harmonising Security Models for Open Services. *Applied Artificial Intelligence J.*, 20:2-4 (2006) 353-379
- [19] Poslad, S., Tan, J.J., Huang, X., Zuo, L.: Middleware for semantic-based security and safety management of open services, *Int. J. Web and Grid Services*, 1: 3-4 (2005) 305 – 327

- [20] Lamport, L., Shostak, R., Pease, M.: The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems, 4:3 (1982) 382-401
- [21] Steinwart I., Hush, D., Scovel, C.: A Classification Framework for Anomaly Detection. J. Machine Learning Research 6 (2005) 211–232

# A Framework for Goal-Based Semantic Compensation in Agent Systems

Amy Unruh, James Bailey, and Kotagiri Ramamohanarao

Dept. of Computer Science and Software Engineering  
The University of Melbourne, VIC 3010, Australia  
{unruh,jbailey,rao}@csse.unimelb.edu.au

**Abstract.** This paper describes an approach to improving the robustness of an agent system by augmenting its failure-handling capabilities. The approach is based on the concept of semantic compensation: “cleaning up” failed or canceled tasks can help agents behave more robustly and predictably at both an individual and system level. Our approach is goal-based, both with respect to defining failure-handling knowledge, and in specifying a failure-handling model that makes use of this knowledge. By abstracting failure-handling above the level of specific actions or task implementations, it is not tied to specific agent architectures or task plans and is more widely applicable. The failure-handling knowledge is employed via a failure-handling support component associated with each agent through a goal-based interface. The use of this component decouples the specification and use of failure-handling information from the specification of the agent’s domain problem-solving knowledge, and reduces the failure-handling information that an agent developer needs to provide.

## 1 Introduction

The design of reliable agent systems is a complex and important problem. One aspect of that problem is making a system more robust to failure. The work described in this paper is part of a Department of CSSE, University of Melbourne project to develop methodologies for building more robust multi-agent systems, in which we investigate ways to apply transactional semantics to improve the robustness of agent problem-solving and interaction. Traditional transaction processing systems prevent inconsistency and integrity problems by satisfying the so-called ACID properties of transactions: Atomicity, Consistency, Isolation, and Durability [1]. These properties define an abstract computational model in which each transaction runs as if it were alone and there were no failures. The programmer can focus on developing correct, consistent transactions, while the handling of recovery, concurrency, and failure is delegated to the underlying engine.

However, for most agent systems, with agents situated in and interacting with their environment, principles of transaction management cannot be directly

applied. The effects of many actions may not be delayed: such actions “always commit”, and thus correction of problems must be implemented by “forward recovery”, or failure *compensation* [1]– that is, by performing additional actions to correct the problem, instead of a transaction rollback. In addition, actions may not be “undoable” nor repeatable. Further, it is often not possible to enumerate how the tasks in a dynamic agent system might unfold ahead of time: it is not computationally feasible, nor do we have enough information about the possible “states of the world” to do so.

In this paper, we focus on one facet of robustness motivated by transactional semantics, that of *failure-handling*, and in particular, how the concept of *semantic compensation* can be used to support a robust approach to failure-handling in an agent context.

A “semantic” compensation need not exactly reverse a given effect, but rather is an “undo” appropriate to the context of the situation. Semantic compensation provides a way for an agent system to recover from task problems by “cleaning up after” its problematic actions, thus approximating failure atomicity. The effective use of semantic compensation in response to agent failure can have several benefits:

- It helps leave an agent in a more stable state, and one from which future actions– such as retries, or alternate methods of task achievement– are more likely to be successful;
- it helps maintain an agent system in a more predictable state: failure propagation is controlled; agent interactions are more robust; and unneeded resources are not tied up;
- the approach can be applied as a useful default method, when no domain-specific plans for patching a given failed activity are available.

However, the system characteristics discussed above mean that traditional transaction management methods are not appropriate in a multi-agent context. We cannot always characterize ‘transactions’ and their compensations ahead of time, nor create “compositions” of compensations by applying subcompensations in a reverse order [1]; in many domains such compensations will not be correct or useful. In addition, in an agent context, failure-handling behavior should be related to the agent’s goals, and take into account which goals are current or have been cancelled.

Thus, to operationalize the use of semantic compensation in an agent context, it is necessary both to define the compensations in a way that is effective, and to usefully specify when to initiate both compensations and goal re-achievements, including scenarios where problems occur with tasks delegated between agents. The key components of our approach, which address these issues, are as follows:

First, for a given domain, and for some subset of the tasks the agent knows how to perform, we associate information about what needs to be achieved to compensate a failure of that task. This information is specified in terms of the *goals* that need to be achieved in order for the compensation to be successful. That is, we specify, at an abstract level, declarative information about *what* to do to handle a failure, not *how* to achieve the implementation.

Second, a *compensation-retry* failure-handling model is defined, which uses the defined compensation knowledge, and is employed to address failures in the agent’s problem solving. Goal re-achievement is incorporated as an intrinsic part of this model: if a goal hierarchy remains current, then by default the agent continues to work towards achieving it at some level. Difficulties in compensating at failure points, or in retrying a failed goal, may cause an agent to “push up” failure-handling to a higher-level goal context.

In our approach, the agent’s failure-handling behaviour is supported by a layer that sits below each agent’s domain logic component. This underlying layer does the bookkeeping necessary to support the failure-handling model and its employment of the compensation information. Thus, agent developers are not required to encode the failure-handling logic at the agent’s “domain” level. By consistent employment of the failure-handling model across agents in a system, the system’s behaviour on failure becomes more robust and predictable.

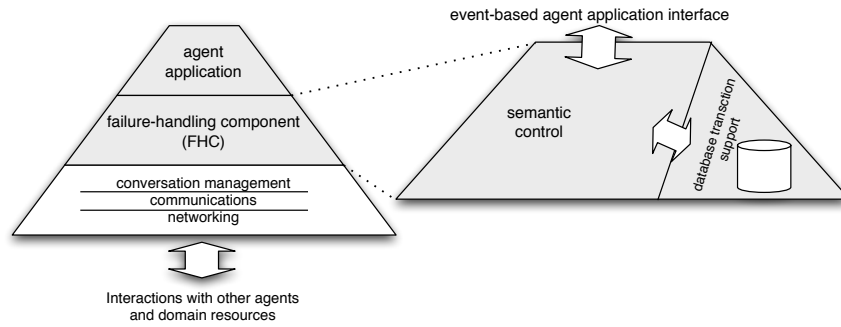


Fig. 1: An agent’s FHC. We refer to the domain logic part of the agent, above the failure-handling component, as the “agent application”.

This underlying failure-handling component, which we term the agent’s FHC, allows incremental specification of knowledge about how to perform compensations, and allows the agent to react to failures in a well-specified way when such information is available. As shown in Figure 1, the FHC interfaces with (in a typical architecture) the conversation protocol layer of the agent [2, 3] in addition to its domain logic component. In the remainder of this paper, we refer to the domain logic part of the agent, above the FHC, as the “agent application”.

A key aspect of this framework is the use of transactionally-supported logging, which allows consideration of execution history in failure-handling, and supports compensation of *already-achieved* goals. In addition, the FHC supports the necessary failure-handling interaction protocols between agents.

In Section 2, we provide a motivating example for our use of compensation, and describe at a conceptual level our failure-handling model and the knowledge it utilizes. Then, in Section 3 we detail the agent’s failure-handling component, or FHC, which makes use of this failure-handling model; and describe a prototype

which implements much of the approach. In Sections 4 and 5 we finish with a discussion of related work, and summarize.

## 2 Compensation-Based Failure-Handling

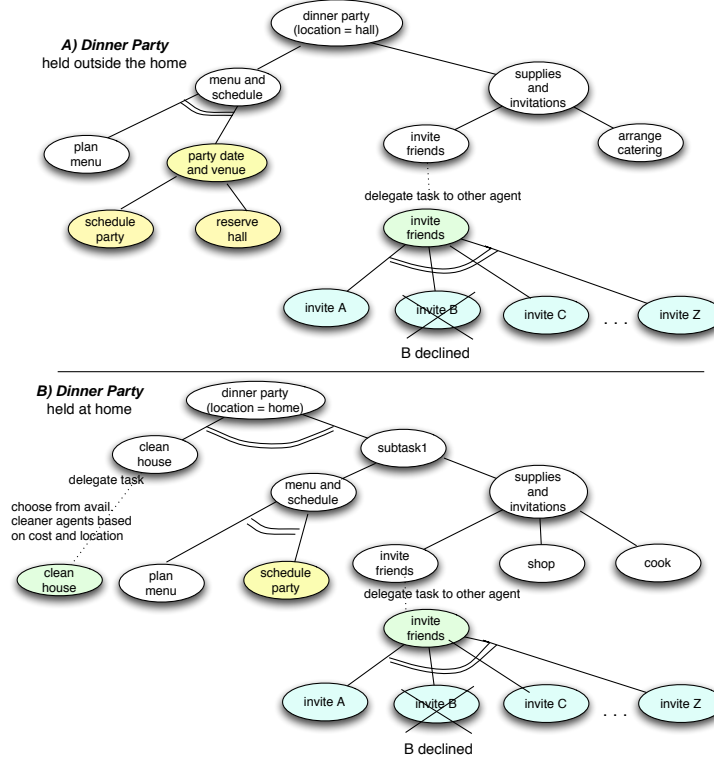


Fig. 2: Planning a dinner party. The figure uses an informal notation, where subtasks that may be executed concurrently are connected by a double bar; otherwise they are executed sequentially.

We motivate our approach to semantic compensation by presenting an example in a “dinner party” domain. (We use this domain because it has a rich semantics and is easily understandable; its issues can be mapped to analogous problems in more conventional domains). Consider two related scenarios, where a group of agents must carry out the activities necessary to prepare for holding a party. First, consider an example where the party will be held at a rented hall, for a business-related event. Fig. 2A shows one task decomposition for such an activity. The subtasks include planning the menu, scheduling the party and arranging to reserve a hall, inviting the guests and arranging for catering. The figure indicates that some of the subtasks (such as inviting the guests) may be



delegated to other agents in the system. Next, consider a scenario which differs in that the party will be held at the host’s house. In this case, while the party must be scheduled, a hall does not need to be reserved. In addition, the hosts will not have the party catered and will shop themselves. Fig. 2B shows a task decomposition for this scenario.

If the party planning fails or the event must be canceled, then a number of things might need to be done to properly take care of the cancellation— that is, to “compensate for” the party planning. However, the specifics of these activities will be different depending upon what has been accomplished prior to cancellation. In the first case (Fig. 2A) we may have to cancel some reservations, but if we have used caterers, then we will not have to deal with any extra food; in the second case (Fig. 2B), we have no reservations to cancel, but we will likely have unused food. In either event, the party cancellation activities can be viewed as accomplishing a *semantic compensation*. The nature of a useful compensation depends upon upon the nature of the task. An exact ‘undo’ is not always desirable— even if possible. Compensations are both context-dependent and usually infeasible to enumerate ahead of time, and employing a composition of subtask compensations is almost always too simplistic.

## 2.1 Goal-Based Compensation Definitions

The example above showed that compensation of a task can typically be achieved in different ways depending upon context. It is often difficult to identify prior to working on a task the context-specific details of how a task failure should be addressed or a compensation performed. It can be effectively impossible to define all semantic compensations prior to runtime in terms of specific actions that must be performed.

Instead, as a key component of our approach, we claim it is more useful to define semantic compensations *declaratively*, in terms of the goals that the agent system needs to achieve in order to accomplish the compensation, thus making these definitions more widely applicable. We associate *compensation definitions* with some or all of the tasks (goals)<sup>1</sup> that an agent can perform. These definitions specify at a goal— rather than plan— level *what* to do in certain failure situations, and the agents in the system then determine *how* a given goal is accomplished. The way in which these goals are achieved, for a specific scenario, will depend upon context.

Figures 3A and 3B illustrate this idea. Suppose that during the party-planning of the examples above, the host falls ill and the party needs to be canceled. Let the compensation goals for the party-planning task be the following:

**all party-related reservations should be canceled; extra food used elsewhere if possible; and guests notified and ‘apologies’ made.**

<sup>1</sup> In this paper, we use ‘goal’ and ‘task’ interchangeably; as distinguished from plans, action steps, or task execution. In our usage, goals describe conditions to be achieved, not actions or decompositions.

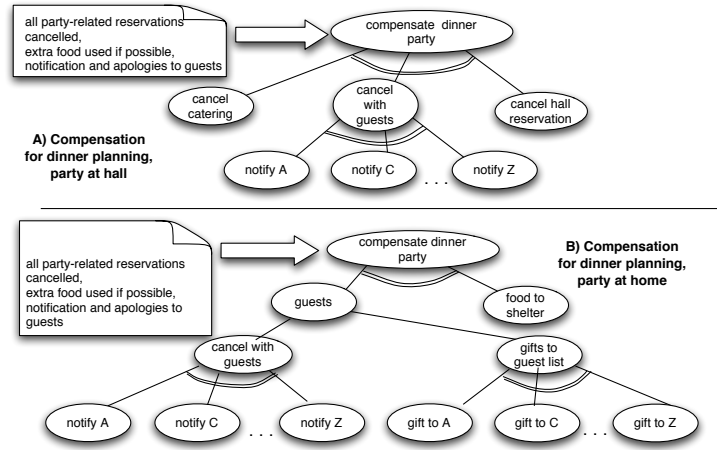


Fig. 3: Compensation of dinner party planning.

The figures show the resulting implementations of these compensation goals for the activities in Figs 2A and 2B. The compensation goals for the high-level party-planning task are the same in both cases, and indicate what needs to be made true in order for the compensation to be achieved. However, the *implementations* of these goals will differ in the two cases, due to the different contexts in which the agent works to achieve them. When the party was to be held at home (Fig. 3B), extra food must be disposed of, and (due to the more personal nature of the party) gifts are sent to the guests. In the case where the party was to be held in a meeting hall (Fig. 3A), there are reservations that need to be canceled. However, there is no need to deal with extra food (the caterers will handle it) nor to send gifts. Some compensation goals may be already satisfied and need no action, and some tasks may be only partly completed at time of cancellation (e.g. not all guests may be yet invited).

A semantic compensation may include activities that do not address any of the (sub)tasks of the original task, such as the gift-giving in the first example. Some compensation activities may “reverse” the effects of previous actions (e.g. canceling the reservation of a meeting hall), but other effects may be ignored (no effort is made to “undo” the effects of the house-cleaning) or partially compensatable (dealing with the extra food) depending upon context. The compensation for a high-level task will likely not correspond to a composition of compensations for the high-level activity’s expected subtasks.

Thus, in defining failure-handling knowledge for a given (sub)task, the developer must specify what must be achieved— what must be true about the state of the world— for compensation of that task to be successful. As described above, this information is specified *only in terms of goals*, not actions or plans— the agent application will determine at runtime how to implement, or achieve, these goals.

A goal-based formulation of failure-handling knowledge has several benefits:

- it allows an abstraction of knowledge that can be hard to express in full detail;
- its use is not tied to a specific agent architecture; and
- it allows the compensations to be employed in dynamic domains in which it is not possible to pre-specify all relevant failure-handling plans.

A compensation definition is specified in terms of the parameters of its associated “forward” task. However, the agent developer will make this specification *independently of the context* in which the task may be achieved. That is, it must be possible to define a task’s compensation semantics independent of the state of the world or parent task; we do not know a priori the circumstances under which the task will be invoked or whether it will have fully completed. Encoding failure-handling knowledge as goals, not plans, allows this criteria to be addressed. The application agent will take current state into account when deciding at runtime how to implement a given compensation goal.

Task	Compensation Goals
plan dinner party	all party reservations canceled & extra food used if possible & notification and apologies to guests
invite guests	guests notified of cancellation
reserve hall	hall reservation canceled
shop for supplies	supplies returned

Table 1: Simplified compensation definitions for the ‘dinner party’ example.

Table 1 shows some example task compensation definitions from the dinner party scenario. We view construction of the definitions as “assisted knowledge engineering”; we can examine the agent’s problem-solving implementation, and in particular its goal-subgoal relationships, and leverage this implementation knowledge to support the definition process. We are researching ways to provide semi-automated support for this process.

## 2.2 Compensation-Retry Model

Given a set of goal-based compensation definitions for an agent domain, we would like the agents to employ this information consistently across a system, in a manner that supports recovery from problems, and helps prevent propagation of failure between agents. Here, we present an approach to failure-handling, based on invocation of compensations and goal re-achievement, that addresses these criteria.

We define a *compensation-retry* failure-handling model that is used by each agent in a system, supported and enforced by the agents’ failure-handling layers. Failure-handling is triggered by a **failure** or **cancellation** event on a goal— as detected by the agent’s domain logic. A **cancellation** event may occur either

in conjunction with a failure or by itself, and has the semantics that the effects of the goal should be compensated but the goal should no longer be pursued.

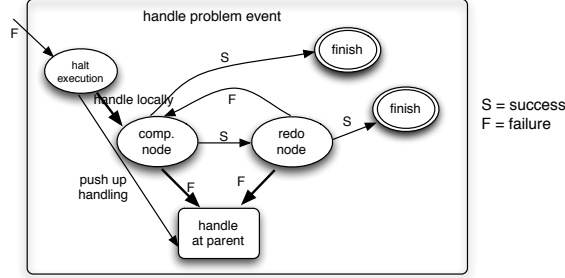


Fig. 4: The FSM for the failure-handling model triggered by a goal failure.

There are two basic building blocks used in our failure-handling process– the invocation of compensation for a failed task, and the invocation of a retry (re-achievement) of a failed task. The failure-handling model controls when and at what goal level these two building blocks may be applied. The model incorporates two important concepts. First, if a goal is not cancelled, there is always an option to persist in goal re-achievement after compensation. Second, if compensation at the failure point is not successful, the agent may “push up” the failure event to the parent goal context. This means that handling of the problem will result in broader and more general cleanup, and pursuit of alternate methods of achieving the higher-level tasks.

The failure-handling model can be viewed as a set of finite-state machines (FSMs), where the transitions between the states are events signaling changes in goal status. Figure 4 shows a simplified version of the FSM initiated by a goal failure. A “compensation” state indicates that the compensation goals associated with the failed task are invoked. A “retry” state means that a new instantiation of the original ‘forward’ task is invoked. As discussed above, the agent’s domain logic determines both how to compensate for a failed goal, and how to work on re-achieving a goal. So, a “retry”– an attempt at goal re-achievement– does not imply that the goal should be (necessarily) re-attempted in the same manner as before. The agent application makes that determination based on the current problem-solving context.

When failure occurs, the FSM of Fig. 4 defines what options are available at each stage of the failure-handling process, and the conditions under which the failure-handling may terminate. A similar FSM, not shown, encodes the agent’s response to a cancellation event<sup>2</sup>. A key difference is that when a goal is cancelled, the agent does not attempt to re-achieve the goal after compensation.

FSM instances may be nested; e.g., a failure within the implementation of a ‘retry’ subgoal can be pushed up to the enclosing FSM as a failure of the retry

<sup>2</sup> In fact, the two FSMs can be combined.

itself. When such failure-handling models are used consistently across an agent system, the system’s response to failure becomes more predictable and robust.

**Refining the Failure-Handling Model with Domain Knowledge.** For a given FSM state and goal status event event, there may be more than one transition possible. For example, the figure above indicated that if a retry fails, the agent may either “push up” the compensation, or re-attempt a compensation-retry sequence. The agent developer can add domain knowledge to refine the system’s failure-handling defaults, by specifying which of the allowable transitions in the failure-handling finite-state machine above, should be followed. As with the compensation definitions, specification of these conditions requires knowledge of the domain semantics.

The next section describes how this failure-handling model, with associated preferences on its state transitions, is implemented by the agent’s failure-handling component, or FHC. For the failure-handling process to be most effective, there are requirements on the capabilities of the agent with which it is used. These are described in Section 3.3.

### 3 An Architecture for Failure-Handling

The previous section described a model for defining compensations, and employing compensations and goal retries, in dynamic and complex agent environments. In this section, we describe an architecture that supports this model, to allow consistent and predictable system behaviour on failure, while reducing what needs to be done by the agent developer to program the agents to handle failure. We separate each agent’s “normative” from failure-handling knowledge by means of a decoupled component which, as introduced in Figure 1, we term the agent’s *FHC*. It is this layer of the agent architecture which implements the failure-handling model described in the previous section.

An agent’s domain logic resides in its agent application layer, which performs planning, task decomposition and execution. A goal-based API between the layers allows the FHC to both monitor and affect the agent application’s problem-solving. Based on its monitoring, an agent’s FHC determines *which* failure-handling activities will be performed— with respect to both task compensations and re-achievements— and *when* they will be requested. The agents’ application logic is then invoked to implement the tasks and determine the details of *how* they are achieved.

The interface between the FHC and its agent application is based on goal events. An agent application reports new (sub)goals to its FHC as it generates its task decompositions, including information about the new goal’s parent; and reports the goal status events it detects— **success**, **failure**, and **cancellation**— as its problem-solving progresses. The agent application also reports information that allows the FHC to keep track of which goals have been delegated to other agents. Based on the information reported via this API, an agent’s FHC determines whether/when an agent can start work on a goal, can instruct it to start

new goals in order to implement compensations and/or re-achievements, and can tell it to halt work on an existing goal.

The use of the FHC reduces the agent developer’s implementation requirements, by providing a model that structures and supports the failure-handling information that needs to be defined. (The FHC implementation must be specific to a given agent framework, including its communication mechanisms, and ‘connects’ the agent application to the lower agent layers). The motivation behind the use of the FHC is analogous to that of the exception-handling mechanism in a language like Java; the developer is assisted in generating desired agent failure-handling behavior, and the result is easier to understand and predict than if the knowledge were added in an ad-hoc fashion. In Section 3.1, we describe the FHC’s task monitoring and logging mechanisms. Then, Section 3.3 discusses the requirements imposed by this approach on the agent applications in a system.

### 3.1 FHC Task Monitoring and Logging

To provide failure-handling support, an agent’s FHC performs high-level monitoring of the agent application via its API, and maintains a transactionally-supported abstract log, or history, of its problem-solving. The maintenance of this abstract history, in conjunction with the domain knowledge described in Section 2, gives the FHC the information it needs to implement the compensation-retry model at failure points<sup>3</sup>.

The FHC logs at a goal level of abstraction: each subgoal is represented by a ‘node’ in the log, with two types of dependency information maintained between nodes: parent-child goal relationships, including those of delegated goals (as will be further discussed below), and failure-handling relationships (*compensation-for*, *retry-of*). Each node stores information about whether its associated goal is current (part of the current execution path), and any goal status events that have been reported for it. Event timestamps are recorded as well. The monitored information, and the logged nodes, do not include plan or action details. Thus, they are a lightweight abstraction of the agent’s actual problem-solving history.

In the following, we refer to (fragments of) the log as *task-monitoring* trees, reflecting the fact that the logging encodes parent-child task relationships. The task-monitoring trees support the FHC’s failure-handling. Figure 5(a) shows a simple example of the way in which a task-monitoring tree is incrementally built as agent problem-solving is tracked. The agent application reports new goals, and waits for the signal from its FHC before starting them<sup>4</sup>. The agent application, using its domain knowledge, performs the actual problem solving to achieve the goal. Note that the agent application need not be using an explicit “tree” representation itself, nor be keeping track of goal parent-child relationships over

---

<sup>3</sup> In our current implementation, each agent maintains its log only locally, though in the future it may be of utility to allow the agents to export some history information to other agents.

<sup>4</sup> While not discussed in this paper, the FHC log may also be used to support task coordination.

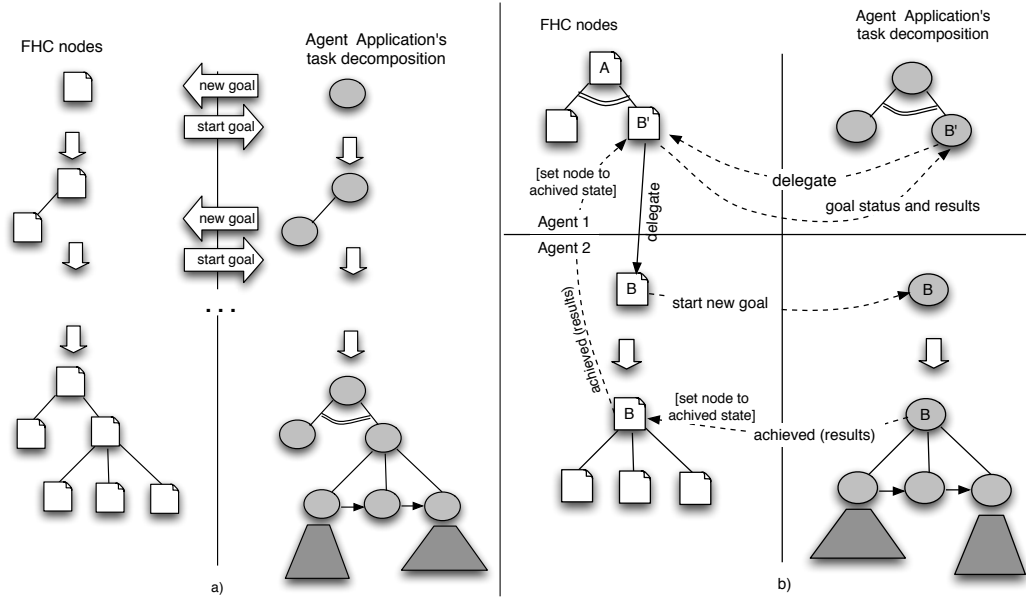


Fig. 5: Constructing an FHC “task tree”, by monitoring problem-solving. Each FHC node contains information about task status and associated failure-handling definitions. Task details are not tracked.

time; it only needs to be able to report its subgoals as they are generated. Part of the role of the FHC is to record this dependency information, required for its failure-handling model.

FHC trees are retained for all tasks that have a currently-executing root or that have a repair relationship to a currently-executing task. Thus the extent of the abstract logging depends on the agent’s activities. (Sub)tasks that have a currently-active ancestor task may not be flushed from the log, even if they themselves are completed. Task trees that are terminated may potentially be removed. However, by maintaining finished tasks on the log, the FHC can support compensations of completed tasks in addition to addressing problems with currently-executing tasks.

### 3.2 Implementation of the Compensation-Retry Model

The failure-handling model described in Section 2.2 is supported by the logging mechanism above. The abstract execution history, and the relationships maintained between history ‘nodes’, allows reasoning about the failure-handling states and the transitions between them.

The failure-handling process is triggered for a given goal by a **failure** and/or **cancellation** event on that goal, detected by the agent application and reported via its API<sup>5</sup>. However, failure-handling is initiated only if compensation infor-

<sup>5</sup> In addition to goal events detected by the agent application, cancellation events on already-achieved goals may also be detected via FHC *cancellation rules*, based

mation exists for the failed or cancelled goal. Thus, this information may be added incrementally to an agent system, and used where available.

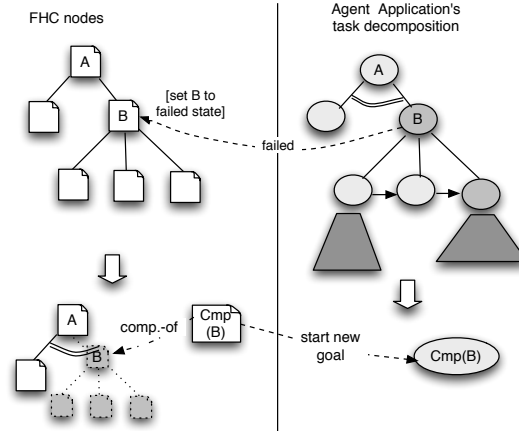


Fig. 6: Initiating a compensation goal for failed task B. B’s history remains in the log. The relationship with its compensation task is recorded in the log.

The FHC’s failure-handling logic is implemented as a set of ECA rules [4]. More specifically, the FSMs described in Section 2.2 are implemented by a set of rules, residing in the FHC, that are triggered by goal events and conditioned on compensation definitions in conjunction with information about the nodes in the FHC’s log. The rule execution results in instructions to the agent application, which tell it to start new goals, or stop work on existing goals. Compensations and retries, initiated by the FHC’s rules, cause the creation of new nodes in the FHC’s log. The new nodes maintain links to the node for which they are compensating or retrying. Figure 6 shows the process of initiating the compensation associated with a “forward” goal in response to a failure detected for that goal.

**Inter-Agent Failure Handling** To support failure-handling in scenarios where multiple agents work to achieve parts of a larger task, it is necessary for the FHCs of the agents involved to be able to ‘connect’ delegated tasks between agents. By doing so, they effectively maintain a distributed task structure which supports failure-related communication between the connected agents [5–7]. Such communication is necessary to support the FHC’s reasoning— not only to propagate changes in the status of delegated goals between agents, but to support interactions to determine which of the agents involved in a task will take responsibility for the failure-handling if a problem occurs.

on information in the log about the completed goal, in conjunction with changes in the current state of the world. The compensation of a completed goal, once its ‘cancellation’ is detected, may be handled using the same basic mechanisms as above. While this is an important feature of our larger research model, for simplicity we do not discuss it here.



The agent application’s ‘regular’ interaction protocols will determine how subtasks are allocated and delegated. These protocols may involve negotiation or bidding [2] or market mechanisms [8], as well as direct sub-task assignments or subscriptions, and typically include the reporting of error/task results.

It is important that the FHC’s ability to “connect” tasks across agents does not make assumptions about the specifics of these regular protocols (or how they may change), so that failure-handling remains decoupled in delegation situations, and failure-handling protocols can be implemented by the FHC in a manner transparent to the agent application. Figure 5(b) suggests the task delegation bookkeeping that needs to be maintained by the FHC. [9] provides details of the API developed to support this process, and describes the failure-handling interaction protocols in more detail.

### 3.3 Requirements on the Agent Applications and Agent System

The model presented here is most effective in the context of a situated agent: one which senses its execution results, and detects changes to its environment. In addition, the agent must be able to explicitly detect goal status changes. To robustly support retries and compensations, an agent must *be able to determine whether or not a given goal needs to be achieved*; whether it has failed, and in what failure mode. (In general, detection of failure can be undecidable; however, the FHC may be instructed to infer failure for a given task if the agent does not respond with task status within a given domain-dependent time interval.)

Further, our goal-based agent/FHC interface imposes certain requirements on the agent application. For an agent to implement the interface correctly, the following must be true. First, the agent must be able to perform goal-based problem-solving and report ‘goal events’. Second, an agent application must attempt to implement the goal-based instructions it receives from its FHC<sup>6</sup>. These instructions may include the *halt* of an existing task as well as new goal initiations.

In addition to the requirements that the agent applications must meet, failure-handling is best supported if the multi-agent system supports some form of brokering or service discovery [10], enabling robust re-delegation of retries of compensations if the original agent assigned the task is no longer available.

### 3.4 Prototype Implementation

We have implemented a prototype multi-agent system in which for each agent, an FHC interfaces with an agent application logic layer. The agents are implemented in Java, and the FHC of each agent is implemented using a Jess core [11], with the ECA rules of the FHC implemented as Jess rule sets. The agent-application components of our prototype are simple goal-based problem-solvers to which an implementation of the FHC interface was added. Currently, the

---

<sup>6</sup> In this paper, we do not explore issues of levels of commitment or goal prioritization.

transaction layer/agent application interface uses Jess “fact” syntax to communicate goal-based events. Inter-agent communication in our prototype system is primitive; in the future we plan to situate the implementation of the FHC on an established agent system framework, which can provide communication, brokering, and interaction protocol support. Using our prototype, we have performed initial experiments in several relatively simple problem domains. Preliminary results are promising, and suggest that our approach to defining and employing goal-based failure-handling information generates useful behavior in a range of situations. Work is ongoing to define failure-handling knowledge and strategies for more complex problem domains in which agent interaction will feature prominently. In the process, the FHC framework will be used as a testbed for evaluating and comparing different failure-handling strategies. Section 5 discusses some of the issues we expect to encounter and investigate in these more complex domains, and discusses future work in the larger project context.

## 4 Related Work

Our approach is motivated by a number of transaction management techniques in which sub-transactions may commit, and for which forward recovery mechanisms must therefore be specified. Examples include open nested transactions [1], flexible transaction [12], SAGAs, [13], and ConTracts [14]. Earlier related project work has explored models for the implementation of transactional plans in BDI agents [15–18], and a proof-of-concept system using a BDI agent architecture with a closed nested transaction model was constructed [19].

In Nagi et al. [20], [21] an agent’s problem-solving drives ‘transaction structure’ in a manner similar to that of our approach (though the maintenance of the transaction structure is incorporated into their agents, not decoupled). However, they define specific compensation plans for (leaf) actions, which are then invoked automatically on failure. Thus, their method will not be appropriate in domains where compensation details must be more dynamically determined.

The SPARK [22] agent framework is designed to support situated agents, whose action results must be sensed, and for which failure must be explicitly detected. ConGolog’s treatment of exogenous events and execution monitoring has similar characteristics [23]. While these languages do not directly address crash recovery, their action model and task expressions are complementary to the recovery approach described here.

Parsons and Klein [24] describe an approach to MAS exception-handling utilizing sentinels associated with each agent. For a given specific domain, such as a type of auction, “sentinels” are developed that intercept the communications to/from each agent and handle certain coordination exceptions for the agent. All of the exception-detecting and -handling knowledge for that shared model resides in their sentinels. In our approach, while we decouple high-level handling knowledge, the agent retains the logic for failure detection and task implementation; agents may be added to a system to handle certain failures if need be.

Chen and Dayal [25] describe a model for multi-agent cooperative transactions. Their model does not directly map to ours, as they assume domains where commit control is possible. However, the way in which they map nested transactions to a distributed agent model has many similarities to our approach. They describe a peer-to-peer protocol for failure recovery in which failure notifications can be propagated between separate ‘root’ transaction hierarchies (as with cooperating transactions representing different enterprises).

[26] describe a model for implementing compensations via system ECA rules in a web service environment— the rules fire on various ‘transaction events’ to define and store an appropriate compensating action for an activity, and the stored compensations are later activated if required. Their event- and context-based handling of compensations have some similarities to our use of strategy rules. However, in their model, the ECA rules must specify the compensations directly at an action/operation level prior to activation (and be defined by a central authority).

WSTx [6] addresses transactional web services support by providing an ontology in which to specify *transactional attitudes* for both a service’s capabilities and a client’s requirements. A WSTx-enabled ‘middleware’ may then intercept and manage transactional interactions based on this service information. In separating transactional properties from implementation details, and in the development of a transaction-related capability ontology, their approach has similar motivations. However, their current implementation does not support parameterized or multiple compensations.

Workflow systems encounter many of the same issues as agent systems in trying to utilize transactional semantics: advanced transactional models can be supported [27], but do not provide enough flexibility for most ‘real-world’ workflow applications. Existing approaches typically allow user- or application-defined support for semantic failure atomicity, where potential exceptions and problems may be detected via domain rules or workflow ‘event nodes’, and application-specific fixes enacted [28, 29]. Recent process modeling research attempts to formalize some of these approaches in a distributed environment. For example, BPEL&WS-Coordination/Transaction [5] provides a way to specify business process ‘contexts’ and scoped failure-handling logic, and defines a ‘long-lived transaction’ protocol in which exceptions may be compensated for. Their scoped contexts and coordination protocols have some similarities to our failure-handling strategy rules and FHC interaction protocols.

## 5 Discussion and Future Work

We have described an approach to improving robustness in a multi-agent system. The approach is motivated by transactional semantics, in that its objective is to support *semantic compensations* for tasks in a given domain; we assume environments in which the agents cannot wait to commit actions. We provide the agents in a system with a failure-handling model based on compensations and retries. Its consistent use makes the semantics of an agent system more

predictable, both with respect to the individual agent and with respect to its interactions with other agents; thus the system can become more robust in its reaction to unexpected events.

An important criteria driving our approach is that it address issues faced by agents situated in a changing environment. For situated agents, exogenous events can cause problems in accomplishing a task or render a task unnecessary. A key aspect of robust agent behaviour is the ability to react appropriately in such situations. Our approach is goal-based, both with respect to defining failure-handling knowledge for agent tasks, and in deciding when to employ it. We separate the definition of failure-handling knowledge from the agents' implementations of those tasks. Our hypothesis, borne out by our initial experiments, is that in a large range situations this knowledge can be separated. One goal of further experiments will be to characterize when this approach works is most effective. Our experiments will also help us to evaluate the ways in which compensation definitions and failure-handling domain knowledge are tied to the problem representation. For example, task/subtask granularity influences retry strategy.

Failure-handling and crash recovery are closely tied, and one focus of current work is the extension of our existing model and framework to support a unified treatment of these two activities. In our model, crash points are treated as failure events, and the effects of a crash are sensed in the same way as unexpected execution events. Compensation post-crash helps 'reset' an agent, approximating a rollback, and post-recovery, the agent continues to work on relevant goals. We will explore these ideas in the context of a larger-scale implementation that extends an existing agent framework. Another focus of current research is the development of a 3APL-based language and semantics for describing our failure-handling model.

**Acknowledgments.** This research is funded by an Australian Research Council Discovery Grant.

## References

1. Gray, J., Reuter, A.: Transaction Processing: Concepts and Techniques. Morgan Kaufmann (1993)
2. FIPA: (Fipa)
3. Nodine, M., Unruh, A.: Facilitating open communication in agent systems. In Singh, M., Rao, A., Wooldridge, M., eds.: Intelligent Agents IV: Agent Theories, Architectures, and Languages. Springer-Verlag (1998)
4. Dayal, U., Hanson, E., Widom, J.: Active database systems. In: Modern Database Systems: The Object Model, Interoperability, and Beyond. (1995)
5. Curbera, F., Khalaf, R., Mukhi, N., Tai, S., Weerawarana, S.: The next step in web services. COMMUNICATIONS OF THE ACM **Vol. 46, No. 10** (2003)
6. Mikalsen, T., Tai, S., Rouvellou, I.: Transactional attitudes: Reliable composition of autonomous web services. In: Workshop on Dependable Middleware-based Systems. (2002)
7. Nodine, M., Unruh, A.: Constructing robust conversation policies in dynamic agent communities. In: Issues in Agent Communication. Springer-Verlag (2000)
8. Walsh, W., Wellman, M.: Decentralized supply chain formation: A market protocol and competitive equilibrium analysis. JAIR **Vol. 19** (2003) 513–567

9. Unruh, A., Bailey, J., Ramamohanarao, K.: Managing semantic compensation in a multi-agent system. In: The 12th International Conference on Cooperative Information Systems, Cyprus, Springer Verlag LNCS (2004)
10. Cassandra, A., Chandrasekara, D., Nodine, M.: Capability-based agent matchmaking. In Sierra, C., Gini, M., Rosenschein, J.S., eds.: Proceedings of the Fourth International Conference on Autonomous Agents, Barcelona, Catalonia, Spain, ACM Press (2000) 201–202
11. Friedman-Hill, E.: Jess in Action. Manning Publications Company (2003)
12. Zhang, A., Nodine, M., Bhargava, B., Bukhres, O.: Ensuring relaxed atomicity for flexible transactions in multidatabase systems. In: Proceedings of the 1994 ACM SIGMOD international conference on Management of data, Minneapolis, Minnesota, United States, ACM Press (1994) 67–78
13. Garcia-Molina, H., Salem, K.: SAGAs. In: ACM SIGMOD Conference on Management of Data. (1987)
14. Reuter, A., Schwenkreis, F.: Contracts - a low-level mechanism for building general-purpose workflow management-systems. Data Engineering Bulletin **18** (1995) 4–10
15. Rao, A.S., Georgeff, M.P.: An abstract architecture for rational agents. In: *Third International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann (1992)
16. Busetta, P., Bailey, J., Ramamohanarao, K.: A reliable computational model for BDI agents. In: 1st International Workshop on Safe Agents. Held in conjunction with AAMAS2003. (2003)
17. Ramamohanarao, K., Bailey, J., Busetta, P.: Transaction oriented computational models for multi-agent systems. In: 13th IEEE International Conference on Tools with Artificial Intelligence, Dallas, IEEE Press (2001) 11–17
18. Smith, V.: Transaction oriented computational models for multi-agent systems. Internal Report, University of Melbourne (2003)
19. Busetta, P., Ramamohanarao, K.: An architecture for mobile BDI agents. In: 1998 ACM Symposium on Applied Computing. (1998)
20. Nagi, K., Nimis, J., Lockemann, P.: Transactional support for cooperation in multiagent-based information systems. In: Proceedings of the Joint Conference on Distributed Information Systems on the basis of Objects, Components and Agents, Bamberg (2001)
21. Nagi, K., Lockemann, P.: Implementation model for agents with layered architecture in a transactional database environment. In: AOIS '99. (1999)
22. Morley, D., Myers, K.: The SPARK agent framework. In: AAMAS '04, NY, NY (2004)
23. de Giacomo, G., Lesperance, Y., Levesque, H.J.: ConGolog, a concurrent programming language based on the situation calculus. Artif. Intell. **121** (2000) 109–169
24. Parsons, S., Klein, M.: Towards robust multi-agent systems: Handling communication exceptions in double auctions. In: Submitted to The 2004 Conference on Autonomous Agents and Multi-Agent Systems. (2004)
25. Chen, Q., Dayal, U.: Multi-agent cooperative transactions for e-commerce. In: Conference on Cooperative Information Systems. (2000) 311–322
26. T. Strandens, R.K.: Transaction compensation in web services. In: Norsk Informatikkonferanse. (2002)
27. Alonso, G., Agrawal, D., El Abbadi, A., Kamath, M., Gunthor, R., Mohan, C.: Advanced transaction models in workflow contexts. In: ICDE. (1996)
28. Casati, F.: A discussion on approaches to handling exceptions in workflows. SIG-GROUP Bulletin **Vol 20. No. 3** (1999)
29. Rusinkiewicz, M., Sheth, A.P.: Specification and execution of transactional workflows. Modern Database Systems: The Object Model, Interoperability, and Beyond (1995) 592–620

# A Guardian Agent Approach to Safety in Medical Multi-agent Systems

S.Modgil and J.Fox

Advanced Computation Lab, Cancer Research UK (CRUK),  
44 Lincolns Inn Field, London

**Abstract.** We present an overview of current work on argumentation based reasoning to ensure safety in medical multi-agent systems. In particular, we describe deployment of a specialised Guardian agent engaging in argumentation based reasoning about safety, and argumentation based dialogues with other agents to cooperatively ensure that agent activities are safe. The work is being conducted as part of a project aiming at development of theoretical models of argumentation, and their implementation in software components for deployment in agent technologies. The project has established a medical multi-agent scenario requiring deployment of the Guardian agent. This scenario provides a focus both for theoretical work and the eliciting of technical requirements. In this paper we describe the scenario and discuss the theoretical issues being addressed.

## 1 Introduction

The issue of ensuring safety is a high priority for multi-agent system (MAS) applications in safety critical domains such as medicine. A number of these systems (e.g., [1], [13]) involve specialised medical domain agents (SMDAs) communicating with each other in order to specify one or more medical actions. Hazards may result from interactions between actions separately planned by SMDAs, or from one or a combination of planned actions contraindicated by a particular patient's clinical circumstances. Hence, in order that a "Guardian Agent" (GA) be in a position to warn of hazards that may arise, requires that such a GA have a global overview of the SMDAs' action recommendations, as well as access to patients' clinical information. However, ensuring safety is not simply a matter of prohibiting a medical action, since the treatment goal of the prohibited action may still have to be realised if the patient's health is not to be compromised. Furthermore, strategies other than prohibition may be appropriate. This suggests a more involved deliberative interaction between a GA and SMDAs to decide a safe course of action; one that involves argumentation.

To illustrate, consider an SMDA deciding over a course of action for realising a treatment goal. Any number of actions may be appropriate (e.g., any number of drugs may be appropriate for reducing hypertension). Deciding a preferred course of action can be facilitated by weighing up the arguments for alternative courses of action. Suppose the SMDA recommends a preferred action  $a_1$ . The GA

may indicate that  $a1$  is contraindicated by the patient's clinical circumstances. There are then a number of possible strategies for ensuring that safety is not compromised. For example,  $a1$  may simply be prohibited by the GA. On the other hand, the SMDA may argue that the treatment goal of  $a1$  is a necessary goal, in which case the GA may advise that the SMDA choose an alternative action - one that, from the SMDA's perspective, may be deemed as less preferred by its decision making process. Or, the GA may recommend another action that ameliorates the hazard resulting from  $a1$ . Deliberating over a safe course of action thus involves a dialogue between the SMDA and the GA, involving exchange of the SMDA's arguments for action and the GA's arguments for alternative strategies to enforce safety. To sum up, we list the following requirements for a Guardian Agent:

- **Global overview:** The GA will have a global view of agent activities so that it can monitor actions proposed or committed to by distributed SMDAs
- **Specialisation:** The GA will be a dedicated agent with specialist safety reasoning capabilities<sup>1</sup>. It should be equipped with as generic a theory of safety as possible so as to facilitate deployment in a wide range of medical MAS applications
- **Argumentation:** The GA will make use of arguments for alternative strategies for ensuring safety, and engage in dialogues with other agents to agree on safe plans of action. These dialogues will involve exchange of arguments.
- **Information access:** The GA will have (possibly privileged) access to patient information

In this paper we focus on the challenges for argumentation theory raised by the deployment of a Guardian agent in medical multi-agent systems. We present an overview of current work addressing these challenges. This work is being conducted as part of the EU 6th framework project ASPIC (Argumentation Service Platform with Integrated Components)<sup>2</sup>. ASPIC aims at development of theoretical models of argumentation, implementation of these models in software components, and development of a platform for component integration with knowledge resources (semantic web) and embedding of components in agent technologies. ASPIC has identified a number of scenarios for steering ASPIC research and development activities. These scenarios consist of agents with ASPIC developed components implementing inference of arguments, argumentation based decision making and argumentation based dialogue. One such scenario - the Guardian Agent scenario - describes a Guardian agent engaging in argumentation based reasoning with other agents to cooperatively ensure that agent activities are safe. In section 4 we describe the scenario and ongoing research addressing the formal modelling of this scenario. Prior to this, we

---

<sup>1</sup> Note that an advantage of centralising safety reasoning in a Guardian Agent is that it will also facilitate auditing of safety after execution of plans. Auditing is difficult if safety reasoning is distributed among SMDAs

<sup>2</sup> CRUK are the scientific coordinators of ASPIC, which is comprised of 8 academic and 2 industrial partners. For further information visit [www.argumentation.org](http://www.argumentation.org)

give some background on argumentation (section 2) and discuss integration of argumentation based reasoning in agent architectures (section 3).

## 2 The Basics of Argumentation

Argumentation captures everyday patterns of practical reasoning that are precluded by the monotonicity and inconsistency intolerance of classical logical approaches. Argumentation begins with logic based inference of arguments. Given a theory denoting an agent's belief or knowledge base, arguments for competing (possibly inconsistent) claims can be inferred. In this view an argument consists of a claim and its support, where the support corresponds to a proof of the claim. The support can be an unstructured set of data from which the claim is derived, or a structured sequence of lines of reasoning or an inference tree. Assuming the former, let  $b, p$  and  $f$  respectively denote that 'Tweety is a bird', 'Tweety is a penguin' and 'Tweety flies'. Given the propositional logic theory  $\{b, p, b \rightarrow f, p \rightarrow \neg f\}$ , then two arguments can be inferred: one for the claim that Tweety flies -  $A1 = \{b, b \rightarrow f\} : f$ ; and one for the claim that Tweety does not fly -  $A2 = \{p, p \rightarrow \neg f\} : \neg f$ .

The next step is to evaluate the status of arguments. Evaluation proceeds in three steps:

Firstly, the interactions (usually conflict based) between arguments are defined. For example, if two arguments have conflicting claims, then they are said to *attack* each other. Hence, in the above example  $A1$  and  $A2$  attack each other. On the other hand, if the claim of one argument conflicts with a datum in the support of another argument, then the former attacks the latter, but not vice versa. This is referred to as an undercutting attack or simply *undercut*. For example, suppose an argument  $A3$  for the claim that 'Tweety is not a bird' ( $\neg b$ ). Then  $A3$  undercuts  $A1$ .

Secondly, pairs of mutually attacking arguments then need to be compared on the basis of some relative valuation. This comparison is manifested as a *defeat* relation. So, in our example, since penguins are a subclass of birds, then by the specificity principle which states that properties of subclasses take priority over properties of superclasses, we have that  $A2$  defeats  $A1$ .

Thirdly, one needs to account for defeat relations between all arguments to determine the arguments with preferred status. This is commonly accomplished by applying Dung's seminal 'calculus of opposition' [3]:



Let  $S$  be a set of arguments and  $Defeat \subseteq S \times S$  (i.e., the set of pairs  $(A, A')$  such that  $A$  defeats  $A'$ ). Let  $E \subseteq S$ . Then  $E$  is an *acceptable* set of arguments iff:

1. no two arguments in  $E$  defeat each other
2. for any  $A \in E$ , if there exists an  $A' \in S$  such that  $A'$  defeats  $A$  then there exists an  $A''$  in  $E$  such that  $A''$  defeats  $A'$
3.  $E$  is set inclusion maximal under the previous two conditions

The preferred arguments are those that are in the intersection of the set  $E_1, \dots, E_n$  of acceptable subsets of  $S$ . For example, suppose we have another argument  $A3 = \{gp, gp \rightarrow f\} : f$  where  $gp$  denotes that ‘Tweety is a genetically modified penguin’. On the basis of the specificity principle,  $A3$  defeats  $A2$  and  $A2$  defeats  $A1$ . Hence  $\{A1, A3\}$  is the single acceptable subset of  $\{A1, A2, A3\}$ , and so  $A1$  and  $A3$  are the preferred arguments.

In a decision making context, evaluating the arguments with preferred status establishes the preferred claims, i.e., the preferred decision candidates. However, as will become apparent in the following sections, a more comprehensive model of argumentation based decision making requires extensions to the basic Dung approach.

There is a growing body of research addressing the uses of argument in multi-agent dialogues ([12]), whereby the contents of an agent’s locution (a claim) is subject to challenge and counter-argument from another participating agent. In response to a challenge, an agent must communicate its supporting argument for a claim. In turn, each component of the supporting argument is itself considered as a claim that can be subjected to challenge and counter-argument. Research has primarily focussed on argumentation based persuasion and negotiation dialogues. These are categorised in Walton and Krabbe’s typology [14], along with enquiry, information seeking and deliberation dialogues. Deliberation dialogues involve reasoning about the appropriate course or courses of action for a group to undertake.

### 3 Agents that Argue

Recent works address argumentation based decision making over goals and actions (e.g., [2, 7]) in an agent setting. However, integration of argumentation with agent architectures is a relatively nascent area of research. As a first step in this direction, we propose the DOMINO model [4] of agent control shown in figure 1.

The DOMINO model extends the Belief-Desire-Intention model [9] to accommodate argumentation. The model can be thought of as depicting the elements of a process in which an agent can respond reactively and purposefully to situations and events. Given a set of *situation beliefs* representing the current state of the environment, the agent infers (1) a problem goal. Realisation of the problem goal will be achieved by updating the situation beliefs into line with the desires encoded in the problem goal. In the first case, the problem goal may encode a

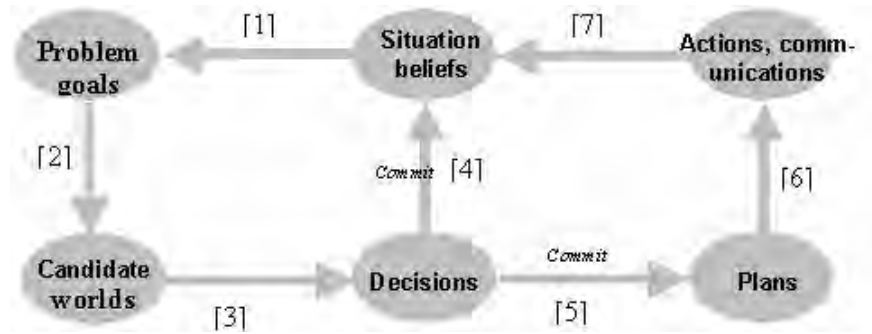


Fig. 1. The DOMINO agent model

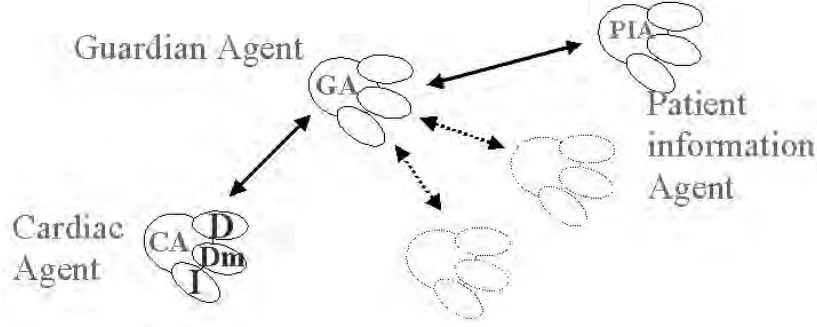
desire to update the agent's beliefs with some new belief obtained by inference from existing beliefs or through inquiry. For example, if the agent's situation beliefs are updated with a patient's clinical symptoms, then this may raise the goal 'diagnose symptoms'. Arguments for candidate new beliefs (e.g., diagnoses) are inferred (2), and argumentation based decision making (3) determines the preferred candidate (diagnosis) which is subsequently committed (4) to the situation beliefs. Of course, if insufficient information is available to infer arguments and/or decide a preference amongst them, an information seeking action is committed to (5), executed (6)<sup>3</sup>, resulting in an update to the situation beliefs (7). Decision making to determine a preferred candidate can then proceed as above. In the second case, the problem goal may encode a desire to realise an intention. For example, if the agent's situation beliefs are updated with the diagnosis 'heart attack', then this may raise the goal 'treat heart attack'. Arguments for candidate actions are inferred (2), and argumentation based decision making (3) determines the preferred candidate which is subsequently committed to (5), executed (6), resulting in a change to the external environment, so updating the situation beliefs (7).

The DOMINO model is partly implemented by the *PROforma* technology [4] which has been successfully used to build a number of medical argumentation based decision support systems ([www.acl.icnet.uk](http://www.acl.icnet.uk)). As part of the ASPIC project, *PROforma* is currently being extended to an agent technology implementing the DOMINO model. The aim is to embed ASPIC built components for inference of arguments, and argumentation based decision making and dialogue, to facilitate deployment of *PROforma* agents in medical multi-agent systems. In what follows, we describe the Guardian agent scenario that illustrates deployment of such agents.

<sup>3</sup> This illustrates the way in which an information seeking dialogue might be initiated from the actions/communications node

## 4 The Guardian Agent Scenario

In a medical MAS, specialist SMDAs are responsible for recommending a diagnosis given a set of patient symptoms, and a course of action given a treatment goal. Examples of SMDAs include agents specialised for reasoning about the cardiac domain, gastrointestinal domain etc. Overlooking the activities of all these agents is a Guardian agent (GA). The GA has specialised knowledge for reasoning about safety, and access to patient information agent mediated by a patient information agent. The GA monitors hazards arising during execution of medical actions, and, if appropriate, initiates remedial courses of action. The GA also monitors courses of action planned by the SMDAs, warns of hazards that may arise, and deliberates with the SMDAs to decide upon a safe course of action.



**Fig. 2.** Medical MAS - each agent has embedded components for argument inference (I) and argumentation based dialogue (D) and decision making (Dm)

In what follows, we describe a scenario involving argument inference, argumentation based decision making by both the cardiac and guardian agent, and an argumentation based deliberation dialogue involving both agents. We discuss the theoretical challenges raised by the scenario, and give an overview of current work addressing these challenges. We assume that the knowledge base of the cardiac agent is encoded as a logic program consisting of first order facts and rules of the form  $p_n(\bar{X}_n) \leftarrow p_1(\bar{X}_1) \wedge \dots \wedge p_k(\bar{X}_k) \wedge \text{not } p_{k+1}(\bar{X}_{k+1}) \wedge \dots \wedge \text{not } p_n(\bar{X}_n)$ , where for  $i = 1 \dots n$ ,  $p_i$  is a predicate name,  $\bar{X}_i$  is a tuple of variables and/or constants, and *not* denotes negation as failure. We assume the knowledge base of the GA is encoded as a second order logic program consisting of facts and rules of the form  $q_n(\bar{Y}_n) \leftarrow q_1(\bar{Y}_1) \wedge \dots \wedge q_k(\bar{Y}_k) \wedge \text{not } q_{k+1}(\bar{Y}_{k+1}) \wedge \dots \wedge \text{not } q_n(\bar{Y}_n)$ , where for  $i = 1 \dots n$ ,  $\bar{Y}_i$  is a tuple of variables and/or constants and/or first order predicates of the form  $p(\bar{X})$ .

## 4.1 The cardiac agent

**4.1.1 Argumentation over beliefs** Let us suppose that the situation beliefs of the cardiac agent (CA) is updated with the clinical symptoms and test results of a particular patient. The goal *diagnose(patient\_condition)* is raised. The CA has specialised medical knowledge about the cardiac domain, and makes use of this knowledge to infer arguments for and against different diagnoses. The arguments are evaluated to decide the preferred decision candidate *myocardial\_infarct* (heart attack).

The differing diagnoses represent candidate beliefs. Inference of arguments for beliefs is based on the *Logic of Argumentation* (LA) [8]. LA defines a natural deduction proof theory for inferring arguments of the form  $(Claim, Warrant, Qualifier)$ , where the *Warrant* is the set of beliefs from which the claim is derived, and the *Qualifier* is the degree of confidence warranted by the argument in the claim. The *Qualifiers* are taken from a variety of dictionaries of numerical coefficients (e.g., confidence measures), symbolic qualifiers (e.g., +, ++), linguistic qualifiers, etc.

Argument evaluation to determine the preferred diagnosis is based on an approach defined for evaluating the status of LA inferred arguments. All arguments for a given claim  $p$  are aggregated to give an aggregated argument for  $p$ . Qualifier specific aggregation functions determine an overall confidence for  $p$  (e.g., if  $A_1 \dots A_n$  are the arguments for  $p$ , where  $Q_1 \dots Q_n$  are the respective confidence measures, then using Bernoulli's rule, the confidence measure for the aggregated argument for  $p$  is  $(Q_1 + \dots + Q_n) - (Q_1 \times \dots \times Q_n)$ ). Aggregated arguments for competing claims are then "flattened" to decide the preferred claim (e.g., if the aggregated confidence measure for  $p$  is greater than the aggregated confidence measure for the competing claim  $\neg p$  then  $p$  is the preferred candidate).

The above model of inference and evaluation has been successfully used in a number of medical decision support systems [4]. More recently, we have implemented an ASPIC component for inference of LA arguments. The Dung model of evaluation is to be implemented as the core argument evaluation component. However, Dung does not account for argument aggregation. This has led to current work on extending Dung to accommodate aggregation.

**4.1.2 Arguments over actions** To continue with the scenario, the preferred candidate belief is updated to the situation beliefs, and thus the goal *treat(myocardial\_infarct)* is raised. Arguments for alternative courses of action are inferred, and these arguments are evaluated to decide the preferred decision candidate *administer(aspirin)*.

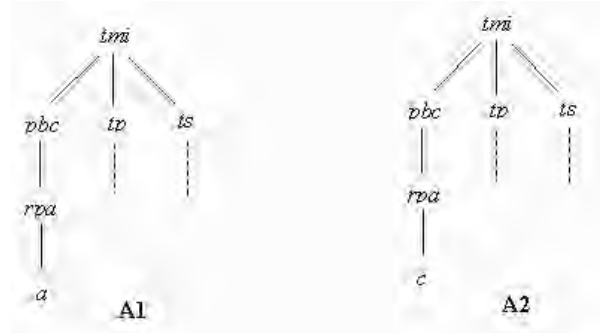
Argumentation over action differs from argumentation over beliefs in that arguments for the former do not so much appeal to 'the truth of the matter' as to the nature of the action's consequences, i.e., the effects of an action, the goals realised by these effects and the utility (be it value based, economic e.t.c) that these actions have in the world. In [10] we formalise inference of arguments for action in an agent context (building on work by Amgoud [2]) and evaluation of such arguments. Briefly, an agent's belief base contains rules representing

causal reasoning from actions to effects, effects to the goals these effects realise, and sub-goals to goals. An argument for a course of actions is then inferred via backward reasoning from the top level goal to be realised, to the actions that realise these goals. These arguments can be represented as AND trees whose root nodes represent the top level goal, and whose leaf nodes are actions.

In the case of the cardiac agent, its belief base contains the following rules relating subgoals to goals (1), effects to the goals these effects realise (2) and actions to their effects (3 and 4):

- (1)  $treat(myocardial\_infarct) \leftarrow treat(pain) \wedge treat(sickness) \wedge prevent(blood\_clotting)$
- (2)  $prevent(blood\_clotting) \leftarrow reduce(platelet\_adhesion)$
- (3)  $reduce(platelet\_adhesion) \leftarrow administer(aspirin)$
- (4)  $reduce(platelet\_adhesion) \leftarrow administer(chlopidogrel)$

On the basis of these rules, arguments for courses of action are inferred. Figure 3 shows two such arguments (ignoring the subtrees indicating actions for treatment of pain and sickness which we assume to be the same in both cases):



**Fig. 3.** Arguments A1 and A2 for alternative courses of action, where: tmi = treat(myocardial\_infarct), tp = treat(pain), ts = treat(sickness), pbc = prevent(blood\_clotting), rpa = reduce(platelet\_adhesion), a = administer(aspirin), c = administer(chlopidogrel)

We now describe evaluation of these arguments. Firstly, A1 and A2 represent alternative courses of action, and so attack each other. To determine which defeats the other requires that we value and thus elicit a preference amongst these arguments on the basis of some criterion. In [10] we show that while an argument A may defeat A' according to some criterion C, it may well be that A' defeats A according to some other criterion C'. One then needs to argue over the relative importance of the criteria to resolve the symmetrical defeat. Hence, in [10] we define a framework formalising 'meta-argumentation' over the

grounds by which one argument defeats another. In our running example, one might have an argument  $B1$  stating that  $A1$  defeats  $A2$  on the basis of the cost criterion, and an argument  $B2$  stating that  $A2$  defeats  $A1$  on the basis of the efficacy criterion. Hence, to determine whether  $A2$  or  $A1$  is preferred will require constructing some argument  $C$  justifying either  $B1$ 's defeat of  $B2$  (in which case  $A1$  will be preferred) or  $B2$ 's defeat of  $B1$  (in which case  $A2$  will be preferred). Such an argument  $C$  will be based on some preference ranking on the criteria. However, let us suppose that aspirin is both more efficacious (w.r.t the treatment goal) and cheaper than chlopidogrel, so that there exist two arguments  $B1$  and  $B2$ , each of which represents an argument for the claim that  $A1$  defeats  $A2$ . Hence,  $A1$  is Dung preferred.

## 4.2 The Guardian Agent

The Guardian agent is equipped with specialist knowledge for reasoning about safety. This knowledge includes generic safety constraints and specific knowledge about medical safety. An example of the former (taken from a list of generic safety principles reported on in [11]) is given below in the standard logic database representation of integrity constraints:

$\leftarrow \text{state\_property}(S, P), \text{planned\_action}(Ac1, G), \text{effect}(Ac1, E), \text{hazardous}(P, E, H)$

The constraint should be read as a requirement that the predicates to the right of “ $\leftarrow$ ” must not all hold for some ground instantiation of the variables. In particular, that it should not be the case that a planned action ( $Ac1$ ) with goal  $G$ , has an effect ( $E$ ) that in combination with some state property ( $P$ ) results in some hazard ( $H$ ). Associated with this constraint are three arguments of the form ( $Support, Claim$ ) where the claims represent alternative strategies for revising a plan to enforce compliance with the constraint:

$A3 = (\{\text{planned\_action}(Ac1, G), \text{effect}(Ac1, E), \text{state\_property}(S, P), \text{hazardous}(P, E, H), \text{not\_necessary}(G)\}, \text{prohibit\_action}(Ac1))$

$A4 = (\{\text{planned\_action}(Ac1, G), \text{effect}(Ac1, E), \text{state\_property}(S, P), \text{hazardous}(P, E, H)\}, \text{choose\_alternative\_action}(Ac1))$

$A5 = (\{\text{planned\_action}(Ac1, G), \text{effect}(Ac1, E), \text{state\_property}(S, P), \text{hazardous}(P, E, H), \text{action}(A2), \text{ameliorate}(Ac2, E)\}, \text{add\_amelioration\_action}(Ac2))$

Note that  $A3$  encodes the default assumption that the goal of the planned action is not a necessary goal. This may well be the case if  $G$  is one among a disjunction of subgoals for realising some goal  $G'$ . For example, the two rules  $G' \leftarrow G, G' \leftarrow G''$  of the type described in section 4.1.2, express that  $G'$  can be realised by subgoal  $G$  or  $G''$ .

As for evaluation of these arguments, note that each argument attacks the other two since the claims represent alternative strategies for revising a plan of

action. However,  $A3$  defeats  $A4$  on the basis that  $A3$  is a more minimal plan revision than  $A4$ , and both  $A3$  and  $A4$  defeat  $A5$  on the basis that the latter only ameliorates the hazard rather than prevents the hazard from arising. Hence  $A3$  is the Dung preferred argument.

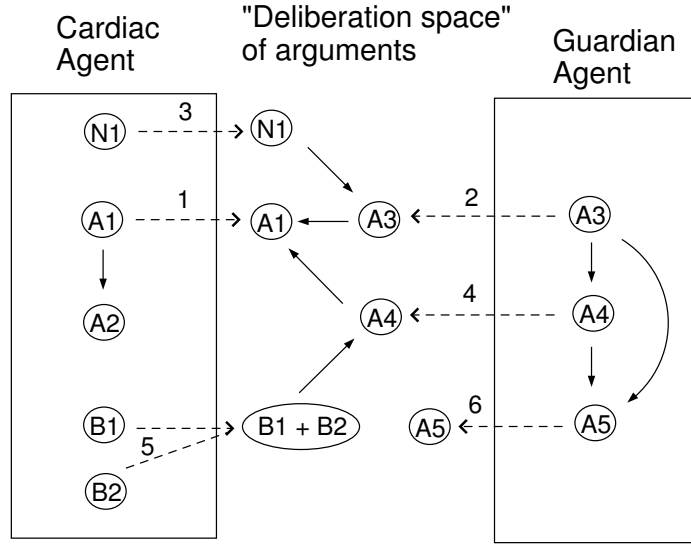
### 4.3 Deliberating over a safe course of action

We have described inference and evaluation of arguments for both the cardiac and guardian agents. We now describe how these agents communicate arguments in a deliberation dialogue to determine a safe course of action.

Currently, no models exist for the conduct of deliberation dialogues. As a first step, Hithcock et.al. [5] propose a generic framework in which to develop such models. In [5], deliberation is described as involving (among other things) communication of goals, proposals for plans of action for realizing goals, and the reasoning underlying how preferences between competing action proposals are judged. In section 4.1.2 we briefly described our current work on inferring arguments for proposals for action, and formalising argumentation over the justifications for why one action argument is preferred to (defeats) another. We believe that this work can form a basis for agent participation in deliberation dialogues.

In what follows we describe a deliberation dialogue between the cardiac and guardian agent, in which each time the CA communicates a claim, the GA challenges the claim, which in turn requires that the CA communicate the argument's support for the claim. We assume the same for any claim communicated by the GA. We can therefore abstract from these individual communications to communication of arguments. For example, when stating that the CA communicates the argument  $A1$ , this corresponds to the CA's claim *administer(asprin)*, followed by a challenge from the GA, followed by the CA communicating the whole of the argument  $A1$ .

Figure 4 shows the CA's arguments  $A1$  and  $A2$  for alternative courses of action *administer(asprin)* and *administer(chlopidogrel)* respectively. Also included are arguments  $B1$  and  $B2$  justifying  $A1$ 's defeat of  $A2$  on the grounds of efficacy and cost respectively. The argument  $N1$  expresses that the goal *prevent(blood clotting)* is a necessary goal for realisation of the top level goal *treat(myocardial infarct)* (i.e., there is no alternative subgoal for realisation of the top level goal). The Guardian agent's plan revision arguments  $A3, A4$  and  $A5$  are also shown. The defeat relations between arguments are indicated by the arrows with unbroken lines. At each stage of the dialogue, each agent can choose to communicate one of its Dung preferred arguments. At the outset, CA's preferred arguments are  $A1, N1, B1$  and  $B2$ , and GA's preferred argument is  $A4$ . These preferences change as a result of arguments communicated by the other agent. At the end of the dialogue it is the preferred arguments in the "deliberation space" that represent the agreed course of action. The deliberation proceeds as follows (as indicated by the numbered dashed arrows):



**Fig. 4.** Both the cardiac and guardian agents' arguments are shown. Also shown are the arguments communicated in the deliberation dialogue (the "deliberation space")

1) CA communicates the goal *prevent(blood\_clotting)* and its preferred argument *A1* for *administer(aspirin)*.

2) The GA has access to the information agent for the patient in question, and based on the state's (patient's) property (history of gastritis), and knowledge that aspirin has an acidic effect that in combination with a history of gastritis can cause the hazard of gastric bleeding, instantiates the argument *A3* for the claim *prohibit\_action(administer(aspirin))*. Recall that this argument is also based on the assumption that the goal is not a necessary one. The safety argument *A3* attacks *A1*, and defeats *A1* given that the safety criterion is ranked the highest. Notice that the CA's Dung preferred arguments are now *A2*, *N1*, *B1* and *B2*.

3) CA communicates *N1* which undercuts and so defeats *A3* (by overriding the default assumption that the goal is not necessary). Hence the GA's Dung preferred argument is now *A4*. Also, *A1* is now a CA preferred argument again (*N1* effectively reinstates *A1* by defeating the defeater *A3* of *A1*)

4) GA communicates its preferred argument *A4*. This argument's claim proposes that the CA choose an alternative action to *administer(aspirin)*, and so *A4* attacks *A1*, and defeats *A1* given the safety criterion's highest ranking. Now



the CA's preferred argument is *A2* again (as well as *B1* and *B2*).

5) The CA can now communicate its preferred argument *A2* for *administer* (*chlopidogrel*), or the two arguments *B1* and *B2*, each of which express that *A1* defeats *A2* and the reasons for the defeat. The CA chooses to communicate the aggregation of the latter two arguments, given that their aggregation has sufficient force (aspirin is preferred to chlopidogrel both on efficacy **and** cost grounds) to defeat the GA's argument *A4* for choosing an alternative. Thus, *A1* is reinstated as the CA's preferred argument, and the GA's preferred argument is now *A5*.

6) The GA communicates *A5* which proposes inclusion of an ameliorating action *administer*(*proton\_pump\_inhibitor*) which reduces the acidity resulting from administration of aspirin. At this stage no further relevant arguments are available. Thus the CA and GA terminate the dialogue, agreeing on the course of action as represented by the claims of the preferred arguments *A1* and *A5* in the deliberation space; i.e., to administer aspirin and a proton pump inhibitor.

## 5 Conclusions

In this paper we have described deployment of a Guardian agent in medical multi-agent systems. This agent deliberates cooperatively with other specialist medical agents to determine a safe course of action. We have presented an overview of existing work and current research into inference of arguments and argumentation based decision making by both the specialist agents and the guardian agent, and argumentation based deliberation dialogues between these agents. This programme of research is being conducted as part of an EU project - ASPIC - aiming at development of theoretical models of argumentation, and implementation of these models in software components for deployment in agent technologies. The eventual aim is that ASPIC technology will enable deployment of Guardian agents in medical systems. To summarise, we have described:

- the DOMINO model as a starting point for integration of argumentation based reasoning with agent architectures
- inference of arguments for beliefs and aggregation based evaluation of these arguments to decide a preferred candidate belief.
- inference of arguments for actions, and argumentation over the justification for one action argument's defeat of another, so as to facilitate decision making as to the preferred course of action.
- a Guardian agent's arguments for alternative plan revision strategies for enforcing compliance with generic safety constraints.
- deliberation between specialist medical agents and the Guardian agent so as to cooperatively decide a safe course of action. This involves communication of agents' arguments whereby the decision as to the choice of which argument to communicate is in part based on the agent's preferred arguments at the given stage in the dialogue. The arguments communicated include arguments

for action, arguments for plan revision, and arguments justifying why one course of action is preferred to another. Once the agents agree to terminate the dialogue, the communicated arguments are evaluated to determine the Dung preferred argument, and hence the agreed safe course of action.

The above research programme is still in its early stages. Further theoretical work remains to be done before executable algorithms can be specified and implemented in argumentation components for deployment in agent technologies. The Guardian agent scenario serves to facilitate this ‘pipeline’ from theory to technology, by providing a focus for both theoretical work and prototyping work. Indeed we are currently prototyping the Guardian agent scenario in the logic programming based COGENT environment [6].

**Acknowledgements** This work was funded by the European Commission’s Information Society Technologies programme, under the IST-FP6-002307 ASPIC project.

## References

1. T. Alsinet, C. Anstegui, R. Bjar, C. Fernndez, and F. Many. Automated monitoring of medical protocols: a secure and distributed architecture. *rtificial Intelligence in Medicine*, 27(3):367–392, 2003.
2. L. Amgoud. A formal framework for handling conflicting desires. In *Proc. 7th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU’2003)*, pages 552–563, 2003.
3. P. M. Dung. On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and  $n$ -person games. *Artificial Intelligence*, 77:321–357, 1995.
4. J. Fox and S. Das. *Safe and Sound. Artificial Intelligence in Hazardous Applications*. AAAI Press, The MIT Press, 2000.
5. D. Hitchcock, P. McBurney, and S. Parsons. A framework for deliberation dialogues. In H. V. Hansen, C. W. Tindale, J. A. Blair, and R. H. Johnson, editors, *Proceedings of the Fourth Biennial Conference of the Ontario Society for the Study of Argumentation (OSSA 2001)*, Windsor, Ontario, Canada, 2001.
6. R. P. Cooper (includes contributions from P. Yule, J. Fox, and D. W. Glasspool). *Modelling High-Level Cognitive Processes*. Lawrence Erlbaum Associates, 2002.
7. Antonis Kakas and Pavlos Moraitis. Argumentation based decision making for autonomous agents. In *Proc. Second international joint conference on Autonomous agents and multiagent systems*, pages 883–890. ACM Press, 2003.
8. P. Krause, S. Ambler, M. Elvang-Gøransson, and J. Fox. A logic of argumentation for reasoning under uncertainty. *Computational Intelligence*, 11(1):113–131, 1995.
9. M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge. The belief-desire-intention model of agency. In *Proceedings of Agents, Theories, Architectures and Languages (ATAL)*.
10. S. Modgil. Fractal argumentation and its application to decision making over actions. Submitted to the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 2005.

11. S. Modgil and P. Hammond. Generating symbolic and natural language partial solutions for inclusion in medical plans. In Andreassen Quaglini, Barahona, editor, *Proc. 8th Conference on Artificial Intelligence in Medicine in Europe, AIME2001*, pages 239–248, Cascais Portugal, 2001. LNAI 2101.
12. T. J. Norman, D. V. Carbogim, E. C. W. Krabbe, and D. Walton. Argument and multi-agent systems, chapter 2. In C. Reed and T. J. Norman, editors, *Argumentation machines: New frontiers in argument and computation*. Kluwer Academic Publishers, 2004.
13. D. Riao, A. Moreno, and A. Valls. Palliasys: Agent based palliative care. In C. Ghidini, P. Giorgini, and W. van der Hoek, editors, *IEEE 4th Conf. on Intelligent Systems Design and Applications (ISDA'04)*, Budapest, Hungary, 2004. EUMAS.
14. D. N. Walton and E. C. W. Krabbe. *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. SUNY Series in Logic and Language. State University of New York Press, Albany, NY, USA, 1995.

# Towards Safe Coordination in Multi-agent Systems

Anita Raja<sup>1</sup>, Michael Barley<sup>2</sup>, and Xiaoqin Shelley Zhang<sup>3</sup>

<sup>1</sup> Department of Software and Information Systems, The University of North Carolina at Charlotte, Charlotte, NC 28223, anraja@uncc.edu

<sup>2</sup> Department of Computer Science, University of Auckland. Auckland, NZ  
mbar098@cs.auckland.ac.nz

<sup>3</sup> Department of Computer Science, The University of Massachusetts Dartmouth, North Dartmouth, MA 02747, x2zhang@umassd.edu

**Abstract.** Conservative design is the ability of an individual agent to ensure predictability of its overall performance even if some of its actions and interactions may be inherently less predictable or even completely unpredictable. In this paper, we describe the importance of conservative design in cooperative multi-agent systems and briefly characterize the challenges that need to be addressed to achieve this goal.

## 1 Introduction

Uncertainty is ubiquitous in complex MAS operating in open environments. A safe multi-agent system is composed of agents that are equipped with "conservative design" [2] capabilities. We define conservative design in cooperative multi-agent systems as the ability of an individual agent to ensure predictability of its overall performance even if some of its actions and interactions may be inherently less predictable or even completely unpredictable. An essential feature of conservative design is the ability of agents to efficiently handle risk. Risk is the potential for realizing unwanted negative consequences of an event [5].

Establishing environmental predictability and enforcing risk management measures are first-class goals of real-world organizations. Corporations are motivated by the need to maintain their reputations while maximizing profit. Reputation can be a precious commodity and corporations can charge more for their products because they have a reputation, say, for reliability. When corporations depend on other organizations that are less reliable than themselves, they must come up with plans that enable them to guarantee that they will still live up to their reputations. In practice, the customers of a reputable corporation assume that the corporation has accounted for the risk of its suppliers failing to deliver within the given constraints, has made contingency plans, and will deliver the product as agreed upon.

In this paper, we describe this concept of conservative design in multi-agent systems specifically as it relates to coordination with other agents. We identify a number of challenge areas along with examples that have to be addressed to achieve this goal.

## 2 Conservative Design

Predictability of both the environment and of the outcomes of agent performance are essential for building safe systems. One way of ensuring conservative design in agents is to identify levels of risk in an agent's goals and to incorporate reasoning about risk into agent planning. More specifically, an important challenge problem for safe coordination in multi-agent systems is the ability to transform an unpredictable and highly complex world into a simpler world where even if the actions are not entirely deterministic, at least there are predictable bounds on the probabilities of their outcomes. The following is an example of a multi-agent application that motivates the need for conservative design. We also identify the questions that need to be addressed by the multi-agent system.

Consider the supply chain scenario described in Figure 1. Each agent represents a company/organization involved in the supply chain. The *PCManufacturer*, the two Chip Producers (*ChipProducer1* and *ChipProducer2*) and the two Transporters (*TransporterA* and *TransporterB*) are represented by individual agents. The *PCManufacturer* makes a commitment to the customer to deliver the product within a deadline  $DL$ . This means the *PCManufacturer* has to negotiate commitments with one of the two Chip Producers and one of the two Transporters such that there is enough time to assemble and deliver the final product to the customer before the deadline  $DL$ . It may be the case that *PCManufacturer* may have to take some risky actions to ensure completing the goal within the deadline. Agent *PCManufacturer* has the achievement goal of satisfying the customer's request within the given constraints. Its maintenance goal could be to keep the costs low and allow for risk within a certain threshold while handling the customer's request.

Suppose the *ChipProducer1* agent is unable to keep its initial commitment to the *PCManufacturer* agent. *PCManufacturer* may choose to re-negotiate a new commitment instead. The re-negotiation is caused by the local dynamics of *ChipProducer1*, and it may result in a change of the content of the existing commitment, such as the quality or the quantity of the parts, the cost, or the delivery time of the parts [8]. Resources invested in searching for alternate commitments in real-time could result in high costs. The *PCManufacturer* agent could avoid these costs by estimating the probability that *ChipProducer1* will need to re-negotiate its commitment and include this information in the decision process while choosing the Chip Producer agent. Furthermore, if the probability of re-negotiation by *ChipProducer1* is within an acceptable threshold, the costs of online re-negotiation can be kept low by identifying contingent commitments. The following are some questions that will help determine the risk of choosing *ChipProducer1* as the supplier:

1. How trustworthy is the agent based on its reputation? How likely is it to drop its commitment?
2. Is it useful to enforce a decommitment penalty for this particular sub-contract?
3. How likely is it for the agent to complete the task within the expected performance profile?

**Fig. 1.** A Supply Chain Scenario

4. Should payment for product be a function of the performance profile of end product delivered?
5. Are there enough resources to establish commitments with both *ChipProducer1* and *ChipProducer2*? This will allow *PCManufacturer* to store one of the chips while using the other in case both agents deliver on schedule. This reduces the risk of not having a chip available by the delivery deadline.

Determination of risk enables a multi-agent system to identify predictable and less-predictable parts of its solution space. The multi-agent system, depending on its goals and its environment, may have to spend resources in ensuring that the less-predictable parts of the solution space are made more predictable in order to provide performance guarantees. We now describe some factors that will facilitate risk handling and leverage the uncertainty in multi-agent environments.

## 2.1 Identifying Risk Thresholds

One way that corporations enforce risk management is by establishing legal thresholds on the amount of risk that they can assume or are willing to tolerate. For instance, the boards of directors of a company cannot act "irresponsibly" or "recklessly". They also may have thresholds on the risk associated with their products. The Federal Aviation Agency (FAA) may set risk thresholds on the products they allow for airplane parts manufacturers. An example of such a risk threshold could be the average number of air miles flown between maintenance checks. The plane would not be allowed on the market until it has been shown

to have a risk lower than the legal threshold. The thresholds can be fuzzy and may depend on a number of variables. Yet they are useful measures that help achieve a entity's overall goal which is to make its environment deterministic, so that planning and scheduling can be made simpler.

Our goal in this work is to build agents that want to maintain their reputation within a multi-agent system while minimizing the cost of ensuring predictability of the environment. The cost of dealing with the unwanted consequences of an event is called *risk impact*. For instance, if the power goes down in a manufacturing company for a number of hours, there is an impact on the productivity for that day. Predictability of the environment involves pre-computing and controlling risk impact for different situations. Suppose risk is defined by  $m$  attributes and  $x_i$  is one such attribute contributing to risk;  $p_i$  is the probability of this attribute occurring;  $imp_i$  is the impact (cost) of the attribute  $x_i$ . Then,

$$Risk = \sum_{i=1}^m p_i * imp_i$$

It is possible to extend this function to contexts where there is uncertainty in the impact. In other words, if a particular risk attribute occurs, various impact outcomes are possible depending on the current environmental state. For example, if the power goes down at 10 a.m. as opposed to 10 p.m., the resulting impact would be very different. Suppose there are  $n$  possible impacts for each attribute  $x_i$  and each impact  $p_{ij}$  occurs with probability  $imp_{ij}$ , then

$$Risk = \sum_{i=1}^m \sum_{j=1}^n p_{ij} * imp_{ij}$$

A risk threshold  $\tau$  may have to be determined in a situation specific fashion and conservative design means that the agents have the following maintenance goal:

$$Risk < \tau$$

The following are examples of risk attributes in the supply chain scenario described earlier:

1. Chip Producer agent delivers product  $x$  units after deadline  $d1$  established in a commitment.
2. Transport agent delivers product  $y$  units after deadline  $d2$  established in a commitment.
3. Storage costs of redundant orders result in higher total costs, thereby lowering total profit.

Consider a situation where *TransporterA* promises to deliver the product by time 10 with cost 20, however there is a probability 0.4 it may be delayed to time 20. *TransporterB* promises to deliver by time 10 with cost 30 and with probability 0.05 it may be delayed to time 12. The following are some issues that affect *PCManufacturer's* decision process:

1. The risk threshold for *PCManufacturer* could be a function of the tightness of the deadline, the cost constraints and the importance of maintaining its reputation. Suppose the price negotiated with the customer for the final product is 60. If the deadline is sufficiently beyond time 12, say time 25, *PCManufacturer* would choose *TransporterA* and pay 20 cost units. If the deadline is 18, then *PCManufacturer* would be willing to pay the higher cost of 30 cost units to have *TransporterB* deliver the Chip with 100% guarantee by time 12.
2. It is also possible for *PCManufacturer* to look at the possible product delivery times and try to move the schedule a little ahead. For instance, the agent could anticipate a potential product request, initiate negotiation to complete the product early and pay for storage. This will reduce the risk threshold of failure to have the product part on time but does increase the cost by paying for storage.

Identifying risk attributes and risk thresholds are important problems that need to be addressed to accurately handle uncertainty.

## 2.2 Using Risk to Prune Solution Space

There are multiple choices in an agent's coordination process including with which agent to cooperate, when to cooperate and how to cooperate. When an agent has to coordinate with several other agents, the order of the coordination is also an important factor. An agent uses its performance criteria to make its coordination choices. The performance criteria is defined by multiple factors including risk threshold, utility and cost. In order to maintain its reputation and reliability, the agent could use risk as a factor to prune the search space. In other words, if a coordination solution carries a risk measure higher than the risk threshold acceptable by that the agent, this solution will not be considered as a candidate. After deleting all solutions with risk beyond the threshold, the agent can evaluate the rest of the solutions using a utility/cost measure, meaning, the agent can select the solution with the highest utility/lowest cost. Another approach to evaluate those solutions is to use a multi-dimensional weighted function that combines both the utility and the risk measurement. For example, given the following performance measures of three solution options:

$$\text{Solution1 : utility} = 100, \text{risk} = 10$$

$$\text{Solution2 : utility} = 120, \text{risk} = 15$$

$$\text{Solution3 : utility} = 145, \text{risk} = 25$$

Suppose the acceptable risk threshold is 20, Solutions 1 and 2 are valid candidates, while Solution 3 is eliminated. The agent can then use a multi-dimensional function to evaluate the remaining two solutions. The following is an example of a possible multi-dimensional evaluation function:

$$F = w1 * \text{utility} - w2 * \text{risk}$$



where  $w_1$  and  $w_2$  represent the importance of utility and risk are to the agent. This simple weighted function would allow the agent to balance utility and risk in its decision-making process. If  $w_2$  is higher than  $w_1$ , the agent tries to find the solution with the highest utility while minimizing risk.

Thus we conjecture that an agent can use risk to reduce its problem solving costs. It has to first enumerate all possible solutions, use its risk threshold to prune the solution space, and then evaluate the remaining solutions using the utility measure or a multi-dimensional function that combines utility and risk.

### 2.3 Contingencies in Coordination

In mission-critical environments, failure can lead to catastrophic consequences. This means that it is not sufficient for an agent to minimize risk in coordination. For instance, the uncertainty in agent outcomes and environment characteristics could necessitate dropping previous commitments and establishing new commitments [7]. However, online re-coordination can be prohibitively expensive. While negotiating, it will be useful for agents both to determine the probability of a commitment completing successfully and to identify contingency commitment contracts in case of commitment failure. There has been significant previous work in studying contingencies for planning and scheduling [1, 3, 4]. We are interested in extending some of these ideas to coordination.

Consider an example where agent *A* should complete task *T1* by deadline 80. Suppose Task *T1* can be achieved by two alternate ways *P1* and *P2* where *P1* is a high risk solution with high utility while *P2* is a low risk solution with low utility. Specifically, *P1* requires that agent *B* completes an enabling action *M1*. However agent *B* is usually heavily loaded and has the reputation of dropping its commitments regularly.

One way of handling this uncertainty would be for agent *A* to establish a contingent commitment that consists of two parts: the details of a commitment along with time of completing the contracted task, and a time at which the contractor agent will confirm the commitment. This confirmation time will lie between the time of establishing the commitment and the time for completing the commitment.

In the above example, suppose the current time is 12 and a commitment between agent *A* and agent *B* is feasible with a probability of 0.4. The contingent commitment would be as follows: Agent *B* commits to complete action *M1* by time 43; and will confirm the commitment at time 25. Agent *A* will choose plan *P2* as the contingent plan. In the case that agent *B* confirms the commitment positively at time 25, agent *A* will continue with Plan *P1*. If on the other hand, agent *B* states that it has to drop its commitment, then agent *A* will resort to plan *P2*. Additionally, there will be a high penalty for agent *B* if it confirms commitment at time 25 but fails to complete action *M1* by time 43.

An alternative way of handling uncertainty would be leveled commitment contracts [6]. These are contracts where each party can decommit by paying a predetermined penalty. This would allow agents to accommodate events which

unfolded since the contract was entered into while allowing the other agents to use the penalty to make alternate plans.

A critical question is to determine how much contingency planning for coordination is enough? The contingent plans themselves may require contingencies in case of failure. The challenge is then to design agents that are equipped with the ability to do cost-benefit tradeoffs for the depth of contingency in coordination.

### 3 Conclusions

In this paper, we defined the concept of conservative design in multi-agent systems. This is based on the premise that safety, like security, should not be an after-thought but an integral part of the design of the agent's decision making capabilities. We also identified factors that contribute to analyzing and responding to risk in agent environments. As future work, we plan to formalize the representation of risk using the supply chain scenario as an example. Then we plan to implement some of the reasoning processes described in this paper. We see this as first step towards conservative design in multi-agent systems.

### References

1. Draper, D., Hanks, S., Weld, D.: Probabilistic planning with information gathering and contingent execution. In: Proceedings of the Second International Conference on Artificial Intelligence Planning Systems (AIPS-94). (1994) 31–36.
2. Mead, C., Conway, L.: Introduction to VLSI Systems. Addison-Wesley, Reading, MA (1980).
3. Onder, N., Pollack, M.: Contingency selection in plan generation. In: Proceedings of the Fourth European Conference on Planning. (1997).
4. Raja, A., Wagner, T., Lesser, V.: Reasoning about Uncertainty in Design-to-Criteria Scheduling. In: Working Notes of the AAAI 2000 Spring Symposium on Real-Time Systems, Stanford. (2000).
5. Rowe, W.D.: An Anatomy of Risk. Robert E. Krieger Publishing Co., Malabar, FL (1988).
6. Sandholm, T., Lesser, V.: Leveled commitment contracting: A backtracking instrument for multiagent systems. *AI Magazine* (2000) 89–100.
7. Xuan, P., Lesser, V.R.: Handling uncertainty in multi-agent commitments. Technical Report UM-CS-1999-005 (1999).
8. Zhang, X., Lesser, V., Wagner, T.: A layered approach to complex negotiations. *Web Intelligence and Agent Systems: An International Journal* **2** (2004) 91–104.

# A Distributed Numerical Approach for Managing Uncertainty in Large-Scale Multi-Agent Systems

Anita Raja<sup>1</sup> and Michael Klibanov<sup>2</sup>

<sup>1</sup> Department of Software and Information Systems, The University of North Carolina at Charlotte, Charlotte, NC 28223, anraja@uncc.edu

<sup>2</sup> Department of Mathematics, The University of North Carolina at Charlotte, Charlotte, NC 28223, mklibanv@email.uncc.edu

**Abstract.** Mathematical models of complex processes provide precise definitions of the processes and facilitate the prediction of process behavior for varying contexts. In this paper, we present a numerical method for modeling the propagation of uncertainty in a multi-agent system (MAS) and a qualitative justification for this model. We discuss how this model could help determine the effect of various types of uncertainty on different parts of the multi-agent system; facilitate the development of distributed policies for containing the uncertainty propagation to local nodes; and estimate the resource usage for such policies.

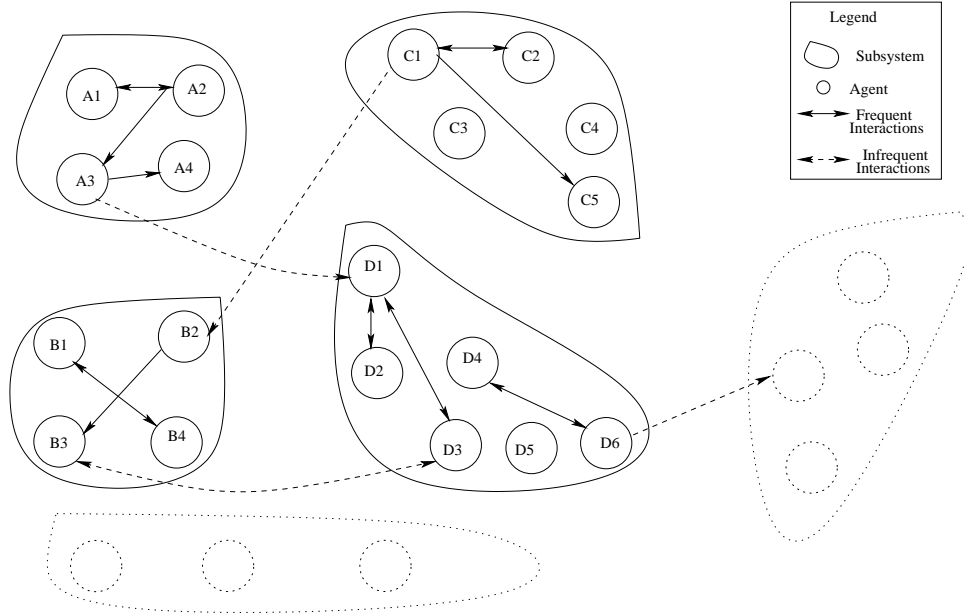
## 1 Introduction

Agents operating in open environments should be capable of handling the uncertainty in their environments in order to provide performance guarantees. This includes determining and controlling the source and propagation of the uncertainty. Mathematical models of complex processes provide precise definitions of the processes and facilitate the prediction of their behavior for varying contexts. In this paper, we present a closed-form numerical method for modeling the propagation of uncertainty in a multi-agent system (MAS) and a qualitative justification for this model. We discuss how this model could help determine the effect of various types of uncertainty on different parts of the multi-agent system; facilitate the development of distributed policies for containing the uncertainty propagation to local nodes; and estimate the resource usage for such policies.

We will first describe the characteristics of the MAS problem we are interested in addressing and then model this problem using a numerical Partial Differential Equation. The solution to this particular Partial Differential Equation (PDE) will describe the behavior of the MAS for varying number of agents and for varying amounts of time. In keeping with the philosophy behind derivations of PDEs governing physical processes, we derive the PDE using only small time intervals and only for agents with direct dependencies to other agents. The advantage of this procedure is that as soon as such a model is derived, the solution of that PDE provides the prediction of system behavior for all times (small

or large) and for different kinds of agent networks. Providing information to predict performance degradation in individual agents due to the propagation of the uncertainty will facilitate the construction and implementation of contingent plans to control the propagation of uncertainty and contain its potential negative effects on the performance characteristics of the cooperative MAS.

In order to apply the mathematical techniques to the study of propagation of uncertainty, we consider large scale multi-agent systems (MAS) that are nearly-decomposable and have hundreds of heterogeneous agents. Simon [11] defines a nearly-decomposable system as a system made up of sub-systems where the interactions between sub-systems are weak and they behave nearly independently. This implies that inter-subsystem interactions are weaker than intra-subsystem interactions. In our work, we assume that most of these interactions can be predicted at design time [5]. Another simplification is that we abstract the details of uncertainty propagation into a probability per unit time for an individual to be influenced by uncertainty. This abstraction will allow for the construction of the PDE-based model.



**Fig. 1.** Nearly-Decomposable Multi-agent System

Figure 1 describes a nearly-decomposable system that we will use to describe the issues addressed by this paper. The nearly-decomposable multi-agent system has a large number of cooperative agents. We assume that each agent is capable of performing a set of tasks. *Agent interactions* are defined as those relationships where the performance characteristics (utility, duration, cost) of a task(s) in

one agent affect the performance characteristics of task in another agent. In other words, uncertainty in the execution of a task in one agent might have adverse effects on the execution of tasks in other agents. An example of such an interaction would be an *enables* relationship [4] where successfully completing a task (or a primitive action) in one agent is a pre-condition to initiating a related task (primitive action) in another agent. In Figure 1, there are a number of agents that have frequent interactions, for instance agent sets  $\{A1, A2, A3, A4\}$ ,  $\{B1, B2, B3, B4\}$ , etc. Agents with frequent interactions are grouped loosely together into sub-systems. There are other sets of agents  $\{A3, D1\}$ ,  $\{B2, C1\}$ ,  $\{B3, D3\}$  that exhibit infrequent interactions.

There is significant previous research in reasoning about uncertainty in multi-agent systems [1, 3, 9, 12, 13]. Many of these use formal models and decision theoretic approaches to describe and reason about uncertainty in scheduling and coordination. The bottleneck in most of these approaches is that as the number of agents and interactions increase, determining the optimal solution becomes very difficult. Hence many of these approaches use approximate techniques to compute the solution. Our work is novel in that we use a linear mathematical model to reason about uncertainty and the advantage of a closed-form approach is that it scales to large MAS. This mathematical model has also been used to reason about other complex processes like heat conduction in materials [2] and propagation of viruses in computer networks [6]. Uncertainty propagation in MAS under certain assumptions is analogous to propagation of computer viruses in computer networks. In this paper, we limit ourselves to studying the uncertainty of an agent completing a task successfully where that task is a pre-condition to the task belonging to another agent. These are also called *enables* relationships [4]. Computer viruses propagate through networks causing node failures while uncertainty/failure propagates through MAS networks causing agents to fail in their tasks. We are currently working on extending this model to the propagation of different levels of uncertainty from one agent to another. An example of such an interaction would be the *facilitates* relationship [4] where if some task A facilitates a task B, then completing A before beginning work on B might cause B to take less time, or cause B to produce a result of higher quality, or both. If A is not completed before work on B is started, then B can still be completed, but might take longer or produce a lower quality result than in the previous case. In other words, completing task A is not necessary for completing task B, but it is helpful.

The paper is structured as follows: In Section 2, we present the issues involved in modeling the problem using a PDE, the derivation of a PDE model of uncertainty propagation in nearly-decomposable MAS and a qualitative analysis of the model. In Section 3, we summarize the results and discuss future work.

## 2 The Model

In this section, we describe the process of modeling the uncertainty propagation problem as a Partial Differential Equation (PDE) as well as the benefits

of using a PDE. The main difference between PDE and Ordinary Differential Equation(ODE)-based mathematical models is that the first one is distributed, whereas the second one is non-distributed. An ODE-based model is used when a certain instance is treated as a “non-divisible whole”. For example, if one wants to describe the acceleration of a car by modeling the whole car as a moving particle, one uses the ODE  $\frac{d^2x}{dt^2} = a(t)$ , where  $t$  is time,  $x(t)$  is the travel distance and  $a(t)$  is the acceleration. However, if one wants to describe processes taking place inside of certain parts of the car during its run, e.g., vibration, then one should use PDEs. Analogously, in the case of multi-agent systems, ODE-based models describe all influenced agents as a *non-divisible* subset of the entire network. The distributed PDE-based models, on the other hand, differentiate between different influenced agents.

A PDE is described as a continuous function in euclidean space. So how do we get a continuous model from a discrete set of agents and their interactions? There is a well-known procedure [2] used in the derivations of PDEs that govern physical phenomena in nature where a continuous model is obtained from a discrete model, provided that the number of discrete points is large. This assumption holds here since we consider only large-scale MAS in this work. The procedure involves uniting a subset of agents (subject to some rules)<sup>3</sup> into a set and assigning a point on the real line to this set. Next, we consider another set of agents, which are in a particular kind of relation with the first one and assign a close point on the real line to this set. Continuing this process, we obtain a grid of points with the step size  $h$  on the real line. A close analog is a grid in a finite difference scheme [7] for the numerical solution of a PDE. Next, we assume that all points in the gaps between grid points also represent similar sets of agents. This produces a continuum of points. The continuous functions in this paper can all be viewed as interpolations of corresponding discrete functions. This derivation procedure allows for a continuous model that is often easier to analyze theoretically than its discrete counterpart. It turns out that a proper theoretical analysis can often lead to optimal numerical schemes.

Although, the continuous model is more convenient for the theoretical analysis, finding solutions for continuous PDEs is computationally intractable except for trivial cases. Trivial cases are those PDEs with constant coefficients where the solutions are given by explicit formulas. So, the concept of finite differences [7] is used to solve the PDE. A main idea behind finite differences is that the derivative of a function  $U(x)$  is  $\frac{U(x+h)-U(x)}{h}$  (approximately), where  $h$  is the grid step size. Thus, the derivatives involved in a PDE can be approximated by discrete finite differences. This produces an algebraic system of equations. The solution to this algebraic system is the approximation of the exact solution of the corresponding PDE at those grid points.

In principle, if the grid step size  $h$  approaches zero, the solution of that algebraic system approaches the solution of the continuous model. This is because the derivatives are approximated more accurately as  $h$  approaches zero. But when

---

<sup>3</sup> The subsystems which are part of the nearly-decomposable MAS could provide a natural mapping to these agent sets.

h decreases, it increases the size of the algebraic system. This could potentially bring instabilities in the solution. The advantage of the continuous PDE is that it guarantees that such instabilities will not occur, specifically it does not occur for the PDE's discussed in this paper. In summary, we use a continuous model because the theory is well-defined and it helps in the derivation of suitable finite difference schemes, which are stable and the solutions of corresponding algebraic systems accurately approximate solutions of those continuous PDEs. Our goal is to perform the following sequence of steps:

1. Embed the original discrete math model into a continuous diffusion PDE;
2. Solve this PDE by defining a finite difference grid with grid step size h.

The model is based on the following three conjectures which are analogs of the Fickian diffusion law, the Fourier law of heat conduction and the conservation law. In classical settings, conjectures about natural phenomena have been always validated experimentally, which led to verifications of corresponding mathematical models. This paper only presents the plausibility of the conjectures and we plan to validate the conjectures experimentally as future work.

## 2.1 Uncertainty Propagation in MAS using a Continuous model

As described earlier, our model works under the assumption that for a given agent  $Y$  in the MAS, there are a certain set of agents  $Z$  that are influenced by the uncertainty in agent  $Y$ 's actions.

Let  $N$  be the total number of agents in the MAS. To construct the continuous model, we take a subset of agents  $M_1 \subset N$  of this set. The set  $M_1$  is represented by a point  $x_1$  on the line of real numbers. Next, we take the set  $M_2$  of all agents whose uncertainty is influenced by  $M_1$ . The set  $M_2$  is represented by a point  $x_2 > x_1$ . Next, take the set  $M_3$  of all agents, that are influenced by the agents in the set  $M_2$ .  $M_{1,2}$  is the set of all agents in  $M_1$  that might be influenced by  $M_2$ , i.e., influence can go both ways: from  $M_1$  to  $M_2$  and backwards. We do not include  $M_{1,2}$  in  $M_3$  in order to guarantee that each agent is in exactly one set. The set  $M_3$  is represented by a point  $x_3 > x_2$ . Next, take the set  $M_4$  of all agents, which are influenced by agents in the set  $M_3$ , but not influenced by those agents in the set  $M_{1,3} \cup M_{2,3}$ . The set  $M_4$  is represented by a point  $x_4 > x_3$ .

This process is repeated assuming that the distance between neighboring points is the same, i.e.,  $x_i - x_{i-1} = h$ , where  $h$  is a small positive number independent of the point  $x_i$ . So, there are  $n \gg 1$  points  $\{x_i\}_{i=1}^n$ ,  $-\infty < x_1 < x_2 < x_3 < \dots < x_n < \infty$  and

$$\bigcup_{i=1}^n M_i = N \quad (1)$$

These set of points describe an embedded continuous model. The interval  $x \in [a, b]$  contains representatives of sets of agents. This is analogous to the Finite Difference Method for numerical solutions of PDEs/ODEs. While the embedded continuous model, is certainly convenient for the theory, in the case of numerical solutions we can consider points  $x_1, x_2, \dots, x_n$  with  $x_i - x_{i-1} = h$

as grid points of a corresponding finite difference scheme. Thus, we will actually find the solution of the corresponding PDE at these points. It is also well-known that a solution for a system of finite difference equations is often interpolated on the entire interval as a sufficiently smooth function.

Suppose  $N(x_i)$  is the number of agents in the set  $M_i$  and  $N_{inf}(x_i, t)$  is the number of agents influenced by uncertainty in this set at the moment of time  $t$ . Then

$$u(x_i, t) = \frac{N_{inf}(x_i, t)}{N(x_i)} \ll 1 \quad (2)$$

is the relative number of influenced agents in the set  $M_i$ . Near-decomposability of the MAS implies that the uncertainty from some agents of  $M_i$  might infect only neighboring agents: either those from the set  $M_{i+1}$  or those from the set  $M_{i-1}$ . In other words, an instantaneous change in the relative number  $u(x_i, t)$  of influenced agents of the set  $M_i$  at the time  $t$  will at most cause an instantaneous change of  $u(x_{i-1}, t)$  and  $u(x_{i+1}, t)$ .

Therefore, in the continuous model, we assume that for a *sufficiently small* time interval, agents of the set  $M_x$  can be influenced only by agents in very close sets  $M_y$  with  $y \approx x$ . This means that a change of the function  $u(x, t)$  at a point  $x$  and at a moment of time  $t$  causes a change of this function only for  $y \approx x$  for times  $t' \approx t$ . However, this does not preclude a change of the function  $u(x, t)$  at a point  $x$  to cause changes of  $u(y, t'')$  for points  $y$  located far from the point  $x$ , as long as the moment of time  $t''$  is far from  $t$  and  $t'' > t$ .

All functions involved in the derivations below are assumed to be sufficiently smooth. For each point  $y \in [a, b]$  and for each time moment  $t > 0$ , let  $M_y$  be the set of all agents represented by the point  $y$ ,  $N(y)$  be the total number of agents in  $M_y$ . Also, let  $\Delta x$  and  $\Delta t$  be two sufficiently small positive numbers, i.e.,  $0 < \Delta x, \Delta t \ll 1$ . Then, one can derive [6], the *density of influence of the interval*  $[y, y + \Delta x]$  at the moment of time  $t$  to be

$$\frac{N(y + \Delta x)u(y + \Delta x, t) - N(y)u(y, t)}{\Delta x} \quad (3)$$

The density of influence is approximately the average change, over the interval  $[y, y + \Delta x]$ , of the number of agents influenced by the uncertainty at the moment of time  $t$ . In a small time interval  $(t - \Delta t, t + \Delta t)$ , agents belonging to sets  $M_x$  with points  $x$  located in the interval  $[y, y + \Delta x]$  might be influenced only by agents from sets  $M_z$  for  $z \approx y \approx y + \Delta x$ . And vice versa: in a small time interval  $(t - \Delta t, t + \Delta t)$  agents belonging to these sets  $M_z$  might influence only agents of sets  $M_x$  with  $x \approx y \approx y + \Delta x$ .

Taking the limit for the above equation, the *density of influence* at the point  $x$  and at the moment of time  $t$  can be derived as

$$N(x) \frac{\partial u}{\partial x}(x, t) \quad (4)$$

Let  $G(x, t, t + \Delta t)$  be the number of hosts, which include: (1) agents influenced by agents of the set  $M_x$  in a small time interval  $(t, t + \Delta t)$  plus (2) those



agents in the set  $M_x$ , which are influenced by agents from neighboring sets  $M_z$  with  $z \approx x$  during the same time.  $G(x, t, t + \Delta t)$  is proportional to the density of influence,

$$G(x, t, t + \Delta t) = D(x)N(x)\frac{\partial u}{\partial x}(x, t) \Delta t \quad (5)$$

where the diffusion coefficient  $D(x)$  is an analog of the influence rate at the point  $x$ , i.e.,  $D(x)$  is the relative number of agents in an infinitesimally small period of time, that are influenced by uncertainty in agents of the set  $M_x$  plus those of the set  $M_x$ , which are influenced by agents in neighboring sets.  $D(x)$  is hence related to the *connectivity* averaged over all agents in the set  $M_x$ . It will be shown below that  $D(x)$  is an analog of the *diffusion coefficient* and that the above equation is an analog of the Fickian diffusion law [8].

In summary,  $G(x, t, t + \Delta t)$  is the *influence* through the point  $x$  in the time interval  $(t, t + \Delta t)$  where  $D(x) \geq \text{constant} > 0$ .

We now describe an analog to our model based on the 3-dimensional version of the Fourier law of heat conduction [?]. Let  $V \subset R^3$  be a certain volume,  $S$  be its boundary and the function  $v(X, t)$  be the temperature at the point  $(x, y, z) = X \in R^3$  at the moment of time  $t$ . Then the heat flow across the closed surface  $S$  in the small time interval  $(t, t + \Delta t)$  is

$$Q = \Delta t \int_S k \frac{\partial v}{\partial n} dS, \quad (6)$$

where  $n(X)$  is the unit outward normal vector on  $S$  and  $k(X)$  is the heat conduction coefficient, which is an analog of the diffusion coefficient. Now consider the 1-dimensional version of this formula. Specifically, in the 1-dimensional case, the interval  $[y, y + \Delta x]$  is taken instead of the volume  $V$  and  $S$  is replaced with two edge points  $y$  and  $y + \Delta x$ . We have then  $n(y + \Delta x) = 1$  and  $n(y) = -1$ .

The conservation law, which states that “the rate of change of the amount of material in a region is equal to the rate of change of flow across the boundary plus that which is created within the boundary” [8]. By analogy with the conservation law, it is shown [6] that the following equation can be derived

$$\int_y^{y+\Delta x} \frac{\partial}{\partial x} (D(x)N(x)u_x(x, t)) dx = \int_y^{y+\Delta x} N(x) \frac{\partial u}{\partial t}(x, t) dx. \quad (7)$$

Suppose  $F(x, t)$  is the *level of impact* of individual uncertainty at the point  $x$  at the time  $t$ . If an agent’s uncertainty is completely due to the propagation of uncertainty from another agent, then the impact is due to an external uncertainty and  $F(x, t)$  is set zero. If the uncertainty in an agent is due to its own actions, then  $F(x, t) > 0$ .

The following derivation of the **diffusion equation** is obtained as shown in [6] :

$$N(x)u_t = (D(x)N(x)u_x)_x + N(x)F(x, t). \quad (8)$$

where  $N(x)$  is number of agents in  $M_x$ ;  $u_t$  is the derivative of the number of influenced agents with respect to time  $t$ ;  $u_x$  is the derivative of the number of influenced agents with respect to point  $x$ ;  $D(x)$  is the diffusion coefficient at the point  $x$  which is related to the *connectivity* averaged over all agents in the set  $M_x$ ;  $F(x, t)$  is the level of impact of individual uncertainty. This equation captures the conservation law in terms of uncertainty where the rate of change of the amount of uncertainty in a subsystem is equal to the rate of change of uncertainty flowing into the subsystem plus that which is created within the subsystem.

The initial condition for the diffusion equation is

$$u(x, 0) = A\delta(x - x_0), \quad (9)$$

where  $A$  is a positive number. This initial condition reflects the fact that the source of the uncertainty was in the set  $M_{x_0}$  and it had the power  $A$ .

If  $N(x)$  is a constant, then the diffusion equation is simplified as

$$u_t = (D(x)u_x)_x + F(x, t). \quad (10)$$

## 2.2 Brief analysis of the model

We plan to perform a careful experimental analysis of this model as future work. We now present a brief qualitative analysis instead. Suppose we assume the diffusion coefficient is a constant, thereby considering a trivial case of the PDE i.e. for  $x \in (-\infty, \infty)$ ,  $D(x) = \text{constant} > 0$ ,  $F(x, t) = 0$ . Then the influence of uncertainty at time  $t$  is the solution to the diffusion equation defined above and is given by

$$u(x, t) = \frac{A}{2\sqrt{\pi Dt}} \exp\left[-\frac{(x - x_0)^2}{4Dt}\right] \quad (11)$$

In the above solution, the decay of the function  $u(x, t)$  is slower for larger values of  $D$ . This means that the larger the diffusion coefficient, i.e. the larger the connectivity among the agents, the wider the influence of the uncertainty at any fixed moment of time  $t$ .

Also, for any fixed value of  $x$ , the function  $u(x, t)$  increases first for small values of  $t$  and then decreases to zero starting at a value  $t = t(x)$ . The initial increase is due to influence in the sets  $M_y$  for values of  $y$  close to  $x$ . Let, for example  $x > x_0$ . Then  $x$  feels the spread of the uncertainty effects, which comes from the left, because  $x$  is located to the right of the initial source of the uncertainty  $x_0$ . Also,  $x$  begins to “sense” the influence of uncertainty at the moment of time  $t = t(x)$ , where  $t(x)$  is such a value of  $t$ , at which the function  $u(x, t)$  becomes significantly different from zero: also,  $\lim_{t \rightarrow 0} u(x, t) = 0$  for  $x \neq x_0$ . This supports our intuition of how uncertainty influences other agents in a nearly-decomposable system where one sub-system completes its task before another begins: the uncertainty in the initial sub-system influences agents of a certain set. This set, in turn influences “neighboring” sets, and so on. But after a while,

no new influence comes to the set  $M_x$  and the relative number of influenced agents eventually becomes negligibly small.

### 3 Conclusions and Future Work

In this paper, we have derived a Partial Differential Equation-based model for uncertainty propagation in large-scale multi-agent systems. We have used the similarities between uncertainty propagation and Fourier law of heat conduction to derive a model that is scalable and linear. The model helps determine the spread of influence of uncertainty over time and can help determine performance degradation caused by the uncertainty. It also gives us information on how best to control the uncertainty by using potential contingency plans which keep track of the possible path of the propagation. In this paper, we considered the case where uncertainty affects hard interactions such as *enables* between agents.

There are number of exciting areas of future work. We first plan to validate this model experimentally in a simulation system. The experiments will validate and help modify the model first for the hard interactions described in this paper, then for soft interactions such as *facilitates* and finally the aggregation of uncertainty as it propagates through the MAS. We also will use the simulations to derive reasonable bounds for coefficients of the diffusion PDEs and their derivatives for networks of varying sizes and with varying interactions.

### References

1. Boutilier, C., Dean, T., Hanks, S.: Planning under uncertainty: Structural assumptions and computational leverage. In Ghallab, M., Milani, A., eds.: New Directions in AI Planning. IOS Press (Amsterdam) (1996) 157–172.
2. Case, K., Zweifel, P.: Linear Transport Theory. Addison-Wesley Publishing Company, Massachusetts (1967).
3. Dean, T., Kaelbling, L.P., Kirman, J., Nicholson, A.: Planning with deadlines in stochastic domains. In Fikes, R., Lehnert, W., eds.: Proceedings of the Eleventh National Conference on Artificial Intelligence, Menlo Park, California, AAAI Press (1993) 574–579.
4. Decker, K.S., Lesser, V.R.: Quantitative modeling of complex computational task environments. In: Proceedings of the Eleventh National Conference on Artificial Intelligence, Washington (1993) 217–224.
5. Jennings, N.: An agent-based approach for building complex software systems. In: Proceedings of the Communications of the ACM. (2001) 35–41.
6. Klibanov, M.: Distributed modeling of propagation of computer viruses/worms by partial differential equations. In: Proceedings of Applicable Analysis, Taylor and Francis Limited (September 2006) 1025–1044.
7. Levy, H., Lessman, F.: Finite Difference Equations. Dover Publications, New York (1992).
8. Murray, J.: Mathematical Biology. Springer-Verlag, New York (1989).
9. Raja, A., Wagner, T., Lesser, V.: Reasoning about Uncertainty in Design-to-Criteria Scheduling. In: Working Notes of the AAAI 2000 Spring Symposium on Real-Time Systems, Stanford. (2000).

10. Simon, H.: The Sciences of the Artificial. MIT Press, Cambridge, MA (1969).
11. Vladimirov, V. S.; Equations of Mathematical Physics. Dekker, New York (1971).
12. Wagner, T., Raja, A., Lesser, V.: Modeling uncertainty and its implications to design-to-criteria scheduling. *Autonomous Agents and Multi-Agent Systems* **13** (2006) 235–292.
13. Xuan, P., Lesser, V.R.: Incorporating uncertainty in agent commitments. In: *Agent Theories, Architectures, and Languages*. (1999) 57–70.

# Security and Privacy Issues in Agent-based Location-aware Mobile Commerce

Athanasios Karygiannis<sup>1</sup>, Emmanouil Antonakakis<sup>1</sup>

<sup>1</sup> National Institute of Standards and Technology  
100 Bureau Drive, Gaithersburg, MD 20899  
{manos, karygiannis}@nist.gov

**Abstract.** Mobile commerce and location-aware services promise to combine the conveniences of both online and offline bricks-and-mortar services. Just as agent-enabled desktop computers can be used to improve a user's e-commerce experience, so can agent-enabled mobile devices be used to improve a user's mobile commerce experience. Agent-enabled mobile devices can perform complex and time-consuming tasks not well-suited for the small and cumbersome user interfaces available on most mobile devices, can interact with other mobile devices over more than one interface, and can accompany users under circumstances in which the desktop computers cannot. Agent-enabled mobile devices, however, present new security challenges and risks. While e-commerce agents run the risk of disclosing one's identity in cyberspace, agent-enabled mobile devices running location-aware applications, run the risk of disclosing one's actual physical location in addition to other personal information. This paper outlines security and privacy issues and provides security guidelines for agent-based location-aware mobile commerce.<sup>1</sup>

## 1 Introduction

Mobile devices equipped with built-in telephony, Internet access, Geographic Positioning Systems (GPS), smart cards, audio and video capture and playback, voice recognition and biometric authentication, are becoming widely available and more affordable. Once viewed as coveted gadgets for early technology adopters, their increased processing power, small size, longer battery life and built-in networking capabilities are making them indispensable tools for employees in almost every industry and a practical necessity for consumers throughout the world. The affordability of these devices, the proliferation of wireless hotspots and the availability of wireless location services have created new business opportunities.

Application areas benefiting from these location-aware and positioning technologies range from sales force automation and supply chain management, to tourism and lost pet recovery services. Smart vehicles of the future will provide location-aware services to provide motorists with improved safety, navigation, toll payment, entertainment and communication services. Location services are being employed today either as a service (enhanced-911) or unbeknownst to the consumer – such as the location tracking of EZ-Pass tag holders by roadside EZ-pass readers. On-board GPS devices have been used by rental car companies to track motorists' compliance with rental agreements; clothing manufacturers have considered using embedded RFIDs in clothing to track garments; and companies are offering wrist watches that allow parents to track their child's movement's from the web.

Agent-enabled mobile devices are a natural fit for helping users manage mobile commerce transactions and services with other e-commerce and mobile commerce agents, or directly with bricks-and-mortar businesses. As the number of domain areas and applications of location-aware services and devices increases, so will the difficulty of effectively managing and interacting with these services and devices. Clearly the privacy and security issues raised by these new technologies require careful examination and

---

<sup>1</sup> Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

appropriate technologies to balance the benefits with the risks of the potential abuse of having access to personal data that correlates online and location-specific offline behavior. Agent-enabled mobile devices and multi-agent systems must be able to adequately address these risks with countermeasures suitable for this resource-constrained environment. The following sections discuss the unique security risks associated with agent-enabled mobile devices and provide guidance on useful countermeasures to mitigate these risks [6,7,8,9,10,11,12].

## **2 Security Requirements for Agents in Mobile Commerce**

In order for agents to reliably participate in mobile commerce transactions, the agent platform must be able to provide support for disconnected operation, support for cryptographic operations, run on resource-constrained clients, resist physical attacks, protect sensitive data even when the device is lost or stolen, not introduce new security risk to the user or the user's organization, and securely communicate from a number of wireless interfaces. The security-relevant differences between mobile devices and desktop computers that directly affect the security of and privacy of agent-enabled mobile devices are summarized below. These security-relevant differences present new risks that must be taken into account by both developers and users of multi-agent systems and the appropriate countermeasures that are suited for the mobile commerce environment must be employed [19,23].

- Agent-enabled mobile devices are subject to physical capture. The small size, relatively low cost, and constant mobility of mobile devices make them more likely to be stolen, misplaced, or lost. Agents' private keys must be able to be physically protected from compromise and their public key revoked when the device is reported as being lost.
- Physical security controls that protect desktop computers do not offer the same protection for mobile devices. Security guards are more likely to check the contents of a laptop carrying case or check the laptop itself for proper identification than to physically search people for mobile devices. A thief can more easily conceal a mobile device than a laptop or desktop computer.
- Mobile devices broadcast "hello"-type messages or beacons to keep the network aware of its presence and service availability. Users of wireless networks must present credentials to prove they are authorized users or the service and in doing so disclose the location of their mobile device. Network service providers need to know the physical location of mobile phones for both billing purposes and to comply with e911 legislation in the US.
- The devices themselves have limited computing power, memory, and peripherals that make existing desktop security countermeasures impractical for mobile devices. Limited processing power, for example, may render encryption with long key lengths too time-consuming and unsuitable for an agent's needs.
- Synchronization software allows PCs to back up and mirror data stored on a mobile device and allows the mobile device to mirror data stored on desktop applications. The PC and the mobile device face different threats and require different security mechanisms to mitigate risk, but both must provide the same level of security to protect sensitive information. Confidential data collected by an agent-enabled mobile device may be compromised via the PC, and confidential data on the PC may be compromised via the mobile device.
- Members of an organization often purchase and use mobile devices without consulting with or notifying the organization's network administrator. Wireless mobile devices are often used for both personal and business transactions. Users that purchase these devices on their own often do not consider the security implications of their use in the work environment. Mobile devices can be the target of attack from outsiders or actually be used by insiders to compromise the security perimeter of an organization.
- Mobile devices offer multiple access points such as the user interface, expansion modules, wireless modems, Bluetooth, IR ports, and 802.11 connectivity. These access points present new risks that must be addressed separately from an existing wired network. Each interface supports different security mechanisms and provides different levels of assurance.
- Mobile devices have a number of communication ports from which they can send and receive data, but they have limited capabilities for authenticating the devices with which they exchange data. Agents

communicating via an Infrared port, for example, may not be able to rely on the same underlying security mechanisms available through Bluetooth.

- User authentication methods to the mobile device itself are limited by the device form factor. An unauthorized user of an agent-enabled mobile device that has no user-to-device authentication or weak authentication can instruct an agent to act upon their behalf without authorization.
- Many users have limited security awareness or training with the use of mobile devices and are not familiar with the potential security risks introduced by these devices. Agent-enabled mobile devices can quickly perform a number of complex tasks in a short amount of time on behalf of an authorized user. The same device in an unauthorized user's hands can also perform a number of unauthorized complex tasks in a short amount of time.
- Mobile device users can download a number of productivity programs, connectivity programs, games, and utilities—including freeware and shareware programs—from untrusted sources. The programs can be easily installed without network administrators being notified. These programs may contain Trojan horses or other “malware” that can affect the user's mobile device, PC, or other network resources.
- There are few, if any, auditing capabilities or security tools available for many of these devices. Mobile commerce transactions may require the location of the transaction to be recorded in addition to information related to the transaction itself. In many cases, neither the user nor the administrator can audit security-relevant events related to the use of these devices. Forensic tools for mobile devices also lack the maturity of their desktop counterparts.<sup>2</sup>
- Users often subscribe to third-party Wireless Internet Service Providers (WISP) and access the Internet through wireless modems. Users can download or upload data to and from other computers without complying with the organization's security policy.
- There are several new mobile operating systems and applications that have not been thoroughly tested by the market to expose potential vulnerabilities.
- Denial of service attacks in a wired network can be launched from anywhere in the world. Denial of service attacks in mobile commerce can be launched remotely and locally by jamming the physical medium. Battery exhaustion attacks can also be launched against mobile devices to drain the devices' power and prohibit any communication. Detailed time and geographical location may be required for the completion of location-specific transactions, collection of appropriate taxes, or the application of proper legislation.
- Mobile devices may conduct Peer-to-Peer commerce using digital cash in an ad hoc environment using networking technologies such as Bluetooth. Agents conducting mobile transactions in an ad hoc networking environment, however, may always have access to an online certificate authority to verify the credentials of their peers. Given that the availability of a certificate authority is not always possible, agents must verify the signatures of their peers along a certificate chain by querying a distributed storage system or by using a trust model whereby trusted nodes can vouch for the identities of peers without traversing and verifying the entire certificate chain which may not always be available.
- Agents operating in a mobile commerce environment cannot always expect the same quality of service as would be expected in a wired network and must be able to conduct transactions with intermittent connectivity. Some transactions may be conducted asynchronously, while others may be time-sensitive and require immediate completion.
- Given the mobile device's limited resources, support for security interoperability between resource-constrained mobile devices and desktop workstations or servers must be addressed. Cryptographic functions that can easily be used on workstations may create unacceptable performance degradation or battery use on mobile devices.

### **3 Wireless Location Services**

Wireless Location Services (WLS) can open up new opportunities for online and offline business alike and new services for mobile device users. Agent-enabled mobile devices can take advantage of location information to, for example, help user's find and purchase local goods and services while traveling in new places, help manage travel service, keep the user informed of ongoing transactions, and request approval or

---

<sup>2</sup> See NIST Special Publication SP-800-72 Guideline on PDA Forensics November 2004.

additional information from the user without constraining the user in the same physical location. In order to benefit from these new services, however, the user's location must be made available to the provider of these services. WLS technologies can be broken down into network-based solution, handset-based or GPS-based solutions, or proximity-based solutions. These technologies differ on the method of determining the mobile device's location, and also differ on the method by which they can disclose the mobile device's location [17].

### **3.1 Network - based WLS**

Network-based location services use triangulation techniques to identify the mobile device's location and require modifications to the network infrastructure. Network-based location services are installed on service provider's networks, operate inside buildings and urban canyons, determine the mobile device location within a few seconds, are more accurate in densely populated or urban areas, and require no special circuitry on or upgrading of the mobile handset. Network-based location services are available for cellular telephony networks as well as 802.11 WLAN networks. Network service providers that offer network-based location services may allow third parties to access this information in order to provide location services to mobile users. In this instance, agents running on mobile devices must consent to the release of their location information to third parties offering location-aware services and must trust the network service providers to disclose their location only to authorized third parties.

### **3.2 GPS - based WLS**

GPS-based location services require an on-board chipset on the mobile device itself, are available worldwide, require a clear path to a GPS satellite, cannot be relied upon for use inside a building, have an accuracy of approximately 5 meters, can take from 30 seconds to 15 minutes to determine the location, and have a higher cost per handset. GPS is also capable of providing altitude information in addition to longitude and latitude. Agents running on GPS-enabled mobile devices may share their location information at their discretion.

### **3.3 Proximity - based WLS**

Another way of establishing the location of a mobile device is to detect when the device is within the transmission range of a beacon or access point whose position is fixed and known in advance. For example, when a mobile device associates with an 802.11 access point, the location of the mobile device can be assumed to be within the transmission range of the access point. The transmission range can depend on environmental conditions and the type of antenna used by the mobile device, but can be used effectively in applications that do not require a great deal of accuracy, for example getting a listing of restaurants in a particular town. Bluetooth Piconets are ad hoc networks designed to easily interconnect mobile devices that are in the same physical area such as a conference room. Bluetooth devices have a typical operating range of about 10 meters, but this range can be extended up to 100 meters. A Bluetooth-enabled mobile handset can query or associate with a nearby Bluetooth beacon to establish its location. Agents relying on these technologies to establish their physical location must be able to remain anonymous, to conceal their physical location, and have their mobile commerce transactions be unlinkable and untraceable when desired.

### **3.4 RFIDs**

Radio Frequency Identification (RFID) tags are poised to replace one of the most successful product identification technologies used today, namely the Universal Product Code (UPC) or bar code. Although RFID technologies are predominantly associated with supply chain logistics, many governments are considering their use in border patrol and immigration processing environments. RFID technology can revolutionize supply chain logistics by allowing a handheld RFID reader to scan several hundred



tags/second without requiring line of sight. The ability to scan objects from a distance offers tremendous improvement in supply chain management, but also raises a number of privacy and security concerns. Applications that are cause for concern include: the tracking individuals and inanimate objects, correlating data from multiple tag reader locations, monitoring social interactions, tracking financial transactions, aggregating data from RFID constellations, and scanning personal belongings from a distance. Security threats include eavesdropping and unauthorized scanning, traffic analysis and tracking through predictable tag responses, spoofing and providing false sensor readings, denial of service and disruption of the supply chain, corporate espionage due to lack of reader access control. RFID technology is not a location service, but it does present a number of opportunities for location-specific tasks and its use in mobile commerce has yet to be fully exploited. Agent-enabled mobile devices can be used to locate RFID-enabled products, query the RFIDs for additional product information, combine online and offline comparison-shopping tools, directly purchase goods using mobile payments systems, actively participate in supply-chain management, and reduce the cost of buying for the user and the cost of selling for the merchant.

## **4 Mobile Commerce Agent Platforms**

Wireless mobile devices range from simple one- and two-way text messaging devices to Internet-enabled PDAs, tablets, and smart phones. The use of these devices in agent-based mobile commerce introduces new security risks. Moreover, as these devices begin having their own IP addresses, the devices themselves can become the targets of attacks. A natural evolution, of course, is to have attacks launched from mobile devices.

### **4.1 Mobile Phones, PDAs, and Smart Phones**

PDAs were first introduced to the market in the 1980s as mobile or palm-size computers that served as organizers for personal information and are gradually replacing the traditional leather-bound organizer. PDAs provide users with office productivity tools for accessing e-mail, enterprise network resources, and the Internet. These capabilities are quickly becoming a necessity in today's business environment. In addition, data that users have entered into their PDAs can be synchronized with a PC. Synchronization allows users to easily back up the information on their PDA and transfer data from the PC to the PDA. PDAs can also conveniently transfer data to other mobile devices by sending, or "beaming," the information through IR ports. This section provides general recommendations that can be applied to all mobile devices using these or other operating systems.

As the emerging mobile and networked workforce began carrying laptops and fumbling with PDAs and cell phones at the same time, mobile device manufacturers began responding by introducing devices that combine a PDA and a cell phone on the same device. These devices are referred to as smart phones. Smart phones incorporate the capabilities of a typical PDA and a digital mobile telephone that provides voice service as well as e-mail, text messaging, Web access, and voice recognition. Many smart phones are available that can run programming languages such as C or Java and offer telephony application programming interfaces (API) that allow third-party developers to build new productivity tools to help the mobile work force. Cell phone security has primarily focused on protecting carriers from fraudulent charges and users from eavesdropping. Typical mobile phones use simplified operating systems that have no information-processing capabilities and therefore present few information security risks. Smart phones, however, have more sophisticated operating systems capable of running applications and supporting network connectivity with other computing devices.

Most of the agent platform operating systems for Personal Digital Assistants (PDA) on the market today are Linux, Palm-OS, Symbian EPOC, or Microsoft PocketPC. Most of the agent platforms for mobile phones on the market are predominantly J2ME-based. Most high-end handheld organizers are both mobile phones and PDAs or at a minimum offer 802.11-connectivity. Handheld organizers offer more computing power than typical mobile phones and provide a good opportunity for agents to help users with their mobile commerce transactions. Although both PDAs and mobile phones are presently considered to be limited in processing power and memory when compared to desktop workstations, as can be expected, new products

are continuously being introduced into the market with enough computing power to allow for more sophisticated agent-based applications.

Even though agent-enabled mobile devices are relatively new, the research and developer communities have released a number of agent platform implementations targeting mobile phones and PDAs. Below we provide a representative snapshot of agent platform development platforms and languages. This list is not intended to be comprehensive and it is understood that some will no longer be supported when research grants have been exhausted while others will continue to flourish. Nonetheless, research and development activity in this area foretells a future where agent-enabled mobile devices are commonplace.

The Foundation for Intelligent Physical Agents FIPA is a non-profit organization that develops interoperability standards heterogeneous software agents. FIPA-OS is an agent development toolkit and a FIPA-compatible platform. Following is a list of publicly available implementations of agent platforms that conform to the FIPA Specifications: Agent Development Kit, April Agent Platform, Comtec Agent Platform, FIPA-OS, JACK Intelligent Agents, JADE, JAS (Java Agent Services API), LEAP, and ZEUS.<sup>3</sup>

MicroFIPA-OS runs on any device that supports the J2ME specification and virtual machine and has sufficient memory. MicroFIPA-OS has been tested on the following devices and operating systems: Compaq iPAQ H3600 series (Linux for iPAQ and PocketPC/WindowsCE), Casio Cassiopeia E115 (PocketPC/WindowsCE 3.0&2.11), Psion Series 5mx (Epoc), and Intel X86 architecture (Linux and Windows95/98/NT/2k). Supported and tested virtual machines and operating systems are: 2ME by Sun on Linux for iPAQ and PocketPC/WindowsCE, JDK 1.1.8/1.2.X/1.3.X, EpocVM (Psion) [13,16,18, 20].

## 4.2 Java

Java is the most popular agent programming language available today. Java is a full-featured, object-oriented programming language compiled into platform-independent byte code executed by an interpreter called the Java Virtual Machine (JVM). The resulting byte code can be executed where compiled or transferred to another Java-enabled platform. The Java programming language and runtime environment enforces security primarily through strong type safety, by which a program can perform certain operations only on certain kinds of objects. Java follows a so-called sandbox security model, used to isolate memory and method access, and maintain mutually exclusive execution domains. Java code such as a Web applet is confined to a sandbox, designed to prevent it from performing unauthorized operations, such as inspecting or changing files on a client file system and using network connections to circumvent file protections or people's expectations of privacy [14].

Security is enforced through a variety of mechanisms. Static type checking in the form of byte code verification is used to check the safety of downloaded code. Some dynamic checking is also performed during runtime. A distinct name space is maintained for untrusted downloaded code and linking of references between modules in different name spaces is restricted to public methods. A security manager mediates all accesses to system resources, serving in effect as a reference monitor. Permissions are assigned primarily based on the source of the code (where it came from) and the author of the code (who developed it), which restricts the access of the code to computational resources. In addition, Java inherently supports code mobility, dynamic code downloading, digitally signed code, remote method invocation, object serialization, and platform heterogeneity.

## 4.3 J2ME

The J2ME platform is the Java platform for resource-constrained mobile devices such as PDAs and mobile phones. The J2ME platform offers the following security features. The base of all security infrastructure is that the mobile device we be able to maintain locally keys with a secure way like within a smart card or locally in the hard drive of the device and that any time we be able to perform cryptographic operations in a secure way. The security environment for the J2ME is called the Security and Trust Services APIs (SATSA). SATSA provides application-level digital signatures; basic user credential management,

---

<sup>3</sup> See FIPA web site URL: <http://www.fipa.org/resources/livesystems.html>

cryptographic operation and smart card communication allowing for secure mobile data transmission by using encryption. The information received can be authenticated using the PKI infrastructure or by using pre-shared symmetric keys using decryption methods. Very important is the reduced size of the footprint that the J2ME and the SATSA especially reflects on the mobile device. This is happening because once the SATSA is loaded in the mobile device memory, for that point on it can serve different cryptographic operations without the making the mobile node reload a new process with new SATSA components. This is happening because a single SATSA process can be shared by all the applications that are running on the mobile device.

The JSR-179 Location API for J2ME specification defines a J2ME optional package that enables mobile location-based applications to request and get a location result. The API is designed to be a compact and generic API that produces information about the present geographic location of the device, or periodic location updates, to Java applications. The location may be obtained by using any number of location technologies, for example, satellite-based methods like GPS, cellular telephony network based methods, or short-range positioning methods like Bluetooth Local Positioning, etc. The location is represented by the geographical coordinates (latitude, longitude and altitude) and includes information about the coordinate's accuracy, a timestamp, and information about speed and course of the mobile device. This J2ME optional package also includes a database of landmarks and allows the user to store commonly used locations in the database. The landmark database may be shared between all Java applications and may be shared with other applications in the device, including native applications. This means that each application can learn about past travel activities, favorite locations, and depending on how each landmark is labeled, applications may be able to learn, for example, the user's home and address and office address [22].

#### **4.4 Linux PDA**

The Linux PDAs presently offer the same advantages and disadvantages encountered in the ongoing Linux open source debate. As technology advances are likely to allow for more processing power, and more on-board memory, the factors influencing the selection of a PDA operating system will become essentially the same as those influencing the choice of operating system for desktop workstations. The success of Linux-based PDAs rests on the open source model and its ability to engage the software development community to produce useful applications.<sup>4</sup> The Linux platform supports a wide variety of security technologies (VPN, Firewalls, Intrusion Detection, etc.) and an extensive set of security toolkits are available online.

#### **4.5 Pocket PC**

Microsoft's first entry into the PDA market was with the Windows CE handheld operating system that is now called Pocket PC. Although the first releases of these PDAs were not as easy to use and considerably more expensive than the Palm PDAs, these devices have shown marked improvement are leveraging all of Microsoft's resource from desktop application compatibility to software development tools. The Pocket PC grew out of the success of the Palm PDA and the realization that a market existed for similar devices that were more capable. Microsoft has included handheld devices in its new .Net strategy to encourage developers to build applications that can be accesses from a wide range of Microsoft platforms. The use of homogeneous operating systems for both mobile devices and workstations introduces new security concerns. On one hand, the homogeneous platforms may allow system administrators to take advantage of enterprise network security tools across all the organization's computing infrastructure, on the other hand, it may introduce new opportunities for the development of viruses and malware that originate on one platform and migrate to another.

#### **4.6 Palm OS**

Palm was able to establish itself as the market leader in the PDA market by focusing on simplicity and ease-of-use. Palm was able to capture 80% of the market by 2002, but strong competition has reduced

---

<sup>4</sup> See URL: <http://www.heldhelds.org> for more information on Linux on handheld devices.

Palm's market share over the last few years. Palm devices use 16- and 32-bit processors based on the Motorola DragonBall MC68328-family of microprocessors. The Palm OS is a proprietary operating system, but the source code is available for licensing. The Palm OS is single-threaded and is not generally considered a secure operating system as a user or application can access program code, data and the processor. The Palm OS is stored in ROM while applications and user data are stored in battery-powered RAM. Data is stored in records and records are grouped into databases where the database is analogous to a file. The Mobile Information Device Profile (MIDP) for Palm OS is a Java runtime environment for Palm OS handheld devices. It allows the Palm OS to run the same Java applications that are available for other MIDP-compliant devices, such as mobile phones and pagers. This means that the MIDP for the Palm OS provides standard APIs for developing Java applications for Palm OS handheld devices.<sup>5</sup>

#### **4.7 Symbian**

Symbian is an operating system that has been developed for the mobile phone market. Symbian offers full-strength encryption and certificate management, secure communications protocols and certificate-based application installation. Symbian supports four main content development options: C++, Java, WAP, and the Organisier Programming Language (OPL), a scripting language for the Symbian platform. Symbian supports the authentication of software components using digital signatures and certificates to provide a measure of confidence that applications being installed onto the mobile phone are from a known reputable vendor. The Symbian certificate management module provides mutual authentication between the mobile phone user and other entities.<sup>6</sup> It supports WTLS certificates and X.509 certificates. The certificate management module provides the following services: storage and retrieval of certificates using the cryptographic token framework, assignment of trust status to a certificate on an application-by-application basis, certificate chain construction and validation, verification of trust of a certificate, and certificate revocation.

### **5 Mobile Commerce Security Requirements**

Multi-agent systems and intelligent agents are a relatively new method of designing and implementing complex systems. The complexity of MAS when combined with the wireless devices and mobile commerce introduces new security risks and requirements. Each agent must assume that other agents are acting in the best interest of the host machine or the user it represents. Each agent must protect the confidentiality of the host user's private data and protect the host's or user's anonymity when necessary. Agents running on mobile devices must be able to mutually authenticate other agents and multi-agent systems. Agents may be downloaded from the web or installed from a PC, the provider of the agents must be authenticated by providing an identity certificate, and the mobile device must be able to enforce access control on the device itself [1,2,3,4,5,21].

Although mobile devices have not generally been viewed as posing security threats, their increased computing power and the ease with which they can access networks and exchange data with other mobile devices introduce new security risks to an organization's computing environment. As mobile devices begin supporting more networking capabilities, network administrators must carefully assess the risks they introduce into their existing computing environment. This section describes how the security requirements for confidentiality, integrity, authenticity, and availability for mobile device computing environments can be threatened.

#### **5.1 Loss of Confidentiality**

Information stored on mobile devices and on mobile device storage modules and mirrored on a PC, by both users and agents, must remain confidential and be protected from unauthorized disclosure. The

---

<sup>5</sup> See Sun's MIDP FAQ URL: <http://java.sun.com/products/midp4palm/faq.html>

<sup>6</sup> For more information on the Symbian security modules see URL: <http://www.symbian.com/technology/symbos-v7x-det.html#t11>

confidentiality of information can be compromised while on the mobile device, the storage module, or the PC or while being sent over one of the Bluetooth, 802.11, IR, USB, or serial communication ports. Moreover, most mobile devices are shipped with connectivity that is enabled by default. These default configurations are typically not in the most secure setting and should be changed to match the organization's security policy before being used.

PDA's can beam information from an IR port to another PDA IR port to easily exchange contact information such as telephone numbers and mailing addresses. This capability is a useful feature, but some concerns might arise about the data being transmitted. The data is unencrypted, and any user who is in close proximity to the mobile device and has the device pointed in the right direction can intercept and read the data. This is known as data leakage. Users familiar with PDA beaming should recognize that they often must have the PDA within a few inches of the other device and also make an effort to align the ports properly. The probability of data leakage occurring without the victim's knowledge is relatively low because it requires the intercepting device to be within a few feet and often within a few inches.

Nonetheless, the threat should not be overlooked because it could result in a compromise of sensitive information. No attack has been documented of a malicious user being able to pull information out of an IR port because the IR beaming protocol can only issue a request to send information that must be approved by the device user before the information is sent. There is no equivalent request to receive information. However, a Bluetooth device that is not configured properly is susceptible to having a user with a Bluetooth-enabled device pull data from the device. An 802.11-enabled device with an insecure P2P setting may also expose data to another 802.11-enabled device.

The ability of either the mobile device or the PC to initiate synchronization presents additional risks. A rogue compromised mobile device may attempt to synchronize with a PC; alternatively, a compromised PC may try to synchronize with a PDA. This type of attack is often referred to as "hijacking" and relies on hijacking software that is available today.<sup>7</sup> A malicious user could obtain personal or organizational data, depending on what is stored on the PDA or PC. For this type of attack to be successful, either the PC or the mobile device has been compromised, or a malicious user has been able to create a rogue mobile device or PC and gain access to the user's network.

PDA's can also remotely synchronize with a networked PC using dial-up connections, dialing either directly to a corporate facility or through a WISP. The modems allow users to dial into an access server at their office or use a third-party WISP. Dial-up capability, however, also introduces risks. Dialing into a corporate facility requires a mobile device synchronization server; otherwise, the remote PDA must derive synchronization service by connecting to a PC that is logged on using the remote client's ID and password. If the PC is not at least configured with a password-protected screensaver, it is left vulnerable to anyone with physical access to the PC. Moreover, since the WISP is an untrusted network, establishing a remote connection requires additional security mechanisms to ensure a secure connection. The PDA would require a VPN client and a supporting corporate system to create a secure tunnel through the WISP to the organization's network. Modem-enabled PDA's are still relatively new, and organizations may not have the security services in place to support them. Organizations may want to restrict their use until they have either adapted their existing VPN capabilities or put the required services in place.

Another means for synchronizing data is through an Ethernet connection. Users can synchronize data from any networked workspace. The data that crosses the network is as secure as the network itself and may be susceptible to network traffic analyzers or sniffers. PDA users can also synchronize through their organization's wireless network. This entails accessing 802.11-compliant APs to connect to an organization's wired network. Many PDA vendors support or are beginning to support VPN connections using 802.11 APs.

Smart phones can support wireless location services by using an on-board GPS integrated circuit or by having service providers analyze the cell phone signal received at cellular antenna sites. GPS-enabled phones can identify the phone's location to within a few meters and also relay position information. Thus, in the case of emergency, a user who may be injured or threatened can relay his location to the proper authorities. These devices are subject to security threats associated with networked computing devices but also have a new set of privacy concerns as the user's location can be disclosed to third parties. Advertisers and other service providers would like to access user location information through agreements with the cellular telephone provider. Users should carefully read cellular phone companies' privacy policies and opt out of any unwanted wireless location services.

---

<sup>7</sup> See "A Whole New World for the 21<sup>st</sup> Century," March 2001, at <http://www.sans.org>.

## 5.2 Loss of Integrity

The integrity of the information on the mobile device and the integrity of the mobile device hardware, applications, and underlying operating system are also security concerns. Information stored on, and software and hardware used by, the mobile device must be protected from unauthorized, unanticipated, or unintentional modification. Information integrity requires that a third party be able to verify that the content of a message has not been changed in transit and that the origin or the receipt of a specific message be verifiable by a third party. Moreover, users must be accountable and uniquely identifiable. The integrity of the information can be compromised while in transit or while stored on the mobile device or add-on storage modules. The integrity of the mobile hardware must be protected against the insertion or replacement of critical read-only memory (ROM) or other integrated circuits or upgradeable hardware. Mobile applications must be ensured to protect against the installation of software from unauthorized sources that may contain malware. The integrity of add-on modules must be ensured to protect the mobile device from rogue hardware add-on modules.

## 5.3 Loss of Availability

Many agent-based transactions must have network service availability for the entire duration of the transaction. The purpose of a Denial of Service (DoS) attack is to make computational or network resources unavailable or to severely limit their availability by consuming their resources with an inordinate amount of service requests. DoS attacks are typically associated with networked devices with fixed IP addresses for attackers to target. Most mobile devices access the Internet intermittently and do not have fixed IP addresses, but as networking technologies become more widespread, “always-on” connectivity will be commonplace within the next few years. As a result, many mobile devices already support the use of personal firewalls to protect themselves against certain DoS attacks and other types of attacks.

Mobile devices can also be the targets of DoS attacks through other means. Trojan horses, worms, viruses, and other malware can affect the availability of a network and, in many instances, also compromise the network’s confidentiality and integrity.<sup>8</sup> A virus that, for example, sends documents from a user’s PC to e-mail addresses found in the user’s electronic address book can burden the network with a flood of e-mails, send out confidential information, and even alter the information sent, all while giving the appearance that it was intentionally sent from the user’s account. Viruses have not been widely considered a security threat in PDAs because of the PDA’s limited memory and processing power.

Moreover, users typically synchronize their data with their PCs, and they can recover any lost or corrupted data simply by synchronizing with their PCs. Consequently, even a virus such as the Liberty Crack, which wipes out data on a PDA, has not been considered a serious threat.<sup>9</sup> PDA antivirus protection programs have only been on the market for a few years, and most PDAs do not have antivirus protection either because they do not support networking or the software simply has not been installed.

However, a virus on a mobile device could contain a payload designed to compromise a desktop PC, which in turn could directly affect the local network. As PDAs become more powerful, malicious users will develop viruses designed to achieve more harmful results. PDAs that share the same operating system as a PC may be particularly susceptible to a new strain of viruses. Although offering users additional benefits of sharing documents developed using the same applications, the common operating systems may invite new security threats. With both of the devices running the same applications, the methods for the virus to launch its attack and spread to other parts of the network increase.

Smart phones may lose network connectivity not only when they travel outside a cell coverage area but also when cell phone jammers are used. Many restaurants and movie theaters, for example, now use commercially available jammers to block cell phone communications often without notifying the cell phone users. Users expecting important messages are not able to receive those messages because the jammers block them from accessing network resources. Malicious users may also use cell phone jamming devices. Jamming devices can carry out these attacks by broadcasting transmissions on cellular frequencies that nullify the actual cellular tower transmissions. The jammed cell phone will not be able to communicate

---

<sup>8</sup> See SP 800-28, Guidelines on Active Content and Mobile Code, October 2001, for more information on malware.

<sup>9</sup> See PDA/Wireless Communication Pains, November 17, 2000, at <http://www.sans.org>.

unless other means of communications are available on the phone or in that region (e.g., a dual-band cell phone that can operate at different frequencies and also operate on an analog signal).

Cell phones, smart phones, and text pagers are able to send text messages, from 110 to 160 characters in length depending on the carrier, to other cell phones by using Short Message Service (SMS). To send and receive SMS text messages, phone users usually have to pay a monthly fee to their service provider or a small fee for each text message beyond a preset monthly limit. Text messages can also be sent from a cellular service provider's Web page, by visiting Web sites that allow users to send text messages free of charge from e-mail applications. Text-messages rely on the service provider's network and are not encrypted, and no guarantees exist on quality of service. Cell phones and text-messaging devices can be spammed with text messages until their mailbox is full, and the user is no longer able to receive new text messages unless previously stored e-mails are deleted.

As 3G development progresses and 3G phones become more prevalent, organizations will need to be aware of the security issues that arise. One potential security issue is that a 3G mobile device, when connected to an IP network, is in the "always-on" mode. This mode alleviates the need for the device to authenticate itself each time a network request is made. However, the continuous connection also makes the device susceptible to attack. Moreover, because the device is always on, the opportunity exists to track users' activities.

## **6 Mobile Device Security Guidelines**

As the use of mobile devices increases and technology improves, attacks can be expected to become more sophisticated. To control and even reduce the security risks identified above, administrators and users of multi-agent systems in mobile commerce need to implement management, operational, and technical countermeasures to safeguard mobile devices, confidential information, and the organization's networks [24].

### **6.1 Management Countermeasures**

Information security officers and network administrators should conduct a risk assessment before mobile devices are introduced into the organization's computing environment. Organizations should educate the users about the proper use of their mobile devices and the security risks introduced by their use by providing short training courses or educational materials to help users use these devices more productively and more securely. Moreover, network administrators should establish and document security policies that address their use and the user's responsibilities.<sup>10</sup> The policy document should include the approved uses, the type of information that the devices may store, software programs they can install, how to store the devices and associated modules when not in use, proper password selection and use, how to report a lost or stolen PDA, and any disciplinary actions that may result from misuse. Organizations should also perform random audits to track whether devices have been lost or stolen.

### **6.2 Operational Countermeasures**

Operational countermeasures require mobile device users to exercise due diligence in protecting the mobile devices and the networks they access from unnecessary risks. Most operational countermeasures are common sense procedures that require voluntary compliance by the users. Operational countermeasures are intended to minimize the risk associated with the use of mobile devices by well-intentioned users. Although a determined malicious user can find ways to intentionally disclose information to unauthorized sources, the mobile security policy and the organization's operational countermeasures should make clear the user's responsibilities.

The back of the PDA device should always be labeled with the owning organization's name, address, and phone number in case it is lost. Mobile device users should be provided with a secure area to

---

<sup>10</sup> See SP 800-30, Risk Management Guide for Information Technology Systems, January 2002, at <http://csrc.nist.gov/publications/nistpubs/index.html>.



store the device when not in use. A desk with drawers that lock or a file cabinet with locks are available in most offices and should provide sufficient physical security against theft from within the office environment.

Galvanized steel cables and locks are also available to secure mobile devices to the user's desktop if other physical controls are not available. Although these measures cannot ensure that a determined thief will not cut these cables or locks, it does prevent an opportunistic thief from walking away with an unattended mobile device. While on travel, room safes should be used, if available, to store mobile devices when not in use.

Security administrators should have a list of authorized mobile device users, to enable them to perform periodic inventory checks and security audits. Individuals that use their mobile devices for other than business uses should comply with the organization's security policy or be restricted from accessing the organization's network. Mobile devices should be distributed to the users with security settings that comply with the organization's security policy and should not be distributed with "out-of-the-box" default settings. A configuration management policy should be established. Such a policy frees security administrators from having to focus on many different configurations and allows them to concentrate on the configurations that have been adopted for the organization. Mobile devices should have a PIN code or password to access the device. Some mobile devices already use voice authentication for authenticating users to the device or to network resources. Voice authentication should be coupled with password authentication. A number of security tools are currently available to help mitigate the risks related to the use of PDAs, including password auditing, recovery/restoration, and vulnerability tools.<sup>11</sup>

In general, users should not store sensitive information on mobile devices. However, if sensitive information is stored on the mobile device, users should be encouraged to delete sensitive information when no longer needed. This information can be archived on the PC during synchronization and transferred back to the PDA when needed. Users can disable IR ports during periods of nonuse to deter them from leaking information from their mobile devices. Users with access to sensitive information should have approval from their management and network security administrators before storing sensitive information on their mobile device to ensure they have the appropriate security countermeasures in place.

Some mobile devices allow users to mark certain records as "private" and hide them unless the device password is entered. Thus, if a malicious user gained access to an unattended device without knowledge of the device password, that malicious user would not be able to see the private data. Depending on the underlying operating system, however, some of these private data fields can be read directly from memory.

### **6.3 Technical Countermeasures**

This section describes technical countermeasures for securing wireless mobile devices. Technical countermeasures should address the security risks identified during the risk assessment and should ensure that the organization's security policy is being enforced.

#### **6.3.1 Authentication**

Identification and authentication (I&A) form the process of recognizing and verifying valid users, processes, or devices. Mobile device users must be able to authenticate themselves to the mobile device by providing a password, a token, or both. At the most basic level, organizations should require mobile devices to be password protected. Security administrators should educate users on the selection of strong passwords. Password-cracking tools for mobile devices are available for network administrators and users to audit their PC's synchronization application password.<sup>12</sup> Password protection is already included with most mobile devices, but is usually not enabled in the default setting. Several Web sites offer software that prompts a user to enter a password when the user has turned the PDA off and turned it back on again.

---

<sup>11</sup> See "Research Tools" at <http://www.atstake.com>.

<sup>12</sup> See <http://www.atstake.com/research/tools/index.html> for PDA security assessment tools.



### 6.3.2 PC Synchronization Software

Biometric user authentication technologies are also available for mobile devices. Fingerprint readers can be attached to the mobile devices through a serial or USB port and can be set to lock the whole device, to lock an individual application, or to connect to a remote database over a network or dial-up connection. Tamper-proof smart cards, which contain unique user identifying information such as a private key, can also be used to authenticate the user to the device. Users insert the smart card into a peripheral slot on the device and provide a password to authenticate themselves. Malicious users must have possession of the smart card and knowledge of the user's password to gain access to the device.

Unique device identifiers, when available, can be used as part of an authorization mechanism to authenticate and provide network access to a mobile device. Mobile devices can take advantage of several methods to identify a unique mobile device, including flash ID, device ID, and Electronic Serial Number (ESN). Unique device identifiers can be used to authenticate the mobile device for network access or allow the mobile device itself to be used as a physical token for two-factor authentication.

Although it might be possible for an unauthorized user to copy the shape of a signature, many handwriting recognition programs measure aspects that are more difficult to copy, such as the rhythm and timing of the signature. The user can select a password to write instead of a signature, which is more widely available on paper documents distributed in the normal course of business.

### 6.3.3 Encryption

Some files on the device may require a higher level of security than password protection can offer. For example, user passwords are required to access all sorts of automated services in our everyday lives.

During the course of a single day, a user may need to use passwords to withdraw money from an automatic teller machine (ATM), to enter a building by typing an access code, to listen to voice mail, to browse favorite Web sites, to purchase goods online, to access online trading accounts, to make a phone call using a calling card, and to access personal and business e-mail accounts. Using the same password to access different services is discouraged because if this single password were compromised, an unauthorized user would be able to access all of the user's accounts. However, many PDA users store many of these passwords in a file on the PDA, possibly even naming the file "mypasswords." Once a single password has been given, other user accounts can be identified through various means ranging from dumpster diving to simply reviewing a user's Web browser history file. Encryption software can be used to protect the confidentiality of sensitive information stored on mobile devices and mirrored on the desktop PC. The information on add-on backup storage modules should also be encrypted and the modules securely stored when not in use. This additional level of security can be added to provide an extra layer of defense to further protect sensitive information stored on mobile devices. Many software programs are freely available to help users encrypt these types of files for an added layer of security. Encrypting the file protects it from brute-force password guessing if the file falls into the wrong hands. Mobile device users may elect to encrypt files and messages before the files and messages are transferred through a wireless port.

Smart phones use digital technologies to deter unencrypted voice traffic from being intercepted. FEC (Forward Error Correction) coding and spread-spectrum techniques add more robust communication error protection and complexity. Organizations should upgrade their analog phones to digital smart phones that offer more capabilities at the application level (e.g., Web browsing, networking) and the ability to use more security mechanisms with those applications.

### 6.3.4 Antivirus Software

Antivirus software is another important security measure for mobile devices.<sup>13</sup> All organizations, regardless of their security requirements, should incorporate PDA antivirus applications to scan e-mail and data files and to remove malware from files. The software should scan all entry ports (i.e., beaming, synchronizing, e-mail, and Internet downloading) as data is imported into the device, provide online signature update capabilities, and prompt the user before it deletes any suspicious files. Organizations should further require regular updates to the antivirus software and require associated workstations (i.e., the PCs with which users synchronize their PDAs) to have current, properly working virus-scanning software. Most major PC antivirus software vendors have mobile device anti-virus software that can be downloaded directly from their Web sites.

---

<sup>13</sup> See <http://csrc.nist.gov/virus/> for useful links for more information on viruses.

### **6.3.5 PKI**

Many mobile devices are beginning to offer support for PKI technologies. PKI is one of the best available methods for meeting confidentiality, integrity, and authenticity security requirements.<sup>14</sup> A PKI uses an asymmetric encryption method, commonly known as the “public/private key” method, for encrypting and ensuring the integrity of documents and messages. A certificate authority issues digital certificates that authenticate the claimed identity of people and organizations over a public network such as the Internet. The PKI also establishes the encryption algorithms, levels of security, and the key distribution policy for users. PKI support is often integrated into common applications such as Web browsers and e-mail programs by validating certificates and signed messages. The PKI can also be implemented by an organization for its own use to authenticate users that handle sensitive information.

The use of PKI counters many threats associated with public networks, but also introduces management overhead and additional hardware and software costs that should be evaluated while performing the risk assessment and selecting the appropriate countermeasures to meet the organization’s security requirements. If PKI has already been deployed to provide security services in the wired network of an organization, users should be provided a separate and distinct public/private key pair for use on PDAs. This will prevent compromise of the organization’s data in the event of a lost or stolen PDA.

### **6.3.6 VPN and Firewalls**

Organizations in a wide variety of industries are using mobile devices for remote access to patient records, merchandise inventory, and shipping logistics. Secure remote access for desktop and laptop computers has been successfully enabled by the use of firewalls and VPN over the last few years.<sup>15</sup>

Mobile devices are beginning to offer support for personal firewalls and VPN technologies and to offer network administrators effective countermeasures against threats to the confidentiality, integrity, and authenticity of the information being transferred. A packet filter firewall, for example, screens Internet traffic based on packet header information such as the type of application (e-mail, ftp, Web, etc.) and by the service port number. A VPN creates a virtual private network between the mobile device and the organization’s network by sharing the public network infrastructure. VPN technology offers the security of a private network through access control and encryption, while taking advantage of the economies of scale and built-in management facilities of large public networks. Network administrators should look for the following features when purchasing PDA-based VPN technologies: interoperability with existing infrastructure, support for wireless and dial-up networking, packet-filtering or stateful-inspection firewall, automatic security updates, and a centralized management console.

### **6.3.7 Location-specific Security Policies for Mobile Devices**

As the mobile device users move from location to location and the device is used under different conditions and exposed to different risks, a location-based security policy can help protect the device from both insider and outsider threats. When agent-enabled mobile devices are used for mobile commerce and location-aware services, there are a number of new policy-related and location-specific security concerns that must be addressed. The problem of how one creates, expresses, administers, distributes, enforces, updates, and monitors location-based security policies on mobile devices is very different from how policies are enforced on a desktop environment [15]. Examples of location-specific policies include disabling communication ports when there is an elevated threat of eavesdropping or signal detection, specifying encryption settings based on risk-level, recording the presence of other mobile devices in a particular meeting place, including the physical location of security-related events in audit logs, enabling capture-resilient protocols in case the mobile device is lost, and tagging and classifying objects based on the location in which they were created.

---

<sup>14</sup> See SP 800-32, Introduction to Public Key Technology and the Federal PKI Infrastructure, February 2001, at <http://csrc.nist.gov/publications/nistpubs/index.html>.

<sup>15</sup> See Special Publication 800-46, Security for Telecommuting and Broadband Communications, at <http://csrc.nist.gov/publications/nistpubs/index.html>.

### **6.3.8 Enterprise Solutions**

Enterprise mobile device management software allows network administrators to discover mobile devices, install and remove applications, back up and restore data, collect inventory information, synchronize data with corporate servers and databases, and perform various configuration management functions from a central location. Enterprise security solutions have been introduced that allow the organization to set policies on all mobile devices under the organization's control. Some of the options that are available include defining the type of encryption to use, which application databases to encrypt, password protection, and port protection.

### **6.3.9 Miscellaneous**

Third-party developers have introduced a number of security tools to help protect mobile devices. These security tools are fairly inexpensive and typically offer simple yet practical security countermeasures to protect against malicious users that are more likely to steal the device than to crack an encrypted file or eavesdrop on their wireless communications. Some of these security tools delete applications and their data after a preset number of unsuccessful login attempts. Authorized users simply have to resynchronize the mobile device with their PCs to recover the deleted information. This countermeasure is particularly effective and applicable in instances where mobile devices are holding sensitive information. Users must be cautioned that all data entered on the mobile device since the last synchronization will be lost. A malicious user could purposely enter several incorrect passwords to delete the data on an unattended mobile device, but this risk can be mitigated by frequent synchronization with the user's PC. Another simple security tool is to add an application that auto-locks the mobile device after it is idle for a selected period of time. The user can usually set this time-out period. This solution mitigates risks that arise when users leave mobile devices unattended. Users simply enter a password to regain access to the mobile device. This solution is similar to a screen saver password for a desktop PC.

## **7 Conclusions**

Mobile commerce is being enabled by new technologies that allow mobile phones and other mobile devices to access the Internet. There are now more mobile phones in use worldwide than PCs and televisions. This rapid growth of wireless data services foreshadows an even larger growth in both portable and embedded devices. Agent technologies can play an important role in helping users manage multiple mobile devices and communicate with other agent-enabled systems throughout their virtual and physical environments. These new technologies offer the promise of new and better ways of conducting business, but also raise a number of new privacy and security challenges and opportunities for research. To improve this situation, several steps must be taken. Wireless security standards must be developed and vetted by industry. Security services across wireless networks must be implemented end-to-end. Authentication services and key exchange technology must become more robust, including a mix of secret key and public-private key management technologies. As more security services are implemented, and as these services become more complex, sufficient policy management tools must exist to help agents negotiate privacy and security protocols between mobile devices. This paper has outlined security risks arising from agent-enabled mobile commerce and has provided practical management, operational, and technical guidelines for mitigating these risks.

## **References**

1. Gritzalis, D., Kyrloglou, N.: Consumer Online-Privacy and Anonymity Protection using Infomediary Schemes. Computer Science Society, 2001, SCCC 2001 Proceedings, XXI International Conference of the Chilean, 7-9 Nov 2001.
2. Geissler, T., Berlin, O.: Applying Security Standards to Multi Agent Systems. W21 Safety & Security in Multiagent Systems held at AAMAS 2004.

3. Patrick, A.S.: Building Trustworthy Software Agents. IEEE Internet Computing, November-December 2002, NRC 44958.
4. Padovan, B., Sackmann, S., Eymann, T., Pippow, I.: A Prototype for an Agent-based Secure Electronic Marketplace including Reputation Tracking Mechanisms. System Sciences, 2001 Proceedings of the 34th Annual Hawaii International Conference on , 3-6 Jan. 2001.
5. Younas, M., Chao, K.M., Anane, R.: M-Commerce Transaction Management with Multi-Agent Support, Advanced Information Networking and Applications. AINA 2003, 17th International Conference on, 27-29 March 2003.
6. Soriano, M., Ponce, D.: A Security and Usability Proposal for Mobile Electronic Commerce. Communications Magazine, IEEE, Volume 40, Issue 8, Aug. 2002.
7. Park, N.J., Song, Y.J.: M-Commerce Security Platform based on WTLS and J2ME. Industrial Electronics, 2001, Proceedings ISIE 2001, IEEE International Symposium on, Volume 3, 12-16 June 2001.
8. Ahluwalia, P., Varshney, U.: A Link and Network Layer Approach to Support Mobile Commerce Transactions. Vehicular Technology Conference, 2003, VTC 2003-Fall 2003 IEEE 58th, Volume 5, 6-9 Oct 2003.
9. Grosche, S.S., Knospe, H.: Secure mobile commerce. Electronics & Communication Engineering Journal, Volume 14, Issue 5, Oct 2002.
10. Li, X., Kou, W.: A Secure M-Commerce Model Based On Wireless Local Area Network. Advanced Information Networking and Applications, 2004, AINA 2004, 18th International Conference on, Volume 2, 29-31, March 2004.
11. Wang, Y.H., Wang, C.A., Chiang, J.S., Lo, W.H., Tamsui: A Secure Model in Agent-Based Marketplace. 17th International Conference on Advanced Information Networking and Applications (AINA'03) March 27 - 29, 2003 Xian, China.
12. Laclavik M., Hluchy L.: Secure inter-agent negotiation and communication. ICETA 2001. International Conference on Emerging Telecommunications Technologies and Applications - Kosice ELFA, 2001.
13. Bergenti, F., Poggi, A., Burg, B., Caire, G.: Deploying FIPA-compliant systems on mobile devices. Internet Computing, IEEE , Volume 5 , Issue 4 , July-Aug 2001
14. Datasheet Security Trust Services APIs For the Java" 2 Platform, Micro Edition: Security and Trust Services APIs For the Java 2 Platform, Micro Edition. [https://sdc.sun.com/kiosk/ViewPDF?pdf\\_id=IG820GGABO](https://sdc.sun.com/kiosk/ViewPDF?pdf_id=IG820GGABO) (Jan-2005).
15. Gavrilu S., Iorga M., Jansen W., Karygiannis A., Korolev V.: Security Policy Management for Handheld Devices. The 2003 International Conference on Security and Management, Las Vegas, Nevada, June 23-26, 2003.
16. Borselius N.: Security in multi-agent systems, Proceedings of the 2002 International Conference on Security and Management (SAM'02), Las Vegas, Nevada, June 2002, CSREA Press.
17. Hristova, N., O'Hare, G.M.P.: Ad-me: wireless advertising adapted to the user location, device and emotions. System Sciences, 2004 Proceedings of the 37th Annual Hawaii International Conference on , 5-8 Jan 2004.
18. Hattangady S., Davis C.: Reducing the Security Threats to 2.5G and 3G Wireless Applications. White Paper SWPY003 January 2002.
19. Chari S., Kermani P., Smith S.W., Tassiulas L.: Security Issues in M-Commerce: A Usage-Based Taxonomy. E-Commerce Agents, Marketplace Solutions, Security Issues, and Supply and Demand, Springer-Verlag LNCS 2033, pp. 264-283 April 2001.
20. Ahonen J.: PDA OS Security: Application Execution. Telecommunications Software and Multimedia TML-C7 ISSN 145597.
21. Spinellis D., Moulinos K., Iliadis J., Gritzalis D., Gritzalis S., Katsikas S.: Deploying a Secure Cyberbazaar by adding Trust on Commercial Transactions, eJETA: The eJournal for Electronic Commerce Tools and Applications, Vol.1, No.2, November 2002
22. Nokia Developer's Suites for J2ME™ and PersonalJava™, Location API for J2ME™, JSR-179, [http://www.forum.nokia.com/files/nds\\_disclaimer/1.6673.3551.00.html](http://www.forum.nokia.com/files/nds_disclaimer/1.6673.3551.00.html) (Jan-2005).

23. Labrou Y., Agre J., Ji L., Molina J., Chen W.L.: Wireless Wallet. Proceedings of the 1st International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2004), August 22-25, 2004 - Boston, Massachusetts, USA.
24. Owens, L., Karygiannis, A.: NIST Special Publication on Wireless Network Security: 802.11, Bluetooth, and Handheld Devices, SP 800-48, October 2002.

# Analyzing Dangers in Multiagent Rescue using DEFACTO

Janusz Marecki<sup>1</sup>, Nathan Schurr<sup>2</sup>, Milind Tambe<sup>3</sup>, and Paul Scerri<sup>4</sup>

<sup>1</sup> University of Southern California  
Los Angeles, CA 90089  
`marecki@usc.edu`

<sup>2</sup> University of Southern California  
Los Angeles, CA 90089  
`schurr@usc.edu`

<sup>3</sup> University of Southern California  
Los Angeles, CA 90089  
`tambe@usc.edu`

<sup>4</sup> Carnegie Mellon University  
Pittsburgh, PA 15213  
`pscerri@cs.cmu.edu`

**Abstract.** Enabling interactions of agent-teams and humans for safe and effective Multiagent rescue is a critical area of research, with encouraging progress in the past few years. However, previous work suffers from three key limitations: (i) limited human situational awareness, reducing human effectiveness in directing agent teams, (ii) the agent team’s rigid interaction strategies that jeopardize the rescue operation, and (iii) lack of formal tools to analyze the impact of such interaction strategies. This paper presents a software prototype called DEFACTO (Demonstrating Effective Flexible Agent Coordination of Teams through Omnipresence). DEFACTO is based on a software proxy architecture and 3D visualization system, which addresses the three limitations mentioned above. First, the 3D visualization interface enables human virtual omnipresence in the environment, improving human situational awareness and ability to assist agents. Second, generalizing past work on adjustable autonomy, the agent team chooses among a variety of ”team-level” interaction strategies, even excluding humans from the loop in extreme circumstances. Third, analysis tools help predict the dangers of using fixed strategies for various agent teams in a future disaster response simulation scenario.

**Key words:** Multiagent Systems, Adjustable Autonomy, Teamwork

## 1 Introduction

Multi agent safety addressed in this paper is interpreted in the way that a team of agents ensures the safety of civilians or buildings in case of an emergency situation. Analyzing the safety of using multi agent teams interacting with humans

is critical in a large number of current and future applications[2, 5, 14, 3]. For example, current efforts emphasize humans collaboration with robot teams in space explorations, humans teaming with robots and agents for disaster rescue, as well as humans collaborating with multiple software agents for training [4, 6].

This paper focuses on the challenge of improving the effectiveness and analysing the dangers of human collaboration with agent teams. Previous work has reported encouraging progress in this arena, e.g., via proxy-based integration architectures[10], adjustable autonomy[13, 4] and agent-human dialogue [1]. Despite this encouraging progress, previous work suffers from three key limitations. First, when interacting with agent teams acting remotely, human effectiveness is hampered by low-quality interfaces. Techniques that provide telepresence via video are helpful [5], but cannot provide the global situation awareness. Second, agent teams have been equipped with adjustable autonomy (AA)[14] but not the flexibility critical in such AA. Indeed, the appropriate AA method varies from situation to situation. In some cases the human user should make most of the decisions. However, in other cases human involvement may need to be restricted. Such flexible AA techniques have been developed in domains where humans interact with individual agents [13], but whether they apply to situations where humans interact with agent teams is unknown. Third, current systems lack tools to analyze the impact of human involvement in agent teams, yet these are key to flexible AA reasoning.

We report on a software prototype system, DEFACTO (Demonstrating Effective Flexible Agent Coordination of Teams through Omnipresence), that enables agent-human collaboration and addresses the three shortcomings outlined above. First, DEFACTO incorporates a visualizer that allows the human to have an *omnipresent* interaction with remote agent teams. We refer to this as the Omni-Viewer, and it combines two modes of operation. The Navigation Mode allows for a navigable, high quality 3D visualization of the world, whereas the Allocation Mode provides a traditional 2D view and a list of possible task allocations that the human may perform. Human experts can quickly absorb on-going agent and world activity, taking advantage of both the brain’s favored visual object processing skills (relative to textual search, [9]), and the fact that 3D representations can be innately recognizable, without the layer of interpretation required of map-like displays or raw computer logs. The Navigation mode enables the human to understand the local perspectives of each agent in conjunction with the global, system-wide perspective that is obtained in the Allocation mode.

Second, to provide flexible AA, we generalize the notion of *strategies* from single-agent single-human context [13]. In our work, agents may flexibly choose among team strategies for adjustable autonomy instead of only individual strategies; thus, depending on the situation, the agent team has the flexibility to limit human interaction, and may in extreme cases exclude humans from the loop. Third, we provide a formal mathematical basis of such team strategies. These analysis tools help agents in flexibly selecting the appropriate strategy for a given situation.

We present results from detailed experiments with DEFACTO, which reveal two major surprises. First, contrary to previous results[14], human involvement is not always beneficial to an agent team— despite their best efforts, humans may sometimes end up hurting an agent team’s performance. Second, increasing the number of agents in an agent-human team may also degrade the team performance, even though increasing the number of agents in a pure agent team under identical circumstances improves team performance. Fortunately, in both surprising instances above, DEFACTO’s flexible AA strategies alleviate such problematic situations.

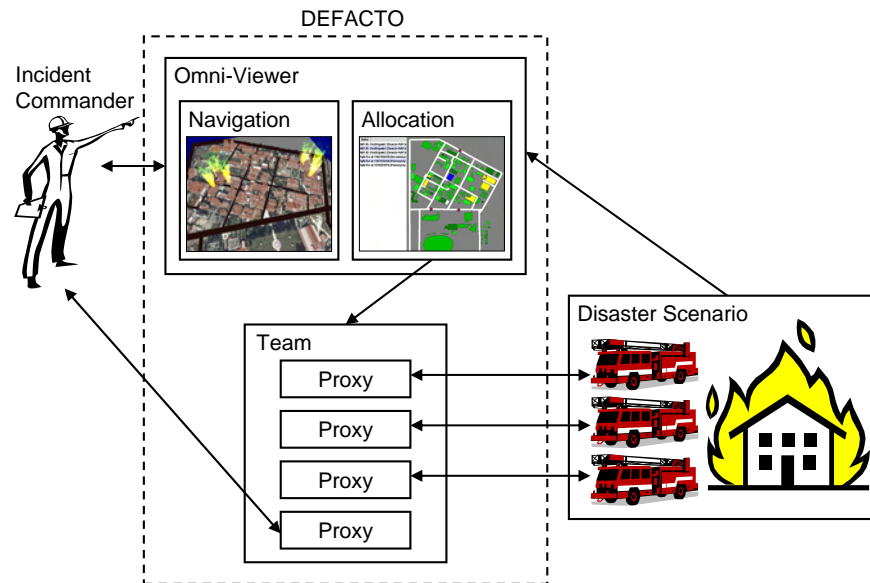
## 2 DEFACTO System Details

DEFACTO consists of two major components: the Omni-Viewer and a team of proxies (see Figure 1). The Omni-Viewer allows for global and local views. The proxies allow for team coordination and communication, but more importantly also implement flexible human-agent interaction via Adjustable Autonomy. Currently, we have applied DEFACTO to a disaster rescue domain. The incident commander of the disaster acts as the *user* of DEFACTO. This disaster can either be “man made” (terrorism) or “natural” (earthquake). We focus on two urban areas: a square block that is densely covered with buildings (we use one from Kobe, Japan) and the University of Southern California campus, which is more sparsely covered with buildings. In our scenario, several buildings are initially on fire, and these fires spread to adjacent buildings if they are not quickly contained. The goal is to have a human interact with the team of fire engines in order to save the most buildings. Our overall system architecture applied to disaster response can be seen in Figure 1. While designed for real world situations, DEFACTO can also be used as a training tool for incident commanders when hooked up to a simulated disaster scenario.

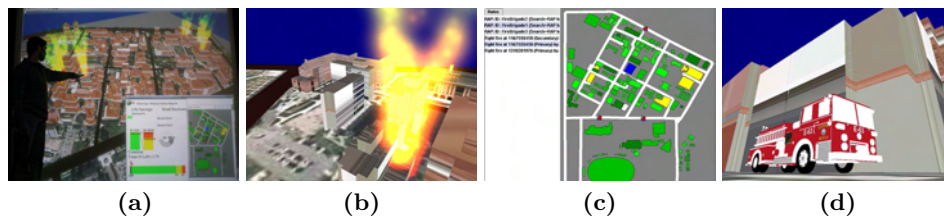
### 2.1 Omni-Viewer

Our goal of allowing fluid human interaction with agents requires a visualization system that provides the human with a global view of agent activity as well as showing the local view of a particular agent when needed. Hence, we have developed an omnipresent viewer, or Omni-Viewer, which will allow the human user diverse interaction with remote agent teams. While a global view is obtainable from a two-dimensional map, a local perspective is best obtained from a 3D viewer, since the 3D view incorporates the perspective and occlusion effects generated by a particular viewpoint. The literature on 2D- versus 3D-viewers is ambiguous. For example, spatial learning of environments from virtual navigation has been found to be impaired relative to studying simple maps of the same environments [11]. On the other hand, the problem may be that many virtual environments are relatively bland and featureless. Ruddle points out that navigating virtual environments can be successful if rich, distinguishable landmarks are present [12].





**Fig. 1.** DEFACTO system applied to a disaster rescue.



**Fig. 2.** Omni-Viewer during a scenario: (a) An Incident Commander using the Navigation mode spots multiple fires (b) The Commander navigates to quickly grasp the situation (c) The Commander is transferred control of the task to fight the fire and uses the Allocation mode to send a fire engine there (d) The fire has been extinguished.

To address our discrepant goals, the Omni-Viewer incorporates both a conventional map-like 2D view, Allocation Mode (Figure 2-c) and a detailed 3D viewer, Navigation Mode (Figure 2-a). The Allocation mode shows the global overview as events are progressing and provides a list of tasks that the agents have transferred to the human. The Navigation mode shows the same dynamic world view, but allows for more freedom to move to desired locations and views. In particular, the user can drop to the virtual ground level, thereby obtaining the world view (local perspective) of a particular agent. At this level, the user can “walk” freely around the scene, observing the local logistics involved as various entities are performing their duties. This can be helpful in evaluating the physical ground circumstances and altering the team’s behavior accordingly. It also allows the user to feel immersed in the scene where various factors (psychological, etc.) may come into effect.

In order to prevent communication bandwidth issues, we assume that a high resolution 3D model has already been created and the only data that is transferred during the disaster are important changes to the world. Generating this suitable 3D model environment for the Navigation mode can require months or even years of manual modeling effort, as is commonly seen in the development of commercial video-games. However, to avoid this level of effort we make use of the work of You et. al. [15] in rapid, minimally assisted construction of polygonal models from LiDAR (Light Detection and Ranging) data. Given the raw LiDAR point data, we can automatically segment buildings from ground and create the high resolution model that the Navigation mode utilizes. The construction of the USC campus and surrounding area required only two days using this approach. LiDAR is an effective way for any new geographic area to be easily inserted into the Omni-Viewer.

We use the JME game engine to perform the actual rendering due to its cross-platform capabilities. JME is an extensible library built on LWJGL (Light Weight Java Game Library), which interfaces with OpenGL and OpenAL. This environment easily provided real-time rendering of the textured campus environment on mid-range commodity PCs. JME utilizes a scene graph to order the rendering of geometric entities. It provides some important features such as OBJ format model loading (which allows us to author the model and textures in a tool like Maya and load it in JME) and also various assorted effects such as particle systems for fires.

## 2.2 Proxy: Teamwork and Adjustable Autonomy

We have built teams based on previous proxy software [13], that is in the public domain. The proxies were extended to our domain in order to take advantage of existing methods of communication, coordination, and task allocation for the team. However, these aspects are not the focus of this paper.

Instead, we focus on another key aspect of the proxies: Adjustable Autonomy. Adjustable autonomy refers to an agent’s ability to dynamically change its own autonomy, possibly to transfer control over a decision to a human. Previous work on adjustable autonomy could be categorized as either involving a single person

interacting with a single agent (the agent itself may interact with others) or a single person directly interacting with a team. In the single-agent single-human category, the concept of flexible transfer-of-control strategy has shown promise [13]. A transfer-of-control strategy is a preplanned sequence of actions to transfer control over a decision among multiple entities, for example, an  $AH_1H_2$  strategy implies that an agent ( $A_T$ ) attempts a decision and if the agent fails in the decision then the control over the decision is passed to a human  $H_1$ , and then if  $H_1$  cannot reach a decision, then the control is passed to  $H_2$ . Since previous work focused on single-agent single-human interaction, strategies were individual agent strategies where only a single agent acted at a time.

An optimal transfer-of-control strategy optimally balances the risks of not getting a high quality decision against the risk of costs incurred due to a delay in getting that decision. Flexibility in such strategies implies that an agent dynamically chooses the one that is optimal, based on the situation, among multiple such strategies ( $H_1A$ ,  $AH_1$ ,  $AH_1A$ , etc.) rather than always rigidly choosing one strategy. The notion of flexible strategies, however, has not been applied in the context of humans interacting with agent-teams. Thus, a key question is whether such flexible transfer of control strategies are relevant in agent-teams, particularly in a large-scale application such as ours.

DEFACTO aims to answer this question by implementing transfer-of-control strategies in the context of agent teams. One key advance in DEFACTO, however, is that the strategies are not limited to individual agent strategies, but also enables team-level strategies. For example, rather than transferring control from a human to a single agent, a team-level strategy could transfer control from a human to an agent-team. Concretely, each proxy is provided with all strategy options; the key is to select the right strategy given the situation. An example of a team level strategy would combine  $A_T$  Strategy and  $H$  Strategy in order to make  $A_TH$  Strategy. The default team strategy,  $A_T$ , keeps control over a decision with the agent team for the entire duration of the decision. The  $H$  strategy always immediately transfers control to the human.  $A_TH$  strategy is the conjunction of team level  $A_T$  strategy with  $H$  strategy. This strategy aims to significantly reduced the burden on the user by allowing the decision to first pass through all agents before finally going to the user, if the agent team fails to reach a decision.

### 3 Mathematical Model of Strategy Selection

We develop a novel mathematical model for these team level adjustable autonomy strategies in order to enable team-level strategy selection. We first quickly review background on individual strategies from Scerri [13] before presenting our team strategies. Whereas strategies in Scerri’s work are based on a single decision that is sequentially passed from agent to agent, we assume that there are multiple homogeneous agents concurrently working on multiple tasks interacting with a single human user. We exploit these assumptions (which capture the

features of our domain) to obtain a reduced version of our model and simplify the computation in selecting strategies.

### 3.1 Background on individual strategies

A decision,  $d$ , needs to be made. There are  $n$  entities,  $e_1 \dots e_n$ , who can potentially make the decision. These entities can be human users or agents. The expected quality of decisions made by each of the entities,  $\mathbf{EQ} = \{EQ_{e_i,d}(t) : \mathcal{R} \rightarrow \mathcal{R}\}_{i=1}^n$ , is known, though perhaps not exactly.  $\mathbf{P} = \{P_{\top}(t) : \mathcal{R} \rightarrow \mathcal{R}\}$  represents continuous probability distributions over the time that the entity in control will respond (with a decision of quality  $EQ_{e,d}(t)$ ). The cost of delaying a decision until time  $t$ , denoted as  $\{\mathcal{W} : t \rightarrow \mathcal{R}\}$ . The set of possible wait-cost functions is  $\mathbf{W}$ .  $\mathcal{W}(t)$  is non-decreasing and at some point in time,  $\Gamma$ , when the costs of waiting stop accumulating (i.e.,  $\forall t \geq \Gamma, \forall \mathcal{W} \in \mathbf{W}, \mathcal{W}(t) = \mathcal{W}(\Gamma)$ ).

To calculate the EU of an arbitrary strategy, the model multiplies the probability of response at each instant of time with the expected utility of receiving a response at that instant, and then sum the products. Hence, for an arbitrary continuous probability distribution if  $e_c$  represents the entity currently in decision-making control:

$$EU = \int_0^{\infty} P_{\top}(t) EU_{e_c,d}(t) .dt \quad (1)$$

Since we are primarily interested in the effects of delay caused by transfer of control, we can decompose the expected utility of a decision at a certain instant,  $EU_{e_c,d}(t)$ , into two terms. The first term captures the quality of the decision, independent of delay costs, and the second captures the costs of delay:  $EU_{e_c,d}t = EQ_{e,d}(t) - \mathcal{W}(t)$ . To calculate the EU of a strategy, the probability of response function and the wait-cost calculation must reflect the control situation at that point in the strategy. If a human,  $H_1$  has control at time  $t$ ,  $P_{\top}(t)$  reflects  $H_1$ 's probability of responding at  $t$ .

### 3.2 Introduction of team level strategies

**$A_T$  Strategy:** Starting from the individual model, we introduce team level  $A_T$  strategy, denoted as  $A_T$  in the following way: We start with Equation 2 for single agent  $A_T$  and single task  $d$ . We obtain Equation 3 by discretizing time,  $t = 1, \dots, T$  and introducing set  $\Delta$  of tasks. Probability of agent  $A_T$  performing a task  $d$  at time  $t$  is denoted as  $P_{a,d}(t)$ . Equation 4 is a result of the introduction of the set of agents  $AG = a_1, a_2, \dots, a_k$ . We assume the same quality of decision for each task performed by an agent and that each agent  $A_T$  has the same quality so that we can reduce  $EQ_{a,d}(t)$  to  $EQ(t)$ . Given the assumption that each agent  $A_T$  at time step  $t$  performs one task, we have  $\sum_{d \in \Delta} P_{a,d}(t) = 1$  which is depicted in Equation 5. Then we express  $\sum_{a=a_1}^{a_k} \sum_{d \in \Delta} P_{a,d}(t) \times W_{a,d}(t)$  as the total team penalty for time slice  $t$ , i.e., at time slice  $t$  we subtract one penalty unit for each

not completed task as seen in Equation 6. Assuming penalty unit  $PU = 1$  we finally obtain Equation 7.

$$EU_{a,d} = \int_0^\infty P_{\top a}(t) \times (EQ_{a,d}(t) - \mathcal{W}(t)).dt \quad (2)$$

$$EU_{a,\Delta} = \sum_{t=1}^T \sum_{d \in \Delta} P_{a,d}(t) \times (EQ_{a,d}(t) - \mathcal{W}(t)) \quad (3)$$

$$EU_{A_T,\Delta} = \sum_{t=1}^T \sum_{a=a_1}^{a_k} \sum_{d \in \Delta} P_{a,d}(t) \times (EQ_{a,d}(t) - W_{a,d}(t)) \quad (4)$$

$$EU_{A_T,\Delta,AG} = \sum_{t=1}^T \left( \sum_{a=a_1}^{a_k} EQ(t) - \sum_{a=a_1}^{a_k} \sum_{d \in \Delta} P_{a,d}(t) \times W_{a,d}(t) \right) \quad (5)$$

$$EU_{A_T,\Delta,AG} = \sum_{t=1}^T (|AG| \times EQ(t) - (|\Delta| - |AG| \times t) \times PU) \quad (6)$$

$$EU_{A_T,\Delta,AG} = |AG| \times \sum_{t=1}^T (EQ(t) - (\frac{|\Delta|}{AG} - t)) \quad (7)$$

**H Strategy:** The difference between  $EU_{H,\Delta,AG}$  and  $EU_{A_T,\Delta,AG}$  results from three key observations: First, the human is able to choose strategic decisions with higher probability, therefore his  $EQ_H(t)$  is greater than  $EQ(t)$  for both individual and team level  $A_T$  strategies. Second, we hypothesize that a human cannot control all the agents  $AG$  at disposal, but due to cognitive limits will focus on a smaller subset,  $AG_H$  of agents (evidence of limits on  $AG_H$  appears later in Figure 5-a).  $|AG_H|$  should slowly converge to  $B$ , which denotes its upper limit, but never exceed  $AG$ . Each function  $f(AG)$  that models  $AG_H$  should be consistent with three properties: i) if  $B \rightarrow \infty$  then  $f(AG) \rightarrow AG$ ; ii)  $f(AG) < B$ ; iii)  $f(AG) < AG$ . Third, there is a delay in human decision making compared to agent decisions. We model this phenomena by shifting  $H$  to start at time slice  $t_H$ . For  $t_H - 1$  time slices the team incurs a cost  $|\Delta| \times (t_H - 1)$  for all incomplete tasks. By inserting  $EQ_H(t)$  and  $AG_H$  into the time shifted utility equation for  $A_T$  strategy we obtain the  $H$  strategy (Equation 8).

**$A_T H$  Strategy:** The  $A_T H$  strategy is a composition of  $H$  and  $A_T$  strategies (see Equation 9).

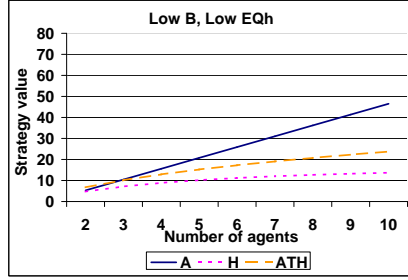
$$EU_{H,\Delta,AG} = |AG_H| \times \sum_{t=t_H}^T \left( EQ_H(t) - \left( \frac{|\Delta|}{|AG_H|} - (t - t_H) \right) \right) - |\Delta| \times (t_H - 1) \quad (8)$$

$$EU_{A_T H, \Delta, AG} = |AG| \times \sum_{t=1}^{t_H-1} \left( EQ(t) - \left( \frac{|\Delta|}{|AG|} - t \right) \right) + |AG_H| \times \sum_{t=t_H}^T \left( EQ_H(t) - \left( \frac{|\Delta| - |AG|}{|AG_H|} - (t - t_H) \right) \right) \quad (9)$$

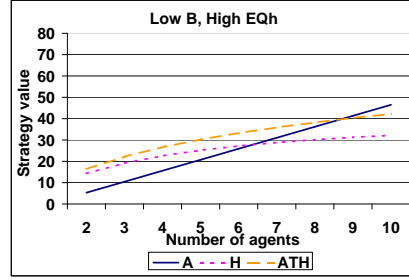
**Strategy utility prediction:** Given our strategy equations and the assumption that  $EQ_{H,\Delta,AG}$  is constant and independent of the number of agents we plot the graphs representing strategy utilities (Figure 3). Figure 3 shows the number of agents on the x-axis and the expected utility of a strategy on the y-axis. We focus on humans with different skills: (a) low  $EQ_H$ , low  $B$  (b) high  $EQ_H$ , low  $B$  (c) low  $EQ_H$ , high  $B$  (d) high  $EQ_H$ , high  $B$ . The last graph representing a human with high  $EQ_H$  and high  $B$  follows results presented in [13] (and hence the expected scenario), we see the curve of  $AH$  and  $A_TH$  flattening out to eventually cross the line of  $A_T$ . Moreover, we observe that the increase in  $EQ_H$  increases the slope for  $AH$  and  $A_TH$  for small number of agents, whereas the increase of  $B$  causes the curve to maintain a slope for larger number of agents, before eventually flattening out and crossing the  $A_T$  line.

## 4 Experiments and Evaluation

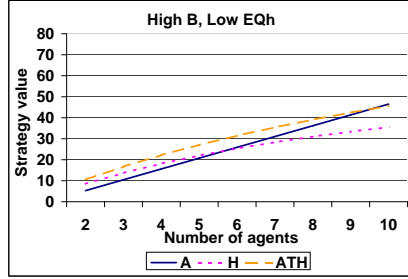
Our DEFACTO system was evaluated in three key ways, with the first two focusing on key individual components of the DEFACTO system and the last attempting to evaluate the entire system. First, we performed detailed experiments comparing the effectiveness of Adjustable Autonomy (AA) strategies over multiple users. In order to provide DEFACTO with a dynamic rescue domain we chose to connect it to a simulator. We chose the previously developed RoboCup Rescue simulation environment [8]. In this simulator, fire engine agents can search the city and attempt to extinguish any fires that have started in the city. To interface with DEFACTO, each fire engine is controlled by a proxy in order to handle the coordination and execution of AA strategies. Consequently, the proxies can try to allocate fire engines to fires in a distributed manner, but can also transfer control to the more expert user. The user can then use the Omni-Viewer in Allocation mode to allocate engines to the fires that he has control over. In order to focus on the AA strategies (transferring the control of task allocation) and not have the users ability to navigate interfere with results, the Navigation mode was not used during this first set of experiments.



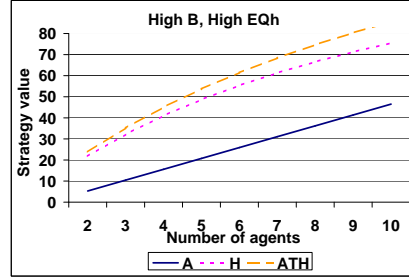
(a)



(b)



(c)



(d)

**Fig. 3.** Model predictions for various users.

The results of our experiments are shown in Figure 4, which shows the results of subjects 1, 2, and 3. Each subject was confronted with the task of aiding fire engines in saving a city hit by a disaster. For each subject, we tested three strategies, specifically,  $H$ ,  $AH$  and  $A_TH$ ; their performance was compared with the completely autonomous  $A_T$  strategy.  $AH$  is an individual agent strategy, tested for comparison with  $A_TH$ , where agents act individually, and pass those tasks to a human user that they cannot immediately perform. Each experiment was conducted with the same initial locations of fires and building damage. For each strategy we tested, varied the number of fire engines between 4, 6 and 10. Each chart in Figure 4 shows the varying number of fire engines on the x-axis, and the team performance in terms of numbers of building saved on the y-axis. For instance, strategy  $A_T$  saves 50 building with 4 agents. Each data point on the graph is an average of three runs. Each run itself took 15 minutes, and each user was required to participate in 27 experiments, which together with 2 hours of getting oriented with the system, equates to about 9 hours of experiments per volunteer.

Figure 4 enables us to conclude the following:

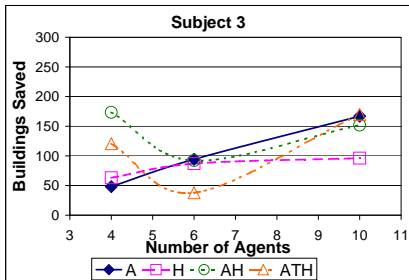
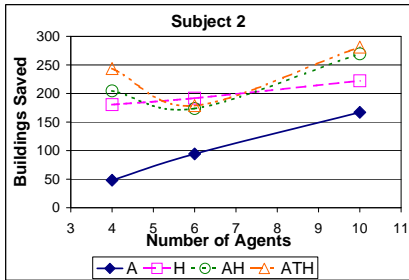
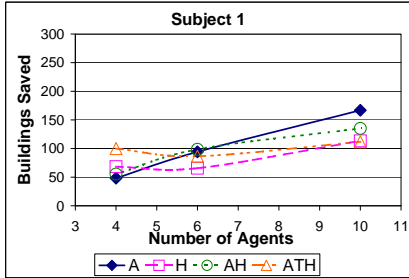


Fig. 4. Performance of subjects 1, 2, and 3.



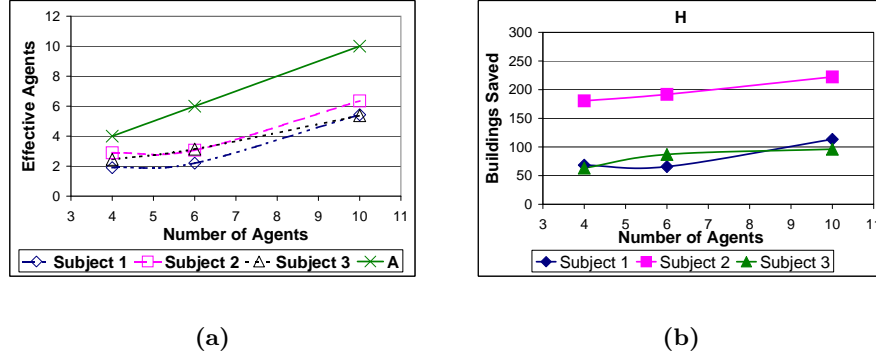
- *Human involvement with agent teams does not necessarily lead to improvement in team performance.* Contrary to expectations and prior results, human involvement does not uniformly improve team performance, as seen by human-involving strategies performing worse than the  $A_T$  strategy in some instances. For instance, for subject 3, human involving strategies such as  $AH$  provide a somewhat higher quality than  $A_T$  for 4 agents, yet at higher numbers of agents, the strategy performance is lower than  $A_T$ . While our strategy model predicted such an outcome in cases of *High B, Low  $EQ_H$* , the expected scenario was *High B, High  $EQ_H$* .
- *Providing more agents at a human’s command does not necessarily improve the agent team performance* As seen for subject 2 and subject 3, increasing agents from 4 to 6 given  $AH$  and  $A_TH$  strategies is seen to degrade performance. In contrast, for the  $A_T$  strategy, the performance of the fully autonomous agent team continues to improve with additions of agents, thus indicating that the reduction in  $AH$  and  $A_TH$  performance is due to human involvement. As the number of agents increase to 10, the agent team does recover.
- *No strategy dominates through all the experiments given varying numbers of agents.* For instance, at 4 agents, human-involving strategies dominate the  $A_T$  strategy. However, at 10 agents, the  $A_T$  strategy outperforms all possible strategies for subjects 1 and 3.
- *Complex team-level strategies are helpful in practice:*  $A_TH$  leads to improvement over  $H$  with 4 agents for all subjects, although surprising domination of  $AH$  over  $A_TH$  in some cases indicates that  $AH$  may also be a useful strategy to have available in a team setting.

Note that the phenomena described range over multiple users, multiple runs, and multiple strategies. The most important conclusion from these figures is that flexibility is necessary to allow for the optimal AA strategy to be applied. The key question is then whether we can leverage our mathematical model to select among strategies. However, we must first check if we can model the phenomenon in our domain accurately. To that end, we compare the predictions at the end of Section 3 with the results reported in Figure 4. If we temporarily ignore the “dip” observed at 6 agents in  $AH$  and  $A_TH$  strategies, then subject 2 may be modeled as a *High B, High  $EQ_H$*  subject, while subjects 1 and 3 modeled via *High B, Low  $EQ_H$* . (Figure 5-(b) indicates an identical improvement in H for 3 subjects with increasing agents, which suggests that B is constant across subjects.) Thus, by estimating the  $EQ_H$  of a subject by checking the “H” strategy for small number of agents (say 4), and comparing to  $A$  strategy, we may begin to select the appropriate strategy.

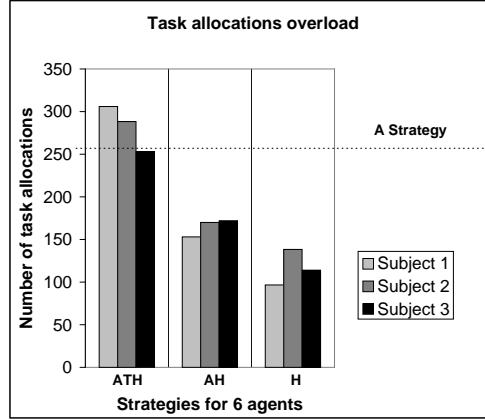
Unfortunately, the strategies including the humans and agents ( $AH$  and  $A_TH$ ) for 6 agents show a noticeable decrease in performance for subjects 2 and 3 (see Figure 4), whereas our mathematical model would have predicted an increase in performance as the number of agents increased (as seen in Figure 3). It would be useful to understand which of our key assumptions in the model has led to such a mismatch in prediction.

**Table 1.** Total amount of allocations given.

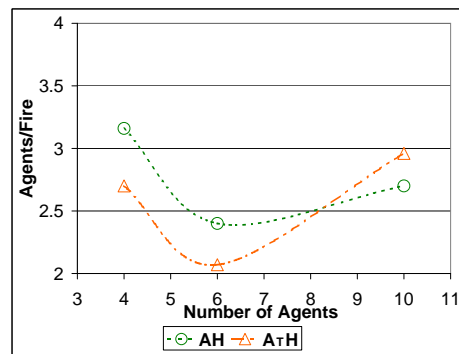
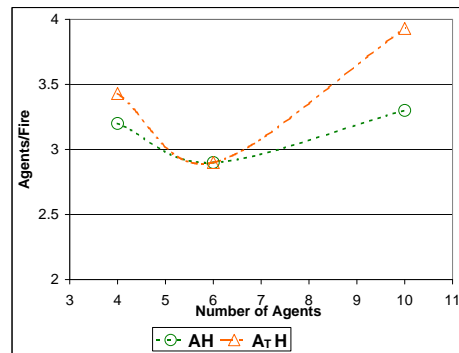
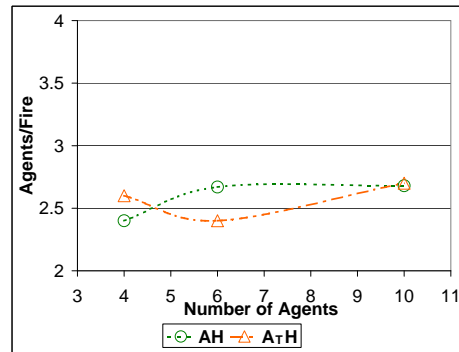
Strategy	$H$			$AH$			$A_TH$		
# of agents	4	6	10	4	6	10	4	6	10
Subject 1	91	92	154	118	128	132	104	83	64
Subject 2	138	129	180	146	144	72	109	120	38
Subject 3	117	132	152	133	136	97	116	58	57



**Fig. 5.** (a) The number  $AG_H$  of agents effectively used and (b) performance of the  $H$  strategy



**Fig. 6.** Task allocation overload for the team of 6 agents



**Fig. 7.** Number of agents per fire assigned by subjects 1, 2, and 3

The crucial assumptions in our model were that while numbers of agents increase,  $AG_H$  steadily increases and  $EQ_H$  remains constant. Thus, the dip at 6 agents is essentially affected by either  $AG_H$  or  $EQ_H$ . We first tested  $AG_H$  in our domain. The amount of effective agents,  $AG_H$ , is calculated by dividing how many total allocations each subject made by how many the  $A_T$  strategy made per agent, assuming  $A_T$  strategy effectively uses all agents. Figure 5-(a) shows the number of agents on the x-axis and the number of agents effectively used,  $AG_H$ , on the y-axis; the  $A_T$  strategy, which is using all available agents, is also shown as a reference. However, the amount of effective agents is actually about the same in 4 and 6 agents. This would not account for the sharp drop we see in the performance. We then shifted our attention to the  $EQ_H$  of each subject. One reduction in  $EQ_H$  could be because subjects simply did not send as many allocations totally over the course of the experiments. This, however is not the case as can be seen in Table 1 where for 6 agents, the total amount of allocations given is comparable to that of 4 agents. To investigate further, we checked if the quality of human allocation had degraded. For our domain, the more fire engines that fight the same fire, the more likely it is to be extinguished and in less time. For this reason, the number of agents that were tasked to each fire is a good indicator of the quality of allocations that the subject makes. Our model expected the number of agents that each subject tasked out to each fire would remain independent of the number of agents. Figure 7 shows the number agents on the x-axis and the average amount of fire engines allocated to each fire on the y-axis.  $AH$  and  $A_TH$  for 6 agents result in significantly less average fire engines per task (fire) and therefore less average  $EQ_H$ .

The next question is then to understand why for 6 agents  $AH$  and  $A_TH$  result in lower average fire engines per fire. One hypothesis is the possible interference among the agents' self allocations vs human task allocations at 6 agents. Table 2 shows the number of task changes for 4, 6 and 10 agents for  $AH$  and  $A_TH$  strategies, showing that maximum occurs at 6 agents. A task change occurs because an agent pursuing its own task is provided another task by a human or a human-given task is preempted by the agent. Thus, when running mixed agent-human strategies, the possible clash of tasks causes a significant increase task changes, resulting in the total amount of task allocations overreaching the number of task allocations for the  $A$  strategy (Figure 6). While the reason for such interference peaking at 6 may be domain specific, the key lesson is that interference has the potential to occur in complex team-level strategies. Our model would need to take into account such interference effects by not assuming a constant  $EQ_H$ .

**Table 2.** Task conflicts for subject 2.

Strategy	4 agents	6 agents	10 agents
$AH$	34	<b>75</b>	14
$A_TH$	54	<b>231</b>	47

The second aspect of our evaluation was to explore the benefits of the Navigation mode (3D) in the Omni-Viewer over solely an Allocation mode (2D). We performed 2 tests on 20 subjects. All subjects were familiar with the university campus. Test 1 showed Navigation and Allocation mode screenshots of the university campus to subjects. Subjects were asked to identify a unique building on campus, while timing each response. The average time for a subject to find the building in 2D was 29.3 seconds, whereas the 3D allowed them to find the same building in an average of 17.1 seconds. Test 2 again displayed Navigation and Allocation mode screenshots of two buildings on campus that had just caught fire. In Test 2, subjects were asked first asked to allocate fire engines to the buildings using only the Allocation mode. Then subjects were shown the Navigation mode of the same scene. 90 percent of the subjects actually chose to change their initial allocation, given the extra information that the Navigation mode provided.

## 5 Related Work and Summary

We have discussed related work throughout this paper, however, we now provide comparisons with key previous agent software prototypes and research. Given our application domains, Scerri et al’s work on robot-agent-person (RAP) teams for disaster rescue is likely the most closely related [13]. Our work takes a significant step forward in comparison. First, the omni-viewer enables navigational capabilities improving human situational awareness not present in previous work. Second, we provide a mathematical model based on strategies, which we experimentally verify, absent in that work. Third, we provide extensive experimentation, and illustrate that some of the conclusions reached in [13] were indeed preliminary, e.g., they conclude that human involvement is always beneficial to agent team performance, while our more extensive results indicate that sometimes agent teams are better off excluding humans from the loop. Human interactions in agent teams is also investigated in [15,2], and there is significant research on human interactions with robot-teams [5,3]. However they do not use flexible AA strategies and/or team-level AA strategies. Furthermore, our experimental results may assist these researchers in recognizing the potential for harm that humans may cause to agent or robot team performance. Significant attention has been paid in the context of adjustable autonomy and mixed-initiative in single-agent single-human interactions [7, 1]. However, this paper focuses on new phenomena that arise in human interactions with agent teams.

This paper addresses the issue of safety in multi-agent systems interpreted in the way that a team of agents ensures the safety of civilians or buildings in case of an emergency situation. To this end, we present a large-scale prototype, DEFACTO, that is based on a software proxy architecture and 3D visualization system and provides three key advances over previous work. First, DEFACTO’s Omni-Viewer enables the human to both improve situational awareness and assist agents, by providing a navigable 3D view along with a 2D global allocation view. Second, DEFACTO incorporates flexible AA strategies, even excluding humans from the loop in extreme circumstances. Third, analysis tools help pre-

dict the behavior of the agent team and choose the safest strategy for the given domain.

We performed detailed experiments using DEFACTO, leading to some surprising results. These results illustrate that an agent team must be equipped with flexible strategies for adjustable autonomy, so that they may select the safest strategy autonomously.

## References

1. J. F. Allen. The TRAINS project: A case study in building a conversational planning agent. *Journal of Experimental and Theoretical AI (JETAI)*, 7:748, 1995.
2. M. H. Burstein, A. M. Mulvehill, and S. Deutsch. An approach to mixed-initiative management of heterogeneous software agent teams. In *HICSS*, page 8055. IEEE Computer Society, 1999.
3. J.W. Crandall, C.W. Nielsen, and M. A. Goodrich. Towards predicting robot team performance. In *SMC*, 2003.
4. G. Dorais, R. Bonasso, D. Kortenkamp, P. Pell, and D. Schreckenghost. Adjustable autonomy for human centered autonomous systems on mars. In *Mars*, 1998.
5. T. Fong, C. Thorpe, and C. Baur. Multi-robot remote driving with collaborative control. *IEEE Transactions on Industrial Electronics*, 2002.
6. R. Hill, J. Gratch, S. Marsella, J. Rickel, W. Swartout, and D. Traum. Virtual humans in the mission rehearsal exercise system. In *KI Embodied Conversational Agents*, 2003.
7. E. Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems (CHI99)*, pages 159166, Pittsburgh, PA, May 1999.
8. H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjoh, and S. Shimada. Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *IEEE SMC*, volume VI, pages 739743, Tokyo, October 1999.
9. A. Paivio. Pictures and words in visual search. *Memory & Cognition*, 2(3):515521, 1974.
10. D. V. Pynadath and M. Tambe. Automated teamwork among heterogeneous software agents and humans. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, 7:71100, 2003.
11. A. Richardson, D. Montello, and M. Hegarty. Spatial knowledge acquisition from maps and from navigation in real and virtual environments. *Memory and Cognition*, 27(4):741750, 1999.
12. R. Ruddle, S. Payne, and D. Jones. Navigating buildings in desktop virtual environments: Experimental investigations using extended navigational experience. *J. Experimental Psychology - Applied*, 3(2):143159, 1997.
13. P. Scerri, D. Pynadath, and M. Tambe. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 17:171228, 2002.
14. P. Scerri, D. V. Pynadath, L. Johnson, P. Rosenbloom, N. Schurr, M. Si, and M. Tambe. A prototype infrastructure for distributed robot-agent-person teams. In *AAMAS*, 2003.
15. U. N. Suya You, Jinhui Hu and P. Fox. Urban site modeling from lidar. In *Proc. 2nd Intl Workshop Computer Graphics and Geometric Modeling (CGGM)*, pages 579588, 2003.

# Towards Using Simulation to Evaluate Safety Policy for Systems of Systems

Robert Alexander, Martin Hall-May, Georgios Despotou, and Tim Kelly

Department of Computer Science  
University of York, York, YO10 5DD, UK.  
{robert.alexander, martin.hall-may, georgios.despotou,  
tim.kelly}@cs.york.ac.uk

**Abstract.** The increasing role of Systems of Systems (SoS) in safety-critical applications establishes the need for methods to ensure their safe behaviour. One approach to ensuring this is by means of safety policy — a set of rules that all the system entities must abide by. This paper proposes simulation as a means to evaluate the effectiveness of such a policy. The requirements for simulation models are identified, and a means for decomposing high-level policy goals into machine-interpretable policy rules is described. It is then shown how the enforcement of policy could be integrated into a simple agent architecture based around a black-board. Finally, an approach to evaluating the safety of a system based using simulation techniques is outlined.

## 1 Introduction

Large-scale military and transport Systems of Systems (SoS) present many challenges for safety. Attempts to define the term ‘SoS’ have been controversial — attempts can be found in [1] and [2]. It is easy, however, to identify uncontroversial examples, Air Traffic Control and Network Centric Warfare being the most prominent. These examples feature mobile components distributed over a large area, such as a region, country or entire continent. Their components frequently interact with each other in an ad-hoc fashion, and have the potential to cause large-scale destruction and injury. It follows that for SoS that are being designed and procured now, safety has a high priority.

In order to ensure the safe behaviour of SoS, the behaviour of the individual system entities must be controlled, as must the overall behaviour that *emerges* from their individual actions and interactions. One way to achieve this is to impose a system-wide *safety policy*, which describes the rules of behaviour which agents in the system must obey. Due to the geographically distributed nature of many entities, the policy typically cannot be directly enforced by some external controller (as in security policy); rather, the entities must comply with it individually. Evaluating the effectiveness of such a decentralised policy is not straightforward in a complex system, since the overall safe behaviour only emerges from the behaviour of the entities themselves.

An SoS is a complex multi-agent system (MAS) in which many entities have a mobile physical presence. The agents within this MAS are in themselves very complex. This complexity means that formal analysis methods and conventional system safety techniques are not adequate for evaluating the safety of an SoS. In this paper, we propose simulation as a viable alternative.

Once the ‘baseline’ model of an SoS has been evaluated, the analysis can be repeated for a range of candidate safety policies. Based on the results of this analysis, candidate policies can be modified or discarded, and the process repeated until a satisfactory safety policy is found. We believe that simulation is a valuable tool for the development of SoS safety policy.

### 1.1 Structure of this Paper

The following section describes the problems faced in analysing and ensuring the safety of SoS. Section 3 introduces the concept of safety policy. Section 4 describes what is required from a simulation engine and simulation model. Section 5 outlines an approach to implementation of an SoS as a multi-agent simulation that satisfies the identified requirements. Section 6 highlights how safety cannot be considered in isolation from other dependability attributes. Section 7 describes some issues and challenges that need to be tackled, and section 8 presents a summary.

## 2 The Problem of SoS Safety Analysis

The Oxford English Dictionary [3] defines safety as “*The state of being safe; exemption from hurt or injury; freedom from danger.*” In system safety engineering, it is common to restrict the definition of ‘hurt or injury’ to the physical injury or death of humans. For the purposes of this paper, we will restrict ourselves to this definition. It can be noted, however, that the approach presented can easily be expanded to cover alternative conceptions of safety, such as those including avoidance of material loss.

The problems faced by safety analysts when attempting to analyse SoS fall into three categories: the difficulty of performing hazard analysis, the restricted means by which safety features can be introduced, and the problem of ‘System Accidents’. In their discussion of functional hazard analysis, Wilkinson and Kelly [4] note that these problems are present in conventional systems. The characteristics of SoS, however, exacerbate them.

### 2.1 Hazard Analysis

In a conventional system, such as a single vehicle or a chemical plant, the system boundary is well-defined and the components within that boundary can be enumerated. Once hazard analysis has been performed to identify events that may cause injury or death, safety measures can be introduced and the risk of



accidents computed from the probabilities of various failures. Conventional techniques such as fault tree analysis are effective in this task.

In an SoS, the necessary hazard analysis is itself very difficult. When hazard analysis postulates some failure of a component, the effect of that failure must be propagated through the system to reveal whether or not the failure results in a hazard. The system boundary is not well defined, and the set of entities within that boundary can vary over time, either as part of normal operation (a new aircraft enters a controlled airspace region) or as part of evolutionary development (a military unit receives a new air-defence system). Conventional tactics to minimise interactions may be ineffective, because the system consists of component entities that are individually mobile. In some cases, particularly military systems, the entities may be designed (for performance purposes) to form ad-hoc groupings amongst themselves. Conventional techniques may be inadequate for determining whether or not some failure in some entity is hazardous in the context of the SoS as a whole.

It follows from this that a hazard analysis approach is needed which can reveal hazards caused by failure propagation through complex systems and that can consider the effect of multiple simultaneous failures.

## 2.2 Ensuring Safety

A purely functional design with no safety features is unlikely to be adequately safe. Therefore, design changes need to be made in order to reduce safety risk to acceptable levels. In a conventional monolithic system, there are many features that can be introduced to prevent or mitigate hazards; examples include blast doors, interlocks, and pressure release valves.

The SoS that are considered here contain many mobile agents with a high degree of autonomy. Such ‘hard’ safety features are therefore not available. Consider, for example, air traffic control. If a controller wants to prevent a given aircraft from entering an airspace region (say, one reserved for an airshow) then he or she can instruct the aircraft to fly around it. The controller cannot, however, physically prevent the aircraft from flying into the region. (In a military scenario there are more drastic measures for dealing with aberrant agents, particularly if they are unmanned.)

Therefore, achieving safety in an SoS will rely to a large extent on responsible behaviour from the individual agents. In order to achieve this, agents need to know what behaviour is acceptable in any given circumstance. It follows from this that system designers and operators need to know how the agents in the system can safely interact.

## 2.3 System Accidents

Perrow, in [5], discusses what he calls ‘normal accidents’ in the context of complex systems. His ‘Normal Accident Theory’ holds that any complex, tightly-coupled system has the potential for catastrophic failure stemming from simultaneous minor failures. Similarly, Leveson, in [6] notes that many accidents have

multiple necessary causes; in such cases it follows an investigation of any one cause *prior to the accident* (i.e. without the benefit of hindsight) would not have shown the accident to be plausible.

An SoS can certainly be described as a ‘complex, tightly-coupled system’, and as such is likely to experience such accidents. It can also be noted that a ‘normal accident’ could result from the combination of apparently safe, normal behaviours which are safe in isolation but hazardous in combination. Imagine, for example, a UAV that aggressively uses airspace and bandwidth under some circumstances. This may be safe when the UAV is operating on its own, but not when it is part of larger SoS.

It follows from this that an SoS safety analysis approach will need to be able to capture the effects of interactions between multiple simultaneous failures and normal agent behaviour.

### 3 What is Safety Policy?

#### 3.1 Background on Policy

The belief that numerous independently designed and constructed autonomous systems can work together synergistically and without accident is naïve, unless they are operating to a shared set of rules which is informed by a high level view of the system. In existing systems of systems such rules already exist, to a degree, because otherwise such systems would be nothing more than an uncoordinated collection of parts. Burns, in [7]: *“The proper functioning of the network as a whole is a result of the coordinated configuration of multiple network elements whose interaction gives rise to the desired behaviours.”*

The problem that we face, however, is that these rules or procedures are often either not explicitly expressed, not well understood or are inconsistent. Similarly, they typically do not consider the inter-operating systems as a whole SoS, or simply do not address the safety aspects arising from this inter-operation. A term that can be used to encompass such rules and procedures is *policy*. Whilst some existing work covers security policy, no work yet deals with a policy for the safe operation of a system of systems.

The Oxford English Dictionary [3] defines ‘policy’ as:

*“A course of action or principle adopted by a government, party, individual, etc.; any course of action adopted as advantageous or expedient.”*

Intuitively, therefore, a policy guides the action of an individual or group according to some criteria. Foreign policy, for example, is a familiar concept from everyday language and sets out ground rules for guiding a nation’s diplomatic interactions with other nations. Similarly, criminal law lays down rules to curtail undesirable behaviour and promote desirable behaviour amongst the populace.

Much of government policy, however, confuses policy with ‘goal-setting’. Although some definitions of policy mention goals, they are in the context of policy goals, or high-level actions, such as “the system is to operate safely at all times”

or “no University applicant should be discriminated against based on his/her ability to pay tuition fees”, as distinct from targets, e.g. “to ensure 50% of school-leavers continue to higher education”. Policy can therefore be thought of as being *orthogonal*, but complementary, to plans and goals.

Policy is defined in the literature in various ways, but the most generally applicable system-oriented definition is given in [8]:

*“A policy is a rule that defines a choice in behaviour of a system.”*

This definition is distinct from that used in, for example, reinforcement learning, where a prescriptive policy maps from perceived internal state to a set of actions. Indeed, it can be seen that policy is *persistent* [9]; policy is not a single action which is immediately taken, because a policy should remain relatively stable over a period of time. Any policy containing one-off actions is brittle, in that it cannot be reused in a different context and quickly becomes out-of-date and invalid.

Most organisations issue policy statements, intended to guide their members in particular circumstances [9]. Some provide positive guidance, while others set out constraints on behaviour. To take a simple example as an illustration, consider a mother who asks her child to go to the corner shop to buy a pint of milk. She may lay down two rules with which the child must comply on this trip:

1. The child must not talk to strangers.
2. The child must use the pedestrian crossing when crossing the road.

The first of these rules defines what the child is allowed to do, specifically it proscribes conversation with people with whom the child is not previously acquainted. The second statement expresses the obligation that the child should take a safe route across the road, namely by using the pedestrian crossing. Together these rules form a policy that guides the behaviour of the child on his journey to the corner shop. The rules are invariant to the child’s ‘mission’; they still hold whether the child is going to buy a loaf of bread or a dozen eggs, or not going to the corner shop at all.

### 3.2 Systems of Systems and Safety Policy

According to Bodeau [10], the goal of SoS engineering is “*to ensure the system of systems can function as a single integrated system to support its mission (or set of missions).*” Among the principle concerns of SoS engineering that Bodeau identifies are interoperability, end-to-end performance, maintainability, reliability and security. Unfortunately, he neglects to mention safety.

Wies, in [11], describes policy as defining the *desired* behaviour of a system, in that it is a restriction on the *possible* behaviour. Leveson extends this sentiment to say that the limits of what is possible with today’s (software-based) systems are very different to the limits of what can be accomplished *safely* [6]. In terms of collaborative groups of systems, SoS, whose behaviour has been observed to be

non-deterministic, a policy is a mechanism to create order or (relative) simplicity in the face of complexity. Sage and Cuppan [12] talk of “*abandoning the myth of total control*”, while Clough [13] describes it as creating a system that is “*deterministic at the levels that count*”, i.e. at the ‘black-box’ level, and Edwards [14] observes the need to “*selectively rein in the destructive unpredictability present in collaborative systems*”.

In discussing policy many different terms are employed, such as rule, procedure, convention, law and code of conduct. The presence of so many terms would seem to suggest a lack of clarity about what policy is, but these terms can be viewed as policy *at different levels of abstraction*. Often policy specifications cause confusion by combining statements at high and low levels of abstraction [11].

Policy statements or goals can be organised into a hierarchy, with the most abstract at the top. There is a need to refine from these abstract policies down to implementable, atomic procedures. Existing goal-oriented techniques and notations, such as GSN [15], KAOS [16] and TROPOS [17], provide a basis for the decomposition of high-level goals. Specifically, the Goal Structuring Notation (described by Kelly in [15]) allows the explicit capture of contextual assumptions, for example assumptions made about other agents’ behaviour, and of strategies followed to perform the decomposition.

At the lowest level of abstraction policies can be expressed in terms of the permissions, obligations and prohibitions of individual and groups of agents. In this paper, an approach is suggested for decomposing and implementing policy goals motivated by safety concerns in a simulation of an SoS. The effect of this policy is to moderate the behaviour of the agents such that no accidents occur in the simulated SoS. The safety policy decomposition process and its relation to agent models are explored further in [18, 19].

## 4 Requirements on the Simulation Engine and Models

Multi-agent Simulation has previously been used in a safety context, for example to evaluate the safety of proposed changes to the US National Airspace System [20] and to study the relationship between road intersection layout and automobile accidents [21]. As noted by Ferber in [22], such simulations “*make it possible to model complex situations whose overall structures emerge from interactions between individuals*”.

However, not all multi-agent simulations are suitable for safety analysis. In order to perform safety analysis using simulation, there are two key requirements that must be satisfied by the simulation environment and the models that it contains. Firstly, the simulation must be able to generate the types of hazards and accidents that are of concern, without the emergent system behaviour being described in advance. Secondly, it must be possible to detect these situations when they occur.

For example, consider a system to be analysed that involves flocking Unmanned Air Vehicles (UAVs). Given a description of how the entities behave, in

terms of flight control, attempting to achieve mission goals, and collision avoidance, it must be possible to run a simulation of a typical mission scenario and see what flight paths the entity behaviour would generate. It must also be possible to detect whether these flight paths would lead to collisions, or hazardous loss of separation.

From the general requirements above, and by looking at the nature of the accidents we are concerned with, a number of more detailed requirements can be derived. These requirements are discussed in the following sections.

#### **4.1 Sharing of Physical Space**

Safety-critical accidents must, by their nature, occur in physical space. At the point of an accident, it is through physical interaction that humans are injured or killed. It follows that a safety-related simulation must have a clearly-defined model of space and time interactions. Models that abstract away such details (e.g. by maintaining only relative time ordering, or by dividing geography into large, arbitrarily shaped regions) will not be able to capture the necessary interactions.

It can be noted that although physical space is needed to actually effect an accident, many accidents have causes which can be traced back to events and interactions at the control system or communication levels.

#### **4.2 Autonomous Entity Behaviour**

The SoS that are of concern to us involve entities with a large degree of autonomy. Many SoS that are being developed now feature unmanned vehicles, and their autonomous behaviour is an important issue for safety analysis. Negative emergent behaviour, resulting from the interaction of many such vehicles, is a particular concern. It is therefore important to model autonomous behaviour. Autonomous agents are also needed in order to simulate deviation from expected scenario courses; entities must be able to make plausible decisions and actions once the situation has departed from the expected course of events.

The simulation cannot, therefore, rely on a single centralised plan of action. The entity models must be capable of some degree of planning and decision-making so as to achieve their goals in the face of unexpected obstacles.

#### **4.3 Local and Shared Entity World Views**

A common cause of accidents in many systems is a discrepancy between the mental model of one agent (be it a UAV or a chemical plant worker) and the actual state of the world. Each agent has a local world model based on the information that they have perceived directly, that they have received in communication from others, and that they have inferred from the information from the other two sources. For example, an airline pilot can observe other aircraft in their immediate area, can receive notification from an air traffic controller of upcoming weather obstacles, and can infer from the ATC's instructions that the course they have been placed on is free from either.

Increasingly, automated systems are used to share data between agents in a system. Examples include air traffic control centres exchanging data on aircraft that are moving from one region to another, and a group of fighter aircraft having access to the combined vision cones of all their radars (this is sometimes referred to as ‘data fusion’). This exchange provides many benefits (potentially including safety benefits, as agent knowledge is increased), but also raises new kinds of hazards. For example, if an agent misidentifies a friendly aircraft as hostile, a data fusion system may propagate that ‘knowledge’ to many other agents, some of whom may be in position to threaten the friendly aircraft.

#### **4.4 Communication Between Entities**

As mentioned above, entities can supplement their world model through communication with other agents. Communication also incorporates command and control relationships, which affect the behaviour of subordinate agents. Errors in communication may, consequently, cause accidents either by modifying an agent’s world model or by instructing the agent to perform an unsafe action.

#### **4.5 Proxy Measures of Safety**

Although a simulation model may generate explicit accidents, a safety analyst cannot rely on this. As in the real world, accidents in a well-modelled simulated SoS will be rare; they will be avoided due to subtleties of time and distance. For example, a collision between a UAV and a manned aircraft may be repeatedly avoided in a series of different runs, with the two aircraft coming close to collision but never actually colliding. For the case of policy, the number and severity of accidents is therefore too crude a measure for the safety of a given policy. There is therefore a need for surrogate measures (e.g. counting near misses rather than just collisions), offering greater resolution than a simple casualty count.

#### **4.6 Introducing Expected Variation**

In a model that is solely concerned with performance, for example, it may be sufficient to capture only average performance over time. For a safety model, this is not sufficient; specific, high-cost events must be captured. Therefore, simulations must not only be performed with idealised models of expected entity behaviour; they must also cover all anticipated failure modes. For example, it must be possible to specify that a UAV included in a simulated system has no functioning IFF (Identify Friend-or-Foe) capability, or that one of its engines is operating at reduced maximum thrust.

Going beyond simple failures, it is also desirable to be able to implement different behaviours. Each entity has a set of default behaviours, and the developer of an entity model may provide a set of optional or alternative behaviours. By swapping behaviours in and out from this larger set, variations in overall entity behaviour can be introduced that may affect the results of the simulation run.

An example would be swapping a cautious target identification behaviour for a more aggressive one that made fewer checks before deciding that an entity was hostile.

## 5 Implementing a Multi-Agent Simulation of an SoS

### 5.1 Describing Policy in a Machine-Interpretable Form

Policy has to be implemented by individual agents — even if there is a central ‘master controller’ for the whole system, in the systems we are dealing with it will not be able to enforce policy by fiat. Therefore policy has to be decomposed into rules that are expressed in terms of individual agent behaviour. It follows that for any given entity type, policy must be expressed such that it involves:

- Responding to states or events that the agent is capable of observing
- Making decisions that are within the scope of the agent’s intelligence and world model (this is particularly important for non-human agents)
- Taking actions that the agent is capable of performing

To this end, policy is decomposed in the context of an SoS model. This model embodies the contextual assumptions of other agents’ behaviour, knowledge and capabilities. Policy decomposition proceeds with increasing specificity, working top-down from a high-level goal to policy statements on individual agents or sets of agents. Goal Structure Notation (see section 3.2) allows the explicit capture of the strategies by which the decomposition is achieved and the context necessary for such a decomposition.

Figure 1 illustrates an excerpt from a possible policy hierarchy for the UK civil aerospace Rules of the Air [23]. The policy decomposition starts from an obvious high-level goal, ‘No collisions shall occur in the civil aviation SoS’, which is at the top of the diagram. This goal is then decomposed hierarchically, leading eventually to a number of low-level goals (leaf nodes on the diagram; due to space limitations, only two of these are shown). These lowest-level goals correspond to policy rules that can be implemented directly by agents; in the diagram, the goal ‘Below1000’ has been annotated by a machine-interpretable version of the corresponding policy rule.

The low-level policy statements are expressed as one of three types:

**Permit** Describes the actions that an agent is permitted to perform or conditions that it is permitted to satisfy.

**Forbid** Describes the actions that an agent is forbidden to perform or conditions that it is forbidden to satisfy.

**Oblige** Describes the actions that an agent is obliged to perform.

In contrast to policy definition languages such as Ponder [24], we do not attempt to define those actions which an agent is forbidden to perform as well as those which it is obliged not to perform. As mentioned in section 2.2 it is not

always possible to prevent agents from performing actions contrary to policy. Unlike security policies, which often assume the presence of an access control monitor, safety policy cannot assume that aberrant agent behaviour can be blocked by an external controller. For example, in air traffic control, there is no external way to stop a wayward aircraft from straying into a forbidden region of airspace. In a sense, the system operator must rely on agents to police themselves.

There must also be a design decision about the overall permissiveness of the SoS. A policy model can either be open or closed: the former allowing all actions which are not expressly forbidden, while the latter forbids all those actions that are not explicitly permitted. The presence of both permit and forbid in this policy model would therefore appear redundant. This is not so, however, given that exceptions to rules can be expressed in the opposite modality. For instance, in an open policy model, a policy rule may forbid the low flying of an aircraft; exceptions to this rule (e.g. for take-off and landing) can be expressed as permissions. The more specific permissions must then take precedence over the more general blanket policy to forbid low flying.

## 5.2 Implementation in an Agent Architecture

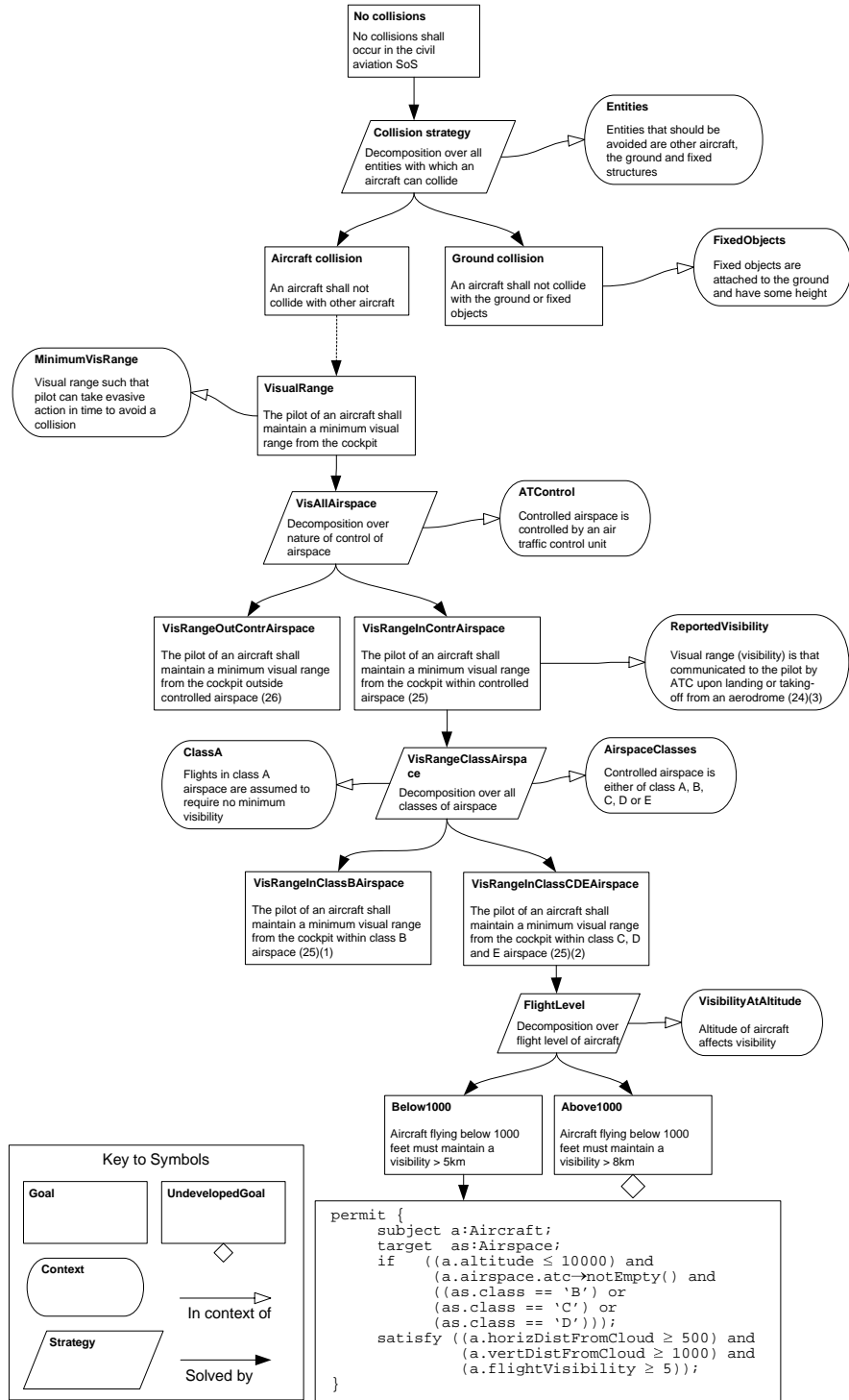
The implementation of policy at the agent level is tied closely to the details of the agent architecture used. In the current work, an architecture is proposed based on the ‘C4’ architecture developed by the Synthetic Characters group at the MIT Media Lab. This is a blackboard architecture, in that it contains a series of subsystems that communicate only through a structured shared memory space. C4 is described by Isla et al in [25]. The blackboard architecture is valuable in that it allows discrete behaviours to be loosely coupled, and hence allows variant behaviours to be easily swapped in and out as described in section 4.6.

Our proposed architecture is depicted in figure 2. The core of the system is the blackboard, which is divided into several sub-boards. Of particular note is the outgoing action board, which determines what the agent actually does after each decision cycle. Each agent has several behaviours, which act by making changes to the blackboard (including the action space). The arbitration strategy is simple — on each time ‘tick’, all behaviours are processed in turn (from top to bottom in the diagram).

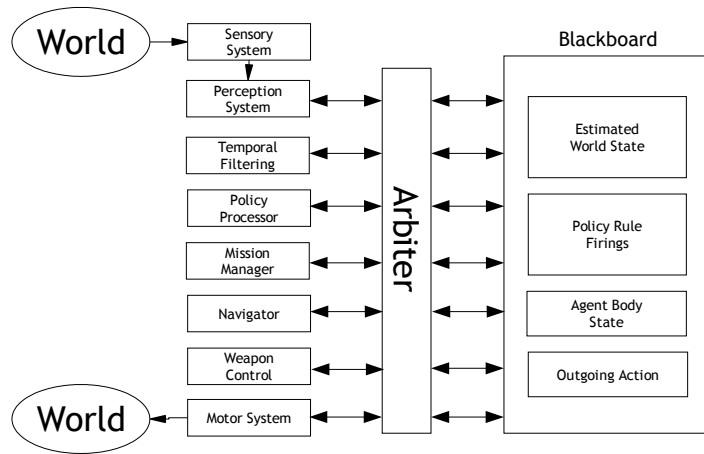
The arbiter also has a role in enforcing adherence to safety policy. It can be seen that one of the first behaviours to be processed is the Policy Processor, which compares the current policy to the current state and ‘fires’ all the policy rules that apply. This generates a set of permitted and forbidden actions, which is written to the policy sub-board. This sub-board is hereafter read-only — the other behaviours can observe it, in order that they might propose only permitted actions, but they cannot change it. The policy sub-board is regenerated on each tick, as changes in the environment may change which policy rules now apply.

Policy rule firings generate tuples of the form (operator, action, list of parameters). For example:





**Fig. 1.** Example Policy Decomposition for Rules of the Air



**Fig. 2.** The Agent Architecture

- (FORBID, change-speed, < 180 knots)
- (FORBID, enter-region, 2000m radius of [15000,5100])
- (PERMIT, attack-target, entity#127)

The rule-firings and the behaviours use the same ontology. As noted above, behaviours can see which PERMIT and FORBID policy rules are active at the current time, and modify their behaviour accordingly. As a supplement to this, or an alternative, the arbiter may check proposed actions (against the policy board) and reject those that are against the rules. This could seem redundant, since the behaviours are part of the agent, and hence as much a trusted source as the arbiter itself. It is easier, however, to build a reliable policy enforcer than it is to build behaviours that always conform to policy. Likewise, it is easier to build a behaviour that chooses its action taking into account what policy currently permits, rather than build one that tries whatever action seems best, then tries to respond when the action is forbidden.

An alternative to the policy enforcement role of the arbiter is that of a monitor, which notes policy violations but does not prevent them. This is particularly valuable during development of an agent.

An advantage of the blackboard model is that the behaviours are loosely coupled to each other; they interact only through the blackboard. This means that behaviours can be added, removed or changed at any time, without changing the other behaviours. This relates to the requirements identified in section 4.6.

### 5.3 Evaluating the Safety Achieved by the Policy

In order to evaluate the level of safety that has been achieved, the SoS agents must be configured to use the policy, then simulation runs must be performed for

a variety of representative scenarios. Further variation can be introduced through failures and variant behaviours applied to agents. The level of safety achieved by the system operating with the policy can be evaluated by measuring the number of accidents and incidents, and the worst near-incidents, that occurred across all runs with that policy.

Once the SoS model has been configured and the set of runs decided on, measures must be put in place to measure the level of safety achieved by the system. From the definition of safety presented in section 2, it can be seen that two types of event need to be counted.

The first type is accidents, which corresponds to “*exemption from hurt or injury*” in the definition. Examples of such accidents include collisions between vehicles and military units firing on friendly forces. The set of possibilities is quite small, and they can be easily detected.

From “*freedom from danger*” we can derive another class of event, the incident or ‘near miss’. Examples include activations of an aircraft’s collision-avoidance system, separation between two aircraft falling below some safe level, and queries to a superior of the form “Is X hostile?” when X is in fact friendly. Unlike actual accidents, a great many types of such incidents can be described. It can be noted that many incidents correspond to *hazards*; when an incident occurs, it may be the case that an accident could happen without any other deviation from normal behaviour.

The great value of counting incidents as opposed to accidents is that accidents are extremely rare — in the real world, accidents are often avoided by (seemingly) sheer chance. Measures that track incidents can be given variable sensitivity, so that they can be adapted to the level of risk that is actually exhibited in the system. For example, it is desirable to calculate actual collisions accurately, so that their effects on the unfolding scenario can be modelled realistically. This is especially true if a multi-criteria analysis is being performed, for example with performance as well as safety being analysed. By comparison, an incident measure based on aircraft proximity can be as sensitive (or insensitive) as is required since triggering it only affects the statistics gathered, not the events that follow it.

For both accidents and incidents, it is possible to weight events by a measure of their severity. Consider one policy that generated a number of minor accidents against another that caused a single accident with massive loss of life. A simple approach is to count the human casualties (or potential casualties, in the case of an incident) that could result from an event. The safest policy is then the one that caused the smallest loss of simulated lives over all the scenarios that were considered. Weighing accidents against incidents is more difficult, however; there is the question here of model fidelity, and the consequent fear that an incident in the simulation might have been an accident in the real system.

The means of detecting accidents during a simulation run are well understood as they are an essential part of many non-safety simulations. Providing a large range of incident detectors is less straightforward, and some of these will raise performance challenges. This is, however, beyond the scope of this paper.

If two policies cannot be compared because their accident and incident counts are zero or very low, a third technique is possible. For a variety of measures, perhaps the same measures as those used for incidents, the worst magnitude achieved could be tracked. An obvious example is violation of aircraft separation; rather than just counting the number of occasions on which this occurred, the minimum separation achieved can be recorded. The minimum for the policy is then the minimum over all runs. An example of this can be seen in Benson [26].

## 6 Dependability Conflicts in Systems of Systems

Safety is an important system attribute, but it is not the only consideration when developing an SoS. There are other important attributes such as availability, performance and security. The term *dependability* is commonly used to encompass all such system attributes [27]. Attempting to address all these different attributes can result in competing objectives; consequently there are conflicts that need to be resolved and trade-offs that need to be made in order to achieve the optimum characteristics for a system.

In SoS, conflicting objectives (and hence trade-offs) are inevitable; probably the most obvious are conflicts between performance and safety. An example is the reduction of minimum aircraft vertical separation (RVSM), within controlled airspace. In RVSM airspace, aircraft fly closer to each other, greatly increasing the number of aircraft that can be served by an ATC centre within a certain period of time. This has obvious performance benefits (reduction of delays, more flights during peak hours), but it raises some serious safety concerns. Special safeguards (changes to either sub-system design or operational policies) are therefore required.

If an SoS is developed with safety as the highest priority, it will be possible to devise policies that constrain the interactions of system agents to the safest that are possible. However, such an approach might unacceptably decrease the overall performance of the system. For example, there is no point in introducing an extremely safe air traffic policy if doing so reduces the throughput of the system to uneconomic levels. In order that safety is not achieved at the unacceptable detriment of other attributes, it is important to model the effect on all attributes of safety-related changes.

Performance acceptability criteria differ depending on the particular system mission. Therefore, the required performance level and its criticality (based on which we determine our willingness to compromise safety in favour of performance) are defined with consideration of the system's context and operational objectives. Establishing a dependability case about the system can be a means of communicating an argument about the acceptable achievement of the required system properties [28].

Simulation provides a way to evaluate the different dependability attributes of the system in different contexts, by running a set of representative operational scenarios. This provides a basis for achieving a satisfactory trade-off between the different attributes.

## 7 Issues and Challenges

### 7.1 Model Fidelity and Statistical Significance

No novel and untried system of systems will enter operation with only simulated evidence of its safety. Simulation, however, gives a guide to the behaviour of the system which informs, and is supplemented by, further analysis. It is particularly valuable in that it can reveal the emergent behaviour of a complex system in a variety of contexts; it is difficult if not impossible to acquire knowledge of this by other means.

Even when the fidelity of a given simulation is considered inadequate to assess the safety of a system, it can provide confidence that a given policy is viable, and help judge relative superiority to other candidates. (For an example of this, see Benson in [26]). Perhaps most importantly, the simulation analysis can reveal flaws in a policy that would not have been apparent in manual analysis.

The problem of model fidelity, and of the validity of any results that are gained through simulation, is a serious one and affects all applications of simulation analysis, not just safety. This is a longstanding controversy in the field of robotics; discussion can be found in Brooks [29] and Jakobi [30]. In the current context, one key requirement for usefulness is that the simulation be able to exhibit emergent behaviour.

### 7.2 Volume of Processing

As noted above in section 5.3, evaluating the safety of the system requires a large number of scenarios to be simulated. For each of those simulations, a large range of failures and variant behaviours need to be considered. Combinations of failures and behaviours are also important.

It follows that the possible set of simulation runs is extremely large. A naïve approach would be to run all possible combinations of scenario, failures and behaviours. However, as discussed by Hoeber in [31], such exhaustive exploration is intractable even for simple simulations and modest numbers of inputs.

There is therefore a need for more targeted exploration of the state space. In [32], Dewar et al discuss some experimental designs that can be used for reducing the number of combinations that need to be run. Many such designs, however, deal poorly with systems in which the interesting phenomena result from combinations of inputs.

One other approach would be to concentrate on and prioritise those combinations of failures that were statistically most likely. A useful selection criteria can be based on the potential for certain types of SoS failures to occur together, as discussed by the authors in [33].

With a large volume of processing comes a large volume of associated output, in the form of event logs from the simulation runs. Policy developers must understand and comprehend the causal relationships expressed in this output if they are to refine their policy, but this not likely manually tractable. Some form of automated assistance will therefore be required. An approach to this using machine learning is presented in [34].

## 8 Summary

In this paper, we have presented the case for using safety policy to ensure the safe behaviour of complex systems of systems, and suggested multi-agent simulation as a means of evaluating the effectiveness of such policies. It is clear that analysing SoS is difficult, particularly when they are highly decentralised. Simulation offers an approach to dealing with some of these difficulties.

An approach to policy evaluation has been proposed, whereby an SoS is exercised through a variety of simulations for each candidate policy, with a range of failures and behaviour modifications being introduced. The level of safety provided by each policy can be assessed by measuring the values of various safety-related parameters. This concept can be extended further, using simulation to consider the trade-off between safety and performance.

A number of challenges remain, such as limitations in the fidelity of models and the number of runs needed to get statistically valid results. The authors are currently working on tools and examples to demonstrate the concepts described in this paper.

**Acknowledgements** The work described in this paper was funded under the Defence and Aerospace Defence Partnership in High Integrity Real Time Systems (Strand 2).

## References

1. Maier, M.W.: Architecting principles for systems-of-systems. In: 6th Annual Symposium of INCOSE. (1996) 567–574
2. Periorellis, P., Dobson, J.: Organisational failures in dependable collaborative enterprise systems. *Journal of Object Technology* **1** (2002) 107–117
3. Simpson, J., Weiner, E., eds.: *Oxford English Dictionary*. Second edn. Oxford University Press (1989)
4. Wilkinson, P.J., Kelly, T.P.: Functional hazard analysis for highly integrated aerospace systems. In: *IEE Seminar on Certification of Ground / Air Systems*, London, UK (1998)
5. Perrow, C.: *Normal Accidents: Living with High-Risk Technologies*. Basic Books, New York (1984)
6. Leveson, N.G.: A new accident model for engineering safer systems. *Safety Science* **42** (2004) 237–270
7. Burns, J., Cheng, A., Gurung, P., Rajagopalan, S., Rao, P., Rosenbluth, D., Surendran, A.V., Martin, Jr, D.M.: Automatic management of network security policy. In: *Proceedings of the DARPA Information Survivability Conference and Exposition*. Volume 2., Anaheim, California, USA, IEEE Computer Society (2001) 1012–1026
8. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: Managing security in object-based distributed systems using Ponder. In: *Proceedings of the 6th Open European Summer School (Eunice 2000)*, Twente University Press (2000)
9. Moffett, J.D., Sloman, M.S.: The representation of policies as system objects. In: *Proceedings of the Conference on Organizational Computing Systems*, Atlanta, Georgia, USA, ACM Press (1991) 171–184

10. Bodeau, D.J.: System-of-systems security engineering. In: Proceedings of the 10th Annual Computer Security Applications Conference, Orlando, Florida, USA, IEEE Computer Society (1994) 228–235
11. Wies, R.: Using a classification of management policies for policy specification and policy transformation. In Sethi, A.S., Raynaud, Y., Fure-Vincent, F., eds.: Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management. Volume 4., Santa Barbara, California, USA, Chapman & Hall (1995) 44–56
12. Sage, A.P., Cuppan, C.D.: On the systems engineering and management of systems of systems and federations of systems. *Information, Knowledge, and Systems Management* **2** (2001) 325–345
13. Clough, B.T.: Autonomous UAV control system safety—what should it be, how do we reach it, and what should we call it? In: Proceedings of the National Aerospace and Electronics Conference 2000, Dayton, Ohio, USA, IEEE Computer Society (2000) 807–814
14. Edwards, W.K.: Policies and roles in collaborative applications. In: Proceedings of the Conference on Computer-Supported Cooperative Work, Cambridge, Massachusetts, USA, ACM Press (1996) 11–20
15. Kelly, T.P.: Arguing Safety—A Systematic Approach to Managing Safety Cases. Dphil thesis, University of York, Heslington, York, YO10 5DD, UK (1998)
16. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. *Science of Computer Programming* **20** (1993) 3–50
17. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: Tropos: An agent-oriented software development methodology. *Journal of Autonomous Agents and Multi-Agent Systems* **8** (2004) 203–236
18. Hall-May, M., Kelly, T.P.: Defining and decomposing safety policy for systems of systems. In Wither, R., Gran, B.A., Dahll, G., eds.: Proceedings of the 24th International Conference on Computer Safety, Reliability and Security (SAFECOMP '05). Volume 3688 of LNCS., Fredrikstad, Norway, Springer-Verlag (2005) 37–51
19. Hall-May, M., Kelly, T.P.: Using agent-based modelling approaches to support the development of safety policy for systems of systems. In Gorski, J., ed.: Proceedings of the 25th International Conference on Computer Safety, Reliability and Security (SAFECOMP '06). Volume 4166 of LNCS., Gdansk, Poland, Springer-Verlag (2006) 330–343
20. Lee, S., Pritchett, A., Goldsman, D.: Hybrid agent-based simulation for analyzing the national airspace system. In Peters, B.A., Smith, J.S., Madeiros, D.J., Rohrer, M.W., eds.: Proceedings of the 2001 Winter Simulation Conference. (2001) 1029–1037
21. Archer, J.: Developing the potential of micro-simulation modelling for traffic safety assessment. In: Proceedings of the 13th ICTCT Workshop. (2000) 233–246
22. Ferber, J.: Multi-Agent Systems: an Introduction to Distributed Artificial Intelligence. Addison-Wesley (1999)
23. Allan, R., ed.: Air Navigation: The Order and the Regulations. third edn. Civil Aviation Authority (2003)
24. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: Ponder: A language for specifying security and management policies for distributed systems. Research Report DoC 2000/1, Imperial College, London (2000) <http://www.doc.ic.ac.uk/deptechrep/DTR00-1.pdf>.
25. Isla, D., Burke, R., Downie, M., Blumberg, B.: A layered brain architecture for synthetic creatures. In: Proceedings of the International Joint Conference on Artificial Intelligence, Seattle, WA (2001)

26. Benson, K.C., Goldsman, D., Pritchett, A.R.: Applying statistical control techniques to air traffic simulations. In Ingalis, R.G., Rosetti, M.D., Smith, J.S., Peters, B.A., eds.: *Proceedings of the 2004 Winter Simulation Conference*. (2004) 1330–1338
27. Avizienis, A., Laprie, J., Randell, B.: Dependability of computer systems: Fundamental concepts, terminology and examples. In: *Proceedings of the IARP/IEEE-RAS Workshop on Robot Dependability*, Seoul (2001)
28. Despotou, G., Kelly, T.: An argument based approach for assessing design alternatives and facilitating trade-offs in critical systems. In: *Proceedings of the 24th International System Safety Conference (ISSC)*, Albuquerque, NM, USA, Systems Safety Society (2006)
29. Brooks, R.A.: Intelligence without representation. *Artificial Intelligence* **47** (1991) 139–159
30. Jakobi, N., Husbands, P., Harvey, I.: Noise and the reality gap: The use of simulation in evolutionary robotics. *Lecture Notes in Computer Science* **929** (1995)
31. Hoerber, F.P.: *Military Applications of Modeling: Selected Case Studies*. Gordon & Breach Science Publishers (1981)
32. Dewar, J.A., Bankes, S.C., Hodges, J.S., Lucas, T., Saunders-Newton, D.K., Vye, P.: Credible uses of the distributed interactive simulation (DIS) system. Technical Report MR-607-A, RAND (1996)
33. Alexander, R., Hall-May, M., Kelly, T.: Characterisation of systems of systems failures. In: *Proceedings of the 22nd International Systems Safety Conference (ISSC 2004)*, System Safety Society (2004) 499–508
34. Alexander, R., Kazakov, D., Kelly, T.: System of systems hazard analysis using simulation and machine learning. In Gorski, J., ed.: *Proceedings of the 25th International Conference on Computer Safety, Reliability and Security (SAFECOMP '06)*. Volume 4166 of LNCS., Gdansk, Poland, Springer-Verlag (2006) 1–14



# Safe Agents in Space: Preventing and Responding to Anomalies in the Autonomous Sciencecraft Experiment

Daniel Tran, Steve Chien, Gregg Rabideau, Benjamin Cichy  
Jet Propulsion Laboratory, California Institute of Technology  
Firstname.Lastname@jpl.nasa.gov

**Abstract.** This paper describes the design of the Autonomous Sciencecraft Experiment, a software agent that has been running on-board the EO-1 spacecraft since 2003. The agent recognizes science events, retargets the spacecraft to respond to the science events, and reduces data downlink to only the highest value science data. The autonomous science agent was designed using a layered architectural approach with specific redundant safeguards to reduce the risk of an agent malfunction to the EO-1 spacecraft. The agent was designed to be “safe” by first preventing anomalies, then by automatically detecting and responding to them when possible. This paper describes elements of the design that increase the safety of the agent, several of the anomalies that occurred during the experiment, and how the agent responded to these anomalies.

## 1 Introduction

Autonomy technologies have incredible potential to revolutionize space exploration. In the current mode of operations, space missions involve meticulous ground planning significantly in advance of actual operations. In this paradigm, rapid responses to dynamic science events can require substantial operations effort. Artificial intelligence technologies enable onboard software to detect science events, re-plan upcoming mission operations, and enable successful execution of re-planned responses. Additionally, with onboard response, the spacecraft can acquire data, analyze it onboard to estimate its science value, and only downlink the highest priority data. For example, a spacecraft could monitor active volcano sites and only downlink images when the volcano is erupting. Or a spacecraft could monitor ice shelves and downlink images when calving activities are high. Or a spacecraft could monitor river lowlands, and downlink images when flooding occurs. This onboard data selection can vastly improve the science return of the mission by improving the efficiency of the limited downlink. Thus, there is significant motivation for onboard autonomy.

However, building autonomy software for space missions has a number of key challenges and constraints; many of these issues increase the importance of building a reliable, safe, agent.

1. Limited, intermittent communications to the agent. A spacecraft in low earth orbit typically has 8 communications opportunities per day. This means that the spacecraft must be able to operate for long periods of time without supervision. For deep space missions the spacecraft may be in communications far less fre-

quently. Some deep space missions only contact the spacecraft once per week, or even once every several weeks.

2. Spacecraft are very complex. A typical spacecraft has thousands of components, each of which must be carefully engineered to survive rigors of space (extreme temperature, radiation, physical stresses). Add to this the fact that many components are one-of-a-kind and thus have behaviors that are hard to characterize.
3. Limited observability. Because processing telemetry is expensive, onboard storage is limited, and downlink bandwidth is limited, engineering telemetry is limited. Thus onboard software must be able to make decisions on limited information.
4. Limited computing power. Because of limited power onboard, spacecraft computing resources are usually very constrained. An average spacecraft CPUs offer 25 MIPS and 128 MB RAM – far less than a typical personal computer.
5. High stakes. A typical space mission costs hundreds of millions of dollars and any failure has significant economic impact. Over financial cost, many launch and/or mission opportunities are limited by planetary geometries. In these cases, if a space mission is lost it may be years before another similar mission can be launched. Additionally, a space mission can take years to plan, construct the spacecraft, and reach their targets. This delay can be catastrophic.

This paper discusses our efforts to build and operate a safe autonomous space science agent. The principal contributions of this paper are as follows:

1. We describe our layered agent architecture and how that enables additional agent safety.
2. We describe our knowledge engineering and model review process designed to enforce agent safety.
3. We describe the process the agent use to detect anomalies and how it responds to these situations.
4. We describe several of the anomalies that occurred during in-flight testing, the response of the agent, and what steps were taken to prevent its occurrence in the future.

This work has been done for the Autonomous Sciencecraft Experiment (ASE) [2], an autonomy software package currently in use on NASA's New Millennium Earth Observer One (EO-1) [5] spacecraft.

In this paper we address a number of issues from the workshop call.

Definition of agent safety and how to build a safe agent – we define agent safety as ensuring the health and continued operation of the spacecraft. We design our agent to have redundant means to enforce all known spacecraft operations constraints. We also utilize declarative knowledge representations, whose models are extensively reviewed and tested. We use code generation technologies to automatically generate redundant checks to improve software reliability. Additionally, our experiment is also designed to fly in a series of increasing autonomous phases, to enable characterization of performance of the agent and to build confidence.

Robust to environment (unexpected) – our agent must be robust to unexpected environmental changes. Our agent uses a classic layered architecture approach to dealing with execution uncertainties.

How to constrain agents – because of strong concerns for safety, our agent architecture is designed to enable redundancy, adjustable autonomy, and fail-safe disabling of agent capabilities. The layering of the agent enables lower levels of the agent to inhibit higher-level agent behavior. For example, the task executive systems (SCL) does not allow dangerous commands from the planner to be sent on to the flight software. The flight software bridge (FSB) can be instructed to disable any commands from the autonomy software or to shutdown components of or the entire autonomy software. The EO-1 flight software also includes a fault protection function designed to inhibit potentially hazardous commands from any source (including the autonomy software, stored command loads from the ground, or real-time commands).

The remainder of this paper is organized as follows. First we describe the ASE software architecture, with an emphasis on how it enhances safe agent construction. Next we discuss the efforts made to prevent, detect, and respond to in-flight anomalies. Finally we present several of the anomalies that have occurred to date. We describe how the software responded to these anomalous situations, and the steps taken to prevent it from occurring in the future.

## **2 ASE**

The autonomy software on EO-1 is organized as a traditional three-layer architecture [4] (See Figure 1.). At the top layer, the Continuous Activity Scheduling Planning Execution and Replanning (CASPER) system [1, 7] is responsible for mission planning functions. Operating on the tens-of-minutes timescale, CASPER responds to events that have widespread (across orbits) effects, scheduling science activities that respect spacecraft operations and resource constraints. Activities in a CASPER schedule become inputs to the Spacecraft Command Language (SCL) execution system [6].

SCL initiates a set of scripts that issue the complete sequence of commands to the flight software. Prior to issuing each command, constraints are checked again to confirm the validity of the command as well as ensure the safety of the spacecraft. After the command is sent, SCL checks for a successful initiation and completion of the command. When a full sequence for a data collection is complete, one or more image processing algorithms are performed which may result in new requests to the planner.

### **2.1 Mission Planning**

Responsible for long-term mission planning, the ASE planner (CASPER) accepts as inputs the science and engineering goals and ensures high-level goal-oriented behavior. These goals may be provided by either the ground operators or triggered by the onboard science algorithms. The model-based planning algorithms enables rapid response to a wide range of operations scenarios based on a deep model of spacecraft

constraints, including faster recovery from spacecraft anomalies. CASPER uses repair-based techniques [1] that allow the planner to make rapid changes to the current plan to accommodate the continuously changing spacecraft state and science requests. During repair, CASPER collects a set of conflicts that represent violations of spacecraft constraints. Generic algorithms are used to select and analyze a conflict to produce a set of potential plan modifications that may resolve the conflict. Heuristics are used to select a potential modification, and the plan is updated and reevaluated for new conflicts. This process continues until no conflicts remain.

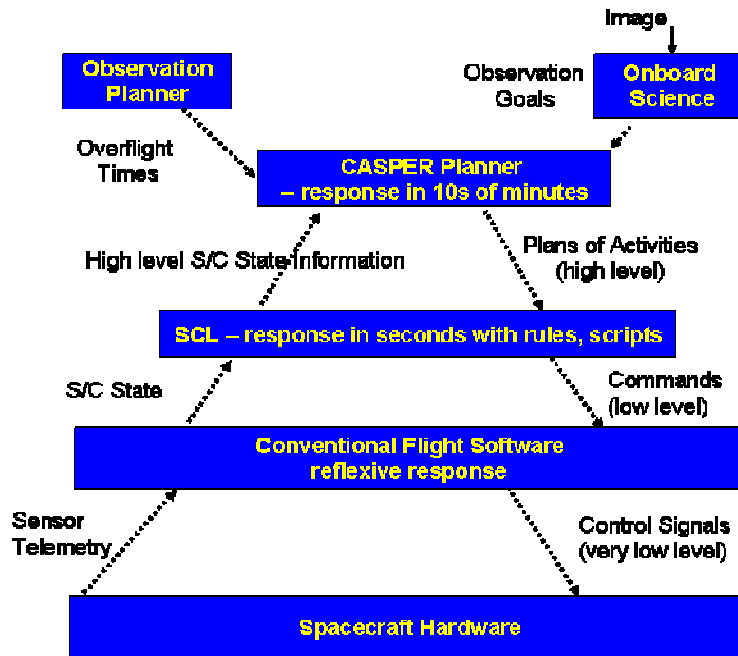


Fig. 1. Autonomy Software Architecture

## 2.2 Robust Execution

At the middle layer, SCL is responsible for generating and executing a detailed sequence of commands that correspond to expansions of CASPER activities. SCL also implements spacecraft constraints and flight rules. Operating on the several-second timescale, SCL responds to events that have local effects, but require immediate attention and a quick resolution. SCL performs activities using scripts and rules. The scripts link together lower level commands and routines and the rules enforce additional flight constraints.

SCL issues commands to the EO-1 flight software system (FSS), the basic flight software that operates the EO-1 spacecraft. The interface from SCL to the EO-1 FSS

is at the same level as ground generated command sequences. This interface is implemented by the Autonomy Flight Software Bridge (FSB), which takes a specified set of autonomy software messages and issues the corresponding FSS commands. The FSB also implements a set of FSS commands that it responds to that perform functions such as startup of the autonomy software, shutdown of the autonomy software, and other autonomy software configuration actions.

The FSS accepts low-level spacecraft commands which can be either stored command loads uploaded from the ground (e.g. ground planned sequences) or real-time commands (such as commands from the ground during an uplink pass). The autonomy software commands appear to the FSS as real-time commands. As part of its core, the FSS has a full fault and spacecraft protection functionality which is designed to:

1. Reject commands (from any source) that would endanger the spacecraft.
2. When in situations that threatens spacecraft health, execute pre-determined sequences to “safe” the spacecraft and stabilize it for ground assessment and reconfiguration.

For example, if a sequence issues commands that point the spacecraft imaging instruments at the sun, the fault protection software will abort the maneuver activity. Similarly, if a sequence issues commands that would expend power to unsafe levels, the fault protection software will shut down non-essential subsystems (such as science instruments) and orient the spacecraft to maximize solar power generation. While the intention of the fault protection is to cover all potentially hazardous scenarios, it is understood that the fault protection software is not foolproof. Thus, there is a strong desire to not command the spacecraft into any hazardous situation even if it is believed that the fault protection will protect the spacecraft.

### **2.3 Science Analysis**

The image processing software is scheduled by CASPER and executed by SCL where the results from the science analysis software generate new observation requests presented to the CASPER system for integration in the mission plan.

This layered architecture for the autonomy SW is designed such that each lower layer is verifying the output of the higher layers. Requests from the science analysis, or from operators on the ground, are checked by the planner prior to being sent to SCL. The planner activities are checked by SCL prior to being sent on to the FSS. Finally, the FSS fault protection checks the SCL outputs as well.

## **3 Anomalies**

As with any large software system and complex science scenarios, anomalous situations are expected to occur during operations. This section will describe how the ASE model was developed to enforce agent safety. We also discuss how the agent was

developed to detect for anomalies and several of the responses encoded within the model. Finally, we describe several of the anomalies that have occurred during in-flight testing, the cause of the anomalies, how the agent responded to these situations, and modifications taken to prevent it from occurring in the future.

### 3.1 Prevention

With the control aspects of the autonomy software embodied in the CASPER & SCL models, our methodology for developing and validating the CASPER and SCL models is critical to our safe agent construction process. These models include constraints of the physical subsystems including: their modes of operation, the commands used to control them, the requirements of each mode and command, and the effects of commands. At higher levels of abstraction, CASPER models spacecraft activities such as science data collects and downlinks, which may correspond to a large number of commands. These activities can be decomposed into more detailed activities until a suitable level is reached for planning. CASPER also models spacecraft state and its progression over time. This includes discrete states such as instrument modes as well as resources such as memory available for data storage. CASPER uses its model to continuously generate and repair schedules, tracking the current spacecraft state and resources, the expected evolution of state and resources, and the effects on planned activities.

**Table 1:** Sample safety analysis for two risks

	<b>Instruments overheat from being left on too long</b>	<b>Instruments exposed to sun</b>
<b>Operations</b>	For each turn on command, look for the following turn off command. Verify that they are within the maximum separation.	Verify orientation of spacecraft during periods when instrument covers are open.
<b>CASPER</b>	High-level activity decomposes into turn on and turn off activities that are with the maximum separation.	Maneuvers must be planned at times when the covers are closed (otherwise, instruments are pointing at the earth)
<b>SCL</b>	Rules monitor the “on” time and issue a turn off command if left on too long.	Constraints prevent maneuver scripts from executing if covers are open.
<b>FSS</b>	Fault protection software will shut down the instrument if left on too long.	Fault protection will safe the spacecraft if covers are open and pointing near the sun.

SCL continues to model spacecraft activities at finer levels of detail. These activities are modeled as scripts, which when executed, may execute additional scripts, ultimately resulting in commands to the EO-1 FSS. Spacecraft state is modeled as a database of records in SCL, where each record stores the current value of a sensor, resource, or sub-system mode. The SCL model also includes flight rules that monitor spacecraft state, and execute appropriate scripts in response to changes in state. SCL uses its model to generate and execute sequences that are valid and safe in the current context. While SCL has a detailed model of current spacecraft state and resources, it does not generally model future planned spacecraft state and resources.

Development and verification of the EO-1 CASPER and SCL models was a multiple step process.

1. First a target set of activities was identified. This was driven by a review of existing documents and reports. This allowed the modeler to get a high-level overview of the EO-1 spacecraft, including its physical components and mission objectives. Because EO-1 is currently in operation, mission reports were available from past science requests. These reports were helpful in identifying the activities performed when collecting and downlinking science data. For example, calibrations are performed before and after each image, and science requests typically include data collection from both the Hyperion (hyperspectral) and Advanced Land Imager (ALI) instruments.
2. Once the activities were defined, a formal EO-1 operations document [3] was reviewed to identify the constraints on the activities. For example, due to thermal constraints, the Hyperion cannot be left on longer than 19 minutes, and the ALI no longer than 60 minutes. The EO-1 operations team also provided spreadsheets that specified timing constraints between activities. Downlink activities, for example, are often specified with start times relative to two events: acquisition of signal (AOS) and loss of signal (LOS). Fault protection documents listing fault monitors (TSMs) were also consulted, using the reasoning that acceptable operations should not trigger TSMs.
3. With the model defined, CASPER was able to generate preliminary command sequences from past science requests that were representative of flight requests. These sequences were compared with the actual sequences for the same request. Significant differences between the two sequences identified potential problems with the model. For example, if two commands were sequenced in a different order, this may reveal an overlooked constraint on one or both of the commands. We were also provided with the actual downlinked telemetry that resulted from the execution of the science observation request. This telemetry is not only visually compared to the telemetry generated by ASE, but it can also be “played back” to a ground version of the ASE software to simulate the effects of executing sequences. The command sequences were aligned with the telemetry to identify the changes in spacecraft state and the exact timing of these changes. Again, any differences between the actual telemetry and the ASE telemetry revealed potential errors in the model. A consistent model was defined after several iterations of generating commands and telemetry, comparing with actual commands and telemetry, and fixing errors. These comparisons against ground generated se-

quences were reviewed by personnel from several different areas of the operations staff to ensure acceptability (e.g. overall operations, guidance, navigation and control, science operations, instrument operations).

4. Model reviews were conducted where the models are tabletop reviewed by a team of personnel with a range of operations and spacecraft background. This is to ensure that no incorrect parameters or assumptions are represented in the model.

Finally, a spacecraft safety review process was performed. By studying the description of the ASE software and the commands that ASE would execute, experts from each of the spacecraft subsystem areas (e.g., guidance, navigation and control, solid state recorder, Hyperion instrument, power) derived a list of potential hazards to spacecraft health. For each of these hazards, a set of possible safeguards was conjectured: implemented by operations procedure, implemented in CASPER, implemented in SCL, and implemented in the FSS. Every safeguard able to be implemented with reasonable effort was implemented and scheduled for testing. An analysis for two of the risks is shown below.

### 3.2 Detection

The EO-1 FSS has a set of Telemetry and Statistics Monitoring (TSM) tasks that monitor the state of the spacecraft. TSMs typically detect anomalies by comparing a state value with expected thresholds for that value.

The FSS also includes processes for transmitting engineering data from the spacecraft subsystems for recording and future playback. ASE tapped into this data stream so that it could automatically monitor spacecraft state and resources. The data is received at 4Hz and the relevant information is extracted and stored into the SCL database. SCL uses the latest information when making decisions about command execution. Spacecraft state and resources are checked:

- Prior to executing the command to verify command prerequisites are met.
- After executing the command to verify the receipt of the command.
- After an elapsed time period when the effects of the command are expected.

The SCL model also contains a set of rules that continuously monitor the state of the spacecraft and relays any change to the database to CASPER. CASPER compares the new data with its predicted values and makes any necessary updates to the predictions. Effects of these updates to the current schedule are monitored and conflicts detected. If a set of conflicts are detected, CASPER will begin modifying the plan to find conflict-free schedule.

During ground contacts, mission operators can monitor the EO-1 spacecraft telemetry in real-time, as well as playback recorded telemetry and messages. Limits are set on the various data points to alarm operators when values fall out of their expected ranges. To manually monitor ASE, we developed a set of telemetry points for each of the ASE modules. This is typically high-priority health and status data that is continuously saved to the onboard recorder and downlinked during ground contacts. The real-



time engineering data for EO-1 is monitored with the ASIST ground software tool developed at GSFC.

The FSB, which acts as a gateway, has several telemetry points to verify that we have enabled or disabled the flow of spacecraft commands and telemetry. It also has command counters for those issued to the ASE software. SCL provides telemetry on its state including counters for the number of scripts executed. CASPER provides statistics on the planning algorithm including the types of conflicts that it addresses and what changes it makes to the plan when repairing the conflicts. It also generates telemetry that identifies any differences it finds between the actual spacecraft state and the state it expects during the execution of the plan.

The telemetry points for each module is useful in providing a high level view of how the software is behaving, but debugging anomalies from this would be difficult. Therefore, each software module also saves more detailed data to log files stored on a RAM disk. As they are needed, these log files are downlinked either to debug new issues or to further validate the success of the test.

### **3.3 Response**

Anomaly detection may trigger a response from any one of the software modules, or from the operations team. At the highest level, CASPER can respond to some anomalies by replanning future activities. For example, CASPER may delete low priority requests to alleviate unexpected over-utilization of resources. At the middle layer, SCL can respond to small anomalies with robust execution. In most cases, it can delay the execution of a command if the spacecraft has not reached the state required by the command. Because of interactions within a sequence, however, a command cannot be delayed indefinitely and may eventually fail. If a critical command fails, SCL may respond with additional commands in attempt to recover from the failure. This is typically a retry of the commands or set of commands that has failed. If the command remains failed, the effects of the command do not propagate to the SCL database, which may trigger CASPER to replan. Finally, at the lowest level, the EO-1 FSS fault protection is used to detect potentially hazardous spacecraft states and trigger commands to transition the spacecraft out of those states. For example, the Hyperion temperature increases while the instrument is in-use. The FSS fault protection monitors this temperature and initiates shut-down commands when the maximum temperature is exceeded.

## **4 Case Study**

In this section, we describe several of the in-flight anomalies that have occurred, including the responses taken by the agent, and the changes performed to prevent future anomalies from occurring.

## 4.1 Anomaly Types

The anomalies that have occurred onboard can be classified into the following types: modeling, software, operator, hardware.

**Modeling** – This is the most common type of error, caused by an incorrect model of the spacecraft within CASPER and SCL. Many of these errors were not detected during testing and validation because the EO-1 mission did not have a high-fidelity testbed, requiring the development of simulators that made several incorrect assumptions of the spacecraft behavior.

**Software** – These are your standard software implementation errors that occur with any large project. The design of the agent needed to be robust to errors that occur between the CASPER, SCL, FSW, and science modules.

**Operator** – Commands are regularly issued from mission operators during ground contacts. These commands may modify the state of the spacecraft, so the agent will need to be robust to these situations.

**Hardware** – The Livingston 2 software component (described in section 5) was designed to detect and diagnose this type of error. However, because hardware errors are rare on spacecraft systems, we chose not to focus on detecting these.

The remainder of this section will describe four anomalies in detail, mainly of the modeling, software, and operator type. We will describe how they were caused, how the agent responded to the anomaly, and what work was done to prevent it from occurring in the future.

## 4.2 Onboard Anomalies

### April 1, 2004

During this early stage of the project, we were testing a single response scenario where the onboard science module would analyze an image, and issue requests to the onboard planner for more images of that target. The scenario went as follows:

- Image Prudhoe Bay, Alaska
- Playback data from the solid state recorder into RAM for image processing (bandstripping)
- Run image classifier to identify snow, water, ice, and land. This was scheduled to run immediately after the bandstripping process.
- Run image classifier to identify for cloud coverage. This was scheduled to run 60 minutes after the bandstripping process.
- If the classification of the scene was above a threshold defined by mission scientists, request additional images of Prudhoe Bay.

Several of the constraints modeled within CASPER and SCL were:

- The bandstripping process cannot begin unless the target Prudhoe Bay was successfully imaged

- The image classifiers cannot begin unless the bandstripping process was successful

During the first ground contact following this scenario, mission operators noticed several warnings from SCL telemetry and that the onboard science module did not perform any image processing. After collecting log files from SCL and CASPER, and replaying back telemetry collected during the test, it was determined that SCL had failed the bandstripping script because of a time out during the verification of the command completion. In actuality, this verification failure was not a result of bandstripping failing, but of a bug within the flight software time code. It is still interesting, however, to examine the response of the agent.

The failure of the bandstripping script resulted in a lack of change to a SCL database record. This record is continuously monitored by CASPER and a conflict with the scheduled image classifier algorithm was recognized. However, because the first image classifier algorithm was scheduled immediately after bandstripping, CASPER had already committed to executing the classifier activity. When making this type of commitment, CASPER locks the activity (preventing any re-scheduling) and sends the execution request to SCL. The command was received by SCL, but failed the pre-requisite check, blocking the command from being sent to the science module.

The second image classifier was scheduled 60 minutes after the end of bandstripping, and thus CASPER was able to modify the plan to resolve the conflict by removing it from the schedule.

This anomaly demonstrated how the layered architecture ensured the safety of the agent. CASPER was not responsive enough to prevent the first image classifier from being issued to SCL, but the SCL pre-requisite check failed and thus the command was not issued. However in the second case, CASPER was able to respond to this failure by removing the subsequent image processing activity from the schedule.

A modification of the agent architecture to prevent these false-positive anomalies from occurring would be to have redundant checks in the completion of the commands. In this example, a single SCL database item indicated that bandstripping had failed, when in fact, it had succeeded. The model could have been updated to check multiple database records for the status of the bandstripping, instead of relying on solely on a single data point to verify completion of the command.

#### **July 15, 2004**

This anomaly demonstrates how SCL was able to respond to a verification failure of command sequence. During this test, the anomaly occurred during normal operations for an X-Band ground contact. The scenario was:

- Using the X-Band transmitter, downlink all images from the solid state recorder
- Clear all images from the solid state recorder

Several of the constraints modeled were:

- The correct voltage/current level of transceiver must be met prior to operating X-Band activities.
- The downlink must complete successfully prior to clearing all the images from the solid state recorder.

During the ground contact, mission operators noticed several warnings from SCL and also that EO-1 had not begun the X-Band downlink of images collected. The operators manually initiated the X-Band contact and completed dumping the data. After analyzing log files, it was determined that a prerequisite failure in the SCL model for the current/voltage of the transceiver prevented the contact from being initiated. As a result of the X-Band failure, SCL also rejected the command to clear all the images from the solid state recorder.

This was actually an error within the SCL model. An early version of the model included a constraint that the transceiver cannot be powered on unless the current/voltage was at the correct level. However, the threshold values for the current/voltage in reality are not valid until the transceiver is powered on.

Unfortunately, this modeling error slipped through our testing and validation process because of the lack of a high fidelity testbed. The EO-1 testbed did not have a transceiver for testing and therefore, the current/voltage values were static (at the “on” levels) in the simulation. Without valid values on the current/voltage prior to powering on the X-Band transceiver, our resolution to this problem was to simply remove the current/voltage constraint from the SCL model.

#### **January 31, 2005**

This anomaly describes CASPER’s response to an unexpected change in the state of the spacecraft. During one of the scheduled ground contacts, the agent did not initiate the command sequence as requested from mission planners. An anomaly had occurred that removed the contact sequence from the mission plan. After analysis of collected telemetry, the cause of the anomaly was due to human intervention with the spacecraft several hours prior. An unscheduled contact had been initiated by mission planners, which was performed externally from the onboard planner. The unscheduled contact required mission operators to perform a blind acquisition of EO-1 and manually power on the transceiver, changing the state of the onboard transceiver to “on”. At the end of this contact, the operators manually powered down the transceiver.

The change to the transceiver state resulted in an update to the SCL database, which propagated to the CASPER schedule and created a conflict with the next ground contact activity. The conflict was with a constraint in the CASPER model that only allowed the transceiver state to transition from “on” to “off” or from “off” to “on”. When the update to the transceiver state was received, it set the current state to the transceiver to “on”. This created a conflict with the next scheduled contact that had planned to turn the transceiver on when the state was already “on”. To resolve this conflict, CASPER removed the next contact from the plan. Once the mission operator powered down the transceiver at the end of the unscheduled contact, subsequent contacts were conflict free, but the deleted contact was not rescheduled due to the risk of inserting the goal too close to its scheduled time.

To prevent this anomaly for future operations, we simply removed the transition constraints from the CASPER model of the transceiver. While not ideal, it was determined that this presented no risk to the spacecraft, and allowed the ASE software to

support real-time contact requests from mission planners without affecting the remainder of the schedule.

In this anomaly, although the update to the state of the transceiver was short-lived as it was eventually powered off by mission operators, its affect on the planner propagated to the next scheduled contact, resulting in the contact being removed from the schedule. One possible solution to prevent this from occurring in the future is to delay resolving conflict until necessary. Some initial work has been started on CASPER to support time-sensitive repair heuristics, but is still experimental and was not deployed on EO-1.

#### **February 26, 2005**

This anomaly occurred during a normal data collect of an area in Peru. During the first contact following this image, SCL telemetry indicated 1 warning from execution. After analysis of log files, this was determined to be caused by a command verification failure when issuing the command to close the ALI cover. The response of SCL was to reissue the command to close the covers, ensuring that the covers would be closed at the end of the collect.

Further investigation into the problem showed that the cover did indeed close after the first command. However, due to a spike in CPU consumption during that time, SCL was not able to process the change in the database record indicating that the cover was closed. While SCL has the highest priority among the ASE tasks, it is not higher than any of the FSS tasks. We are still investigating the cause for the increase in CPU load.

In this situation, the actions of the agent were correct in preserving the safety of the spacecraft. However, a change to the model can be used to ensure this anomaly does not occur in the future. Again, similar to the first anomaly described in this paper, redundant checks to multiple SCL database items can be used to determine the true state of the covers. For example, the EO-1 Hyperion instrument covers have two data-points representing the state of the cover. One data-point indicates if the cover is either open or closed, while the other is a continuous value, representing how far the cover has been opened. A check that reasons using both of these data-points would be less prone to false-positives.

## **5 Livingston 2**

More recently (Fall 2004), in collaboration with Ames Research Center, we have begun flying the Livingstone 2 (L2) [12] diagnosis system. Both L2 and CASPER use models of the spacecraft separate from the reasoning engine: the models are tailored for a particular application without the need to change the software, allowing reuse of the advanced reasoning software across applications. The diagnostic capability of an on-board agent can then use the models to monitor the health of the spacecraft and detect faults. Early development of the L2 model currently does not support responding to anomalous situations, only detection of them.

However, during the times of the described anomalies, L2 was not operational. Also its current model only supports monitoring the operations of the spacecraft and not the CASPER or SCL software. Therefore, anomalous situations within CASPER or SCL would not be detected by L2.

## **6 Related Work**

In 1999, the Remote Agent experiment (RAX) [10] executed for a several days onboard the NASA Deep Space One mission. RAX is an example of a classic three-tiered architecture [4], as is ASE. RAX demonstrated a batch onboard planning capability (as opposed to CASPER's continuous planning) and RAX did not demonstrate onboard science. PROBA [11] is a European Space Agency (ESA) mission demonstrates onboard autonomy and launched in 2001. However, ASE has more of a focus on model-based autonomy than PROBA.

The Three Corner Sat (3CS) University Nanosat mission used CASPER onboard planning software integrated with the SCL ground and flight execution software [13]. The 3CS mission was launched in December 2004 but the spacecraft were lost due to a deployment failure. The 3CS autonomy software includes onboard science data validation, replanning, robust execution, and multiple model-based anomaly detection. The 3CS mission is considerably less complex than EO-1 but still represents an important step in the integration and flight of onboard autonomy software.

## **7 Conclusions**

This paper has described the design of a safe agent for the Autonomous Sciencecraft Experiment along with several of the anomalies and the software's responses that have occurred during in-flight testing. First, we described the salient challenges in developing a robust, safe, spacecraft control agent. Second, we described how we used a layered architecture to enhance redundant checks for agent safety. Third, we described our model development, validation, and review. Fourth, we described how the agent responds and detects anomalous situations. Finally, we described several case studies of anomalies that have occurred in-flight and the response taken by the agent to maintain the safety of the spacecraft.

## **8 Acknowledgements**

Portions of this work were performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

## References

1. S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Using Iterative Repair to Improve Responsiveness of Planning and Scheduling," *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling*, Breckenridge, CO, April 2000. (see also <http://casper.jpl.nasa.gov>)
2. S. Chien et al., "The Earth Observing One Autonomous Science Agent", *Proceedings of the Autonomous Agents and Multi-Agent Systems Conference*, New York City, NY, July 2004.
3. S. Chien et al, EO 1 Autonomous Sciencecraft Experiment Safety Analysis Document, 2003.
4. E. Gat, Three layer architectures, in *Mobile Robots and Artificial Intelligence*, (Kortenkamp, Bonasso, and Murphy eds.), Menlo Park, CA: AAAI Press, pp. 195-210.
5. Goddard Space Flight Center, EO-1 Mission page: <http://eo1.gsfc.nasa.gov>
6. Interface and Control Systems, SCL Home Page, <http://www.interfacecontrol.com>
7. G. Rabideau, R. Knight, S. Chien, A. Fukunaga, A. Govindjee, "Iterative Repair Planning for Spacecraft Operations in the ASPEN System," *International Symposium on Artificial Intelligence Robotics and Automation in Space*, Noordwijk, The Netherlands, June 1999.
8. G. Rabideau, S. Chien, R. Sherwood, D. Tran, B. Cichy, D. Mandl, S. Frye, S. Shulman, R. Bote, J. Szwaczkowski, D. Boyer, J. Van Gaasbeck, "Mission Operations with Autonomy: A preliminary report for Earth Observing-1", *International Workshop on Planning and Scheduling for Space*. Darmstadt, Germany, June 2004
9. D. Tran, S. Chien, G. Rabideau, B. Cichy, "Flight Software Issues in Onboard Automated Planning: Lessons Learned on EO-1", *International Workshop on Planning and Scheduling for Space*, Darmstadt, Germany. June 2004
10. NASA Ames, Remote Agent Experiment Home Page, <http://ic.arc.nasa.gov/projects/remote-agent/>. See also Remote Agent: To Boldly Go Where No AI System Has Gone Before. Nicola Muscettola, P. Pandurang Nayak, Barney Pell, and Brian Williams. *Artificial Intelligence* 103(1-2):5-48, August 1998
11. The PROBA Onboard Autonomy Platform, <http://www.estec.esa.nl/proba>
12. J. Kurien and P. Nayak. "Back to the future for consistency-based trajectory tracking." *In Proceedings of the 7th National Conference on Artificial Intelligence (AAAI'2000)*, 2000.
13. S. Chien, B. Engelhardt, R. Knight, G. Rabideau, R. Sherwood, E. Hansen, A. Ortiz, C. Wilklow, S. Wichman, "Onboard Autonomy on the Three Corner Sat Mission," *Proc i-SAIRAS 2001*, Montreal, Canada, June 2001.

# Coordinating Randomized Policies for Increasing Security in Multiagent Systems

Praveen Paruchuri<sup>1</sup>, Milind Tambe<sup>2</sup>, Fernando Ordóñez<sup>3</sup>, and Sarit Kraus<sup>4</sup>

<sup>1</sup> University of Southern California  
Los Angeles, CA 90089  
`paruchur@usc.edu`

<sup>2</sup> University of Southern California  
Los Angeles, CA 90089  
`tambe@usc.edu`

<sup>3</sup> University of Southern California  
Los Angeles, CA 90089  
`fordon@usc.edu`

<sup>4</sup> Bar-Ilan University  
Ramat-Gan 52900, Israel  
`sarit@cs.biu.ac.il`

**Abstract.** Despite significant recent advances in decision theoretic frameworks for reasoning about multiagent teams, little attention has been paid to applying such frameworks in adversarial domains, where the agent team may face security threats from other agents. This paper focuses on domains where such threats are caused by unseen adversaries whose actions or payoffs are unknown. In such domains, action randomization is recognized as a key technique to deteriorate an adversary's capability to predict and exploit an agent/agent teams actions. Unfortunately, there are two key challenges in such randomization. First, randomization can reduce the expected reward (quality) of the agent team's plans, and thus we must provide some guarantees on such rewards. Second, randomization results in miscoordination in teams. While communication within an agent team can help in alleviating the miscoordination problem, communication is unavailable in many real domains or sometimes scarcely available. To address these challenges, this paper provides the following contributions. First, we recall the Multiagent Constrained MDP (MCMDP) framework that enables policy generation for a team of agents where each agent may have a limited or no (communication) resource. Second, since randomized policies generated directly for MCMDPs lead to miscoordination, we introduce a transformation algorithm that converts the MCMDP into a transformed MCMDP incorporating explicit communication and no communication actions. Third, we show that incorporating randomization results in a non-linear program and the unavailability/limited availability of communication results in addition of non-convex constraints to the non-linear program. Finally, we experimentally illustrate the benefits of our work.

**Key words:** Multiagent Systems, Decision Theory, Security, Randomized Policies



## 1 Introduction

Decision-theoretic models like the Multiagent Markov Decision Problem's (MMDPs) [2], Decentralized Markov Decision Problem's (Dec-MDPs) [3] and the Decentralized Partially Observable MDP's (Dec-POMDPs) [4] have been successfully applied to build agent-teams acting in uncertain environments. These teams must often work in an adversarial environment. For example, when patrolling, UAV (Unmanned Air Vehicles) teams might often be watched by adversaries such as unobserved terrorists [5] or robotic patrol units trying to detect intruders in physical security sites [6, 7]. Security, commonly defined as the ability of the system to deal with intentional threats from other agents [8], becomes a critical issue for these agent teams acting in such adversarial environments. Often, the agents cannot even explicitly model the adversary's actions and capabilities or its payoffs. However, the adversary can observe the agents' actions and exploit any action predictability in some unknown fashion. For example, consider the team of UAVs [9] monitoring a region undergoing a humanitarian crisis. Adversaries may be humans intent on causing some significant unanticipated harm, e.g. disrupting food convoys, harming refugees or shooting down the UAVs. Further, the adversary's capabilities, actions or payoffs are unknown or difficult to model explicitly. However, the adversaries can observe the UAVs and exploit any predictability in UAV surveillance, e.g. engage in unknown harmful actions by avoiding the UAVs' route.

Given our assumption that the agent team acts in an adversarial domain where the adversary cannot be explicitly modeled, policy randomization becomes crucial for the teams to avoid the action predictability [5]. We also assume that the agent team is acting in accessible environments and hence can be modeled using MMDPs. We further make the following three assumptions about the adversary in our work. First, we assume that the adversary can also observe the agents' state exactly. The second assumption is that the adversary knows the agents' policy, which it may do by learning over repeated observations. Policy randomization would then ensure that even if the adversary knows the agents' state exactly at each instant and also the agents' policy from that state, the adversary would still be unable to predict the agent's action correctly and hence significantly cut down the chances of unanticipated harm. The third assumption is that agent teams cannot communicate in general or limited communication bandwidth is available. If available, we assume that communication is encrypted and also being a private resource for the team, is unobservable for the adversary i.e. communication is safe. However, if communication is observable it can be easily masked by using simple deception techniques like sending some meaningless data for non-communication acts, thus making it safe [10].

While policy randomization avoids action predictability, simply randomizing an MDP policy as mentioned above can degrade the expected team reward significantly and hence we face a randomization-reward tradeoff. The difficulty in generating randomized policies that provide the appropriate randomization-reward tradeoff is further exacerbated by the fact that randomization creates miscoordination in team settings. We wish to enable our agents to perform ran-

domized actions without any type of coordination whatsoever, or any type of synchronization.<sup>5</sup>

For real world teams, communication resources are usually unavailable or severely limited, e.g., members of a UAV team might not be able to communicate due to bandwidth/environmental restriction or have limited communication bandwidth [11] allocated. Hence, the agent teams face resource (bandwidth here) constraints. Constraints involving averaging a quantity, in general, are soft constraints because as long as the average is maintained, there is no hard bound on the resource amount to be used at each timestep [12, 13]. In our example, we model bandwidth as a soft constraint [11] because exceeding bandwidth in any single run is not a disaster; but if the team consumes more than its bandwidth limit on an average, it jeopardizes the communications of other agents on the same network. The importance of such soft constraints is seen by continued work in operations research literature on constrained MDPs (CMDPs) that reason about expected resource consumption [14].

Our work focuses on increasing security using policy randomization for agent teams with no/limited bandwidth while ensuring fixed reward thresholds. Although, such randomized policies have occurred as side effect [14] and turn out to be optimal in some stochastic games [15], work on intentional policy randomization has received focus only recently. For example, [5] intentionally randomizes MDP/POMDP policies for increasing security but their work provides heuristic solution assuming that the agents cannot communicate. Work that has been done on developing agent teams with resource constraints [14, 11, 16] has not paid attention to the issue of security in such teams. To address these concerns, we therefore solve a multicriterion problem that maximizes the team policy randomization while ensuring that the average bandwidth consumption is below a threshold and the team reward is above a threshold. The problem we solve is general enough and other soft resource constraints can be considered without any modifications to the structure of the problem.

This paper provides three key contributions to solve the problem described. First, we recall MCMDP as multiagent MDP framework where agents reason not only about their rewards but also about resource constraints. We then introduce the entropy metric to quantify policy randomization for MCMDP and formu-

---

<sup>5</sup> One particular method to avoid miscoordination, is to assume that the agents (say the UAV's) use a pseudo-random number generation process with an initial shared seed, but it suffers from many drawbacks. First, this technique doesn't work when the agents cannot communicate because the agents need to communicate their random seeds. Second, different UAV's need not use the same random number generation algorithms which is quite likely due to the various manufacturers involved, making seed sharing an impractical approach. Third, the random seed sharing method assumes that the hardware clocks of all the agents involved are synchronized which can be unrealistic in some domains. Fourth, the agents need to establish protocols beforehand for the seed sharing method to work which gets complicated as the number of agents increase. On the other hand, our technique works even if we assume that the agents cannot communicate, thus making it a general-purpose algorithm without any of these hardware assumptions.

late a nonlinear program that maximizes policy randomization while ensuring threshold rewards. We then identify a novel coordination challenge that occurs due to randomized policies in multiagent settings, i.e agents miscoordinate if there are randomized policies in team settings. Second, we provide a novel polynomial time transformation algorithm that converts the MCMDP into a transformed MCMDP incorporating explicit communication and no communication actions to alleviate such miscoordination. Third, we developed a non-linear program with non-convex constraints for the transformed MCMDP that randomizes team policy while attaining a threshold reward without violating the communication constraints. We further show that the value of entropy for MCMDP and the transformed MCMDP remains the same for the same policy, thus showing that our transformation is correct. In our experimental section, we show results after evaluating the new non-linear program we developed for the transformed MCMDP. The rest of the paper begins with MCMDP and a non-linear program for it that captures policy randomization. An automated method of transformation is provided that converts this MCMDP to a transformed MCMDP. We then provide our solution approach to solve this new model. We then briefly describe the various transformations possible. Lastly, we provide experimental results that clearly show the interdependence between the important factors of our domain namely policy randomization, reward and bandwidth.

## 2 Randomization: MCMDP

MCMDP is a useful tool for users, providing a layer of abstraction to model agent-teams with resource constraints in uncertain domains. For purposes of this paper, the only resource being modeled is the bandwidth. We first recall a 2-agent MCMDP for expository purposes. A 2-agent MCMDP is defined as a tuple,  $\langle S, A, P, R, C1, C2, T1, T2, N, Q \rangle$  where:  $S$  is a finite set of states. Given two individual actions  $a_l$  and  $a_m$  of the two agents in our team, the team's joint action  $\hat{a} = (a_l, a_m) \in A$  i.e  $A$  represents the set of all possible joint actions.  $P = [p_{ij}^{\hat{a}}](\equiv p(i, \hat{a}, j))$  is the transition matrix, providing the probability of transitioning from a source state  $i$  to a destination state  $j$ , given the team's joint action  $\hat{a}$ ,  $R = [r_{i\hat{a}}]$  is the vector of joint rewards obtained when an action  $\hat{a}$  is taken in state  $i$ .  $C1 = [c1_{i\hat{a}k}]$  is the vector to account for cost of resource  $k$  when action  $\hat{a}$  is taken in state  $i$  by agent 1 i.e it models cost for individual resource of agent 1. ( $C2$  is similarly defined.)  $T1 = [t1_k]$  and  $T2 = [t2_k]$  are vectors of thresholds on the availability of the individual resources  $k$  for agents 1 and 2 respectively.  $N = [n_{i\hat{a}}]$  is the vector of joint communication costs incurred by the agents when an action  $\hat{a}$  is taken in state  $i$ .  $Q$  is a threshold on communication costs that can be used by the team of agents. A MCMDP is thus similar to a CMDP [14] with multiple agents.

### 2.1 Randomization due to resource constraints

The goal in a MCMDP is to maximize the total expected reward, while ensuring that the expected resource (bandwidth here) consumption is maintained below

threshold. Formally, this requirement can be stated as a linear program, extending the linear program for CMDPs [13] to a two agent case, as shown below.  $x_{i\hat{a}}$  is the expected number of times an action  $\hat{a}$  is executed in state  $i$  and  $\alpha_j$  is the initial probability distribution over the state space.

$$\begin{aligned}
& \max \sum_i \sum_{\hat{a}} x_{i\hat{a}} r_{i\hat{a}} \\
& \text{s.t.} \quad \sum_{\hat{a}} x_{j\hat{a}} - \sum_i \sum_{\hat{a}} x_{i\hat{a}} p_{ij}^{\hat{a}} = \alpha_j \quad \forall j \in S \\
& \quad \sum_i \sum_{\hat{a}} x_{i\hat{a}} c_{i\hat{a}k} \leq t_{1k}, \quad \sum_i \sum_{\hat{a}} x_{i\hat{a}} c_{2i\hat{a}k} \leq t_{2k} \\
& \quad \sum_i \sum_{\hat{a}} x_{i\hat{a}} n_{i\hat{a}} \leq Q, \quad x_{i\hat{a}} \geq 0 \quad \forall i \in S, \hat{a} \in A
\end{aligned} \tag{1}$$

If  $x^*$  is the optimal solution to (1), optimal policy  $\pi^*$  is given by (2) below, where  $\pi^*(s, \hat{a})$  is the probability of taking action  $\hat{a}$  in state  $s$ .

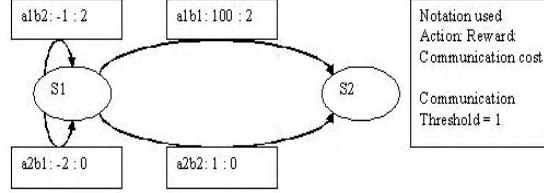
$$\pi^*(s, \hat{a}) = \frac{x^*(s, \hat{a})}{\sum_{\hat{a} \in A} x^*(s, \hat{a})}. \tag{2}$$

It turns out that  $\pi^*$  is a randomized policy in the above case due to the resource constraints. Since, bandwidth is the only resource under consideration and is modeled as a team resource we set the individual resources and their thresholds i.e.,  $C1, C2, T1, T2$  to zero. Such randomization leads to miscoordination in team settings as shown in section 2.2. Further, the randomization occurred as sideeffect due to the communication constraint and hence not optimized for policy randomness as needed by our domain.

## 2.2 Miscoordination: Effect of Randomization in Team Settings

For illustrative purposes, Figure 1 shows a 2 state MCMDP with two agents A and B with actions  $a_1, a_2$  and  $b_1, b_2$  respectively, leading to joint actions  $\hat{a}1 = (a_1, b_1)$ ,  $\hat{a}2 = (a_1, b_2)$ ,  $\hat{a}3 = (a_2, b_1)$ ,  $\hat{a}4 = (a_2, b_2)$ . We also show the transition probabilities, rewards and communication costs for each of the actions. The optimal policy for this MCMDP is to take joint actions  $\hat{a}1$  and  $\hat{a}4$  with 0.5 probability. Suppose, agent A chooses its own actions such that  $p(a1) = .5$  and  $p(a2) = .5$ , based on the joint actions. However, when A selects  $a1$ , there is not guarantee that agent B would choose  $b1$ . In fact, B can choose  $b2$  due to its own randomization. Thus, the team may jointly execute  $\hat{a}2 = (a1, b2)$ , even though the policy specifies  $p(\hat{a}2) = 0$ . Therefore, a MCMDP, a straightforward generalization of a CMDP to a multiagent case, results in randomized policies, which a team cannot execute without additional coordination. One simple solution is to add a communication action before each joint action. However, forcing a communication action before every single action can violate communication constraints, since communication itself consumes resources. Thus, a solution that limits communication costs is essential. Further, equation 1 maximizes the expected reward obtained for the MCMDP while we are interested in maximizing the randomness of our policy. Below, we first introduce an entropy measure to

quantify randomness and then develop an algorithm that maximizes the measure while we threshold on reward and constrain the communication. However, the problem of miscoordination still remains which we solve in section 3.



**Fig. 1.** Simple MCMDP [(a1b1:100:2)- Action a1b1 gives reward 100 with communication cost 2]

### 2.3 Randomness of a policy

For a discrete probability distribution  $p_1, p_2, \dots, p_n$  the only function, upto a multiplicative constant, that captures the randomness is the entropy, given by the formula  $H = -\sum_{i=1}^n p_i \log p_i$  [17]. For quantifying the randomness of a single agent MDP policy, we borrow the weighted entropy concept developed in [5]. For purposes of clarity we reproduce the formula here ( $\pi$  is the CMDP policy which defines a probability distribution over actions for each state  $s$ )-

$$H_W(x) = -\sum_{s \in S} \frac{\sum_{\hat{a} \in A} x(s, \hat{a})}{\sum_{j \in S} \alpha_j} \sum_{a \in A} \pi(s, a) \log \pi(s, a) = -\frac{1}{\sum_{j \in S} \alpha_j} \sum_{s \in S} \sum_{a \in A} x(s, a) \log \left( \frac{x(s, a)}{\sum_{\hat{a} \in A} x(s, \hat{a})} \right).$$

Extending this formula for a 2-agent MCMDP is quite straightforward in the sense that instead of calculating the weighted entropy over a single agent policy we calculate it over the joint policy of both the agents for the 2-agent MCMDP. Hence, in the weighted entropy formula above,  $\pi$  refers to the joint policy of the agents.

### 2.4 Intentional Randomization: Maximal entropy solution

We can now obtain maximal entropy policies with a threshold expected reward meeting the communication requirements by replacing the objective of Problem (1) with the definition of the weighted entropy  $H_W(x)$ . (Note that the problem of miscoordination still remains which we will tackle in section 3). The reduction in expected reward can be controlled by enforcing that feasible solutions achieve at least a certain expected reward  $E_{\min}$  and the communication constraint remains unchanged. The following problem maximizes the weighted entropy while

maintaining the expected reward above  $E_{\min}$  and a communication consumption below  $Q$ :

$$\begin{aligned}
& \max H_W(x) \\
& \text{s.t.} \quad \sum_{a \in A} x(j, a) - \sum_{s \in S} \sum_{a \in A} p(s, a, j) x(s, a) = \alpha_j \quad \forall j \in S \\
& \quad \sum_{s \in S} \sum_{a \in A} r(s, a) x(s, a) \geq E_{\min}, \quad \sum_{s \in S} \sum_{a \in A} x(s, a) n(s, a) \leq Q \\
& \quad x(s, a) \geq 0 \quad \forall s \in S, a \in A
\end{aligned} \tag{3}$$

where  $E_{\min}$  is an input domain parameter ( $E_{\min}$  can vary between 0 and  $E^*$  where  $E^*$  is the maximum expected reward obtained by solving (1)). Solving Problem (3) is our first algorithm to obtain a randomized policy that achieves at least  $E_{\min}$  expected reward while meeting the communication constraints (Algorithm 1).

---

**Algorithm 1** MAX-ENTROPY( $E_{\min}, Q$ )

---

- 1: Solve Problem (3) with  $E_{\min}$  and  $Q$ , let  $x_{E_{\min}}$  be optimal solution
  - 2: **return**  $x_{E_{\min}}$  (maximal entropy, expected reward  $\geq E_{\min}$ , communication required  $\leq Q$ )
- 

Unfortunately, the problem of miscoordination introduced in section 2.2 still remains.

### 3 Solving Miscoordination: From MCMDP to Transformed MCMDP

This section presents an automatic transformation of a MCMDP to a transformed MCMDP, where the resulting optimal policies can be executed in multi-agent settings, via appropriate communication (with communication costs within resource limits). We illustrate the key concepts in MCMDP transformations by focusing on one specific transformation namely the *sequential transformation*, given in Figure 2-a. While the transformation introduced is similar to [11], there are two key differences in that work and the present work: (i) The solution for policy randomization we develop for the transformed MCMDP needs a non-linear objective with non-linear constraints unlike earlier work where the reward maximization needed linear objective with non-linear constraints. Hence, the basic problem being solved is different. (ii) Maximizing entropy is the focus of our present work. The transformation requires addition of new states and actions and hence entropy would get affected. Our transformation has to ensure that maximizing entropy for the transformed MCMDP would be equivalent to the problem of maximizing entropy for the original MCMDP. We provided a mathematical proof later to show that indeed this property holds.

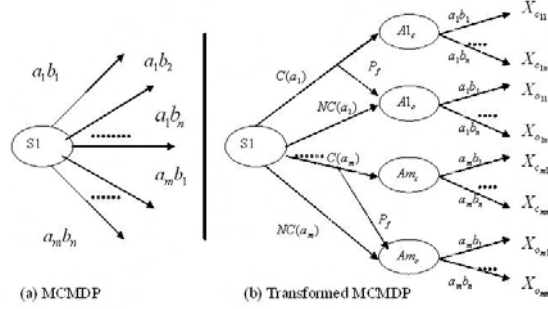
### 3.1 Transformation Methods: Sequential and Others

Figure 2-a shows a portion of a MCMDP, where agent A with actions  $a_1$  to  $a_m$  and B with actions  $b_1$  to  $b_n$  act jointly ( $a_i b_j$ ). Figure 2-b shows the transformation of this MCMDP into transformed MCMDP. This transformation is sequential in that one of the agents, in this case agent A, first chooses one of its actions  $a_i$  and also decides whether to communicate this choice to its teammate, agent B. Thus,  $C(a_i)$  in Figure 2-b refers to A's selection and communication of action  $a_i$  to B, incurring the cost of communication, and going to  $Ai_c$  (with probability  $1-p_f$  where  $p_f$  is the probability with which communication may fail); while  $NC(a_i)$  results in state  $Ai_o$ , where agent A selected  $a_i$  but decided not to communicate this choice to B to avoid communication costs. Note, since communication may fail with a probability  $p_f$ ,  $C(a_i)$  may transition to  $Ai_o$  with a probability  $p_f$ . Once in state  $Ai_c$  or  $Ai_o$ , agent B chooses its action  $b_j$ , and the agents now jointly execute the action  $a_i b_j$ . When choosing its action, B observes which of the different  $Ai_c$  state it is in, since any such state is reached only after A's communication. Unfortunately, agent B cannot distinguish between states  $Ai_o$  reached without A's communication. Thus, B's action  $b_j$  in such non-communication states must be taken without observing which of the  $m$  states  $A1_o$  to  $Am_o$  B is in. Thus, B will be unable to execute any randomized policy which requires it (agent B) to select an action  $b_j$  with a different probability in a state say  $Ai_o$  vs a state  $Ak_o$ . To avoid this problem, we require that for any two states reached after non-communication, the probability of B's action selection must be identical, i.e., for any action  $b_j$  and states  $Ai_o$  and  $Ak_o$ ,  $P(b_j|Ai_o) = P(b_j|Ak_o)$ . This restriction on probability of action execution in the transformed MCMDP translates into the addition of the following non-linear constraints into our Problem 3 applied for the transformed MCMDP, to solve the original MCMDP. Specifically, in terms of the state action variables, given any two states  $Ai_o$  and  $Ak_o$ , and any action  $b_j$ , it is necessary that:

$$Xo_{ij}/(\sum_{u=1}^n Xo_{iu}) = Xo_{kj}/(\sum_{u=1}^n Xo_{ku}) \Rightarrow Xo_{ij} * (\sum_{u=1}^n Xo_{ku}) = Xo_{kj} * (\sum_{u=1}^n Xo_{iu}) \quad (4)$$

Thus, to obtain an optimal randomized policy in the MCMDP, we must solve problem 3 for the transformed MCMDP with these non-convex constraints included in the problem. The optimal policy for a transformed MCMDP thus obtained will require a random selection at state  $S1$  by agent A alone, and then in the next state (either  $Ai_c$  or  $Ai_o$ ) by agent B alone, thus avoiding the problem faced in the MCMDP. The non-linear constraints in the transformed MCMDP affect only the actions taken from states  $A1_o, A2_o, \dots, Am_o$  (from figure 2-b) and ensure that  $P(b_j|A1_o) = P(b_j|A2_o) = \dots = P(b_j|Am_o)$  for  $j \in 1, 2, \dots, n$ . This is because for agent B, states  $A1_o, A2_o, \dots, Am_o$  are indistinguishable, as they are reached without A's communication.

Apart from the addition of these non-linear constraints, the entropy function also undergoes change as the transformed MCMDP has new states and actions



**Fig. 2.** Transformation

added to it. The entropy function for Figure 2-a would be the straightforward  $H_W(x)$  as developed in section 2.1. In the transformed MCMDP, it would still be the  $H_W(x)$  with a small change in the way entropy is calculated at each state. The entropy function at each state as calculated over the probability distribution of all actions at that state is  $H = -\sum_{i=1}^n p_i \log p_i$  where  $p_i$  is the probability of taking action  $a_i$  at that state. Therefore, for state S1 in figure 2-a the entropy is -

$$H(S1) = -1 * (p(a_1b_1) * \log(p(a_1b_1)) + \dots + P(a_1b_n) * \log(p(a_1b_n)) + \dots + P(a_mb_1) * \log(p(a_mb_1)) + \dots + p(a_mb_n) * \log(p(a_mb_n))).$$

If we notice state S1 of Figure 2-b, the probability with which agent A would take action say  $a_1$  would be the sum of the probabilities with which it takes  $C(a_1)$  and  $NC(a_1)$ . This is because whether agent 1 communicates that it would take action  $a_1$  or does not communicate that it would take  $a_1$  is internal to the system because of our assumption (as explained in introduction) that communication is safe. Therefore, only the fact that agent A will take action  $a_1$  (independent of whether it is known to agent B) with certain probability is important to our entropy equation since the enemy gets to observe that as the policy of agent A. Therefore the new entropy function for state S1 in figure 2-b would be

$$H(S1) = -1 * (p(C(a_1) + NC(a_1)) * \log(p(C(a_1) + NC(a_1))) + \dots + p(C(a_m) + NC(a_m)) * \log(p(C(a_m) + NC(a_m))))$$

instead of the entropy function

$$H(S1) = -1 * (p(C(a_1)) * \log(p(C(a_1))) + p(NC(a_1)) * \log(p(NC(a_1))) + \dots + p(C(a_m)) * \log(p(C(a_m))) + p(NC(a_m)) * \log(p(NC(a_m)))).$$

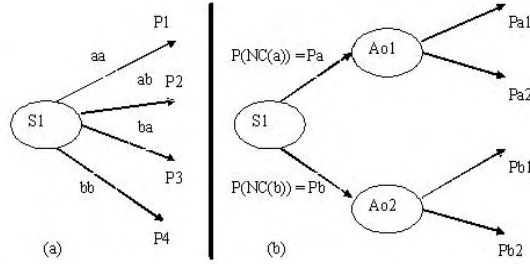
Hence to solve our original MCMDP we solve Problem 3 for the transformed MCMDP using the modified entropy function with the addition of non-linear constraints we described earlier. One interesting fact in Figure 2-a is that the entropy calculation would undergo such a change only for actions of agent A while no such addition of probabilities of C and NC actions is needed for agent B. Given that we now have a new entropy function (calculated using probability of an action of an agent as sum of communication and non-communication probabilities of that action), and also new states and transitions, it might not be nec-



essary that optimizing the entropy function for the transformed MCMDP would automatically mean that we are increasing security for the original problem we were solving. We therefore prove the following lemma below for two cases of communication namely no communication and full communication. The lemma basically states that the under conditions of no communication or full communication the entropy obtained for the MCMDP and the transformed MCMDP would be the same if there are no changes in the reward thresholds to be met and the bandwidth constraints. Under conditions of limited communication, we experimentally verified over a large set of points and found the lemma still holds true although we do not provide a formal proof.

**Lemma 1.** *If in a state say S1 of MCMDP (Figure 3-a), the entropy is defined over the probability distribution of the actions over the state, then the entropy would remain the same in sequential transformation over the whole system of states generated.*

**proof:** For simplicity of proof, let's assume a two agent case where the bandwidth present is zero (no communication case) in the domain. In 3-a, we show the MCMDP where there are four joint actions obtained from the two individual actions a,b of agents 1 and 2. We now transform the MCMDP using our sequential transformation into a transformed MCMDP. We assume agent 1 decides on the communication/non-communication issue. Figure 3-b shows the transformed MCMDP. If the policy of the MCMDP and the transformed MCMDP is the same, then the flows and hence the path probabilities for the four corresponding paths in both figures are equal.



**Fig. 3.** Illustrative Example

Entropy from 3-a:  $Entropy1 = P_1 \log P_1 + P_2 \log P_2 + P_3 \log P_3 + P_4 \log P_4$   
Entropy from 3-b:  $Entropy2 = P_a \log P_a + P_b \log P_b + P_a * (P_{a1} \log P_{a1} + P_{a2} \log P_{a2}) + P_b * (P_{b1} \log P_{b1} + P_{b2} \log P_{b2})$   
Let's consider the terms  $P_a \log P_a + P_a * (P_{a1} \log P_{a1} + P_{a2} \log P_{a2})$   
 $= P_a * (\log P_a + P_{a1} \log P_{a1} + P_{a2} \log P_{a2})$   
Since  $P_{a1} + P_{a2} = 1$ ,  
 $= P_a * ((P_{a1} + P_{a2}) \log P_a + P_{a1} \log P_{a1} + P_{a2} \log P_{a2}) = P_a * (P_{a1} * (\log P_a + \log P_{a1}) + P_{a2} * (\log P_a + \log P_{a2}))$   
 $= P_a P_{a1} * \log(P_a P_{a1}) + P_a P_{a2} * \log(P_a P_{a2})$   
Since the path probabilities are equal,  $P_a P_{a1} = P_1$  and  $P_a P_{a2} = P_2$ . Hence proved

equal to  $P_1 \log P_1 + P_2 \log P_2$ . Similar math applies to the other terms making it equal to  $P_3 \log P_3 + P_4 \log P_4$ . Therefore the entropies of both transformations is the same. The same reasoning as above follows if there is full communication also.

While we showed one particular method of transformation called the sequential transformation with one particular order of communication actions, as shown in Figure 4, there are other methods of transforming a MCMDP into a transformed MCMDP. First, as shown in Figure 4-a, the order of communication actions in the sequential transformation can be changed. If one agent has fewer actions than another (e.g., if  $n < m$ ), such a change in the order of communication may improve the optimality of the resulting policy or reduce communication costs. Second, as shown in Figure 4-b, in a *hierarchical* transformation, an agent first decides which action to select, and only later whether to communicate this choice (C) or not (NC). By choosing an action first, an agent's communication decision may be improved, potentially improving policy optimality. Our third *extra-communication* transformation is similar to the sequential transformation, except that agent A chooses actions for itself and for agent B and communicates the choice of both to agent B. As discussed earlier, this would lead to extra overheads in communication. Finally, our *simultaneous* transformation, is shown in Figure 4-d. Here, while the choice of communication is done sequentially, no communication by A results in state S2; and in S2, agent A and B simultaneously and randomly select their actions. Additionally, combinations of these transformations are also feasible. Typically, we must select from these multiple transformations the one that provides the most optimal policies. However, in this paper, we just introduce the sequential transformation and its properties and leave such an analysis of various transformations for future work.

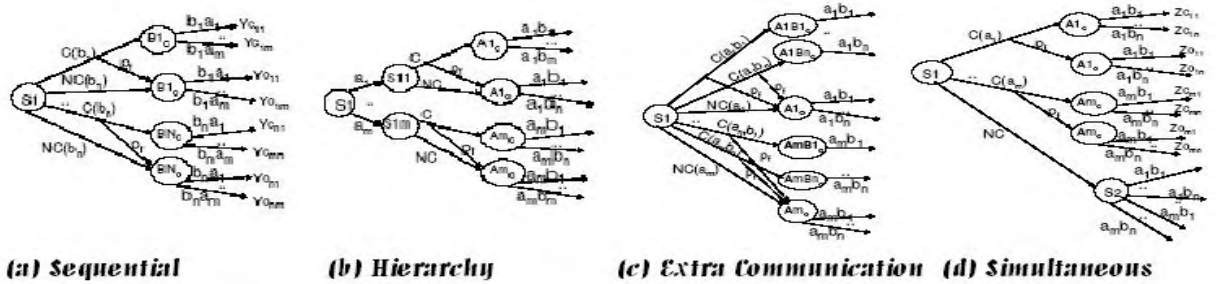


Fig. 4. Other methods of transformation

In all the above transformations, one of the agents selects an action without observation of its actual state, leading to non-linear constraints, e.g., in simultaneous transformation at state S2, agents A and B act simultaneously. Once again, non-linear constraints arise and hence non-linear constraints must be added in the simultaneous case also. Indeed, irrespective of the style of transformation, non-linear constraints must be added. This is because expressing probabilities

of events in MCMDPs requires divisions via Xia variables. And regardless of the transformation that we choose for the MCMDP, we need to express constraints using probabilities. Indeed, all transformations either involve sequential action selection or simultaneous, and we showed non-convex constraints in each case [11]. Thus:

o **Proposition 1:** It is necessary to add non-convex constraints to solve the actual MCMDP.

### 3.2 The Sequential Transformation Algorithm

Since sequential transformation is the basis of our work in this paper, we describe the transformation algorithm for it. We now present Algorithm 2 that achieves this sequential transformation of MCMDP into a transformed MCMDP automatically. (In fact our implementation creates problem 3 with the non-linear constraints as an output). The algorithm works by first adding intermediate states with (and without) communication in *SrcToComm* and then adding transitions from the intermediate states to the destination states in *CommToDest*. We assume that joint actions are processed in increasing order of the index  $i$  ( $1 \leq i \leq m$ ) for  $a_i$ , and  $j$  for  $b_j$  ( $1 \leq j \leq n$ ). In *SrcToComm*, communication actions  $a_i\_c$  leads to state  $sa_i\_c$  with probability  $1-P_{fn}$  (and state  $sa_i\_nc$  with probability  $P_{fn}$ ); and non-communication action  $a_i\_nc$  deterministically transitions to state  $sa_i\_nc$ , where the first agent has decided not to communicate its choice to its teammate. Line 13 in the *Conversion* algorithm adds the constraints on probabilities of outgoing actions from  $sa_i\_nc$  — because of transitivity of equality, it is sufficient to add probability constraints with respect to just the first non-communication state  $sa_1\_nc$ . From line 4 and line 7 of the algorithm, the number of probability constraints can be seen as  $(m-1)*n$  to be later translated into non-linear constraints using equation 4. Thus, this is a polynomial time algorithm, with a complexity of  $O(|S|^2 * |A|)$ , where  $|A| = n * m$  gives us the number of joint actions. In the worst case, the resulting MCMDP has  $2 * |S| * m$  additional states inserted. Given that the output of the transformation algorithm is a nonlinear program with nonlinear constraints our polynomial transformation algorithm does not add anything to the complexity of the problem.

## 4 Experimental Results

Based on the UAV example we described earlier, we first constructed a MCMDP with joint states, actions, transitions and rewards. We then transformed the MCMDP into a transformed MCMDP with the appropriate communication and non-communication actions. We then present results using the transformed MCMDP (Figure 5) to provide key observations about the impact of reward and communication thresholds on policy randomization. Figure 5-a shows the results of varying reward threshold (x-axis) and communication thresholds (y-axis) on the weighted entropy of the joint policies (z-axis). Based on the figure, we make

two key observations. First, with extreme (very low or very high) reward thresholds, communication threshold makes no difference on the value of the optimal policy. In particular, in extreme cases, the actions are either completely deterministic or randomized. On one extreme (maximum reward threshold), agents choose the best deterministic policy and hence communication makes no difference and entropy hits zero. At the other extreme, with low reward threshold (reward threshold 0) agents gain an expected weighted entropy of almost 2 (the maximum possible in our domain), since the agents can choose highest entropy actions and thus communication does not help. Second, in the middle range of reward thresholds, where policies are randomized, communication makes the most difference; indeed, the optimal entropy is seen to increase as communication threshold increases. For instance, when reward threshold is 7, the weighted entropy of the optimal policy obtained without communication is 1.36, but with high communication threshold of 6, the optimal policy provides a weighted entropy of 1.81.

Figure 5-b zooms in on one slice in Figure 5-a (reward threshold fixed at 7). It shows the changes in probability of communication and non-communication actions in the optimal policy (y-axis), with changes in communication threshold (x-axis).  $P(\text{comm } a_i)$  denotes the probability of executing the action to communicate  $a_i$  (similarly for non-communication actions). The graph illustrates the following: when there is no communication in the system, action  $a_1$  gets preferred over  $a_2$  because of reward constraints. Action  $a_1$  would have been chosen with probability 1 but for the fact that entropy needs to be maximized. As communication is increased, most communication is allocated to  $a_1$  as opposed to  $a_2$  because of the high reward to cost ratio for  $a_1$ . The interesting issue that arises here is, at the highest communication point even though after all the communication was used up but action  $a_1$  accounted to only .4 of the total probability (i.e 1), the no communication action  $a_2$  was chosen for the rest of the probability even though  $a_1$  would have provided higher reward. This is due to our assumption that communication is safe, i.e both communication and non-communication actions appear the same to our adversary. If this assumption was not there, most possibly non communication of  $a_1$  should have been chosen with higher probability. In the highest communication threshold case, increasing probability of  $NC(a_1)$  is actually detrimental to entropy since  $P(C(a_1)) + P(NC(a_1))$  might then add up to near 1 making it more deterministic which seems counterintuitive. The other interesting issue is that, as communication threshold increases, the probability of communicative actions increase say  $P(a_1)$  increases from 0 to 0.4. At the same time, the probability of the non-communication actions decreases.

Table 1 compares the weighted entropies of different joint policies with changes in communication threshold for the same example we showed our results on earlier (using a fixed reward threshold of 5). In the first row we show the three settings of the communication thresholds (0,3 and 6 respectively) we use for deriving the entropy values for the various cases in the table. Row 2 shows the entropies obtained by an optimal MCMDP policy. The entropy (1.9) is an ideal upper-bound for benchmarking and the entropy is unaffected by the communica-

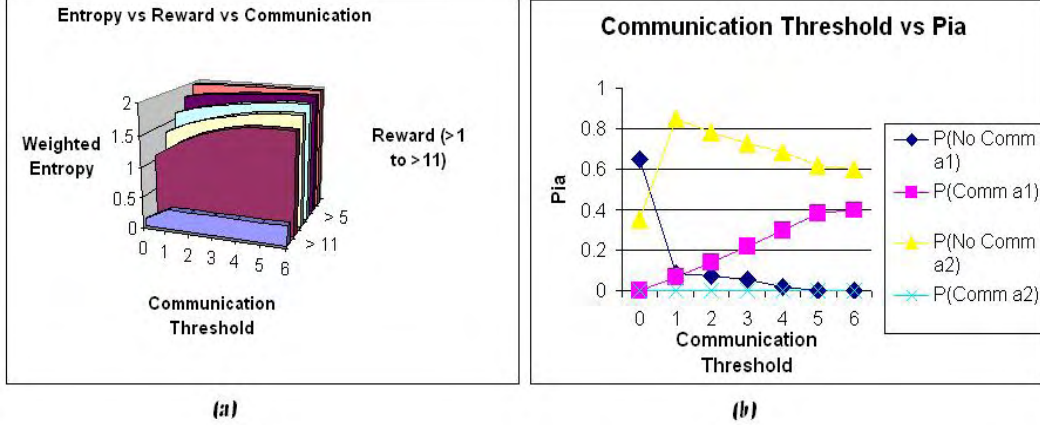


Fig. 5. Effect of thresholds

Table 1: Comparing Weighted Entropies.

<i>Comm Threshold</i> $\rightarrow$	0	3	6
MCMDP	1.9	1.9	1.9
Deterministic	0	0	0
Miscoordination	Yes	Yes	No
Transformed MCMDP	1.6	1.83	1.9

tion threshold. Row 3 illustrates that deterministic policies exist in our domain but their entropy be 0 and hence there would be no security. Row 4 shows the results, where agents take the optimal policy of the MCMDP and attempt to execute it without coordination. Unfortunately, communication constraints are violated in columns 1 and 2. Only when communication resource of 6 units is available the MCMDP policy becomes executable without any miscoordination. Finally, row 5 shows the entropy of the transformed MCMDP for comparison. It is able to avoid the problems faced by policies in row 3 and 4. However, with communication threshold of 0, the transformed MCMDP must settle for an entropy of 1.6; as the communication threshold increases, it finally settles at an entropy of 1.9 which also shows why the MCMDP policy(row 1) becomes executable when communication threshold is 6.

## 5 Summary and Related Work

This paper focuses on coordinating randomized policies for increasing security of multiagent teams acting in observable domains. The issue of security arises here because of intentional threats that are caused by unseen adversaries, whose actions and capabilities are unknown, but the adversaries can exploit any predictability in our agent's policies. Policy randomization with guaranteed rewards

meeting communication constraints becomes critical in such domains. To this end, this paper provides three key contributions. First we recall the MCMDP framework where agents not only maximize their expected team rewards but also bound the expected team consumption of the communication resource. We then developed a non-linear program for this MCMDP that maximizes policy randomization while bounding communication consumption at the same time providing guarantee on the expected team reward obtained. We then show how randomized policies in team settings lead to miscoordination and hence the policies obtained from our non-linear program can be inexecutable. Our second contribution is the introduction of a novel transformation algorithm called the sequential transformation where we can explicitly incorporate communication and non-communication actions. Thus problems may be formulated using our abstract transformed MCMDP and our transformation ensures that the resulting randomized policies avoid miscoordination. We also show the existence of many other such transformations. Third, we showed that despite the fully observable domains, transformed MCMDPs necessitate programs using our non-convex constraints. We then solved our non-linear program with the non-convex constraints on our UAV domain, initially modeled as a MCMDP on which we applied our transformation algorithm to obtain the transformed MCMDP. From these experiments, we showed the various tradeoffs involved between the three key factors namely entropy, reward and communication resources. Finally, while our techniques are applied for analyzing randomization-reward-communication tradeoffs, they could potentially be applied more generally to analyze different tradeoffs between competing objectives in MCMDPs.

Decision-theoretic literature has focused on maximizing total expected reward [18, 3, 19] but maximizing policy randomization as a goal has received little attention in the literature. Randomization is mostly seen as a means or side-effect in attaining other objectives, e.g., in resource-constrained MDPs [14] or limited memory POMDP policy generators [20–24]. In [11] coordination of multiple agents executing randomized policies in a MDP team setting is discussed, but there randomization occurs as a side-effect of resource constraints. The work in [5] explicitly emphasizes on maximizing policy entropy but no resource constraints are considered. In contrast, our work focuses on policy randomization while explicitly ensuring that the communication constraints of the team are met. The effect of communication in multiagent teams has been analyzed extensively [25–28]. However, none of this work focuses on using communication to counter the miscoordination arising due to randomized policies in team settings. Further we model communication as a resource with a cost which is independent of the reward i.e communication costs and rewards cannot be compared and the focus is to make optimal usage of the limited communication unlike heuristic techniques developed earlier for adding communication actions. Significant attention has been paid to learning in stochastic games, where agents must learn dominant strategies against explicitly modeled adversaries [15, 29]. Such dominant strategies may lead to randomization, but randomization itself is not the goal. Our work in contrast does not require any model of the adversary and un-

der this worst case assumption hinders any adversary's actions by increasing the policy's weighted entropy. Thus, we focus on agent teams using Decentralized MDPs with communication constraints doing intentional policy randomization.

**Acknowledgments :** This research is supported by NSF grants #0208580, #0222914 & ISF #8008. It is also supported by the United States Department of Homeland Security through Center for Risk and Economic Analysis of Terrorism Events (CREATE). Sarit Kraus is also affiliated with UMIACS.

## References

1. M. H. Burstein, A. M. Mulvehill, and S. Deutsch. An approach to mixed-initiative management of heterogeneous software agent teams. In HICSS, page 8055. IEEE Computer Society, 1999.
2. C. Boutilier. Sequential Optimality and Coordination in Multiagent Systems. In IJCAI, 1999.
3. R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Transition-Independent Decentralized Markov Decision Processes. In AAMAS, 2003.
4. R. Nair, D. Pynadath, M. Yokoo, M. Tambe, and S. Marsella. Taming Decentralized POMDPs: Towards Efficient Policy Computation for Multiagent Settings. In IJCAI, 2003.
5. P. Paruchuri, M. Tambe, F. Ordonez, and S. Kraus. Security in Multiagent Systems by Policy Randomization. In AAMAS, 2006.
6. D. Carroll, K. Mikell, and T. Denewiler. Unmanned Ground Vehicles for Integrated Force Protection. In SPIE Proc. 5422, 2004.
7. P. J. Lewis, M. R. Torrie, and P. M. Omilon. Applications suitable for unmanned and autonomous missions utilizing the Tactical Amphibious Ground Support (TAGS) platform. <http://www.autonomoussolutions.com/Press/SPIE%20TAGS.html>, 2005.
8. Call for Papers: Safety and Security in Multiagent Systems. <http://www.multiagent.com/dailist/msg00129.html>.
9. R. Beard, and T. McLain. Multiple UAV Cooperative Search under Collision Avoidance and Limited Range Communication Constraints. In IEEE CDC, 2003.
10. A. Serjantov. On the Anonymity of Anonymity Systems. PhD Dissertation, University of Cambridge, 2004.
11. P. Paruchuri, M. Tambe, F. Ordonez, and S. Kraus. Towards a Formalization of Teamwork With Resource Constraints. In AAMAS, 2004.
12. M. H. Rahimi, H. Shah, G. S. Sukhatme, J. Heidemann, and D. Estrin. Studying the Feasibility of Energy Harvesting in a Mobile Sensor Network. In ICRA, 2003.
13. D. Dolgov, and E. Durfee. Approximating Optimal Policies for Agents with Limited Execution Resources. In IJCAI, 2003.
14. E. Altman. Constrained Markov Decision Process. Chapman and Hall, 1999.
15. M. Littman. Markov Games as a Framework for Multi-Agent Reinforcement Learning. [citeseer.ist.psu.edu/littman94markov.html](http://citeseer.ist.psu.edu/littman94markov.html), 1994.
16. D. Dolgov, and E. Durfee. Resource Allocation and Policy Formulation for Multiple Resource-Limited Agents Under Uncertainty. In ICAPS, 2004.
17. C. Shannon. A Mathematical Theory of Communication. In The Bell Labs Technical Journal, 1948.
18. D. Pynadath, and M. Tambe. The communicative multiagent team decision problem: analyzing teamwork theories and models. JAIR, 2002.

19. C. V. Goldman, and S. Zilberstein. Optimizing Information Exchange in Cooperative Multi-agent Systems. In AAMAS, 2003.
20. T. Jaakkola, S. Singh, and M. Jordan. Reinforcement learning algorithm for partially observable markov decision problems. In Advances in NIPS, 1994.
21. R. Parr and S. Russel. Approximating Optimal Policies for partially observable stochastic domains. In IJCAI, 1995.
22. L. Kaelbling, M. Littman, and A. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. In Technical Report, Brown University, 1995.
23. P. Poupart, and C. Boutilier. Bounded finite state controllers. In NIPS, 2003.
24. D. S. Bernstein, E. A. Hansen, and S. Zilberstein. Bounded Policy Iteration for Decentralized POMDPs. In IJCAI, 2005.
25. P. Xuan, and V. Lesser. Multi-Agent Policies: From Centralized Ones to Decentralized Ones. In AAMAS, 2002.
26. R. Becker, V. Lesser, and S. Zilberstein. Analyzing Myopic Approaches for Multi-Agent Communication. In Proceedings of IAT, 2005.
27. M. Ghavamzadeh, and S. Mahadevan. Learning to Communicate and Act in Cooperative Multiagent Systems using Hierarchical Reinforcement Learning. In AAMAS, 2004.
28. R. Nair, M. Roth, M. Yokoo, and Milind Tambe. Communication for Improving Policy Computation in Distributed POMDPs. In AAMAS, 2004.
29. J. Hu, and P. Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. In ICML, 1998.



---

**Algorithm 2** CONVERT()

---

```
1: Input:  $\langle S, A, P, R, N, Q \rangle$ 
2: Output:  $\langle S', A', P', N', Q' \rangle$ 
3: Conversion()
4: Create Problem 3 from Output.
1: Conversion(){
2: Initialize:  $S' = S, A' = A, P' = P, R' = \phi, N' = \phi, Q' = Q$ 
3: for all  $s \in S$  do
4:   for all  $(\hat{a} = (a_i, b_j)) \in A$  do
5:     if  $sa_{i-nc} \notin S'$  then
6:       SrcToComm( $s, \hat{a}, sa_{i-nc}, a_{i-nc}$ )
7:        $p'(s, \hat{a}, sa_{i-nc}) \leftarrow 1$ 
8:       if  $(|p(s, \langle a_i, * \rangle, *) > 0| > 1)$  then
9:         SrcToComm( $s, \hat{a}, sa_{i-c}, a_{i-c}$ )
10:         $n'(s, a_{i-c}) \leftarrow \text{Communication\_Model}$ 
11:         $p'(s, a_{i-c}, sa_{i-c}) \leftarrow 1 - P_f$ 
12:         $p'(s, a_{i-c}, sa_{i-nc}) \leftarrow P_f$ 
13:      if  $i \neq 1$  then
14:         $prob(b_j | sa_{i-nc}) = prob(b_j | sa_{1-nc})$ 
15:        CommToDest( $s, \hat{a}, sa_{i-nc}, a_{i-nc}$ )
16:      if  $(|p(s, \langle a_i, * \rangle, *) > 0| > 1)$  then
17:        CommToDest( $s, \hat{a}, sa_{i-c}, a_{i-c}$ )
18:      for all  $s' \in S'$  do
19:         $p'(s, \hat{a}, s') \leftarrow 0$ 
20: }
1: SrcToComm( $S_{parent}, A_{parent}, S_{current}, A_{current}$ ){
2:  $S' \leftarrow S' \cup S_{current}$ 
3:  $A' \leftarrow A' \cup A_{current}$ 
4:  $r'(S_{parent}, A_{current}), n'(S_{parent}, A_{current}) \leftarrow 0$ 
5: }
1: CommToDest( $S_{parent}, A_{parent}, S_{current}, A_{current}$ ){
2: for all  $s' \in S'$  do
3:    $p'(S_{current}, A_{parent}, s') \leftarrow p(S_{parent}, A_{parent}, s')$ 
4:    $r'(S_{current}, A_{parent}) \leftarrow r(S_{parent}, A_{parent})$ 
```

---

# Uncertain Agent Verification through Probabilistic Model-Checking

Paolo Ballarini<sup>1</sup>, Michael Fisher<sup>2</sup>, and Michael Wooldridge<sup>2</sup>

<sup>1</sup> University of Glasgow, Glasgow, G12 8RZ, UK,  
paolo@dcs.gla.ac.uk

<sup>2</sup> University of Liverpool, Liverpool L69 7ZF, UK,  
{michael,mjw}@csc.liv.ac.uk

**Abstract.** In many situations an agent’s behaviour can sensibly be described only in terms of a distribution of probability over a set of possibilities. In such case (agents’) decision-making becomes probabilistic too. In this work we consider a probabilistic variant of a well-known (two-players) Negotiation game and we show, first, how it can be encoded into a Markovian model, and then how a probabilistic model-checker such as PRISM can be used as a tool for its (automated) analysis. This paper is meant to exemplify that verification through model-checking can be fruitfully applied also to *uncertain* multi-agent systems. This, in our view, is the first step towards the characterisation of an automated verification method for probabilistic agents.

## 1 Introduction

Because of their notorious complexity, multi-agent systems’ run-time behaviour is extremely hard to predict and understand and, as a result, agents’ verification is a hard task. In recent years significant research effort has been invested in the development of *formal methods* for the specification and verification of multi-agent systems [15] and model-checking has been proved to be a possibility in that respect [3]. Bordini *et al.*, for example, have shown how LTL model-checking techniques [14] can be applied, by means of the SPIN model-checker [10], to the verification of a specific class of *non-probabilistic* rational agents (i.e. BDI agent-systems expressed as *AgentSpeak* programs) [4]. In [6], Dix *et al.*, argue that in several real-life situations an agent behaviour (i.e. the state an agent is in at a certain time) may be known with a given degree of uncertainty. As a consequence a specific language for *probabilistic* agents (i.e. a language that allows for a quantification of such uncertainty) is needed and that is what the authors develop throughout their work. Given that probability distributions are sensible means to describe the uncertainty between several possibilities (i.e. several potential successor states of a given, current state) then the type of analysis involved must also be a *probabilistic* one. In a probabilistic framework, the analyst is interested in discovering the measure with which certain properties (either negative or positive ones) are likely to be true. If we refer to a simple negotiation game where two players, a seller and a buyer, bargain over a single item, then a player’s

decision may well be considered a probabilistic one, as an expression of the player’s uncertainty towards their opponent’s behaviour. In such a situation it is relevant to assess how uncertainty affects the negotiation outcome (hence players’ payoff). Questions like “what is the probability that an agreement will be reached at all?”, “how likely is it that the bargained item will be sold to a given value  $x$ ?”, and “what type of strategy gets a player a better *expected payoff*?”, are amongst the ones an analysis of an uncertain negotiation framework is meant to tackle.

In this work we show how a probabilistic model-checker, such as PRISM[13], can be used as a tool to find answers to those type of questions, once the considered probabilistic agent system has been mapped onto a Markovian model, specifically a Discrete Time Markov Chain (DTMC). So as LTL model-checking, through SPIN, has been proved to be suitable for the analysis of *non-probabilistic* multi-agent systems, we show here that PCTL model-checking [9], through PRISM, is a proper tool for verifying *probabilistic* agent systems. What is still lacking, at present, is an automatic translation of probabilistic agent programs, such as, for example, the ones proposed in [6], into the Reactive Module language, the input formalism for PRISM. This will be the focus of our future work. The remainder of the paper is organised as follows: in the next Section we introduce the Negotiation game and describe the probabilistic variant we have considered. In Section 3 a brief introduction to the PCTL model-checking and to the PRISM tool is presented, whereas in Section 4 results obtained by verification of specific PCTL formulae through PRISM are presented. The final section summarises our analysis and lists directions for future work.

## 2 Alternating-Offers Negotiation Framework

In this section we present the Negotiation framework we have considered in our work and describe the probabilistic extension we have introduced and studied. The Bargaining process we refer to is the Alternating-Offers one discussed in [12] and [7]. In the basic formulation of such a game, two players bargain over a single item by alternatively throwing an agreement proposal and making a decision over the opponent’s last proposed value. Bargainers’ interest is clearly conflicting with the *seller* aiming to maximise the outcome of negotiation (i.e. the *agreed-value*) and the *buyer* aiming to minimise it. Players’ behaviour is characterised by a strategy which essentially determines two aspects: a player’s next offer value and a player’s *acceptance condition* (which describes if the opponent’s most recent proposal is going to be accepted/rejected). Negotiation analysis aims to study properties of the players’ strategies and, in particular, is interested in addressing the existence of dominant strategies and strategic equilibrium. However this type of analysis is not tractable in incomplete information settings, which is what we are considering in this paper. Nonetheless assessing the effectiveness of strategies for automated negotiation is relevant from an AI perspective. In the following we briefly describe the family of time-dependent strategies for automated negotiation between agents introduced by Faratin *et al.* in [7], which

is the one we use in our probabilistic framework. Players' strategies depend on 2 intervals, respectively  $[min_b, max_b]$ , where  $max_b$  is the buyer's reservation price and  $min_b$  is the buyer's lowest acceptable offer (i.e.  $min_b=0$ ), and  $[min_s, max_s]$ , where  $min_s$  is the seller's reservation price and  $max_s$  represents the seller's upper bound of a valid offer. A player's initial offer is his most profitable value, which is  $min_b$ , for the buyer and  $max_s$  for the seller. The basic idea of this family of strategies is that a player concedes over time and the pace of concession determines the type of negotiation strategy. Time is thought of as a discrete quantity indicating negotiation periods (i.e. players' turn),  $T$  denotes the time-deadline and a player's offer at time  $t \in [0, T - 1]$ , depicted, respectively, as  $x_b^t$  and  $x_s^t$ , is defined as:

$$x_b^t = min_b + \alpha_b(t)(max_b - min_b), \quad (1)$$

$$x_s^t = min_s + (1 - \alpha_b(t))(max_s - min_s). \quad (2)$$

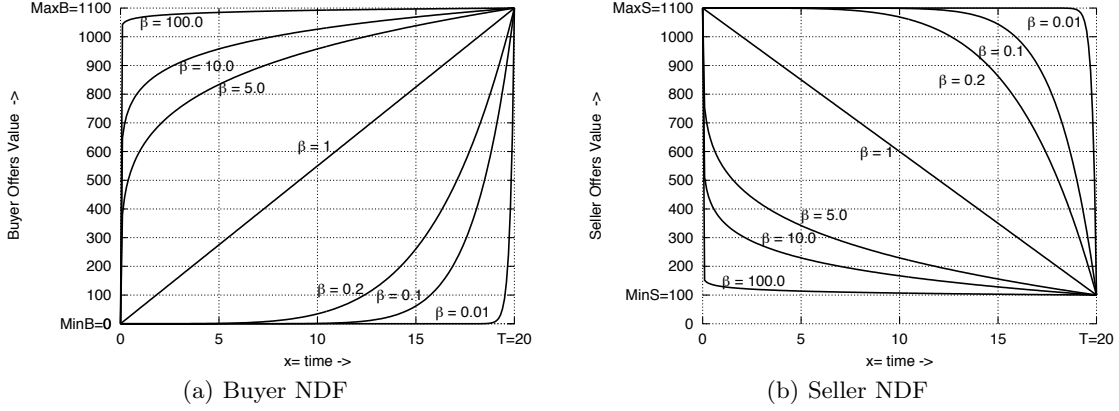
where  $\alpha_i(t)$ ,  $i \in \{b, s\}$  is the time-dependent function in which the constant  $\beta$  determines the conceding pace of player  $i$ :

$$\alpha_i(t) = (\frac{t}{T})^{(\frac{1}{\beta})}. \quad (3)$$

By varying  $\beta \in (-\infty, \infty)$  in (3) a whole family of offer functions, also called Negotiation Decision Functions (NDFs), can be obtained (see Figure 1(a) and Figure 1(b)). For  $\beta > 1$ , strategies are referred to as *Conceder* strategies whereas, for  $\beta < 1$ , strategies belong to the so-called *Boulware* class. With  $\beta = 1$ , on the other hand, we have linear strategies for which a player's offer is monotonically incremented (decremented) over time. Finally, the decision making on a received offer is driven by profitability. Hence the buyer will accept the seller's offer at time  $t$  if, and only if, it is more profitable than his next offered value. Formally:  $x_s^t$  is accepted by  $b$  if, and only if,  $x_s^t \leq x_b^{t+1}$  (similarly  $x_b^t$  is accepted by  $s$  if, and only if,  $x_b^t \geq x_s^{t+1}$ ). For the sake of simplicity we have chosen specific settings for the bargainer's bounds, namely:  $min_b=0$ ,  $max_b=1100$  and  $min_s=100$ ,  $max_s=1100$ . Such a choice is motivated by the constraints of the input language of the PRISM model-checker, in which the only numerical type allowed is integer. As a consequence, in order to be able to observe the effect that different strategy slopes have on the negotiation outcome, a sufficiently wide interval (i.e. corresponding to 1000 units in our case) should be considered. Finally we consider  $T = 20$  as the time deadline for the negotiation process.

## 2.1 Probabilistic Decision Making

As a result of the decision mechanism introduced in [7], an agreement between the buyer and the seller is reached only if the offer functions (Figure 1(a) and Figure 1(b), respectively) cross each other within the time deadline (i.e.  $T$ ). If



**Fig. 1.** Buyer-Seller Negotiation Decision Functions

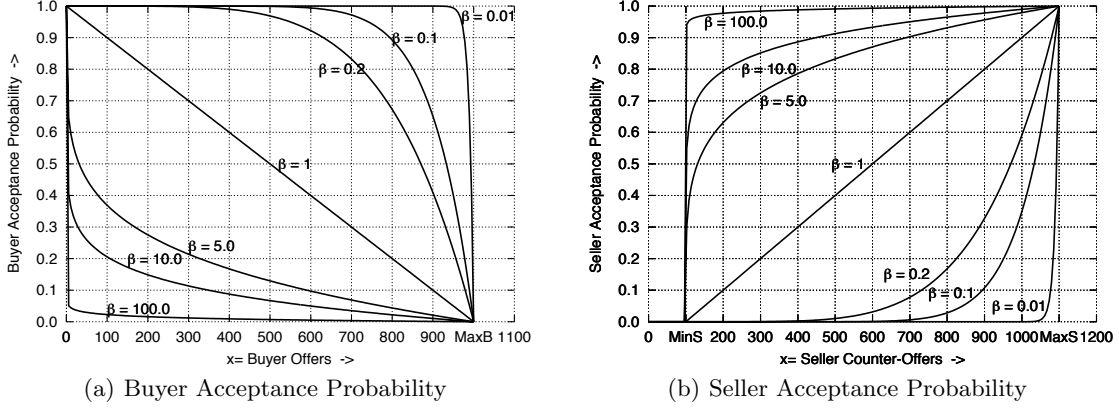
this is the case the result of negotiation is positive and a deal is implemented at the crossing point.<sup>3</sup>

In our framework we introduce uncertainty by making a player's decision a probabilistic function of the offered value. By doing so, we enlarge the semantics of the negotiation model so as to cope with the inherent uncertainty that may characterise a player's behaviour when it comes to deciding over a deal proposal. Such uncertainty may reflect several factors such as: lack of information about the opponent's preferences/strategy, a change of a player's attitude as a result of environmental changes and/or time passing. In order to reduce the complexity of our model, uncertainty, rather than explicitly being a function of time, is quantified with respect to the offered value only (however, since we are adopting time-dependent offer functions the acceptance probability is also, indirectly, dependent on time). Taking inspiration from the NDFs (1) and (2), we introduce the acceptance probability functions,  $S\_AP()$  for the seller and  $B\_AP()$  for the buyer, in the following way:

$$S\_AP(x_b^t) = \begin{cases} 0 & \text{if } (x_b^t \leq \min_s) \\ \left( \frac{x_b^t - \min_s}{\max_s - \min_s} \right)^{\frac{1}{\beta_s}} & \text{if } (x_b^t > \min_s) \wedge (x_b^t < x_s^{t+1}) \\ 1 & \text{if } (x_b^t \geq x_s^{t+1}), \end{cases} \quad (4)$$

$$B\_AP(x_s^t) = \begin{cases} 0 & \text{if } (x_s^t \geq \max_b) \\ 1 - \left( \frac{x_s^t}{\max_b} \right)^{\frac{1}{\beta_b}} & \text{if } (x_s^t < \max_b) \wedge (x_s^t > x_b^{t+1}) \\ 1 & \text{if } (x_s^t \leq x_b^{t+1}). \end{cases} \quad (5)$$

<sup>3</sup> In a monetary negotiation the actual agreement would correspond to rounding up/down to the closest unit of exchange (a penny valued price, if we are referring to the UK market) of such crossing point, depending on the accepting agent.



**Fig. 2.** Buyer-Seller Acceptance Probability Functions

Each definition depends on the parameter  $\beta_i$ ,  $i \in \{b, s\}$ , whose value determines the type of function. A player's attitude (in decision making) is called *conservative* if the likelihood of accepting low profitable offers is very small. Thus a conservative attitude, for the buyer, corresponds to  $\beta_b \gg 1$ , whereas, for the seller, is given by  $\beta_s \ll 1$ . The parametric definition for the acceptance probability functions allows us to assess the effect of different player's attitudes on the negotiation outcome (as we will see we verify several configurations of our model corresponding to different combinations,  $(\beta_s, \beta_b)$ , of the acceptance probability functions). Finally we point out that, by replacing the decision-making model of Faratin *et al.* with the probabilistic mechanism determined by (4) and (5), we still maintain the semantics of the original; in fact the acceptance of an offer corresponding to the crossing point of buyer's and seller's NDFs is certain.

### 3 The PRISM Model of Negotiation

In this section, we describe the Markovian model of the negotiation framework (with probabilistic behaviour) introduced in the previous section. We developed a Discrete Time Markov Chain (DTMC) which represents the behaviour of two bargainers alternatively throwing offers according to the NDF families (1) and (2) and adopting the probabilistic decision mechanism described by the family of functions (4) and (5). We have used the PRISM model-checker to implement and verify the DTMC model of negotiation. Before describing some details of the DTMC model we provide brief background to the PRISM tool and to the Probabilistic Computational Tree Logic (PCTL), the temporal logic used for the verification of DTMC models (for a more detailed treatment of the subject the interested reader is referred to the vast literature, examples of which are [1, 9, 11, 13]).

### 3.1 The PCTL Logic and PRISM

Markov chains [8] are stochastic processes suitable for modelling systems such that the probability of possible future evolutions depends uniquely on the current state rather than on its *past history*. A system's timing is also taken care of with Markov chain models leading to either DTMCs, for which time is considered as discrete quantity, or Continuous time Markov chains (CTMC), where time is continuous. Both DTMC and CTMC models can be encoded in PRISM by means of a variant of the *Reactive Modules* formalism of Alur and Henzinger [1].

For issuing queries, PRISM uses either the *Probabilistic Computational Tree Logic* (PCTL) [9], if the underlying model is a DTMC, or the Continuous Stochastic Logic (CSL) [2] for referring to CTMC models. These languages are variations of the temporal logics used for more conventional Labelled Transition System model checking (i.e. the CTL of Clarke *et al.* [5]). Some basic concepts concerning PCTL model-checking, which is what we consider in this work, are briefly introduced in the following definitions.

**Definition 1.** *Given a set of atomic propositions  $AP$ , a labelled DTMC  $\mathcal{M}$  is a tuple  $(S, \mathbf{P}, L)$  where  $S$  is a finite set of states,  $\mathbf{P}: S \times S \rightarrow [0, 1]$  is the transition probability matrix such that  $\forall s \in S, \sum_{s' \in S} \mathbf{P}(s, s') = 1$  and  $L: S \rightarrow 2^{AP}$  is a labelling function.*

**Definition 2.** *For  $\mathcal{M} = (S, \mathbf{P}, L)$  a labelled DTMC, a path  $\sigma$  from state  $s_0 \in S$  is an infinite sequence  $\sigma = s_0 \rightarrow s_1 \rightarrow \dots \rightarrow s_n \rightarrow \dots$  such that  $\forall i \in \mathbb{N}, \mathbf{P}(s_i, s_{i+1}) > 0$ . Given  $\sigma$ ,  $\sigma[k]$  denotes the  $k$ -th element of  $\sigma$ .*

**Definition 3.** *For  $\sigma$  a path of a labelled DMTC  $\mathcal{M} = (S, \mathbf{P}, L)$  and  $n \in \mathbb{N}$ , let  $\sigma \upharpoonright n = s_0 \rightarrow \dots \rightarrow s_n$  be a finite path of  $\mathcal{M}$ . The probability measure of the set of (infinite) paths prefixed by  $\sigma \upharpoonright n$  is*

$$Prob(\sigma \upharpoonright n) = \prod_{i=0}^{n-1} \mathbf{P}(s_i, s_{i+1})$$

*if  $n > 0$ , whereas  $Prob(\sigma \upharpoonright n) = 1$  if  $n = 0$ .*

**Definition 4 (PCTL syntax).** *For a set of atomic propositions  $AP$ , the syntax of PCTL state-formulae ( $\phi$ ) and path-formulae ( $\varphi$ ) is inductively defined as follows:*

$$\begin{aligned} \phi &:= a \mid tt \mid \neg\phi \mid \phi \wedge \phi \mid \mathcal{P}_{\leq p}(\varphi) \\ \varphi &:= \phi \ U^{\leq t} \ \phi \end{aligned}$$

*where  $a \in AP$ ,  $p \in [0, 1]$ ,  $t \in \mathbb{N}^* \cup \{\infty\}$  and  $\leq \in \{\geq, >, \leq, <\}$ ,*

The PCTL semantics is as the CTL one except for probabilistic path-formulae. The formula  $\mathcal{P}_{\leq p}(\varphi)$  is satisfied in a state  $s$  if, and only if, the probability measure

of paths starting at  $s$  and satisfying  $\varphi$ , denoted  $Prob(s, \varphi)$ , fulfils the bound  $\leq p$ . Formally:

$$s \models \mathcal{P}_{\leq p}(\phi' U^{\leq t} \phi'') \text{ iff } Prob(s, (\phi' U^{\leq t} \phi'')) \leq p, \quad (6)$$

where the semantics of  $(\phi' U^{\leq t} \phi'')$  with respect to a path  $\sigma$  is defined as:

$$\sigma \models \phi' U^{\leq t} \phi'' \text{ iff } \exists i \leq t : \sigma[i] \models \phi'' \wedge \forall j < i, \sigma[j] \models \phi'. \quad (7)$$

Essentially, PCTL extends CTL's expressiveness in two ways: by allowing a continuous *path-quantification* (i.e. CTL *existential* and *universal* path quantifiers are replaced by a single *continuous* quantifier, namely  $\mathcal{P}_{\leq p}$ )<sup>4</sup> and by introducing a *discrete* time-bounding for Until-formulae. For a complete treatment of PCTL we refer the reader to [9].

### 3.2 Modelling Probabilistic Negotiation with PRISM

In this section we describe how we have built the DTMC model of negotiation through PRISM. For the sake of space we do not include any sample from the actual PRISM source file. However we discuss the most relevant characteristics of the resulting models, some of which are a consequence of the expressiveness constraints of PRISM input language. The model consists of three modules: a *Timer* (to keep track of turns); a *Buyer*; and a *Seller*. The modules' parallel composition is synchronised over two events: time-elapsing (driven by the timer) and offer proposals (alternatively driven by the buyer and seller).

**Piecewise approximation of the NDFs:** PRISM input language allows for representing (finitely many) integer values only (i.e. only finite-states system can be modelled). As a result the continuous NDFs (1) and (2) are, in our PRISM models, approximated by (sampling from) piecewise linear functions consisting of two pieces<sup>5</sup> (see, for example, Figure 3). The offer function has three parameters: the slope of the first piece, the slope of the second piece and the boundary (switch time) between the pieces (hence a setting for a NDF approximation is given by a triple  $(n_1, n_2, n_3)$  where the first 2 elements represent the slopes of the first and second piece while the third element is the switch-time; for example, the curves in Figure 3 refer to a setting  $(500, 1, 2)$ ). The desired setting (boulware/conceder strategy) is chosen through model configuration so that different *tactic profiles* are verified.

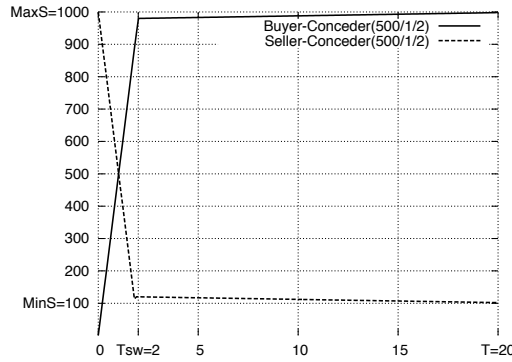
**Model's configuration:** the model we developed is designed to be highly configurable through a number of constants. Before running a verification experiment a configuration is chosen by setting up: the Buyer and Seller NDFs (each

<sup>4</sup> PCTL is a superset of CTL's as:  $E(\phi U \psi) \equiv \mathcal{P}_{>0}(\phi U \psi)$  and  $A(\phi U \psi) \equiv \mathcal{P}_{\geq 1}(\phi U \psi)$

<sup>5</sup> A two piece line is a good approximation for *extreme* bargaining tactics (i.e.  $\psi \sim 0$  or  $\psi \gg 1$ ), which is the type of non-linear tactics we address in this work. Less extreme strategies may be better approximated by multi-piece lines, which would require minimal modifications to our model in order to be coped with.



of which requires three parameters, first-piece-slope, second-piece-slope, switch-time), the buyer and seller reservation-price and initial-offer, the buyer and seller acceptance attitude (i.e. the parameter  $\beta$  for both acceptance probability functions (4) and (5)) and the time-deadline (which we set to 20 turns for all experiments). In order to be able to assess the effect of substantially different NDFs (i.e. to compare how NDF's slopes affect the result of negotiation) we had to allow for a wide enough *acceptance interval* (i.e. the interval  $[min_s, max_b]$ ). As a result we have chosen a default settings of  $[1, 1000]$  for such an interval. This allows us to sensibly verify the effect of slopes values up to 2 orders of magnitude different (i.e. 1-10-100). This is the reason why the graphs reporting the results of our analysis (see next section) refer to such a setting (the x-axis range is referred to  $[0, 1000]$ ).



**Fig. 3.** NDFs' piecewise linear approximation

## 4 Analysis

In this section we describe the result of the verification performed through the PRISM model-checker on the DTMC model of Negotiation.

We observe that, the probabilistic decision making mechanism encoded within the bargaining DTMC model results in a probability distribution over the set of possible outcomes of the negotiation (i.e. the interval  $[min_s, max_s] \subset \mathbb{N}$ ), hence over a strategy profile's payoff.

We have used the verification facilities of PRISM to evaluate the distribution of probability of two distinct aspects of the model behaviour: the value at which an agreement is reached, and the delay for reaching an agreement. These measurements can be automatically derived by running a number of PCTL formulae verifications (the so-called PRISM 'experiment' facility, by means of which a parametric PCTL formula is iteratively verified for each specified value of its

parameters). The corresponding PCTL-like PRISM temporal formula for verifying the probability of reaching an agreement at value  $x$  is as follows:

$$P=? (tt \ U(agreement) \wedge (PURCHASE = x)), \quad (8)$$

whereas the temporal formula for determining the probability of an agreement to be reached at time  $t$  is:

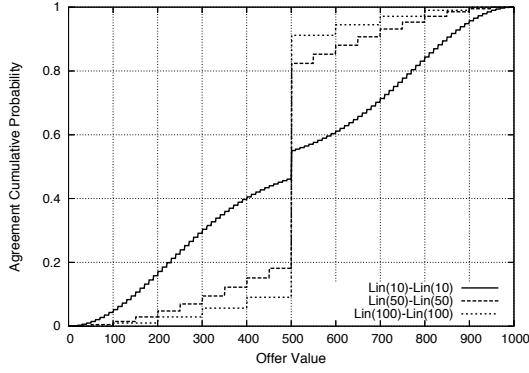
$$P=? (tt \ U(agreement) \wedge (TIME = t)). \quad (9)$$

The reader should not be misled by formulae (8) and (9), in which the PCTL syntax described by Definition 4 is slightly abused by means of the ‘=?’ notation. This is the way PRISM allows for specifying an experiment with respect to one or more parameters. For example, the result of running a PRISM experiment on (8), is that the probability of reaching (at some point in the future) a state in which an agreement is implemented at  $x$ , is computed for every value of the  $x$ -range which has given as input of the experiment (hence by choosing  $[min_s, max_b]$  as input range for an experiment over (8), we end up deriving the distribution of probability over the set of possible agreement values).

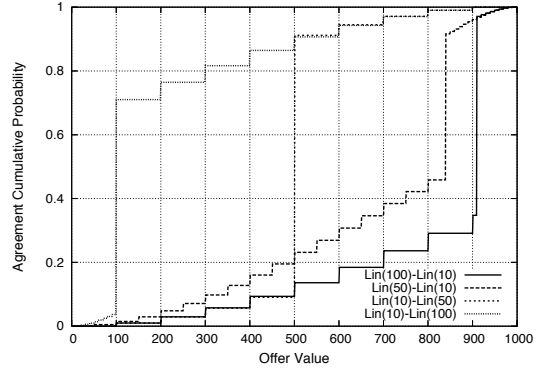
In the following we report about the results of the model analysis obtained by verification of (8) and (9) through PRISM experiments. These are grouped in three different categories, each one of which copes with a different aspect of the model analysis.

**Agreement distribution as a function of players NDFs:** here we discuss experiments which aim to assess how players’ NDFs affect the probabilistic outcome of negotiation. For this reason, in these experiments we have compared model’s configurations by varying combinations of NDFs while sticking with a specific (fixed) combination of Acceptance Probability Function (specifically the one corresponding to  $\beta_b = 0.2$  and  $\beta_s = 5$ ). The results we present are grouped according to different combination of NDFs. Each combination is denoted by a pair  $F_b(x_b/y_b), F_s(x_s/y_s)$  where  $F_a$  ( $a \in \{b, s\}$ ) denotes the type of function for player  $a$  (i.e. either *Lin*, *Boul*, or *Conc*), whereas  $x_a$  and  $y_a$  denote, respectively the first and second piece’s slopes.<sup>6</sup> Hence, for example, *Lin*(10)-*Lin*(100) denotes a profile for which both players are using a linear offer function, the buyer with slope 10 and the seller with slope 100, whereas the profile *Boul*(1/100)-*Conc*(100/1) corresponds to the buyer using a Boulware offer function with first slope 1 and second slope 100 and the seller using a conceder tactic with first and second slope respectively 100 and 1. Figure 4(a) and 4(b) compares the cumulative probability distribution (over the default interval,  $[1, 1000]$ , of possible agreements), for several configurations of, respectively, symmetrical and asymmetrical, linear NDFs. Some general indications can be drawn from these pictures. For example if players use symmetrical NDFs then a higher concession pace tends to favour the seller. This is also confirmed by the expectation values which for the symmetrical case show a (slow) increasing trend  $Exp(Lin(10)) \sim 498$ ,

<sup>6</sup> as linear functions consist of a single piece, then for *Lin<sub>a</sub>* only  $x_a$  is used.



(a) Linear Symmetric NDF

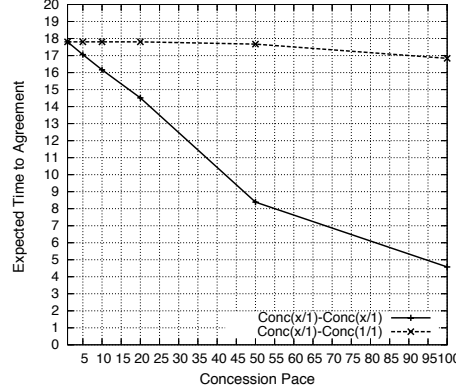


(b) Linear Asymmetric NDF

**Fig. 4.**  $Lin(x)$ - $Lin(y)$  NDF profiles

$Exp(Lin(50)) \sim 499$  and  $Exp(Lin(100)) \sim 500$  confirming the seller's advantage with fast concession behaviour. In case of asymmetrical NDFs (Figure 4(b)), instead, it is generally more convenient for a player to minimise his concession pace, as this is going to get him a more profitable expected outcome. For example if we compare the curves for profiles  $Lin(50)$ - $Lin(10)$ , and  $Lin(100)$ - $Lin(10)$  the probability tend to cumulate closer to the supremum of the interval (which is good for the seller). Again this is confirmed by looking at the expectation values, which show the same tendency, with  $Exp(Lin(50)Lin(10)) \sim 692$ , whereas  $Exp(Lin(100)Lin(10)) \sim 804$ .

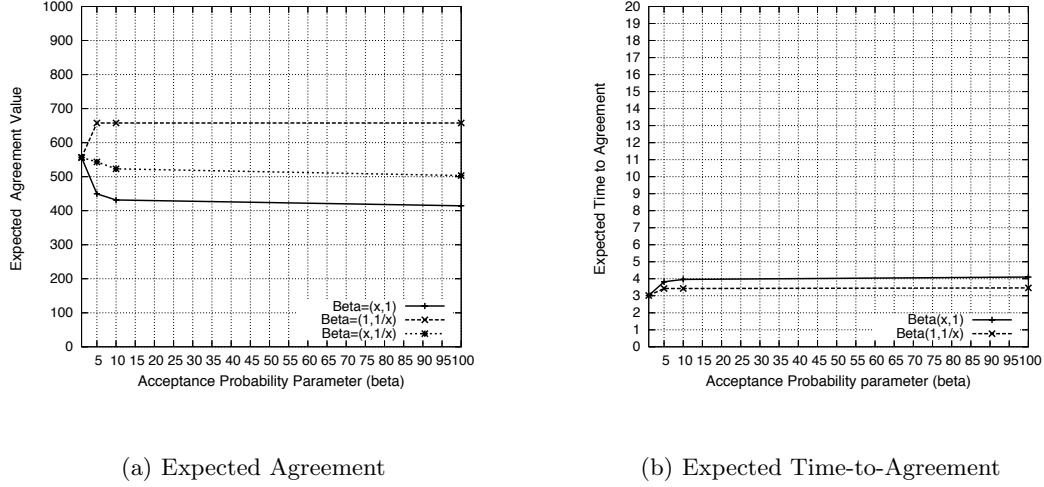
**Agreement delay as a function of players NDFs:** here we discuss the effect that players' NDFs have on the delay for reaching an agreement. Again we consider several combinations of NDFs, but this time we deal with the verification of (9). Figure 5 depicts the expected time-to-agreement as a function of the players' concession pace. It indicates that a faster concession results in a quicker agreement. It should be noted that the discrepancy between the symmetrical case (i.e.  $Conc(x/1) - Conc(x/1)$ ) and the asymmetrical one (i.e.  $Conc(x/1) - Conc(1/1)$ ) is due to the early crossing of NDF curves. In fact, when both player speed up concession at the same pace (symmetrical), NDFs curves intersect earlier (and for  $x > 25$  within the time-deadline  $T = 20$ ) than they do when only one player (asymmetrical) is increasing the pace of concession.



**Fig. 5.** Expected Time-to-Agreement as a function of concession pace

**Agreement distribution as a function of players uncertainty:** here we discuss the effect of players' uncertainty (i.e. Acceptance Probability functions) on the outcome of negotiation. We performed a number of experiments for the verification of (8), this time comparing combinations of values for the parameters  $(\beta_b, \beta_s)$  of (4) and (5), while imposing a constant configuration for the NDFs. Figure 6(a) reports about the expected value of an agreement as a function of the  $(\beta_b, \beta_s)$  parameters. We recall that, in that respect, a *conservative* attitude corresponds to  $\beta_b > 1$ , for the buyer, and to  $\beta_s < 1$  for the seller. The curves in Figure 6(a) allows for comparing the effect of increasing the *conservativeness* of a player while the other one's is maintained constant (in this specific case, linear). As expected, we can conclude that, conservativeness (in probabilistic decision making) is desirable for a player, as it improves the expected payoff of a deal (decreasing it when the buyer becomes more conservative and increasing it when the seller becomes more conservative).

**Agreement delay as a function of players uncertainty:** similarly, here we discuss the effect that players' probabilistic decision making has on the delay to reaching an agreement. Figure 6(b) reports about the expected time-to-agreement as a function of the  $(\beta_b, \beta_s)$  parameters of (4) and (5). Again the indication of these results is confirms the intuitiveness, which is: a more conservative attitude results in a (slightly) longer expected delay. It should be pointed out that the small variability of curves in Figure 6(a) and Figure 6(b) is a direct



**Fig. 6.** Effect of Uncertainty on Expected Agreement and Expected Agreement Delay

consequence of the small variability of the acceptance probability functions (4) and (5) corresponding to extreme values of  $\beta$  (i.e.  $\beta_b \gg 1$  and  $\beta_s \ll 1$ ). Finally from Figure 6(b) one may conclude that the seller's conservativeness affects the delay for reaching an agreement slightly less than the buyer's does. In fact that difference is only due to the fact that results therein reported refer to experiments in which the buyer is the starting player (i.e. when the seller starts the negotiation results are symmetrical).

## 5 Conclusions

In this paper we have shown a different approach to the analysis of multi-agent system which can be used as an alternative to (or in conjunction with) analytical methods and/or simulation. We have illustrated how the analysis of specific negotiation (mixed) strategies can be performed by means of probabilistic model checking when players' decision is made through a probabilistic device. This is achieved by developing an *ad hoc* probabilistic model which is then verified against probabilistic properties with the PRISM model checker. This analysis has helped in comparing the effect that several strategic variables has on two relevant features of the negotiation process: the (expected) value and the (expected) time at which an agreement is reached. The results of our verification provided us with some insights about the behaviour of a system with uncertain players.

Specifically we have shown that, a slower concession pace is generally preferable for both players (unless in the case of symmetrical strategies for which the seller is slightly advantaged by faster concession) as well as a conservative attitude in (probabilistic) decision making. With respect to the delay for reaching an agreement our analysis proves that faster concession paces (probabilistically) speed up the negotiation whereas a conservative decision making (slightly) slows it down.

The model of Negotiation we have developed here is strongly related to the one introduced by Li *et al.* in [12]. In [12], the authors study an alternating offers “uncertain” dynamic framework, where several agents may compete with each other (by setting aggressive reservation prices) for selling a single item/service to a (single) buyer. The uncertainty in that framework refers to two aspects: the availability of competing sellers (i.e. the time of arrival of a seller) and the reservation price of a newly arrived seller. The authors then introduce specific heuristics which allows the buyer to dynamically adjust his strategy as a consequence of changes in the environment (arrival of new sellers) and use simulation to obtain estimations of the framework behaviour. We would like to point out the main difference between our approach to model uncertainty and the one in [12]. In [12], what is uncertain is the environment behaviour (competitors may enter the market with a given probability after each time unit, but players’ behaviour is not inherently probabilistic), whereas in our framework the uncertainty is represented into the players behaviour (by means of the probabilistic decision making mechanism). In a sense our model is more general as it abstracts away from the specific cause of uncertainty (a change in the environment such the arrival of a new player, for example), and encodes uncertainty directly within players’ strategies. Furthermore our approach differs from the one studied in [12] with respect to the verification technique: we use model checking, through which an exhaustive/exact verification of the model is achieved, as opposed to simulation, which is based on estimations derived from non-exhaustive verification of the system model.

Following [12], future developments of this work include the extension of our approach to studying the effect of multiple, uncertain, competing players on the outcome of negotiation (such a model is currently under development). Finally we would like to stress that, to the best of our knowledge, this work provides a novel contribution which shows how a well established and effective automated verification technique such as model-checking can be used for the analysis of game-like scenarios.

**Acknowledgement:** This work was supported by an EC Marie Curie Fellowship under contract number HPMF-CT-2001-00065.

## References

1. R. Alur and T. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.

2. Christel Baier, Boudewijn Haverkort, Holger Hermann, and Joost-Pieter Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Trans. on Software Eng.*, Vol. 29(6):pp. 524–541, June 2003.
3. Rafael H. Bordini, Michael Fisher, Willem Visser, and Michael Wooldridge. Model checking rational agents. *IEEE Intelligent Systems*, 19(5):46–52, September/October 2004.
4. Rafael H. Bordini, Michael Fisher, Willem Visser, and Michael Wooldridge. Verifying multi-agent programs by model checking. *Journal of Autonomous Agents and Multi-Agent Systems*, 12(2):239–256, March 2006.
5. Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 2000.
6. Jürgen Dix, Mirco Nanni, and V. S. Subrahmanian. Probabilistic agent programs. *ACM Trans. Comput. Logic*, 1(2):208–246, 2000.
7. P. Faratin, C. Sierra, and N. R. Jennings. Negotiation decision functions for autonomous agents. *Int. Journal of Robotics and Autonomous Systems*, 24(3-4):159–182, 1998.
8. W. Feller. *An introduction to probability theory and its applications*. John Wiley and Sons, 1968.
9. H. A. Hansson and B. Jonsson. A framework for reasoning about time and reliability. In *Proc. 10th IEEE Real -Time Systems Symposium*, pages 102–111, Santa Monica, Ca., 1989. IEEE Computer Society Press.
10. Gerard J. Holzmann. *The Spin Model Checker*. Addison Wesley, 2003.
11. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with prism: A hybrid approach. *International Journal on Software Tools for Technology Transfer (STTT)*, 2004.
12. C. Li, J. Giampapa, and K. Sycara. Bilateral negotiation decisions with uncertain dynamic outside options. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 36, Part C: Special Issue on Game-theoretic Analysis and Stochastic Simulation of Negotiation Agents(No. 1), 2006.
13. D. Parker. Prism web site. [www.cs.bham.ac.uk/~dyp/prism](http://www.cs.bham.ac.uk/~dyp/prism).
14. A. Pnueli. The temporal logic of programs. In *Proceedings of the Eighteenth IEEE Symposium on the Foundations of Computer Science*, pages 46–57, 1977.
15. M. Wooldridge. Agent-based software engineering. *IEE Proceedings on Software Engineering*, 144(1):26–37, 1997.

# MLBPR: MAS for Large-Scale Biometric Pattern Recognition

Ram Meshulam, Shulamit Reches, Aner Yarden, and Sarit Kraus

Bar-Ilan University, Ramat-Gan 52900, Israel,  
`meshulr1@cs.biu.ac.il`

**Abstract.** Security systems can observe and hear almost anyone everywhere. However, it is impossible to employ an adequate number of human experts to analyze the information explosion. In this paper, we present a multi-agent framework which works in large-scale scenarios and responds in real time. The input for the framework is biometric information acquired at a set of locations. The framework aims to point out individuals who act according to a suspicious pattern across these locations. The framework works in large-scale scenarios. We present two scenarios to demonstrate the usefulness of the framework. The goal in the first scenario is to point out individuals who visited a sequence of airports, using face recognition algorithms. The goal in the second scenario is to point out individuals who called a set of phones, using speaker recognition algorithms. Theoretical performance analysis and simulation results show a high overall accuracy of our system in real-time.

## 1 Introduction & Problem Description

In this paper we address the general problem of Large-scale Biometric Pattern Recognition (LBPR). LBPR problems include situations where there are streams of biometric information acquired at a set of locations, e.g. a set of cameras positioned in several airports. The goal of LBPR solvers is to point out individuals who act according to a predefined suspicious pattern in real-time. The combination of multiple locations, streams of input, inaccuracy of biometric tests and real-time constraints make LBPR problems hard to solve. Furthermore, we assume that suspects do not actively cooperate with the system (as opposed to fingerprint or iris-scanning tests which require the "suspect's" cooperation). In this paper we deal with the following two representative scenarios:

**The airport problem** - Assume a sequence of  $L$  airports denoted  $ap_1, ap_2, \dots, ap_L$ . At each airport there are cameras which take one picture of each passenger. The task is to identify and detain at airport  $ap_L$  every passenger who flew from  $ap_1$  to  $ap_2, \dots$  to  $ap_L$ .

**The tapping problem** - Assume a tapping system which listens to  $L$  phones. The task is to identify in real time anyone who called all  $L$  phones within a specified period of time (e.g. a week), in any possible order. The system should trigger an alarm as soon as the last call ends. Since we assume that the caller might call from several different phones, we can not use the caller's phone number



for identification. The LBPR problem lies at the junction of well known research areas [1–4] which are discussed below.

To solve our problem we use two independent comparison algorithms as a black box which we denote in this paper as  $C_1$  and  $C_2$  [5,6]. As input, each comparison algorithm receives two biometric items (a pair of pictures or a pair of call recordings). The comparison algorithm returns true if the items match, i.e., both items belong to the same person. Otherwise the algorithm returns false. Each comparison algorithm is associated with two measurements: *false acceptance rate* (FAR) which is the probability that the algorithm receives two items that do not belong to the same person but it returns true and *false reject rate* (FRR) which is the probability that the algorithm receives two items which belong to the same person but it returns false. There is a trade-off between FAR and FRR. We assume that  $C_1$  is "FAR oriented", meaning that minimizing FAR is more important. Similarly, we assume that  $C_2$  is "FRR oriented". Formally we assume that  $FAR(C_1) < FAR(C_2)$  and that  $FRR(C_2) < FRR(C_1)$ .

There are many computation difficulties in LBPR problems. First, the LBPR problem space is very large even when a relatively small number of biometric items is considered. For example, assume that in the airport problem we would like to capture every person who visited only two different airports ( $ap_1$  and  $ap_2$ ) and that pictures of 1000 people in each airport were taken. A brute force solver must compare  $1,000 \times 1,000 = 1,000,000$  pairs of pictures. Note that the number of 1000 images per airport is realistic because the cameras' locations ensure that only pictures of passengers who board a certain flight are taken.

Moreover, biometric comparison algorithms are not perfect. For example, current state-of-the-art face recognition algorithms have an error margin that is unacceptable in large-scale situations [2]. Suppose that all the pictures in the previous example are of different people. Even an algorithm with an FAR of 1% will, on average, mistakenly positively identify 10,000 of the 1,000,000 pairs of pictures. Therefore, a picture of a person taken at  $ap_2$  is compared 1,000 times, each time against a different picture acquired at  $ap_1$ . On average, 10 of those comparisons will be falsely identified as a match. This will cause all the passengers of the second airport to be (falsely) detained.

To overcome the above challenges and provide real-time solutions, we present the multi-agent solver that solves the LBPR problem. We call this solver MLBPR<sup>1</sup>. Each agent is responsible for data from one biometric information source (airport or phone).

The MLBPR solver enables control over the global FRR and FAR of the system. Moreover, using a theoretical analysis of the solver we are able to anticipate its expected accuracy. Thus the user may choose a desired FRR/FAR trade-off. For example, if the system is used as a fully autonomous system, without human experts to cross-examine any alarm, we would prefer a very low false alarm rate. On the other hand, if human experts examine each alarm before any action is

---

<sup>1</sup> We use the term 'agent' although the agents might be executed on the same computer, or even as threads belonging to the same program.

---

**Algorithm 1** MLBPR framework: local module

---

**Local-module**(Path  $p$ , item  $i$ )

1. Let  $r, q = \mathbf{Phase1}(p, i)$
2. If  $r = \text{'borderline'}$  Then return  $\mathbf{Phase2}(q, i)$
3. return  $r, q$

**Phase1**(Path  $p$ , item  $i$ )

1. Let  $p' = \mathbf{AddItemToPath}(p, i, C_1)$
2. If  $e(p') > T_1$  Then return 'prune',  $null$
3. If  $(|p'| < L)$  Then
  - 3.1. return 'continue',  $p'$
4. If  $(e(p') < T_1)$  Then return 'trigger alarm',  $p'$
5. return 'borderline',  $p'$

**Phase2**(Path  $p$ )

1. Let  $p'$  be an empty path
2. For each item  $i$  in  $I(p)$ 
  - 2.1.  $p' = \mathbf{AddItemToPath}(p', i, C_2)$
  - 2.2. If  $e(p') > T_2$  Then return 'prune'
3. return 'trigger alarm',  $p'$

**AddItemToPath**(Path  $p$ , item  $i$ , comparisonAlg  $\mathbf{C}$ )

1. Let  $p' = p$
  2. Add  $i$  to  $I(p')$
  3. For each item  $i'$  in  $I(p')$ 
    - 3.1. If  $\mathbf{C}(i, i')$  is false Then increment  $e(p')$
  4. return  $p'$
- 

made, we can afford a low false reject rate at the cost of a higher false alarm rate.

This paper is structured as follows. First we describe the MLBPR in detail and provide a theoretical performance analysis. Then we present the simulation results which are matched against the theoretical analysis. Next we discuss related work. A discussion and summary are presented at the end of the paper.

## 2 MLBPR Framework

The MLBPR uses multiple agents to incrementally build partial biometric sets of candidate suspects. The agents use two modules: the *local* module which is responsible for classifying and pruning current candidates, and the *communication* module which is responsible for communication and knowledge sharing between the agents. Each agent independently runs a communication module that invokes a local module at different points.

## 2.1 Local Module

An agent receives new information from two sources. The first source is his biometric information unit (e.g. a phone call recording). The second source of information is *paths* forwarded from other agents. Each agent maintains a database of suspected partial paths *PDB*.

A path is a data structure defined as  $p = \{I, e\}$ .  $I(p) = \{i_1, i_2, \dots, i_k\}$  is a set of  $k$  items (e.g. face images or call recordings).  $e(p)$  is the number of negative comparison results between any two items in  $I(p)$ . There are  $\binom{k}{2}$  comparison results in a path of length  $k$ . The length of a path, denoted by  $|p|$  is the number of items in  $I(p)$ . A path  $p$  and a new item  $i$  can be joined together to create a new path of length  $|p| + 1$ . The method *AddItemToPath* (Algorithm 1) receives a path  $p$  and an item  $i$  and returns a new path  $p'$  such that  $I(p') = I(p) \cup i$  and  $e(p') = e(p) + f$  where  $f$  is the number of false results of comparisons of  $i$  and items from  $I(p)$ .

Suppose we have a perfect comparison algorithm,  $C_P$ , in which  $FAR = FRR = 0$ . In this case, the system triggers an alarm only when it detects a path  $p$  of size  $L$  (one biometric item from each biometric stream) which has the maximum number of positive comparison results, i.e.,  $e(p) = 0$ . For example, if  $L = 3$ , the system should trigger an alarm only if  $C_P(1, 2) = C_P(1, 3) = C_P(2, 3) = true$ . In practice, however, comparison algorithms have non trivial FRRs and we must consider cases where all items are of the same person, but some of the comparisons are falsely rejected. For each comparison algorithm  $C_1, C_2$  we define a threshold  $T_1, T_2$  where  $0 \leq T_1, T_2 \leq \binom{L}{2}$ . The algorithm triggers an alarm when a path of size  $L$  is created and  $e(p) < T$ . In other words, the algorithm allows a path to have  $e(p) < T$  false comparisons and still treats it as a path with matching items which might lead to a target person.

Algorithm 1 describes the local module of the MLBPR framework. As an input the module receives a path  $p$  and a new biometric item  $i$ . It returns two values: (1) a new joint path  $q$  which contains all the items in  $p$  and  $i$  and (2) a classifier for  $q$  which has three possible values:

- ‘prune’ - There is no point in keeping the joint path (and the returned path is *null*).
- ‘continue’ - The joint path has a potential to become a target path.
- ‘trigger alarm’ - The joint path is a target path and the system must notify about it.

The module may return ‘continue’ only if the size of the joint path is smaller than  $L$ . It may return ‘trigger alarm’ only for full paths - paths of size  $L$ .

Comparing a pair of pictures is time consuming. Thus, in most cases the algorithm uses  $C_1$  alone to classify the new path (lines 3-5 in function **Phase1**). The only scenario in which  $C_2$  is used is a borderline scenario. A borderline scenario occurs when a full path ( $|q| = L$ ), has exactly  $T_1$  negative results. The algorithm then uses  $C_2$  (function **Phase2**) as follows. It calculates  $e(q)$  again but this time using  $C_2$ . The same classification rules are applied but this time the threshold  $T_2$ , associated with  $C_2$ , is used.

---

**Algorithm 2** Tapping problem - agent  $A_i$ 


---

**IncomingCall**(Call  $call$ )

- 1 Add  $call$  to calls database  $CDB$
- 2 For each path  $p$  in  $PDB$ 
  - 2.1 Let  $r, q = \text{Local-module}(p, c)$
  - 2.2 If  $r == \text{'trigger alarm'}$  Then trigger alarm
  - 2.3 If  $r == \text{'continue'}$  Then forward  $q$  to agent  $A_{i+1}$

**IncomingPath**(Path  $p$ )

- 1 Add  $p$  to  $PDB$
  - 2 For each call  $c$  in  $CDB$ 
    - 2.1  $r, q = \text{Local-module}(p, c)$
    - 2.2 If  $r == \text{'continue'}$  Then
      - 2.2.1 Forward  $q$  to agent  $A_{i+1}$
- 

$C_2$  has a higher FAR (and has a greater chance of accepting false candidates). But this is of less concern since it is used for borderline cases of  $C_1$ . At this point the number of paths has already been significantly reduced and only highly suspicious individuals have been selected by the earlier phase of  $C_1$ . Conversely, the low FRR of this algorithm makes it very effective in verifying the identity of true suspects. Note that phase 2 is significantly more time consuming than phase 1. In phase 1 each agent adds one item. The new item  $i$  is compared with items in  $p$ , which totals  $L - 1$  comparisons (worst case). In phase 2, the last agent matches all pairs of pictures using  $C_2$ , which totals  $\binom{L}{2}$  comparisons, all executed by one agent.

The pruning made by the local-module is lossless in the sense that a system which uses the module will have the same accuracy with and without the pruning mechanism. For example, on the first phase, the algorithm only prunes partial paths which have exceeded the number of errors defined by the threshold  $T_1$ . This fact allowed us to ignore the pruning mechanism when we analyzed the algorithm performance in terms of accuracy. However, pruning is crucial in terms of run-time.

Note that the same picture might appear in many paths. For example, consider two paths  $\langle a, b \rangle$  and  $\langle a, c \rangle$  forwarded to airport 3. Suppose a picture  $d$  was acquired at airport 3. The agent will be required to match the following pairs of pictures:  $\langle a, d \rangle$ ,  $\langle b, d \rangle$  for the first path and  $\langle a, d \rangle$ ,  $\langle c, d \rangle$  for the second path. Instead of activating the comparison algorithm twice on  $\langle a, d \rangle$ , the results can be cached. Thus the outcome of the above is that the main time-consuming action of the MLBPR algorithm is comparison-results *lookup* rather than actual *item-comparison*. This notion is important for the time performance analysis discussed later.

## 2.2 Communication Module

The communication module is implemented differently for the two scenarios as follows<sup>2</sup>:

**Tapping Problem:** Here we force a linear order on the agents. The first agent  $A_1$  acquires phone calls from phone 1 and sends them to  $A_2$ . Any other agent,  $A_k$ ,  $1 < k \leq L$  maintains a database of paths sent by  $A_{k-1}$  denoted  $PDB$ . Each path in  $PDB$  is of length  $k - 1$ . Each agent also maintains a call database denoted  $CDB$  which holds calls acquired from phone  $k$ . When a call  $c$  is acquired by agent  $k$ , each path in  $PDB$  is matched against the new call (see function **IncomingCall** in algorithm 2). When a new path is forwarded, it is matched against all calls in  $CDB$  (function **IncomingPath**). Paths that have sufficient positive comparison results are sent to  $A_{k+1}$  or trigger an alarm (if the path was created by the last agent).

**Airport Problem:** Compared to the tapping problem, the airport problem is easier to solve. While the tapping problem allows any order of calls, and any order of calls must be considered, the airport problem considers passengers who visited the airport in a sequential order  $ap_1 \rightarrow ap_2 \rightarrow \dots ap_L$ . A passenger who visited  $ap_k$  and then  $ap_{k-1}$  can not be a target person. Thus, since we respond in real-time, we do not match a new forwarded path (just sent from agent  $k - 1$ ) with passenger images acquired earlier (by agent  $k$ ). Instead, we store it in  $PDB$ .

## 3 Performance Analysis

The FAR and FRR of the comparison algorithms are known in advance. However, a user of such a system would be interested in the *global* FRR and FAR, denoted  $\overline{FRR}$  and  $\overline{FAR}$ , respectively. Referring to the airport problem, for example, one would like to know 1) What the chances are that a suspect who visited all airports will not be detained at the last airport ( $\overline{FRR}$ ); 2) What the chances are that a person who did not visit all the airports will be falsely detained ( $\overline{FAR}$ ).

The thresholds  $T_1$  and  $T_2$  affect the relationship between  $\overline{FRR}$  and  $\overline{FAR}$ : choosing high thresholds will increase the number of innocent people detained, and decrease the number of target people who get away. Choosing low thresholds will have the opposite effects.

The following paragraphs describe a formula which expresses the relationship between  $\overline{FRR}$ ,  $\overline{FAR}$ ,  $T_1$  and  $T_2$ . The formula allows users of the MLBP algorithm to determine  $\overline{FRR}$  and  $\overline{FAR}$  through  $T_1$  and  $T_2$  adjustments. In order to limit the errors of the system we would like to find the upper bound for the global false reject rate and for the global false accept rate. Given the FRR and FAR of both biometric comparison algorithms, the number of biometric sources  $L$ , the number of biometric items acquired from each biometric source  $N$  and the a priori probability of an individual to match  $m$  biometric sources, we will calculate an upper bound for  $\overline{FRR}$  and the  $\overline{FAR}$ .

<sup>2</sup> We start with the tapping problem since the communication module of the airport problem is a specific case of the communication module of the tapping problem.

### 3.1 Global FRR

In order to estimate the upper bound of  $\overline{FRR}$ , we find the lower bound of  $1 - \overline{FRR}$ , i.e., we calculate the probability that our system will trigger an alarm when an individual visits all airports/call all tapped phones. Assume there is such a target suspect  $s$ . What is the probability that  $s$  will trigger an alarm using only the first phase? In other words what is the probability that at least one path containing a biometric item of  $s$  in its last stage, will have at least  $\frac{L(L-1)}{2} - T_1$  positive results?

To simplify our formula, we assume all individuals except  $s$  do not “appear” in more than one location (e.g., in the tapping problem, all callers made only one phone call each). This assumption gives us a lower bound for  $1 - \overline{FRR}$  and therefore an upper bound for  $\overline{FRR}$ . We assume that  $FAR, FRR < 0.5$ . Thus the probability of having a path with at least  $\frac{L(L-1)}{2} - T_1$  positive results is larger when several individuals appear in more than one location, since there are more paths which contain matching items and the probability of a positive result when the items match ( $1 - \overline{FRR}$ ), is higher than the probability of a positive result when they don’t match ( $\overline{FAR}$ ).

First, we will calculate the probability of a specific path to reach the final stage i.e., to have at least  $\frac{L(L-1)}{2} - T_1$  positive results. Then, we will estimate the probability that at least one path will reach the final stage. Finally we will find the complimentary probability i.e., the probability that our system will trigger an alarm when a person is a target person - a person who has visited all airports or called all tapped phones.

**Step I** We will calculate the probability of a specific path to have exactly  $T_1$  negative results (or  $\frac{L(L-1)}{2} - T_1$  positive results).

The number of tests in a full path of length  $L$  is  $n(L) = \frac{L(L-1)}{2}$ . We denote  $A_u^{j,K}$  as the probability of a path, which contains  $L - j$  matching items i.e., items acquired from one person and  $j$  non-matching items, to have exactly  $K$  positive results in phase  $u$  ( $u$ ’s possible values are 1 or 2).

We denote  $FRR_u$  to be the  $FRR$  of the  $u$  pass and  $FAR_u$  to be the  $FAR$  of the  $u$ -th pass. The probability of a path in which all items belong to person  $s$  to have exactly  $K$  positive comparison results after the **first** pass is:

$$A_1^{0,K} = \binom{n(L)}{K} (1 - FRR_1)^K FRR_1^{n(L)-K}.$$

Similarly, consider a path of size  $L$  which contains  $L - 1$  matching items. This time there are  $L - 1$  comparisons of non-matching items. The  $K$  positive comparison-results are the sum of matching items accurately (true acceptance) and falsely positive results by comparing pairs of non-matching items (false acceptance). The probability of exactly  $K$  positive results using the first phase is the sum of the probabilities for each such scenario. One extreme scenario is that all  $L - 1$  non-matching comparisons will falsely return true and  $K - (L - 1)$  of the matching items comparisons will return true. The next scenario to consider is  $(L - 1) - 1$  of the non-matching items comparisons which return (falsely) true

and  $K - (L - 2)$  comparison results of matching items which return true and so on. Then consider the case of  $A_1^{1,K}$  is:

$$A_1^{1,K} = \sum_{i=0}^{n(L)-K} \binom{L-1}{i} FAR_1^{L-1-i} (1 - FAR_1)^i \cdot \binom{n(L) - (L-1)}{K - (L-1-i)} (1 - FRR_1)^{K-(L-1-i)} FRR_1^{n(L)-K-i} . \quad (1)$$

The formula sums up the probabilities of all the combinations of true positive results and false positive results which create a path with exactly  $K$  positive comparison results. The first half of the formula calculates the probabilities related to the comparison results of the non-matching items. The second half of the formula calculates the probabilities related to the comparison results of the matching items. For example,  $i = 0$  represents the first scenario mentioned in the previous paragraph: all  $L - 1$  non matching comparison results return falsely true and the rest of the true comparison results are associated with the truly matching comparisons results.

The formula of  $A_1^{2,K}$  is similar to the formula of  $A_1^{1,K}$ . This time however the number of non matching comparison-results is  $(L - 1) + (L - 2) = 2L - 3$ :

$$A_1^{2,K} = \sum_{i=0}^{n(L)-K} \binom{2L-3}{i} FAR_1^{2L-3-i} (1 - FAR_1)^i \cdot \binom{n(L) - (2L-3)}{K - (2L-3-i)} (1 - FRR_1)^{K-(2L-3-i)} FRR_1^{n(L)-K-i} . \quad (2)$$

In general a path with  $L - j$  matching items and  $j$  different non-matching items has  $S_j = \sum_{i=1}^j (L - i)$  comparisons of non matching items. Generalizing the above formulas, the probability that such a path will contain exactly  $K$  positive results (using only the first phase) is:

$$A_1^{j,K} = \sum_{i=0}^{n(L)-K} \binom{S_j}{i} FAR_1^{S_j-i} (1 - FAR_1)^i \cdot \binom{n(L) - S_j}{K - (S_j-i)} (1 - FRR_1)^{K-(S_j-i)} FRR_1^{n(L)-K-i}$$

where  $S_j = \begin{cases} 0 & j = 0 \\ \sum_{i=1}^j (L - i) & j \neq 0 \end{cases}$  .

**Step II**  $A_2^{j,k}$  represents the probability that a path which contains  $L - j$  matching items and  $j$  different non-matching items will have exactly  $K$  positive results using the **second phase**.  $A_2^{j,k}$  is calculated exactly like  $A_1^{j,k}$ , using the  $FRR_2$  and  $FAR_2$  of the second comparison algorithm.

A path must have at least  $n(L) - T_1 + 1$  positive comparisons in the first phase in order to trigger an alarm. Alternatively, a path may have exactly  $n(L) - T_1$

positive comparison results in the first phase and at least  $n(L) - T_2$  positive comparison results in the second phase. The probability for a path with  $j$  items different from  $s$ , to trigger the alarm on the first pass is :

$$\sum_{i=n(L)-T_1+1}^{n(L)} A_1^{j,i} ,$$

The probability that such a path will trigger the alarm as a border-line case is:

$$A_1^{j,F(L)-T_1} \cdot \sum_{i=n(L)-T_2}^{n(L)} A_2^{j,i} .$$

Thus, the probability for a path with  $L - j$  matching items and  $j$  non-matching items, to trigger an alarm is:

$$A_j = \sum_{i=n(L)-T_1+1}^{n(L)} A_1^{j,i} + A_1^{j,F(L)-T_1} \sum_{i=n(L)-T_2}^{n(L)} A_2^{j,i} .$$

We will calculate the probability that **at least one** path will trigger an alarm. Let  $Acc_i$  be the probability that at least one path with  $L - i$  matching items and  $i$  different non-matching items will trigger an alarm. Let  $F_i = \binom{L-1}{i}$ . Since there is only one path in which all items match,  $Acc_0 = A_0$ . As mentioned above,  $N$  is the number of biometric items acquired from each biometric source. There are  $(N-1)F_1$  different paths with only one non-matching item (a matching item has to appear in the last biometric information source) since the non-matching item can "appear" in every biometric source except the last one, and in that biometric source, every item can be a non-matching item. The probability that such a path will **not** trigger an alarm is  $(1 - A_1)$ .  $(1 - A_1)^{(N-1)F_1}$  is the probability that **none** of the paths will trigger an alarm. The probability for at least one of the above paths to trigger an alarm is:

$$Acc_1 = 1 - (1 - A_1)^{(N-1)F_1} .$$

Similarly we calculate  $Acc_2$ . There are  $(N-1)^2 F_2$  different paths with two different non-matching items. The probability that at least one of the paths will trigger an alarm is:

$$Acc_2 = 1 - (1 - A_2)^{(N-1)^2 F_2} .$$

In general, there are  $(N-1)^i F_i$  different paths with  $L - i$  matching items and  $i$  different non-matching items. The probability that at least one of these paths will trigger an alarm is:

$$Acc_i = 1 - (1 - A_i)^{(N-1)^i F_i} . \quad (3)$$

The probability that none of the paths of the above form will trigger an alarm is  $(1 - Acc_i)$ . The probability that none of the paths, of **any** form, will trigger



an alarm is the product of all the above formulas. Thus, the upper bound for  $\overline{FRR}$  is:

$$\prod_{i=0}^{L-1} (1 - Acc_i) \quad (4)$$

### 3.2 Global FAR

In this section we will focus on the calculation of  $\overline{FAR}$ , i.e., the probability that an individual  $s$  who is not a target will be detained because a path in which the last acquired item of  $s$  has falsely triggered the alarm. We consider a group of individuals which "appear" in exactly  $m < L$  locations, and calculate the probability that a specific person will falsely trigger the alarm. We then multiply this result by the a priori probability of an individual to match  $m$  biometric sources (e.g. call from  $m$  different phones).

In order to find an upper bound for  $\overline{FAR}$  we consider the path which gives us the highest probability of having maximum positive results among the paths in which every item is located in  $m$  information sources. This is the case in which the items are distributed throughout all the information sources they match, i.e., the items in the path are divided into  $\lfloor \frac{L}{m} \rfloor$  groups of matching items of size  $m$ . The number of comparisons between matching items in such a path is:  $c(m) = \binom{m}{2} \lfloor \frac{L}{m} \rfloor + \binom{L \bmod m}{2}$ . Let  $P_l^{m,K}$  be the probability that a path in which every item is located in  $m$  information sources will contain exactly  $K$  positive results in the  $l$ -th phase. The probability that such a path will contain exactly  $K$  positive results in the first pass, for  $m > 1$  is:

$$\begin{aligned} P_1^{m,K} &= \sum_{i=0}^{n(L)-K} \binom{c(m)}{i} (1 - FRR_1)^{c(m)-i} FRR_1^i \\ &\quad \cdot \binom{n(L) - c(m)}{K - (c(m) - i)} FAR_1^{K - (c(m) - i)} (1 - FAR_1)^{n(L) - (K + i)} \\ &\text{where } c(m) = \binom{m}{2} \lfloor \frac{L}{m} \rfloor + \binom{L \bmod m}{2} \\ &\text{For } m = 1, P_1^{1,K} = \binom{n(L)}{K} FAR_1^K (1 - FRR_1)^{n(L) - K} \end{aligned}$$

Similar to the calculation of  $A_1^{j,K}$ , we summarize the probabilities of  $K$  scenarios, where scenario  $i$  represents a case in which  $i$  out of  $c(m)$  matching items comparisons were falsely rejected. The first half of the formula calculates the probabilities of accepting  $c(m) - i$  matching-items comparisons and falsely rejecting  $i$ . The rest of the formula expresses the probability of raising the number of positive comparisons to  $K$  with falsely accepting comparisons of non-matching pairs of items.

The probability that a path in which the items are grouped into matching items of size  $m$  ( $m < L$ ) will trigger an alarm is:

$$P_m = \sum_{i=n(L)-T_1+1}^{n(L)} P_1^{m,i} + P_1^{m,n(L)-T_1} \cdot \sum_{i=n(L)-T_2}^{n(L)} P_2^{m,i}$$

The previous formula is an upper bound for  $\overline{FAR}$  of a path which has at most  $m$  matching items. However, each suspect on a full path is attached to many paths (e.g. the picture of a passenger taken in the last airport appears as the last item of many paths). A person is detained even if only one path has triggered an alarm. We must calculate how many paths are attached to each passenger and incorporate it in our global FAR formula. Let  $M$

$leq L \cdot N$  be the overall number of people and  $p_j$  the a priory probability that a person's biometric items appear in  $j$  biometric sources. Let  $v_K$  be the number of people who visit  $K$  airports ( $v_K = M \cdot p_k$ ). Let  $N_i$  be the maximal number of paths with  $i$  matching items. Initially we would like to calculate  $N_{L-1}$ , i.e., paths of size  $L$  in which there are  $L - 1$  matching items and the last picture belongs to a certain passenger  $s$ . There are  $v_L$  people who visited all airports. Each one of them contributes one path in which all items match and an item of  $s$  as the last item. There are  $v_{L-1}$  individuals who visited  $L - 1$  airports (or made  $L - 1$  calls to different phones). In the worst case, all of them visited airports  $ap_1, ap_2, \dots, ap_{L-1}$ . Thus, they contribute  $L - 1$  paths in which all items match and the last item is a picture of  $s$ . In the worst case,  $s$  is a passenger who visited  $L - 1$  airports and  $N - 1$  different paths will be created which contain his pictures (e.g., if  $L = 5$ ,  $N = 1000$  and a passenger  $s$  visited airports 2-5, there are 1000 paths which contain pictures of  $s$  from airports 2-5 and a picture from airport 1). Thus, the formula of  $N_{L-1}$  is:

$$N_{L-1} = \binom{L-1}{L-1} i_L + \binom{L-1}{L-1} v_{L-1} + (N-1)$$

The formula for  $N_{L-2}$  is:

$$\begin{aligned} N_{L-2} = & \binom{L-1}{L-2} v_L (N-1) + \binom{L-1}{L-2} v_{L-1} (N-1) \\ & + \binom{L-2}{L-2} v_{L-2} N + N \binom{L-2}{L-3} (N-1) \end{aligned}$$

$\binom{L-1}{L-2} v_L (N-1)$  is the number of paths which contain  $L - 2$  matching items out of  $v_L$ .  $\binom{L-1}{L-2} v_{L-1} (N-1)$  is the number of paths which contain  $L - 2$  matching items out of the  $v_{L-1}$ .  $\binom{L-2}{L-2} v_{L-2} N$  is the number of paths which contain  $L - 2$  matching items out of the  $v_{L-2}$ .  $N \binom{L-2}{L-3} (N-1)$  refers to the worst case in which

a passenger has visited  $L - 1$  biometric sources. Similarly we calculate  $N_{L-3}$ :

$$N_{L-3} = \binom{L-1}{L-3} v_L (N-1)^2 + \binom{L-1}{L-3} v_{L-1} (N-1)^2 + \binom{L-2}{L-3} v_{L-2} N (N-1) + v_{L-3} N^2 + N \binom{L-2}{L-4} (N-1)^2$$

The general formula for  $N_m$ ,  $m > 1$  is:

$$N_m = N^{L-m-1} \left( \binom{L-1}{m} \cdot v_L + \sum_{i=0}^{L-m-1} \binom{L-i-1}{m} \cdot v_{L-i-1} \right) + N(N-1)^{L-m-1} \binom{L-2}{m-1}$$

To calculate  $N_1$  we subtract  $N_2, N_3 \dots N_{L-1}$  from the total number of paths attached to a specific passenger in the last biometric source. Our goal is to find an upper bound. The probability of a path which contains  $m > 1$  matching items to have positive results is higher than the probability of a path in which every item is different. Thus, we set an upper bound of the number of paths in which  $m \neq 1$  and a lower bound of the number of paths in which  $m = 1$ :

$$N_1 = N^{L-1} - \sum_{m=2}^{m=L-1} N_m$$

Now we combine the number of different paths attached to each passenger with the probability of false acceptance.  $(1 - P_m)$  is the probability that a specific path will not trigger an alarm.  $(1 - P_m)^{N_m}$  is the probability that no path of that form will trigger an alarm. The upper bound of the probability that at least one path attached to a passenger who visited at most  $m$  airports, will trigger an alarm is:

$$Rej_m = 1 - (1 - P_m)^{N_m} \quad (5)$$

Since there is a dependency between the  $Rej(m)$ ,  $1 \leq m \leq L - 1$  the above formula provides an upper bound. The final formula for the upper bound of  $\overline{FAR}$  is:

$$1 - \prod_{m=1}^{L-1} (1 - Rej_m) \quad (6)$$

In section 4 we will compare results obtained by simulations and theoretical results obtained by applying the above formulas on the airport problem and on the tapping problem.

## 4 Experimental Results

In this section we address the following questions:

1. How does the problem's size i.e.: the number of people  $N$  and the number of information sources  $L$  affect the accuracy of the algorithm over different thresholds (this question is answered by using both simulation formulas presented in the theoretical analysis section)
2. How do these parameters affect the running time of the algorithm?
3. How close are the theoretical analysis predictions to the simulation results?

In order to answer these questions we implemented simulations of the airport problem and the tapping problem. For statistical significance we averaged results from 100 trials of each parameter studied. For each trial of the airport problem, we created a database which contains the flight plan for each passenger. We assumed a distribution function where most of the passengers only visit one or two airports of the possible  $L$  airports. We modeled this behavior as an exponential decay function with the fewest people visiting all  $L$  airports. We made similar preparations for the tapping problem simulation.

#### 4.1 The Airport Problem

L	N	$T_1, T_2$	Simulation		Analysis	
			$\overline{FRR}$	$\overline{FAR}$	$\overline{FRR}$	$\overline{FAR}$
4	500	1,0	0.1500	0.0001	0.1900	0.0010
4	500	1,1	0.0900	0.0002	0.1200	0.0001
4	500	1,2	0.0700	0.0011	0.1050	0.0004
4	500	2,0	0.0100	0.0682	0.0230	0.0710
5	500	2,3	0.0700	0.0022	0.0690	0.0004
5	500	2,4	0.0650	0.0055	0.0580	0.0010
6	500	0,1	0.8400	0.0000	0.8180	0.0000
6	500	1,2	0.4700	0.0000	0.4570	0.0000
6	500	2,3	0.1900	0.0000	0.1840	0.0000
4	1000	1,1	0.0967	0.0012	0.1180	0.0010
4	1000	1,2	0.0900	0.0037	0.0894	0.0019
5	1000	2,3	0.0588	0.0040	0.0490	0.0003
5	1000	2,4	0.0575	0.0108	0.0230	0.0010
6	1000	2,1	0.2500	0.0000	0.2150	0.0000
6	1000	2,2	0.2225	0.0000	0.1890	0.0000
4	1500	1,0	0.1500	0.0001	0.1940	0.0035
4	1500	1,1	0.0900	0.0002	0.1200	0.0040
4	1500	1,2	0.0700	0.0011	0.0940	0.0088

**Table 1.** Airport problem - Various scenarios and results.

We studied scenarios of 4, 5 and 6 airports. We assumed 500, 1000 and 1500 pictures of passengers at each airport. For each scenario, we experimented with increasing values of  $T_1$  and  $T_2$ . We stopped when the resulting  $\overline{FAR}$  was no longer reasonable. We used the performance rates of the state-of-the-art face

recognition algorithms [5] as comparison algorithms  $C_1$  and  $C_2$ .  $C_1$  had a 10% FRR and 1% FAR.  $C_2$  had a 4% FRR and 10% FAR.

Table 1 shows selected simulations and the theoretical bounds<sup>3</sup>. Each value represents an average of 100 simulations over one scenario using a certain set of parameters. The first and second columns show the number of airports and the number of pictures taken at each airport, respectively. The third column shows the number of false matches we allow a path to have and still consider it a target path. There are two numbers for the first and second phase of the algorithm i.e.,  $T_1$  and  $T_2$ . Columns 4-5 show the simulation results and columns 6-7 show the theoretical predictions for each scenario. Performance is measured in terms of  $\overline{FRR}$  and  $\overline{FAR}$ . Though the table shows only a portion of the simulations, the following paragraphs refer to all the data collected.

In most cases, the theoretical results quite accurately predicted the results obtained in the simulations. The differences between the simulation results and the analysis did not exceed 6% for both  $\overline{FAR}$  and  $\overline{FRR}$ . In many cases the difference was less than 1%. This implies that the user may use the theoretical analysis to choose a desired  $\overline{FRR}$  and  $\overline{FAR}$  by assigning the thresholds  $T_1$  and  $T_2$  without actually running any experiments. For example, given a set of 4 airports and 500 pictures taken at each airport, one may choose to assign  $T_1 = 1$ ,  $T_2 = 0$  (first line in table 1). The outcomes of such an assignment risk a 15% false rejection rate while avoiding the need to detain innocent passengers. Alternately, assigning  $T_1 = 2$ ,  $T_2 = 0$  (line 4) might be chosen to decrease the false rejection rate to 1% while increasing the false alarm rate to 6.8%. This means that about 34 passengers would require manual examination. Very encouraging results were obtained across all scenarios. In many cases the MLBPR has obtained a global  $\overline{FRR}$  of less than 10% with a global  $\overline{FAR}$  of less than 1%. For example, if  $L = 5$ ,  $N = 1000$  (5,000 passengers total) if we choose  $T_1 = 2$ ,  $T_2 = 3$  we get  $\overline{FRR} = 5.9\%$ ,  $\overline{FAR} = 0.4\%$ . Similarly, if  $L = 5$ ,  $N = 1500$  (7,500 passengers total) if we choose  $T_1 = 1$ ,  $T_2 = 2$  we attain  $\overline{FRR} = 7\%$ ,  $\overline{FAR} = 0.1\%$ .

## 4.2 The Tapping Problem

We studied scenarios of 4 and 5 phone lines. We assumed 100 or 200 calls from each line which is a reasonable estimation for the number of calls made over several days. We used state-of-the-art relevant speaker recognition algorithms (two speaker conversations with limited data) as  $C_1$  and  $C_2$  [6].  $C_1$  had a 20% FRR and 5% FAR.  $C_2$  had a 15% FRR and 10% FAR.

The number of phone calls of the tapping problem is smaller than the number of pictures taken in the airport problem. However, the comparison-algorithms are less accurate. Thus, reaching a reasonable global  $\overline{FRR}$  and  $\overline{FAR}$  is harder. Table 2 shows selected simulations and the theoretical bounds. Each value represents an average of 100 simulations over one scenario using a certain set of parameters. The table structure is similar to table 1.

<sup>3</sup> We decided to avoid a graphical representation such as the ROC curve since a continuous line representation has no meaning when the number of possible results is small (in this case the number of assignments for the  $T_1$  and  $T_2$  thresholds.).

L	N	$T_1, T_2$	Simulation		Analysis	
			$\overline{FRR}$	$\overline{FAR}$	$\overline{FRR}$	$\overline{FAR}$
4	100	1,2	0.2900	0.0206	0.2510	0.0199
4	100	1,3	0.2400	0.0473	0.1110	0.0512
4	100	1,4	0.1900	0.1693	0.0680	0.1990
5	100	2,4	0.3800	0.0070	0.0800	0.0140
5	100	3,3	0.1300	0.1373	0.0001	0.1950
6	100	4,3	0.2100	0.0045	0.0186	0.0222
6	100	4,2	0.2100	0.0038	0.0380	0.0152
6	100	4,3	0.2100	0.0045	0.1860	0.0222
4	200	1,1	0.3400	0.1137	0.2750	0.1160
4	200	1,2	0.2800	0.1302	0.1600	0.1330
5	200	2,4	0.2600	0.0347	0.0210	0.0410
5	200	2,5	0.2400	0.0401	0.0130	0.0460
6	200	4,3	0.1400	0.0430	0.0060	0.0570
6	200	4,2	0.1500	0.0380	0.0077	0.0480

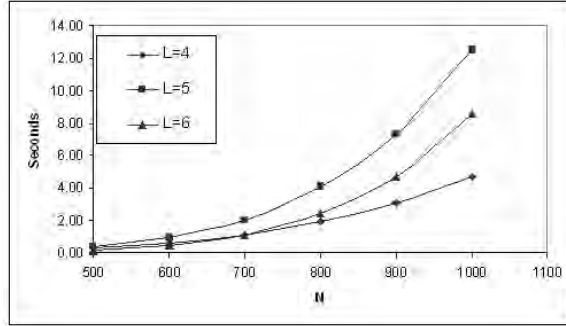
**Table 2.** Tapping problem - Various scenarios and results.

Here too, the theoretical analysis yielded successful predictions of the  $\overline{FAR}$ . However, the  $\overline{FRR}$  predictions of the theoretical analysis at times did not match the results obtained by the simulations. The  $\overline{FRR}$  measures the relative number of rejections of a total number of target persons. For the tapping problem, we assumed that the number of people calling all the phones was only one or two. Thus, the number of possible values for each simulation is extremely low. For example, if the scenario includes one target person, the possible values of the  $\overline{FRR}$  are either 0% or 100%. The theoretical analysis assumes a valid statistical domain and fails to predict such a scenario. In spite of the high inaccuracy of the voice-recognition algorithms, the algorithm produced reasonable rates in various scenarios. For example, if  $L = 5$ ,  $N = 100$ , and we choose  $T_1 = T_2 = 3$  we attain  $\overline{FRR} = 13\%$ ,  $\overline{FAR} = 13.7\%$  which means that about 14 callers will need further examination during the whole tapping session. Note that a relatively high percentage of  $\overline{FAR}$  is still acceptable since the tapping problem deals with a relatively small number of items (100 and 200 in our simulations).

### 4.3 Time Performance

We also measured the time performance of the MLBPR algorithm. As mentioned above, the comparison results are cached. Thus, we did not execute a comparison algorithm more than once on a pair of items. The time performance is dominated by the number of lookup matches. Using a comparison-algorithm with a high FAR in phase 1 of the local-module will generate many paths. Obviously, a high number of items  $N$  also increases the number of paths. A high number of biometric sources  $L$  mainly influences the last agent. The last agent is the only agent who uses the costly second pruning phase. This agent has the most time intensive decision to make, namely deciding in real time which passenger to

single out. The algorithm uses  $\binom{L}{2}$  tests (in the worst case) in the second phase. Therefore the longer the path, the more time needed by the last agent of each path.



**Fig. 1.** Time of last agent per  $N, L = 5$

All tapping simulations ended in a matter of seconds, so we focused our time analysis on the airport problem. Figure 1 shows the time performance of the last agent for 4, 5 and 6 airports over an increasing number of pictures taken at each airport. When  $L = 4$ ,  $T_1 = 2$ , when  $L = 5$ ,  $T_1 = 2$  and when  $L = 6$ ,  $T_1 = 1$ . Note that even when  $N = 1000$  and  $L = 5$ , the final agent was able to render a judgment under 12.5 seconds using a Pentium IV 3.0 Ghz computer with 1 GB of memory.

Two interesting phenomena are depicted in figure 1: (1) The five-airports simulation is slower than the six-airports simulation. (2) Moreover, when there are less than 700 pictures the four-airports is slower than the six-airports simulation. Both phenomena can be explained considering the previous paragraph. As mentioned above,  $T_1$  has a large influence on time performance. Only one error was allowed for the six-airports simulation while two errors were allowed in the five (and four) airports simulations. Thus, more paths were generated and forwarded in the five-airports simulation. The reason the six-airports simulation became slower than the four-airports simulation as the number of pictures increased is the increasing number of paths which reached phase 2. Phase 2, which is executed on borderline paths of size  $L$ , involves  $L(L - 1)/2$  picture matches which are 6 matches for the four-airports simulation and 15 matches for the six-airports simulation. As the  $N$  grew, the number of borderline cases also grew, which slowed down the six-airports simulation.

## 5 Related Work

LBPR problems lie at the junction of well known research areas: multi-modal biometric identification [1, 2, 7], data mining [3, 4], bioinformatics [8] and activity

recognition [9–11]. In the following paragraphs we will present a brief overview of these subjects and explain the LBPR uniqueness. We use the airport problem as a representative problem of the domain.

The solutions to all LBPR problems mentioned above must involve a biometric system [1, 2, 7]. One sub-domain of biometrics is biometric *identification* [12]. A biometric identification system tries to identify an individual by matching his/her biometric feature set to all feature sets in its database. Three main differences make it impossible to use identification biometric algorithms to solve LBPR problems. First, there is no template database to which the acquired picture can be compared. The person who traveled through  $L$  airports in a sequence was not known to the system a priori. His recent actions made him a target. Second, there is not a single test which can classify a person as a target. Even if the biometric tests were perfect, only a series of tests with positive results makes a person a target. The last difference is the time issue. The results must be calculated in real-time, before the suspect leaves the last airport.

The local-module uses two phases of processing to classify paths. *Multi-modal* biometric systems use multiple biometric modalities [1, 2, 13, 7]. Using more than one type of evidence or algorithm can improve robustness to noise, and increase security and accuracy. The tests are either merged together or used in an ordered way as a main and a secondary classifier. For example, in [13] iris and face biometric data are jointly used to achieve a higher accuracy of identity recognition. Hong and Jain developed multi-modal systems which combine face and fingerprint information [14]. Face recognition was first used to create the best  $n$  possible matches. They chose to use this method first due to its speed. Then, fingerprint information was used to identify the person from the  $n$  possible matches. To solve the LBPR problem we use two comparison algorithms to compare some of the pictures. In this sense our proposed system can be categorized as a multi-modal system. Note however, that multi-modal systems use multiple tests to increase accuracy over one query. To the best of our knowledge, none of the multi-modal algorithms aim to find a *set* of matching items over multiple sources.

One of the main targets of data mining research is pattern discovery. Pattern discovery algorithms try to discover interesting patterns in a given database. Although LBPR problems also focus on finding a pattern in a large amount of data, they cannot be classified as traditional pattern discovery problems. The imperfect tests (the biometric comparison algorithms) create noise in the system which does not allow us to use regular data-mining algorithms.

A melding of molecular biology with data-mining has created the field of bioinformatics. One of the main problems of the field is homology-search [8]. Two proteins are homologous if they have related folds and related sequences. Popular homology search algorithms receive a new found protein as input, and search in protein databases for matches [15, 16]. Homology search shares several attributes with MLBPR. Both problems search for target patterns in a large database. Both problems also have to positively classify sequences which are not a perfect match to the target pattern. However, while bioinformatics problems



use static databases, the MLBPR database rapidly changes. Popular sequence matching algorithms in bioinformatics [15, 16] assume cached databases in which extensive preprocessing has been done.

Another related domain is *Plan recognition* [17, 18]. Plan recognition is the process of inferring another’s plans, based on observations of his interaction with the environment. Usually, the agent is provided with a plan-library which holds a number of possible goals, sub-goals and the actions which might be executed for each goal. The agent then tries to match it’s observations on the other agent (the other agent might be a human user) to the plan library in order to infer the other agent’s plans. Both LBPR and plan-recognition domains seek a known pattern (a plan) from opponent(s) actions. Time performance is important in both domains [18]. However, LBPR concentrates on finding one rare pattern. Its target plan library holds only one plan. Moreover, the problem involves a relatively large number of independent agents.

Recently there has been an increasing interest in the field of *activity recognition* [9–11]. The input of activity recognition problems is usually one or more video streams from video cameras or a GPS device. The goal is to identify and learn patterns of behavior of the object under surveillance. The challenges of activity recognition research are related to the detection, representation and recognition of motions. LBPR problems, however, do not involve any motion detection or fusion of video streams. Instead, it copes with finding a target pattern across multiple independent sources with independent inputs, e.g. cameras located in *different* airports as opposed to a set of video cameras which perceive different angles of the same location.

## 6 Conclusions

In this paper we have addressed the domain of large-scale biometric pattern-recognition. The goal in LBPR problems is to single out, in real-time, any individual who acts in according to a suspicious pattern. The combination of multiple locations and streams of input, inaccuracy of biometric tests and real-time constraints make LBPR problems hard to solve.

We have presented an innovative approach - the MLBPR - a multi-agent solver for the LBPR problem. The MLBPR uses multiple agents to incrementally build solution candidates. Each agent consists of a local-module and a communication-module. The local-module prunes candidates which do not reach a certain threshold. Borderline cases go through a secondary process of classification. Theoretical analysis of the local-module provides upper bounds of the algorithm’s global performance. Simulation results of the problems show a high accuracy of the solver while the number of innocent people detained is kept low. All results were obtained in real-time.

We would like to extend our research in three directions. First, we would like to solve new LBPR problems using the MLBPR framework. Second, we would like to generalize the algorithm to exploit more than two comparison algorithms. We are interested in substantiating the generalization with theoretical

analysis. Intuitively, increasing the number of comparisons will improve overall performance. A third direction for future work is to include time performance in current theoretical analysis.

## References

1. Ross, A., Jain, A.K.: Multimodal biometrics: An overview. In: Proc. of 12th European Signal Processing Conf. (2004) 1221–1224
2. Jain, A.K., Ross, A., Prabhakar, S.: Introduction to biometric recognition. *Transactions on Circuits and Systems for Video Technology* **14**(1) (2004)
3. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., eds.: *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, Menlo Park, CA (1996)
4. Hand, D., Mannila, H., Smyth, P.: *Principles of Data Mining*. MIT Press (2001)
5. Phillips, P.J., Grother, P., Micheals, R.J., Blackburn, D.M., Tabassi, E., Bone, J.M.: FRVT 2002: Overview and summary. In: Proc. of Face Recognition Vendor Test 2002, Virginia (2002)
6. Przybocski, M., Martin, A.: The NIST speaker recognition evaluation series (2002)
7. Chang, K.I., Bowyer, K.W., Flynn, P.J.: An evaluation of multimodal 2d+3d face biometrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(4) (Apr. 2005) 619–624
8. Baxevanis, A., Ouellette, B.: *Bioinformatics. A Practical Guide to the Analysis of Genes and Proteins*. Wiley-Interscience, New York, NY, USA (1998)
9. Liao, L., Fox, D., Kautz, H.A.: Location-based activity recognition using relational markov networks. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. (2005) 773–778
10. Wu, G., Wu, Y., Jiao, L., Wang, Y.F., Chang, E.Y.: Multi-camera spatio-temporal fusion and biased sequence-data learning for security surveillance. In: *Proceedings of the 11th ACM International Conference on Multimedia (MM-03)*, Berkley, California, ACM Press (November 2003) 528–538
11. Ashbrook, D., Starner, T.: Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing* **7** (2003) 275–286
12. Wayman, J.L.: Fundamentals of biometric authentication technologies. *INT J. of Imaging and Graphics* **1**(1) (January 2001) 93–97
13. Wang, Y., Tan, T., Jain, A.K.: Combining face and iris biometrics for identity verification. *Audio and Video-based Biometric Person Authentication* (2003) 805–813
14. Hong, L., Jain, A.K.: Integrating faces and fingerprints for personal identification. *IEEE transactions PAMI* **20**(12) (1998) 1295–1307
15. Lipman, D.J. & Pearson, W.: Rapid & sensitive protein similarity searches. *Science* **227** (1985) 1435–1441
16. Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25** (1997) 3389–3402
17. Carberry, S.: Techniques for plan recognition. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research* **11**(1-2) (2001) 31–48
18. Avrahami-Zilberbrand, D., Kaminka, G.A.: Fast and complete symbolic plan recognition. (2005) 653–658

# Safe Stochastic Planning: Planning to Avoid Fatal States

Hao Ren<sup>1</sup>, Ali Akhavan Bitaghsir<sup>2</sup>, and Mike Barley<sup>1</sup>

<sup>1</sup> University of Auckland, New Zealand  
haoren@cs.ubc.ca, barley@cs.auckland.ac.nz

<sup>2</sup> University of Toronto, Canada  
akhavan@cs.toronto.edu

**Abstract.** Markov decision processes (MDPs) are applied as a standard model in Artificial Intelligence planning. MDPs are used to construct optimal or near optimal policies or plans. One area that is often missing from discussions of planning under stochastic environment is how MDPs handle safety constraints expressed as probability of reaching threat states. We introduce a method for finding a value optimal policy satisfying the safety constraint, and report on the validity and effectiveness of our method through a set of experiments.

## 1 Introduction

*The First Law of Robotics* [1]

A robot may not injure a human being, or, through inaction, allow a human being to come to harm.

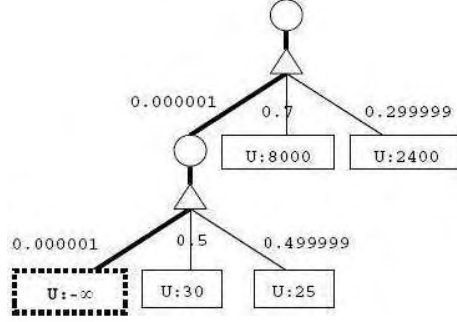
—Isaac Asimov 1942

### 1.1 Motive

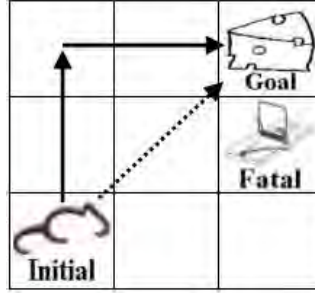
Autonomous agents need to behave safely in the real world. Weld and Etzioni [2] dealt with this issue for agents in a deterministic world with “dont-disturb” conditional constraints. Under real world situations, an agent would have only partial information and there would be exogenous events to be dealt with which are not addressed in their paper.

By applying utilities to address the safety issue and to avoid fatal states, one could use negatively infinite rewards (or *punishments*). This however would mean, that right from the initial state, the agent cannot perform any actions which have even a minute probability of reaching the infinite punishment as shown in Figure 1. In fact, inaction can be just as dangerous, since an agent cannot prevent a human from harming herself for instance. As an example, the state drawn in dotted line of Figure 1 is an unsafe state with infinitely negative reward, and we want to avoid it at all costs. Therefore we want the probability of going to the unsafe state under a certain *Risk Tolerance Threshold*. With this explicit measurement technique, the concern of safety can be tackled, and

actions can still be undertaken from the initial state. With a high risk tolerance, an agent may choose the path indicated by dotted line in Figure 2, where a low risk tolerance may lead to the path drawn in solid line.



**Fig. 1.** The problem is demonstrated in the bolded route leading to infinite punishment



**Fig. 2.** Resulting policy with Risk Tolerance Threshold

The safety issue is critical because the possible tasks that agents are involved in go beyond performing uncritical tasks such as information retrieval [3]. Vital and serious situations such as drug prescription and planning for the control of space vehicles will require reliable solution policies.

## 1.2 Notion of Risk

When generating a policy, without the use of an explicitly specified Risk Tolerance Threshold, one needs to specify both value and risk preferences in the utility function. This is what one would do without thinking about risk, see Figure 1 where the negative infinite utility portrays an extreme case of blending

risk preferences into utility function. However, it is easier to capture user “preferences” by dividing them into acceptable risk tolerance level to enter into fatal states and value preferences (reward functions for goal states).

The separation of risk from value simplifies the utility function and provide a direct description for the ceiling acceptance level of reaching the states that we try to avoid. Two examples are provided here. Firstly, in the United States, a commercial aircraft needs to satisfy federal safety constraints, e.g. fatal accident rate needs to be below 0.022 per 100,000 departures. This kind of benchmark provides a clear guideline for commercial airliner companies to follow, and so the companies would devote a tremendous amount of resources to prove that the restriction is satisfied. Without such an explicit restriction, the businesses would not be concerned to spend millions of dollars to confirm with the safety constraint, and may have their own interpretation of safety. Secondly, when a client invests into a portfolio, the portfolio manager would request the client’s preferences such as maximum tolerable *risk* of losing the investment and minimum expected *return*. According to the preferences, funding could be allocated to different ventures to generate returns as requested by the client.

As the above two examples have shown, the separation of risk tolerance threshold (RTT) from utility function provides a genuine measure of user’s aversion towards certain fatal states.

## 2 Background

The general planning problems that we are considering involve an environment that can be modeled by a stochastic process. Readers who are interested in further readings about planning based on MDPs can use [4], [5], and [6] as excellent introductory references on the topic. The system’s current state and the decision maker or agent’s choice of action together determine the system’s possible next states. The agent prefers to be in certain states (i.e. goal states) over others, and therefore needs a course of actions called a “plan” or “policy” that will lead the agent to those target states, while attempting to avoid undesirable threat states (i.e. fatal states) along the way.

### 2.1 Planning based on Markov Decision Processes

The safe planner is based on Markov Decision Processes (MDPs) to model a stochastic environment. The model would consist of:

- a set of *states*  $S$ ,
- a set of *actions*  $A$ ,
- *cost function*  $c : S \times A \times S' \rightarrow \mathbb{R}$ ,
- a set of *initial states*  $S_{init} \in S$ ,
- a set of *goal states*  $\Gamma \in S$ ,
- *reward function*  $R : \Gamma \rightarrow \mathbb{R}$ ,
- a set of *fatal states*  $\Phi \in S$ , and  $\Gamma \cap \Phi = \emptyset$ , with Punishment  $\varphi$  for all  $s \in \Phi$ ,

- *state transition function* or  $T(s, a, s')$  is the probability of making a transition from state  $s$  to state  $s'$  using action  $a$ .

We assume Fully Observable Markov Decision Processes (FOMDPs) with finite states and finite actions for a finite time horizon.

The *goal* and *fatal* states are *absorbing* states, and so they are terminal states during policy execution. Each goal state can have its own reward, and it is mapped by the reward function  $R(s)$ . Fatal states, which are not used conventionally in the planning literature, are defined as disastrous and destructive states that an agent considers dangerous and tries to avoid. We assume that fatal states are absorbing, as the set of actions  $A$  cannot be applied to  $s \in \Phi$ . All fatal states are given one single punishment (i.e. negative reward), and this enables the safe planner to modify one punishment and regenerate a policy rather than changing a set of punishments. If there is a distinct punishment for every  $s \in \Phi$  during a policy update, the solution space will be multi-dimensional and will be computationally expensive to calculate.

## 2.2 Dynamic Programming

We are given a fully-observable MDP or full knowledge of the tuple defining the problem:  $\langle S, A, S_{init}, \Gamma, \Phi, T, R, c \rangle$ . Suppose we are given a policy  $\pi$ , Bellman [7] shows that the expected value of such a policy at any state,  $V^\pi(s)$ , can be computed by:

$$V^\pi(s) = R(s) - \Phi(s) + \gamma \sum_{s' \in S} \{T(s, a, s') \times [c(s, a, s') + V^\pi(s')]\} \quad (1)$$

where  $\Phi(s) = \varphi$  if  $s \in \Phi$ , and  $\Phi(s) = 0$  otherwise. A policy is optimal if  $V^\pi(s) \geq V^{\pi'}(s)$  for all policies  $\pi'$  and all  $s \in S$ . Bellman's principle of optimality [7] forms the basis of the stochastic dynamic programming algorithms used to solve MDPs, establishing the following relationship between the current optimal value function and the optimal value function at the previous time step.

$$V_t^*(s) = R(s) - \Phi(s) + \gamma \max_{a \in A} \left\{ \sum_{s' \in S} T(s, a, s') \times [c(s, a, s') + V_{t-1}^*(s')] \right\} \quad (2)$$

## 2.3 Value Iteration

Equation 2 forms the basis of the value iteration algorithm for finite-horizon problems. Value iteration (VI) begins with the value function  $V_0^* = \varphi$  for each  $s \in S$ , so that the initial values of the states are minimum values possible. This would ensure that the agent would choose an action  $a \in A$  rather than selecting inaction. VI then uses equation 2 to iteratively refine the value for each state, by selecting an action that maximizes reward. At each step  $t$ , the value of the

expected reward  $V_t(s)$  is computed for each state from the value  $V_{t-1}(s)$  that was computed at the previous step. The algorithm greedily finds an action  $a$  such that  $V_t(s)$  is maximal, and stores it in the policy. The algorithm stops by a maximum number of iterations needed to guarantee that Value-Iteration returns an optimal policy. Another stopping criterion could be:

$$\max_{s \in S} |V_t(s) - V_{t-1}(s)| \leq \epsilon \quad (3)$$

When doing value iteration, a Q-value [8] function notation is normally used. We define  $Q(s, a)$ , the expected cost in a state  $s$  when an agent execute action  $a$ :

$$Q(s, a) = \Phi(s) + R(s) + \gamma \sum_{s' \in S} T(s, a, s') \times [c(s, a, s') + V_{t-1}^*(s')] \quad (4)$$

Then  $V_t^*(s)$  can be obtained by equation:

$$V_t^*(s) = \max_{a \in A} Q(s, a), \quad \forall s \in S \quad (5)$$

## 2.4 Other Related Research

Neuneier and Mihatsch [9] have built the Q-Function to be risk sensitive. Research about goal reaching probability above a certain threshold is complementary to our fatal state reaching probability below the Risk Tolerance Threshold (RTT). The C-BURIDAN planner from Draper *et al* [10] [11] is based on the Bayesian framework, and the planner produces a plan that makes the goal expression true above a certain threshold. Fulkerson *et al* [12] considered probabilistic planning and described safety as the probability of reaching a goal. Weld and Ezioni [2] produced a pioneering work about making autonomous agents safe. Yet, Weld and Ezioni [2] have assumed the world to be static, and did not consider a stochastic environment with actions having probabilistic outcomes.

## 2.5 Directly Related Research

Peter Geibel [13] had the same definitions of risk and safety as defined in this paper, and used a reinforcement learning algorithm on the Markov Decision Process (MDP) to generate policies.

**Risk and Safety** The main contribution of Geibel's work [13] is to provide the first available framework to consider undesirable states (i.e. fatal states) based on MDPs. He defined risk and safety in terms of the probability of reaching fatal states and offered a solution to find a safe policy above a certain safety tolerance threshold, while maximizing value of states from the resulting policy.

**Definition 1.** *Given a policy  $\pi$ , **risk** is defined as the probability  $\rho^\pi(s_{init})$  that the agent will end up in a fatal state  $s \in \Phi$  by starting at an initial state  $s_{init}$ , and so  $\rho^\pi(s_{init})$  is the risk of  $s_{init}$  according to  $\pi$  [13].*

If  $s \in \Phi$ , then  $\rho^\pi(s) = 1$ ; else if  $s \in \Gamma$ , then  $\rho^\pi(s) = 0$  because  $\Phi \cap \Gamma = \emptyset$ . So for states  $s \notin \Phi \cup \Gamma$ , the risk depends on the choice of actions. We have,

$$\sigma^\pi(s, a) = \sum_{s'} T(s, a, s') \rho^\pi(s') \quad (6)$$

$\sigma^\pi(s, a)$  represents the risk that an agent terminates in a fatal state, when it begins in state  $s_{init}$ , first executes action  $a$  and then follows policy  $\pi$  [13].

**Risk Evaluation** In essence, the evaluation algorithm uses VI, but we are given a policy and so at every state, the agent takes an instructed action given by the policy and evaluate the risk of taking that action:  $a \leftarrow \pi(s)$ ;  $\rho'(s) \leftarrow \sum_{s' \in S} [T(s, a, s') \times \rho(s')]$ , rather than calculating the  $\sigma$ -value for all actions with a state and obtaining a greedy action that minimizes  $\rho^\pi(s)$  as done for risk minimizing policy generation:  $\sigma(s, a) \leftarrow \sum_{s' \in S} [T(s, a, s') \times \rho(s')]$ ;  $\rho'(s) \leftarrow \min_{a \in A} \sigma(s, a)$ ;  $\pi(s) \leftarrow \arg \min_{a \in A} \sigma(s, a)$ .

**Learning for Value Optimal  $\omega$ -Safe Policy** Geibel has suggested an algorithm to acquire a value optimal  $\omega$ -safe policy, where  $\omega$  specifies the amount of risk that the agent is willing to accept when executing policy  $\pi$ , and called it  $\pi_\omega^*$ . The agent will try to maximize  $Q(s, a)$ , and minimize  $\sigma(s, a)$  with a *greedy* policy. The greedy policy is defined as follows: if the agent is in a state  $s$ , then it selects a greedy action  $a^*$  by

$$a^* \in \arg \max_a (\lambda Q(s, a) - \sigma(s, a)) \quad (7)$$

$0 \leq \lambda \leq 1$  is used to control the influence of the  $Q$ -values compared to the  $\sigma$ -values.

### 3 Framework

One major contribution of our work is that we considered a different set of problems from Geibel, so we had different definitions for the features of the framework such as risk. We have identified several classes of problems relating to different formations of *initial states*  $S_{init}$  and *fatal states* (Section 3.2) with associated punishment values. Our framework could be used to solve a more general class of problems, e.g. each of the fatal states having a unique punishment value. Also, we have identified the generation of value optimal safe policy  $\pi$  as an interpolation problem, and the most efficient interpolation algorithm should be used for the task.

#### 3.1 Problem Introduction

This section will define some terminology that is defined for the Safe Stochastic planning problem, and the underlying principles of the framework.



## Terminology

**Definition 2.** *Risk Tolerance Threshold (RTT) is the maximum acceptable risk given by the system user. We use  $\omega$  to represent RTT.*

RTT is the concept that identifies the importance of *fatal states*. It will be used to identify whether a policy  $\pi$  is safe or not.

Another important concept to consider in a framework with fatal states is *risk*. Our risk definition differs from Definition 1 provided by Geibel:

**Definition 3.** *The risk  $\rho^\pi(S_{init})$  of a policy  $\pi$  is the weighted average risk of a set of initial states  $S_{init}$ .*

Definition 1 considers risk as the risk of a single state, but our definition of risk considers the risk of a policy given the MDP with a set of initial states. A policy  $\pi$  is said to be *safe* if the risk of that policy is below the RTT specified by the system user, or  $\rho^\pi(S_{init}) < \omega$ .

**Framework Propositions** To generate value optimal safe policies, we have identified several properties about the MDP framework:

**Proposition 1.** *Let  $\pi_i$  be the optimal policy of an MDP with fatal states' punishment equal to  $\varphi_i$ . If  $\varphi_1 < \varphi_2$ , then:*

$$\forall s \notin (\Gamma \cup \Phi) : V(s)_{\pi_1}^{\varphi_1} \geq V(s)_{\pi_2}^{\varphi_2} \quad (8)$$

where  $V(s)_{\pi_i}^{\varphi_j}$  is the value of state  $s$  given policy  $\pi_i$  and punishment  $\varphi_j$  for the fatal states.

*Proof.* For all  $s$  except goal and fatal states, if we increase the value of punishment, with the same setting for all other parameters for an MDP including policy, the value of that state is bound to decrease.

$$\forall s \notin (\Gamma \cup \Phi) : V(s)_{\pi_2}^{\varphi_1} \geq V(s)_{\pi_2}^{\varphi_2} \quad (9)$$

Now assume that after increasing the fatal states' punishments from  $\varphi_1$  to  $\varphi_2$ , the optimal policy's value for at least one state ( $s'$ ) increases:

$$V(s')_{\pi_1}^{\varphi_1} < V(s')_{\pi_2}^{\varphi_2} \quad (10)$$

Given equations 9 and 10, for the state  $s'$  we'll have:

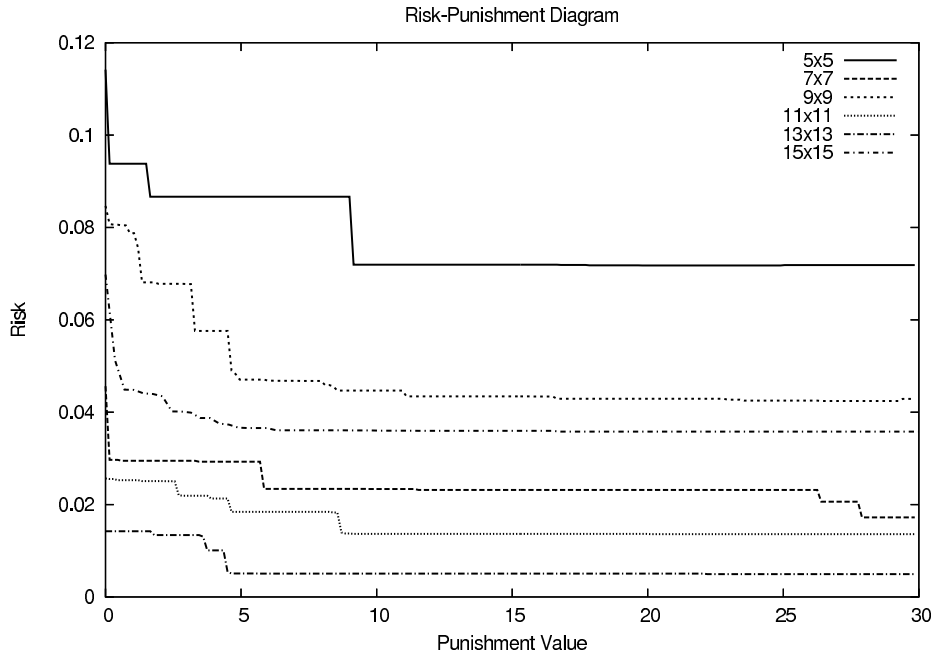
$$V(s')_{\pi_2}^{\varphi_1} > V(s')_{\pi_1}^{\varphi_1} \quad (11)$$

Now, we can use  $\pi_2$  for  $\varphi_1$  and obtain better value for some states (including  $s'$ ), but  $\pi_1$  is supposed to be the optimal policy. Thus, Proposition 1 is proven by contradiction.

Similar to Proposition 1, we recognized another property of the MDP:

*Conjecture 1.* The relationship between the punishment  $\varphi$  of fatal states  $\Phi$  and risk  $\rho^\pi$  of initial states<sup>3</sup> is inverse, and the function of the relationship is monotonically decreasing. So the higher the value of punishment, the lower the value of  $\rho^\pi$ .

The above conjecture is valid because a resulting policy would be trying to avoid fatal states more eagerly as punishment is increased, since it would be more severe to end up in  $s \in \Phi$ . Therefore, risk would be decreased with higher punishment  $\varphi$ . The proposition is shown experimentally to be correct over a range of randomly generated experiments. See Figure 3.



**Fig. 3.** Experimental confirmation of inverse relation between policy risk and punishment value  $\varphi$ . The details of the experiments are discussed in Section 4. The simulations are executed on grids with equal number of rows and columns, ranging from 5 to 15 (with a step-size of 2). In each experiment,  $d/5$  of states were randomly chosen to be goal states,  $d$  random states were set as fatal states and  $d$  other random states were initial states, where  $d$  is the number of rows in the corresponding grid.

<sup>3</sup> risk varies depending on the formulation of the initial states as defined in Section 3.2

### 3.2 Classes of Problems

Several different classes of problems are identified, and each needs a distinct planning algorithm – either in terms of a different risk evaluation algorithm (change in definition of initial state(s)) or in terms of a different interpolation algorithm to find the correct punishment value  $\varphi$  for the optimal value safe policy  $\pi_\omega^*$ .  $\pi_\omega^*$  represents the value optimal policy that satisfies the RTT  $\omega$ .

**Initial State(s)** When a set of different initial states needs to be safe, we can either ensure the safety of a policy, by looking into:

1. The absolute number of safe initial states, as done by Geibel [13]. We initially obtain the risk-minimal policy  $\pi^{r*}$ , from it, we can obtain the maximum number of safe initial states. Then, policy is updated to improve the expected value while not decreasing the number of safe initial states. The algorithm for this method differs from our value optimal safe policy generation algorithm, in that the stopping condition for the algorithm is in terms of the number of safe initial states, rather than comparing the risk of the initial states with RTT.  
or
2. The weighted average risk of initial states.

**Definition 4.**  *$I$  is the probability function that stores the probability of a state to be an initial state.  $I : S_{init} \rightarrow \mathbb{R}$ . Since  $I$  is a probability function,  $\sum_{s \in S_{init}} I(s) = 1$ , then we will have:*

$$\rho^\pi(S_{init}) = \sum_{s \in S_{init}} I(s) \times \rho^\pi(s) \quad (12)$$

We will use this definition of initial states for our framework of planning. So Equation 12 and Risk Evaluation from Section 2.5 forms the basis of our risk evaluation algorithm. A probability distribution over a set of initial states would allow an agent to select which states are more likely to be initial states and gives the agent more degrees of freedom and control over the system.

**Fatal State(s)** The main issue arises when we need to find out the minimal punishments that gives us the optimal policy with greatest  $V^\pi$  function while satisfying RTT  $\omega$ . There are two approaches of setting the punishment for fatal states:

1. Same punishment for all fatal states  $s \in \Phi$ . We search in a one-dimensional space for the minimal  $\varphi$  that still makes the policy  $\pi$  to satisfy  $\omega$  the RTT. We have used this definition of fatal states for our planning framework, for its simplicity and validity in real examples. So the assumption is that fatal states are homogeneous, and the agent consider them to be all disastrous and destructive regardless of which  $s \in \Phi$  is of concern.

2. Different punishments for fatal states  $s \in \Phi$ . This will complicate the root finding algorithm to determine optimal punishments for every  $s \in \Phi$ , because now the root for the different punishment values resides in a multi-dimensional space.

### 3.3 Value Optimal Safe Policy Generation

One observation of critical importance is that the function between punishment and risk is a *step function*, as shown in Figure 3. This has profound impacts on how the framework works.

By Propositions 1 and 1, it can be deduced that in order to generate a value optimal safe policy  $\pi_\omega^*$  that maximizes value while satisfying the RTT constraint (i.e.  $\omega$ -safe), we need a punishment  $\varphi$  that is on the step of the function between “Punishment and Risk” that is nearest and below the RTT’s  $\omega$ -line, shown as the line in bold below the dotted line in Figure 4. At this interval of the punishment values, we still satisfy the RTT restriction, and yet that is the minimum possible punishment value possible to achieve this limitation. So this punishment will give us the maximum value function with the generated policy.

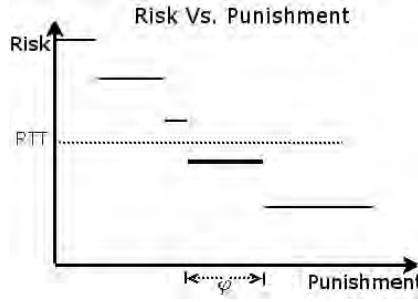


Fig. 4. Position of Optimal  $\varphi$

**Regula Falsi** For discussion purposes, let  $f(x)$  be a function that accepts a  $\varphi$  value as input parameter, and generates a value optimal policy  $\pi_\omega^*$  based on the RTT  $\omega$ . Then  $f(x)$  evaluates the risk of the policy based on the Algorithm stated in Section 2.4 for Risk Evaluation and returns that risk as output of the function.

The Regula Falsi [14] algorithm (false position) is a method suitable for finding function roots based on linear interpolation. Its convergence is linear, but it is usually faster than the bisectioning method. At each step, a line is drawn between the endpoints  $(a, f(a))$  and  $(b, f(b))$  and the point where this line crosses the RTT’s  $\omega$ -line is taken as a “midpoint”. The function’s value is calculated at the “midpoint” and its sign is used to determine which side of the

interval contains a root. That side is a smaller interval containing the root. The method is applied recursively until the target interval is sufficiently small (say with precision  $\epsilon$ ).

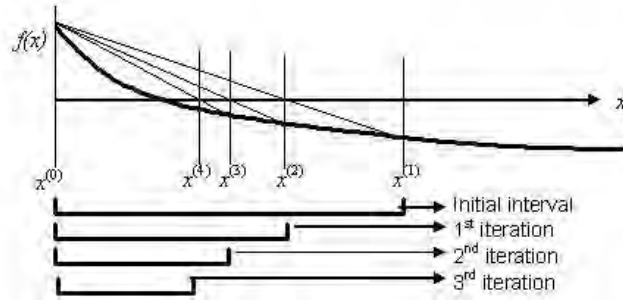
Here is a description of how Regula Falsi is done for root finding of punishment  $\varphi$  by interpolation: First we bracket the root. That is, we find two numbers  $a$  and  $b$  so that  $[a, b]$  encloses the root (or  $f(a)$  and  $f(b)$  are on different sides of RTT). Then we find the point  $(c, \omega)$  on the RTT's  $\omega$ -line where the straight line  $L$  joining the points  $(a, f(a))$  and  $(b, f(b))$  crosses the  $\omega$ -line. The value of  $c$  is computed by the following equation:

$$c = \frac{(a - b) \times w - a \times f(b) + b \times f(a)}{f(a) - f(b)} \quad (13)$$

There are three possibilities:

1. If  $(\omega - \epsilon) < f(c) < \omega$ ,  $c$  is the root, and we are done.
2. If  $f(c) \times f(a) < 0$ , the root lies in  $[a, c]$ . Let  $b = c$  and recurse
3. If  $f(c) \times f(b) < 0$ , the root lies in  $[c, b]$ . Let  $a = c$  and recurse

This algorithm is demonstrated in Figure 5, note that  $x^{(i)}$  indicates the position of the new punishment value at the  $i$ th iteration.



**Fig. 5.** The Regula Falsi process

**Root Finding for Punishment  $\varphi$**  To take into account of all cases, Regula Falsi interpolation is insufficient. Initially two end points will be chosen for the root finding algorithm. The point with smaller punishment  $(a, f(a))$  will be  $(0, f(0))$  where  $f(x)$  is as defined in Section 3.3 previously, and the point with larger punishment  $(b, f(b))$  will be the maximum reward out of all goal states multiplied by 10 which will hopefully give a sizable interval so that Regula Falsi can be done. Thus the larger punishment is:

$$\varphi = \max_{s \in \Gamma} (R(s)) \times 10 \quad (14)$$

Where  $\Gamma$  is the set of all goal states.

However, if  $f(b) > RTT$ , then we should increase  $b$  until we have a point below the RTT line. In this case, at each step, we set the next punishment value by Extrapolation (which will be discussed in Case 4). Also, if after a certain number of computation steps, we don't see such a point, "No Policy" is returned as the result.<sup>4</sup>

Now we have two initial values for root finding, we will consider each possible case for root finding in turn:

- **Case 1:** If  $f(0) < \omega$ , then RTT is higher than the risk for the value maximal policy that does not consider fatal states as a threat. So all policies are safe, and we will return the punishment for the policy that has the maximum value or  $\varphi = 0$ .
- **Case 2:** If  $f(0) > \omega > f(b)$ , RTT is in between the two end points. We will start doing interpolation by Regula Falsi, with initial punishments of 0, and  $f(b)$ . Return the result from the Regula Falsi algorithm as the optimal  $\varphi$ .
- **Case 3:** If by doubling the punishment for a limited number of times, we could never have a policy with risk less than RTT, then there is no policy that will satisfy the RTT. So signal the agent by returning  $-1$  to indicate that there there was no  $\varphi$  that will generate a policy that satisfies RTT.
- **Case 4:** The only other case left is that if there is a solution but it is not on the interval  $[f(a), f(b)]$ . We will do extrapolation to find optimal  $\varphi$ . Extrapolation is done in the same fashion as Regula Falsi interpolation to find a new point  $c$ , but the difference is that the condition of which point  $((a, f(a))$  or  $(b, f(b)))$  to drop is changed. We will drop the point with  $f(x)$  having further distance to the RTT's  $\omega$ -line.

**Final Policy Generation** After running the root finding algorithm, we would have the minimal  $\varphi$  punishment value that would still assure the RTT. With this  $\varphi$ , we could generate the final optimal policy  $\pi_\omega^*$  by using Value Iteration. One exception is that if the  $\varphi$  returned from Section 3.3's root finding algorithm is  $-1$ , then none of the policies will satisfy the RTT. So, in this case, the Risk Minimal Policy will be returned.

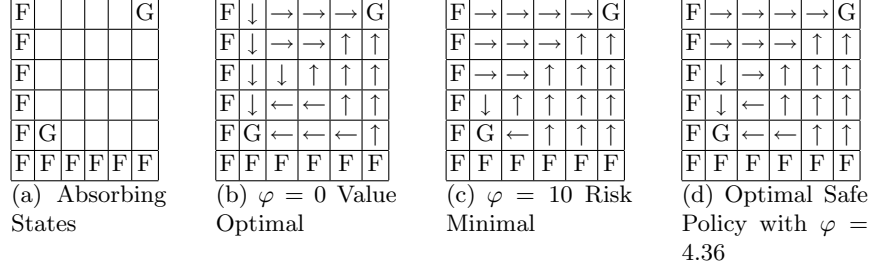
## 4 Experiment

### 4.1 Grid world environment

We used a similar Grid world environment as stated by Geibel. The only difference is that at the border states, actions that would lead the agent off the grid are not applicable. We consider a  $6 \times 6$  grid world that is depicted in Figure 6(a).

---

<sup>4</sup> If the two points were located on the same step, we increase the right punishment arbitrarily until we find a point under RTT



**Fig. 6.** Grid World and different types of Policies [13]

Each grid is a state. F's denote Fatal states, and G's denote goal states positioned at (2,2) and (6,6).

The fatal states are selected to be on the left and bottom boundaries, and goal states are positioned on the two corners to create dangerously positioned (2,2) and safely positioned (6,6) goals.

The given actions are right left  $\leftarrow$ ,  $\rightarrow$ , up  $\uparrow$ , and down  $\downarrow$ . In the border states, actions that would lead the agent off the grid are not applicable. Geibel allowed those actions, and by doing such an action, the agent does not change current state. But if such an action is allowed, when punishment gets extremely enormous, and if the state is beside a fatal state, the agent may prefer to keep choosing that action and stay in the same state rather than moving.

With a probability of 0.21, the agent is not moved to the direction specified by the selected action but to one of the three other directions. As an illustration: when the initial state is (3,4) and the agent chooses action  $\uparrow$ , then it advances upwards with probability of 0.79 to (3,5), or one of the other three directions with probability 0.21. So with probability of 0.07, it will be transported to (2,4), (3,3) or (4,4).

We have chosen the reward to be 1 for both goal states, and no cost on transition between different states. The discount factor was selected to be 0.9 ( $\gamma = 0.9$ ), so that the agent would try to find the shortest path to reach a goal state.

## 4.2 Results

**Value Maximal Policy  $\pi^*$**  Value maximal policy  $\pi^*$  should be generated with punishment  $\varphi = 0$ . The policy  $\pi^*$  is shown in figure 6(b).

The policy simply finds actions that lead the agent to a goal state on the shortest path available, and completely ignores fatal states. Even a state like (2,6) has optimal action as down  $\downarrow$  and will try to move towards goal (2,2) rather than goal (6,6), although it will be highly probable for the agent to fall into a fatal state starting at the initial state (2,6).

**Risk minimal policy  $\pi^{r*}$**  The near Risk minimal policy is generated by punishment value of  $\varphi = \max_{s \in \Gamma} (R(s)) \times 10$ . In this case, because a reward of 1 is

given to all goal states, the punishment is 10 (as a result of  $1 \times 10$ , see equation 14).

The generated policy  $\pi^{r*}$  is reasonable as only states (2, 3) and (3, 2) are trying to reach the goal state (2, 2), and all other states are more focused on attempting to avoid fatal states rather than finding a shortest path to a goal state. All those other states try to reach goal (6, 6) at the safe region of the state space.

**Final Value Optimal Safe Policy  $\pi_{\omega}^{r*}$**  Let us select (2, 4), (3, 4), (2, 3) and (3, 3) to be our possible initial states, and a probability of 0.25 is given to all initial states<sup>5</sup>. Now from the evaluation of risk by Equation 12, we can see that the policy generated with punishment of 0 has a risk of 0.1039 and value of 0.675<sup>6</sup>, and the policy generated with punishment of 10 has a risk of 0.0491 and value of 0.095.

If we define RTT to be 0.08, after our root finding algorithm is used, the optimal punishment  $\varphi$  is found to be 4.36. The final policy is shown in Figure 6(d).

This final policy  $\pi_{\omega}^{r*}$  has a risk of 0.075 and value of 0.3425 (given (2, 4), (3, 4), (2, 3) and (3, 3) as our initial states).

If we list the policies  $\{\pi^*, \pi_{\omega}^{r*}, \pi^{r*}\}$ , their risks  $\{0.1039, 0.075, 0.0491\}$ , and their values  $\{0.675, 0.3425, 0.095\}$ . We can see that  $\pi_{\omega}^{r*}$  has the risk closest to RTT, and a much higher value than can be achieved by  $\pi^{r*}$ . These results obtained through our algorithm are in accordance with Geibel’s experiment [13].

Lastly, only one interpolation step was needed to find the optimal  $\varphi$ , so the root finding algorithm performs efficiently in this experiment.

## 5 Conclusions and Future Research

We provided a solution to avoid fatal states during planning with probability above a certain Risk Tolerance Threshold (RTT). Given an initial state probability distribution, the risk of a policy can be calculated based on the evaluation algorithm. If the risk of the policy is much less than RTT, the value of the policy will be increased by decreasing punishment given to fatal states, although this will also result in increased risk. A root finding algorithm is used to find the maximum punishment that allows the Value Iteration algorithm to still generate a safe policy, so that we can have the maximum value possible while satisfying RTT.

The most immediate extension to the work is to have multiple fatal states to have different punishment values, instead of a single punishment value for all  $s \in \Phi$ . This extension would involve solving for a root in a multi-dimensional space. Another possible extension is that we currently assume complete knowledge of

<sup>5</sup> The probabilities need not necessarily be the same.

<sup>6</sup> Value is calculated the same as risk by using equation 12, except that now we replace individual risk with individual value of initial state.



the problem domain, which is not valid in many situations, so one can extend our work to Partially Observable MDPs or POMDPs.

## References

1. I. Asimov. Runaround. *Astounding Science Fiction*, March 1942.
2. D. Weld and O. Etzioni. The first law of robotics: (a call to arms). In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, Washington, 1994. AAAI Press.
3. O. Etzioni. Intelligence without robots (a reply to brooks). In *AI Magazine*. 1993.
4. C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.
5. L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.
6. M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*, chapter 16, pages 411–433. Morgan Kaufmann Publishers, draft edition, June 2003.
7. R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.
8. C. Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge, England, 1989.
9. R. Neuneier and O. Mihatsch. Risk sensitive reinforcement learning. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 1031–1037, Cambridge, MA, USA, 1999. MIT Press.
10. D. Draper, S. Hanks, and D. Weld. Probabilistic planning with information gathering and contingent execution. Technical report, Department of Computer Science and Engineering, Seattle, WA, December 1993.
11. D. Draper, S. Hanks, and Daniel Weld. Probabilistic planning with information gathering and contingent execution. In K. Hammond, editor, *Proceedings of the Second International Conference on AI Planning Systems*, pages 31–36, Menlo Park, California, 1994. American Association for Artificial Intelligence.
12. M. S. Fulkerson, M. L. Littman, and G. A. Keim. Speeding safely: Multi-criteria optimization in probabilistic planning. In *AAAI/IAAI*, page 831, 1997.
13. P. Geibel. Reinforcement learning with bounded risk. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML’01)*, pages 162–169, 2001.
14. MathPath. The regula falsi method for square-roots, January 2004. <http://www.mathpath.org/Algor/algorsquare.root.regula.falsi.htm>.

# Dependable Multi-Agent Systems: Layered Reference Architecture and Representative Mechanisms<sup>\*</sup>

Peter C. Lockemann and Jens Nimis<sup>\*\*</sup>

Fakultaet fuer Informatik, Universitaet Karlsruhe (TH)  
Postfach 6980, D-76128 Karlsruhe, GERMANY  
[lockemann, nimis]@ipd.uka.de

**Abstract.** Layered architectures are a proven principle for the design of software systems and components. The paper introduces a layered reference architecture for multi-agent systems which assigns each agent property to select layers. It demonstrates how the same reference architecture provides a framework for a dependability model that associates the sources of failures and the ensuing error handling with a specific layer, thus integrating dependability directly into the design of agents. The architectural approach is illustrated by several dependability mechanisms that assume transactional conversations.

## 1 Introduction

Dependability is not one of the favorite topics in the agent literature. Agents never seem to fail, at worst they behave in ways not expected or desired by their peers. But other disturbances do occur, disturbances that are outside the control of the agent system. Consider a job-shop situation where an order-agent negotiates with several machine-agents following the Contract Net Protocol (CNP). Suppose that the protocol has progressed to the point where some machine-agent A submits an offer to the order-agent. Suppose further that the order-agent's accept message never reaches machine-agent A due to a communication failure. If no precautions are taken, machine-agent A forever holds the resources reserved for the order-agent, while the order-agent never gets his order fulfilled. Other failures are conceivable as well. Suppose that towards the end machine-agent A breaks down while processing the order, or takes longer than promised because it encounters some shortage. The order-agent would be expected to find some way around the problem if it were designed to handle such a contingency.

---

<sup>\*</sup> This is a revised and consolidated version of two contributions to the 2004 and 2006 workshops on Safety and Security of Multi-Agent Systems (SaSeMAS) [1, 2]. The reference architecture has also been presented as a short paper at the 5th International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS2006) [3].

<sup>\*\*</sup> This work was in part funded by the Deutsche Forschungsgemeinschaft (DFG, German research council) within the priority research program (SPP) no. 1083.

Generally speaking, we call a software agent dependable if it maintains its service according to specifications even if disturbances occur within the agent. We call a multiagent system dependable if its constituent agents are dependable *and* the system as a whole meets its objectives at all times. To achieve dependability the agent or multiagent system must include internal mechanisms that guarantee that all services are rendered according to specification no matter what happens to the agent.

Engineers are trained to design systems that take into account that something out of the normal can occur, and so should software engineers. Our introductory example seems to indicate that failures may occur, and occur on various levels of abstraction all the way from technical communication to peer collaboration. Correspondingly, their remedies will involve different levels of abstraction. One concept of software engineering that recognizes levels of abstraction is the design of software systems in the form of a layered architecture. It is the main thesis of this paper that if such an architecture exists for software agents one should be able to associate a specific failure and its remedies with an individual layer, and as a consequence can develop a well-structured error model where each kind of error handling can be isolated and localized.

To support the thesis the paper proceeds as follows. Section 2 introduces and motivates a layered reference architecture for software agents. Section 3 presents a general dependability model and tailors it to the specifics of a layered architecture. The results of these two sections are fused in Section 4 into a layered dependability model for agents. Section 5 discusses some novel dependability solutions within the model and treats one—the transactional conversation approach—in considerable detail. Section 6 concludes the paper.

## 2 Reference Architecture

### 2.1 Agent Properties

We follow the widely accepted doctrine that the non-functional properties decide the internal organization of a software component [4]. Following Wooldridge there are four properties that any agent should meet at its minimum [5]:

- (1) *A software agent is a computer system that is situated and continuously operates in some environment.*
- (2) *A software agent offers a useful service. Its behavior can only be observed by its external actions.*
- (3) *A software agent is capable of autonomous action in its environment in order to meet its design objectives, i.e., to provide its service.*
- (4) *As a corollary to Property 3, the autonomy of a software agent is determined, and constrained, by its own goals, with goal deliberation and means-end assessment as parts of the overall decision process of practical reasoning.*

Properties 3 and 4 let the observed behavior appear non-deterministic or—more benignly—flexible and, hence, set software agents apart from object-oriented software [6]. Wooldridge claims that flexibility is particularly needed when

the environment appears non-deterministic as well. He calls a software agent intelligent if it is capable of operating in a non-deterministic environment, and associates four more properties with it:

- (5) *An intelligent software agent is reactive, that is, it continuously perceives its environment, and responds in a timely fashion to changes that occur.*
- (6) *An intelligent software agent achieves an effective balance between goal-directed and reactive behavior.*
- (7) *An intelligent software agent may be proactive, that is, take the initiative in pursuance of its goals.*
- (8) *An intelligent software agent may have to possess social ability, that is, is capable of interacting with other agents to provide its service.*

We note that—apart from Property 8—Wooldridge’s properties say little about multiagent systems. Therefore, we extend *Property 4* to include among the *goals both the agent’s own and those of the entire agent community*. Other properties such as technical performance and scalability of multiagent systems are important but will not be considered in the remainder.

## 2.2 Layering

It would be most convenient if we could develop an architecture that covers all agents no matter what their service and where only certain details must still be filled in for a specific agent. Such an architecture is called an architectural framework or a *reference architecture*. It provides a set of guidelines for deriving a concrete architecture and, consequently, can only be defined on the basis of generic properties. And indeed, the eight properties from above apply to an agent no matter what its application domain is.

Reference architectures follow specific architectural patterns. One of the most popular patterns is the layered architecture. An architecture is layered if one can repeatedly construct the functionality and the qualities (non-functional properties) of a more abstract higher layer from those found on a more detailed lower layer [7]. The layers pattern has in the past been applied to agents as well (e.g. [8, 9]). However, the proposals always seem specialized towards the BDI framework, do not—as is usual for distributed systems—separate internal processing and communication, and avoid the layers for the computing infrastructure. Our proposal below intends to be more complete.

Since Properties 1 to 4 are the ones shared by all agents, our goal is to determine the basic layered structure of our reference architecture from these properties. Properties 5 to 8 of intelligent agents should then have an influence only on the internals of some of the common layers.

We note in passing that layered architectures are first of all design architectures. During implementation one may elect, for performance reasons, to add, fuse, omit or rearrange components and layers.

### 2.3 Individual Agent Architecture

Properties 1 and 2 suggest a first division, with a lower part providing a computing infrastructure and the upper part reflecting the service characteristics. In addition, properties 1 and 3 indicate that the lower part be further subdivided into two layers L1 and L2. L1 should supply all the ingredients necessary for an executable component whereas L2 adds all mechanisms specifically geared towards agents. Both layers should further be subdivided vertically to reflect the agent's internal operability (the “computer system” in property 1 and the “autonomy” in property 3) and its interaction with the environment. Consequently, layer L1 includes operating systems functionality and data management facilities as offered, e.g., by relational database systems, but also the sensors and effectors necessary for the interaction. Layer L2 contains the central parts of a typical agent development framework such as the life cycle management of an agent (Property 1) or the individual control threads for each agent (Property 3), but must also translate the sensory input into perceptions and provide the actions that drive the actuators.

The upper part can be subdivided as well. To meet Property 2 the agent must have some understanding of its environment, i.e., the application domain and its active players. The understanding is reflected on layer L3 by the world model of the agent. Ontologies are the modern way to provide a deeper knowledge. Services for statically analyzing a given ontology and dynamically updating and adapting it are generic in nature and are tailored to an application domain by schema mechanisms.

A next higher layer L4 realizes the agent behavior according to Property 4. The layer must include all mechanisms necessary for goal deliberation and means-end assessment. Goal deliberation has more of a strategic character, means-end assessment is more tactical in nature and results in a selection among potential actions. Since both make use of the world model of layer L3, the rules for practical reasoning and the world model must match in structure, e.g., distinction between facts, goals and means, and in formalization, e.g., use of a symbolic representation. We restrict L4 to the individual goals. A separate layer L5 derives its actions from the more abstract social and economical principles governing the community as a whole. Fig. 1 summarizes the reference architecture for agents.

### 2.4 Locating Agent Intelligence

Given the layers of the reference architecture, we must now assign Properties 5 to 8 of intelligent agents to these layers (see again Fig. 1).

Property 5—agent reactivity—and Property 6—effective balance between goal-directed and reactive behavior—affect layers L3 and L4: The two may interact because the balance may result in the revision of goals and may thus also affect the world model.

Proactivity of an agent (Property 7) is closely related to autonomy and, hence, is already technically solved by the runtime environment of layer L2.

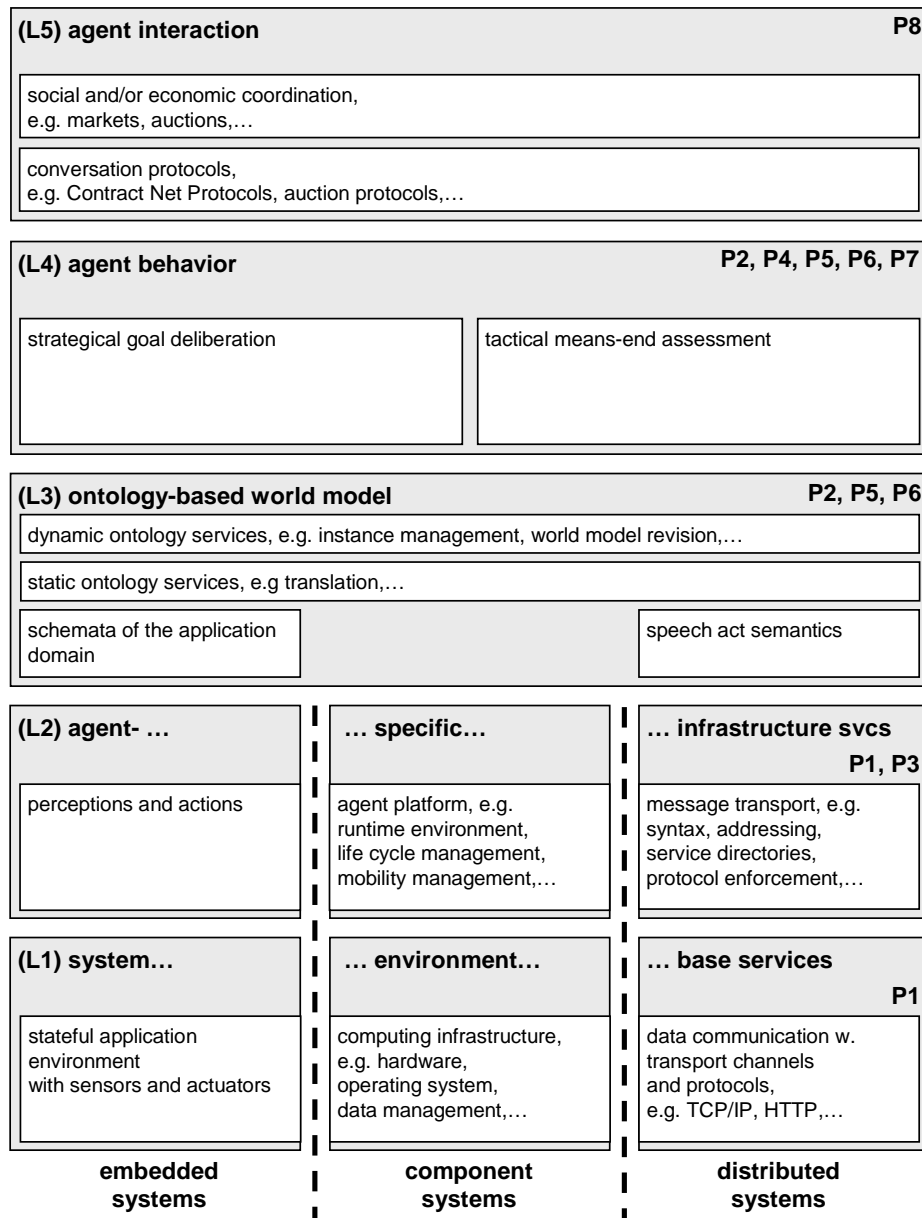


Fig. 1. Reference architecture for intelligent agents in multi-agent systems

Having control over its own progression as a result of reconciling its long-term and short-term goals implies the need for additional mechanisms on layer L4.

Property 8 directly affects layer L5: While pursuing common goals the agent may communicate with other agents to elicit certain responses or to influence their goals, i.e., it may take a more proactive role than just adjusting its own behavior.

## 2.5 Incorporating Agent Interaction

Properties 5 through 8 are also the ones that make multiagent systems a specific kind of distributed systems.

Technically speaking, a multiagent system is a distributed system where system control is entirely distributed, data is completely decentralized, and agents are loosely coupled. Hence, the basic mechanisms for system control, the interaction protocol, must itself be realized in a distributed fashion across the agents. Since communication between the agents is asynchronous, the communication mechanism is message exchange.

As is well-known from telecommunications, the corresponding communication software in each participant follows the layers pattern [7] and, hence, is referred to as a protocol stack. Consequently, the reference architecture ought to include a protocol stack as a further vertical division. Taking our cues from the well-known ISO/OSI model [10], its lower four layers—the transport system—are oblivious to the purpose of the message exchange and may thus be collapsed into layer L1, with data communication services such as HTTP or IIOP visible to layer L2.

However we know more about agent interaction. Communication as specified by Property 8 is purposeful, and is reflected by the interaction protocols of layer L5. In between, layer L2 accounts for the sequence of message exchanges between the agents. Interaction depends on some common understanding among the agents, i.e., a shared world model and ontology. Layer L3 thus must provide suitable mechanisms for proper message encoding, e.g., speech act semantics. By contrast, layer L4 controls the behavior of the individual agent and, hence, does not seem to contribute anything to the coordination per se.

Fig. 1 shows the complete reference architecture by extending the protocol stack across all layers. We note in passing that we were able to demonstrate the higher generality of our reference architecture by refining layers L3 through L5 to BDI [11], and by relating the implementation architectures of JADEX [12] and InteRRap [13] directly to it [14].

## 3 A Dependability Model

### 3.1 Failures and Dependability

We base our conceptual dependability model on (some of) the definitions of Laprie [15].

A computer system, in our case a multiagent system, is **dependable** if it *maintains its service according to specifications* even if disturbances occur that are due to *events endogenous to the system* such that *reliance* can *justifiably* be placed on this service. The **service** delivered by a system is the system behavior *as it is perceived* by another special system(s) interacting with the considered system: its user(s). The service **specification** is an *agreed description of the expected service*. A system **failure** occurs when the *delivered service deviates from the specified service*.

The *cause*—in its phenomenological sense—of any failure is a (unintentional) **fault**. We should make sure that only very few faults occur in an agent (**fault avoidance**). A fault may not always be visible. If it does, i.e., if it is *activated* in the course of providing a service, it turns into an **effective error**. For example, a programming fault becomes effective upon activation of the module where the error resides and an appropriate input pattern that activates the erroneous instruction, instruction sequence or piece of data. Only effective errors can be subject to treatment.

A **dependability model** describes the *service*, the *effective errors* occurring within the system, which ones are *dealt with and how* (**error processing**), and what, if any, the *ensuing failures* are.

### 3.2 Dependability Model for a Layer

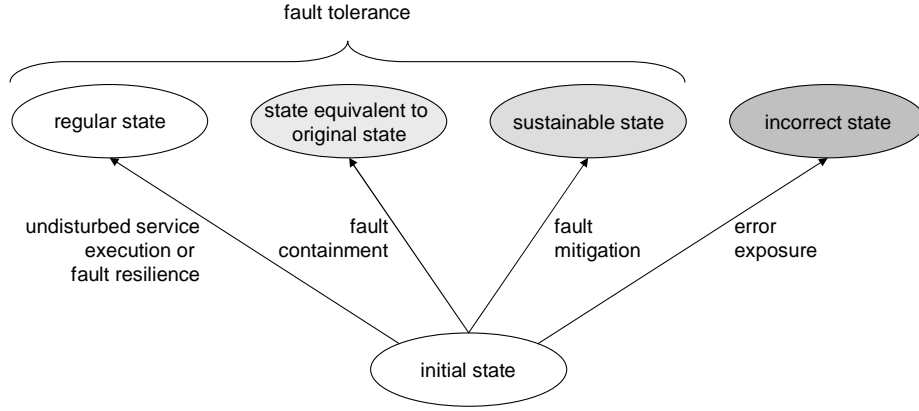
It follows from Section 2 that error processing ultimately takes place within individual agents. Hence, to make our model operational we need to refine how processing of errors takes place in the agent. The processing will have to take note of the nature of the effective error which in turn has something to do with the nature of the underlying fault.

While all faults we consider are endogenous to the entire system, from the perspective of the individual agent some may be exogenous, henceforth called external. The other faults are internal. For example, if we consider a layer in the reference architecture of Section 2.3, an internal fault may be due to a programming fault or faulty parameter settings within the code of that layer. *External* faults are either *infrastructure failures*, i.e., failures of the hardware and software the agent depends on to function properly (basically layers L1 and L2), or *peer failures*, i.e., faults from agents on the same level of service, that fail to deliver on a request due to connection loss, their own total failure, or inability to reach the desired objective due to unfavorable conditions (*unfavorable outcome*, see Pleisch and Schiper [16]).

Error processing may itself be more or less successful, and this will affect the result of a service request. From the service requestor's viewpoint the returned states fall into different classes (Fig. 2). At best, the fault may have no effect at all, i.e., the state reached after servicing the request is identical to the regular state. Or the state is still the old state, i.e., the state coincides with the state prior to the request because the fault simply caused the request to be ignored. A bit worse, servicing the request may reach a state that while no longer meeting the specification may still allow the service requestor a suitable continuation of



its work (sustainable state). All three states meet the definition of dependability. At worst, the outcome may be a state that is plainly incorrect if not outright disastrous. We introduce the following notions. We speak of *fault tolerance* if

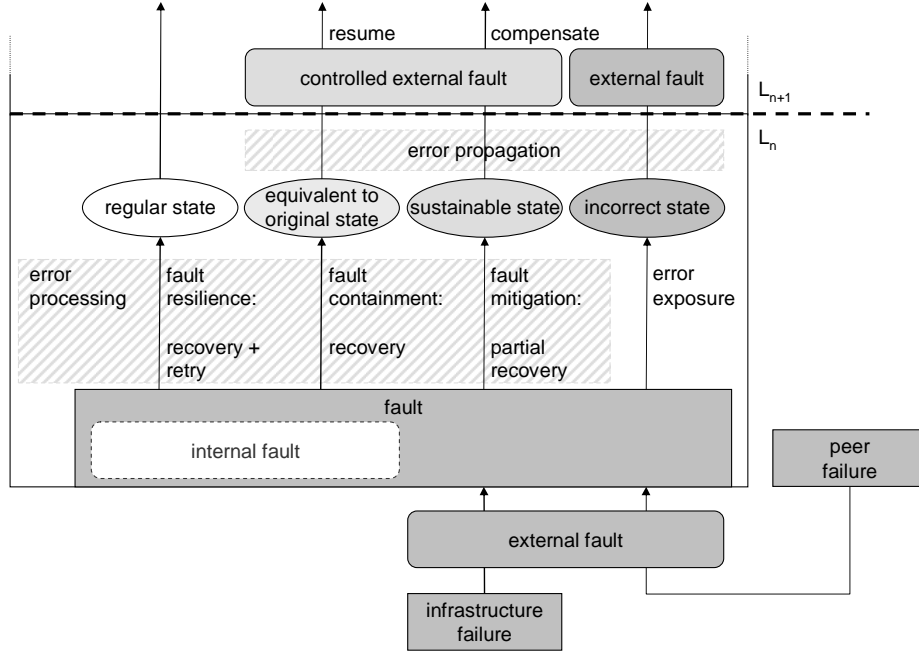


**Fig. 2.** States after service provision

a service reaches one of the three dependability states. Specifically, under *fault resilience* the correct state is reached, under *fault containment* the old state and under *fault mitigation* the sustainable state. Otherwise we have *error exposure*.

Fig. 3 relates the dependability model to the layered reference architecture, by showing two layers where the lower layer provides a service requested by the upper layer. Clearly, fault resilience and fault containment fall to the provider layer. Techniques to be used are indicated in Fig. 3 by enclosing rectangles. Fault containment is achieved by *recovery*. Typical techniques for fault resilience are recovery followed by *retry*, or *service replication*. Fault mitigation requires some action on the provider's part as well, such as partial recovery. Since fault containment, fault mitigation and fault exposure lead to irregular states, the requestor must be informed by *error propagation*. An old or sustainable state is well-defined and thus results in a controlled fault. In the first case the requestor will simply resume work in some way, in the second the requestor may have to transfer by *compensation* to a state from where to resume regular work. Error exposure leaves the requestor with no choice but to initiate its own error processing.

If no fault is detected normal processing takes place in the layer (not shown in Fig. 3). Notice that this does not imply that no fault occurred, just that it remains unprocessed as such and it is up to a higher layer to detect it.



**Fig. 3.** Dependability model: Error processing and responsibilities

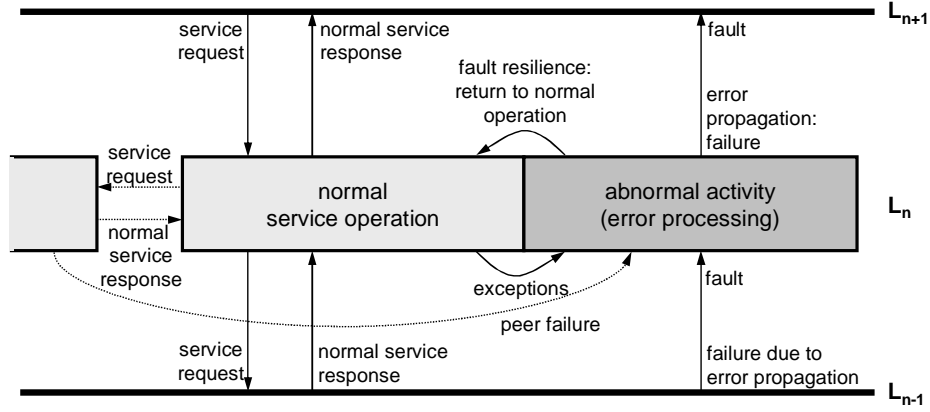
## 4 Layered Dependability Model

### 4.1 Compressed Layer Model

Fig. 3 demonstrates that error processing may not always be successful when confined to a single layer. Instead it must often be spread to higher layers where a wider context is available, e.g., a sequence of statements rather than a single one, or a larger database. Consequently, a layered architecture should help to organize and localize the various dependability issues. In particular, we use the layered reference architecture of Section 2 for organizing the dependability of agents.

In order to discuss the responsibilities of each layer we use an abstraction of the dependability model of Fig. 3 that borrows from the Ideal Fault Tolerant Component of Anderson and Lee for software component architectures [17, p.298].

As Fig. 4 illustrates, a layer can be in one of two operation modes. During normal activity the component receives service requests from layers higher up, eventually accomplishes them using the services of the next lower layer components and returns normal service responses. It may also draw on the services of other components on the same layer or receive service requests from such components.



**Fig. 4.** Compressed dependability model for a single layer

If the lower layer fails to service the request, this can be interpreted on the current layer as the occurrence of a fault. The mode changes to abnormal activity where the fault is interpreted and treated. Error processing follows the model of Fig. 3. Propagated errors are signaled as a failure to the next higher layer. Otherwise the layer returns to normal mode and tries to resume normal operation.

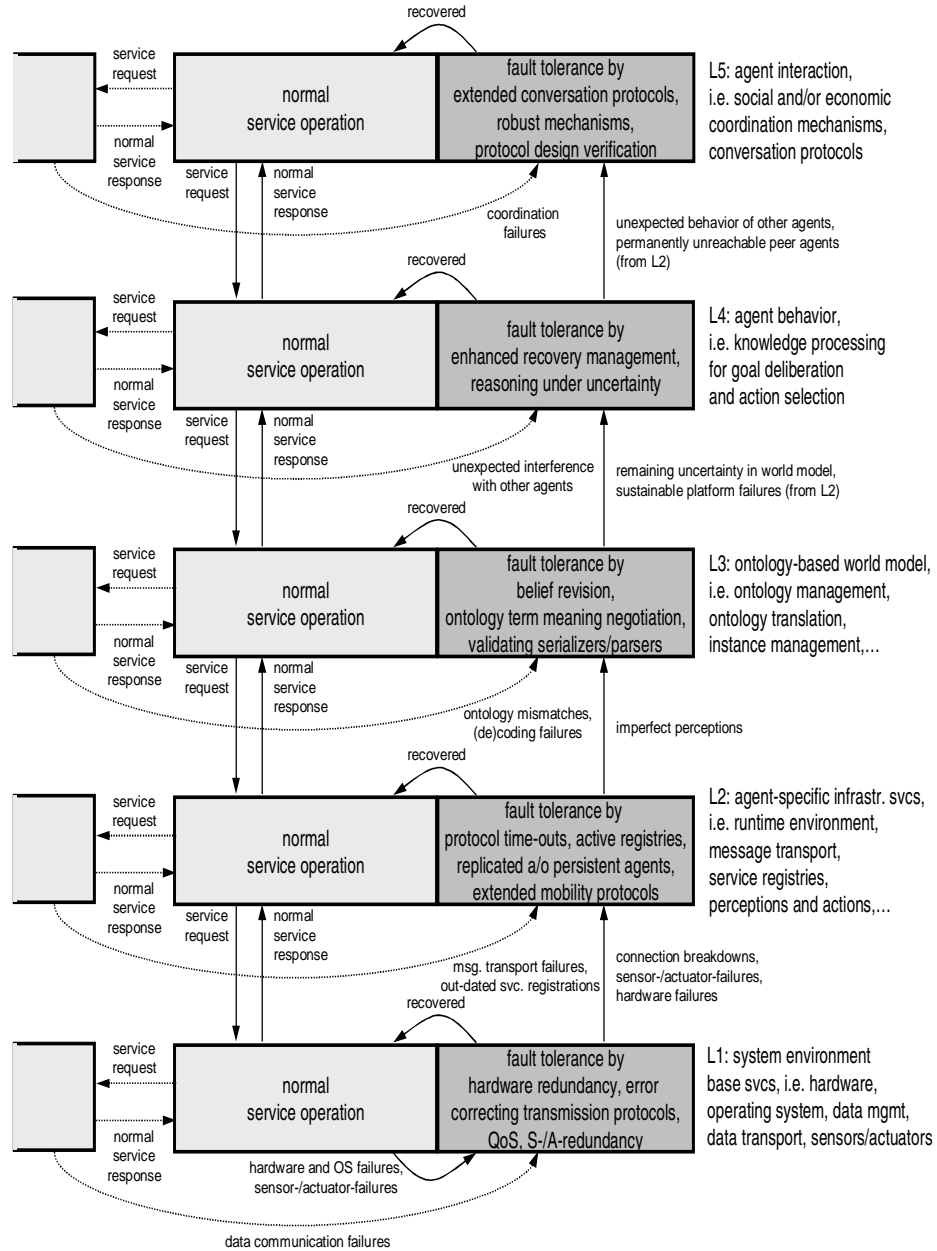
Abnormal activity mode may also be entered due to abnormal events on the same layer. One such event is a malfunction within the same component—an internal fault. A typical mechanism for signaling such an event is (in Java terminology) by throwing exceptions. Exceptions can also be used to model a situation where a fault was not activated further down and thus remains unrecognized while crossing a layer in normal mode and moving upwards as a normal service response. Only on a layer higher up the fault is detected as such.

Another kind of abnormal event is peer failures. If they are recognized as such they are being dealt with, otherwise they propagate upwards as part of a normal service response only to induce an exception higher up, or directly as a fault.

For the rest of this section we give a structured overview of the various faults that may occur on each layer, and short summaries of some of the approaches to cope with them. Fig. 5 depicts failures and approaches and thus summarizes the following subsections. For the sake of clarity, peer failures that are propagated upwards are still shown as peer failures on the layer where they are processed.

## 4.2 System Environment Base Services (L1)

Layer L1 encompasses all services general to a computing environment including data communication. The individual agent is affected by infrastructure failures such as hardware and operating system failures. Operating systems have a built-in dependability, usually by redundancy, that allows them to recover from hard-



**Fig. 5.** Dependability in the layered agent architecture

ware failures. The operating system will show a failure only if such measures are ineffective. If the failure is catastrophic the system will stop executing altogether. If the failure is non-catastrophic such as excessive waiting or unsuccessful disk operation, a failure will be propagated upwards with a detailed description of the cause.

Similarly, most of the faults typical for classical data communication (e.g., package loss/duplication/reordering, or network partition) are internally resolved by any of the numerous protocols, i.e., modern data communication is inherently dependable (see for example [18]). This leaves very few peer failures to be propagated upwards, such as network delays, connection loss, non-correctable transmission failures. In particular, a catastrophic event at one agent can only be detected by other agents where it manifests itself as an unreachable destination. Some dependability aspects are subject to agreement and become quality-of-service (QoS) parameters, e.g., transmission reliability with respect to loss and duplication (at-least-once, at-most-once, exactly-once, best effort), or order preservation. Qualities outside the agreement must be controlled on the next higher layer and if deemed unsatisfactory will raise an exception there.

### **4.3 Agent-specific Infrastructure Services (L2)**

Layer L2 covers the infrastructure services that are specialized towards the needs of agents. Typical platforms are agent development/runtime frameworks like JADE [19] or LS/TS (Living Systems® Technology Suite [20]).

The foremost platform mission is agent interoperability. Typical services concern message handling and exchange, trading and naming, or agent mobility. To give an example, message handling is in terms of the FIPA ACL messages with their numerous higher-level parameters, where in particular the message content may be expressed in terms of the FIPA ACL communicative acts. Now take an uncorrectable data transmission failure on layer L1. It can result in a message loss on the platform layer. Some of the multi-agent platforms such as JADE allow to specify a time-out interval within which a reply to a given message must be received, and once it expires invoke an exception method such as a retry by requesting a resend of the original message.

Another important function is trading, locating an agent that offers the services needed by another agent. The yellow pages functionality of FIPA's Directory Facilitator implements such a function. A fault internal to layer L2 may be an invalid agent service registration. Outdated entries can internally be avoided by an active Directory Facilitator that periodically checks for availability of the registered services. Otherwise the upper layers must be notified of the failure, where most probably the coordination layer (L5) would have to handle it by finding an alternative peer to provide a similar service.

### **4.4 Ontology-based Domain Model (L3)**

From layer L3 on upwards the domain semantics come into play. The only problems that may arise on layer L3 are peer failures in connection with ontologies.

For example, in a message exchange a partner agent may not share the ontology of a given agent so that there is a mismatch in the meaning attributed to a certain term. Provided the mismatch can be detected, one could take approaches like Mena et al. [21] that try to negotiate the meaning of the ontological terms during the course of the agents' interaction. But even if sender and receiver share the same ontology, different coding into and decoding from the transport syntax can be a source of errors.

#### 4.5 Agent Behavior (L4)

Layer L3 deals foremost with static aspects. By contrast, layer L4 deals with the agent functionality and, hence, with the domain dynamics. Therefore, it has by far the most complicated internal structure of all layers, that is based on certain assumptions, i.e., the way knowledge and behavior of the agents are presented and processed. Clearly then, layer L4 should be the layer of choice for all faults—many originating on layer L2—whose handling requires background knowledge of the domain, the goals and the past interaction history. Consequently, most of the error processing will fall under the heading of fault resilience. This is true even for catastrophic infrastructure failures. Suppose the agent temporarily loses control when its platform goes down, but regains it when the platform is restarted. Recovery from catastrophic failure is usually by backward recovery on the platform layer. The resulting fault containment should now be interpreted in light of the agent's prior activities. If these were continuously recorded the agent may add its own recovery measures, or judge itself to be somewhat uncertain of its environment but then act to reduce the uncertainty.

#### 4.6 Agent Interaction (L5)

Whereas Layer L4 is the most critical for dependable functioning of the single agent, layer L5 must ensure the dependable performance of the entire multi-agent system. We refer to the coordinated interaction of agents as a conversation. Conversations follow a—usually predefined—conversation protocol. A number of approaches deal with this issue by supporting the protocol design process. Paurobally et al. [22], Nodine and Unruh [23], Galan and Baker [24] and Hannebauer [25] present approaches in which they enrich a specification method for conversation protocols by a formal model that allows them to ensure the correctness of the protocol execution analytically or constructively.

At runtime conversations are threatened by uncorrectable interoperability failures that are passed upwards from lower layers. The infrastructure layers may pass up network splits and unreachable peers, the ontology layer semantic disagreements, the agent behavior layer unforeseen behavior of peer agents. Layer L5 may itself discover that peers respond in mutually incompatible ways. Techniques of fault resilience seem essential if the user is not to be burdened. For example, the agents may search for similar services, or they may attempt to agree on a new protocol with different peer roles or peers.

#### 4.7 Errors Passed through to the User

A multiagent system may not be able to automatically handle each and every fault, and probably should not even be designed to do so, particularly if the outcome of the failure has a deep effect on the MAS environment and it remains unclear whether any of the potential therapies improves the environmental situation. Consequently, a careful design should identify which errors should be passed as failures up to what one could conceptually see as an additional “user” layer.

### 5 Dependable Agent Interaction

#### 5.1 Transactional Dependability

Given some idea of the background knowledge needed to alleviate a fault and obtain fault resilience, one should be able to determine the layer where to place the error correction mechanisms. If there is more than one potential layer, other factors such as the effect on the other properties should be studied in order to take a final decision. Indeed, we claim that a layered architecture for software agents exactly allows rational decisions of this kind. We use this section to try to support the claim by examining several solutions for peer failures that result in a faulty agent interaction.

We study four novel dependability mechanisms. All four are based on the classical dependability instrument of transaction. Transactions are a widely used behavioral abstraction that includes recovery. In its purest form the transaction has the ACID properties (see [26]). In particular, it is *atomic*, i.e., is executed completely or not at all, hence in the failure case, assumes backward recovery; and it is *consistent*, i.e., if executed to completion it performs a legitimate state transition.

#### 5.2 L2: Recovery by Transactional Conversations

Classical transaction systems guarantee the ACID properties without any knowledge of the algorithms underlying the transactions (they are “generic”). Consequently, one wonders whether recovery of agents and agent interaction can be confined to layer L2 because no further background knowledge is needed. In other words, we look for an abstraction that does without domain knowledge.

Agent interaction follows a conversation protocol that structures the interaction. In the course of the conversation the agents change their states and carry out actions to change their environment. In the introductory job-shop example that satisfied the Contract Net Protocol, a change of the agents’ states could result in an adjusted pricing of machine slots for the following rounds of negotiation. An action in the environment could be a reservation or booking of a certain time slot on a machine, which causes a change to the machine’s schedule.

The transactional abstraction of a conversation is the distributed transaction. Conversations spawn local transactions in the participating agents, that

are coordinated by the initiating agent. This requires some technical prerequisites. Both, the agents' states and the state of the environment must be stored in databases. The implementation of the distributed transactions should follow a standard such as the X/Open DTP Reference Model [27] that introduces two architectural elements: Resource Managers and a Transaction Manager. The later enforces the so-called 2-Phase-Commit Protocol [26], which guarantees atomicity for distributed transactions. It coordinates the involved data sources and decides whether a distributed transaction was successful or not, and as a consequence initiates global commit or rollback. It therefore orchestrates the Resource Managers that locally enforce the ACID-properties and manage the access to the data items, in our case the state of the agents and the environment. For our implementation we relied on commercial products, the Oracle Database 10g<sup>1</sup> for Resource Manager and Atomikos<sup>2</sup> for Transaction Manager<sup>3</sup>.

For standards purposes, our approach was integrated into the FIPA-compliant [29] agent development framework JADE [19]. JADE has the additional advantage of a clear internal structure, its good documentation and its widespread use in research and commercial projects. The following description of the implementation refers to typical characteristics of JADE. Nevertheless, the basic conception of the approach should be applicable to other FIPA-compliant platforms as well<sup>4</sup>.

In JADE the different kinds of functionality of an agent are accomplished by so-called *behaviors*. Roughly speaking, an agent has a special behavior to handle the protocol execution for each conversation type it participates in. To allow for a structured agent design JADE provides the ability to split the functionality of a behavior into so-called *sub-behaviors*. A *parent-behavior* initiates its sub-behaviors, can track their execution, and is informed, when a sub-behavior has finished its work.

Now, behaviors have to be made transactional. We introduce generic *transactional behaviors* (*ta-behaviors*) as parent-behaviors of the so-called *payload-behaviors*. The latter carry out the actual conversations. The resulting system architecture of the robustness mechanism is shown in Fig. 6. The course of action between the different architectural elements is depicted in Fig. 7 for the introductory job-shop scheduling example<sup>5</sup>.

In the initialization phase when an agent A wishes to start a transactional conversation, its ta-behavior generates a transaction context and sends it to the ta-behavior of the peer agent (B) together with an identifier for the requested type of conversation. The local and the remote ta-behavior then initiate the re-

<sup>1</sup> Oracle Corporation. <http://www.oracle.com/database/>

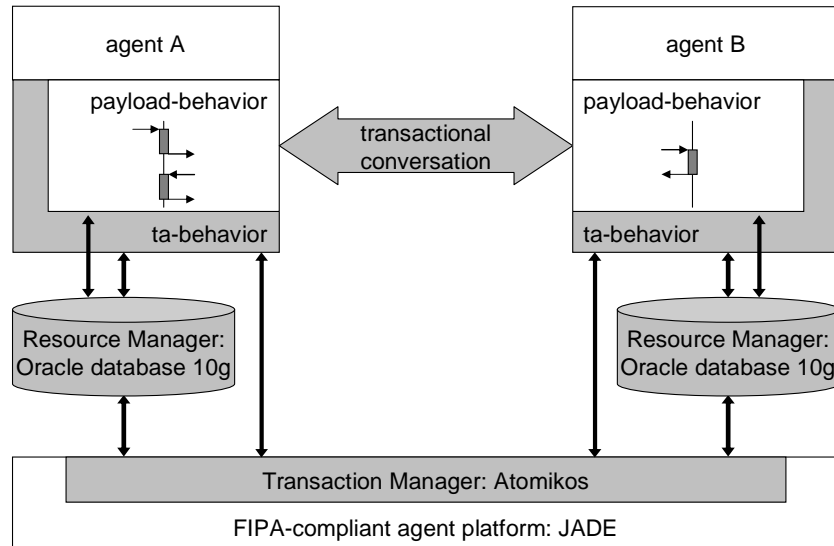
<sup>2</sup> Atomikos. <http://www.atomikos.com/products.html>

<sup>3</sup> More technical details of the implementation concepts are given in [28].

<sup>4</sup> Actually, our first implementation of the approach described in [1, 28] makes use of the FIPA-compliant platform FIPA-OS [30], the Oracle 9i RDBMS for Resource Manager and the Orbacus OTS for Transaction Manager.

<sup>5</sup> For the sake of clarity we restrict the scenario to a single machine-agent (the Contract Net-responder).





**Fig. 6.** Integration of transactional conversations in JADE: system architecture

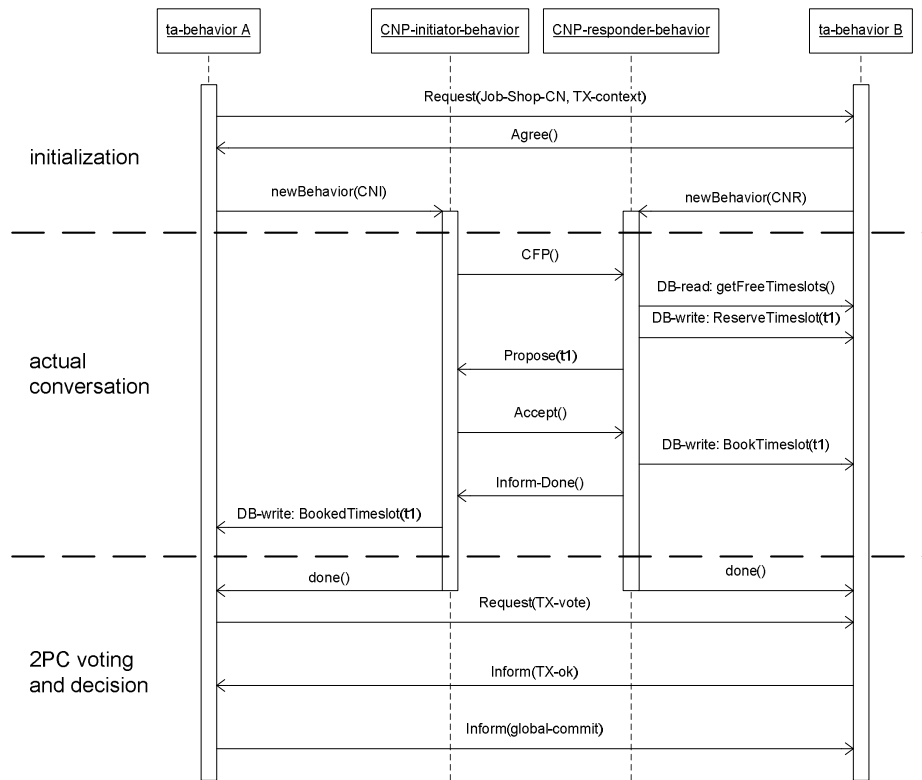
quired payload-behaviors—here: CNP-initiator-behavior resp. CNP-responder-behavior—to carry out the actual conversation.

In the second phase the payload-behaviors follow the normal conversation protocol. The ta-behaviors observe the execution and the occurrence of local state changes. Because not all of these changes must be transactional the payload-behaviors have to indicate to the ta-behaviors which data items should be handled as a transactional resource. This allows the developer a certain degree of control over what should be restored in case of a transaction rollback.

The decision on rollback or commit of the distributed transaction is taken in the third phase. The changes in the states of the communicating agents and the environment now are synchronized via the success or failure of the overall distributed transaction. Therefore, the ta-behaviors of the communicating agents first vote and then distribute the decision according to the 2-Phase-Commit (2PC) Protocol, with the conversation-initiating ta-behavior as the coordinator node.

The presented approach for transactional conversations allows the developer to have guarantees on consistent agent states while still using the framework he is accustomed to. All he has to do is to annotate, with a few additional lines of code, those states of the agents and of the environment that he wishes to be under transaction control. To give an example, in job-shop scheduling no outdated reservations of the machines' time slots would be left in the agents' or environment's states even in the case of errors like communication disturbances.

However, the solution comes at a price. Least serious is that transactional conversations are a technique for fault containment so that higher layers may have to get involved as well. Extremely serious is that 2PC has a negative effect



**Fig. 7.** Integration of transactional conversations in JADE: course of action

on agent autonomy because in case of serious delays or node failures all other agents are held up. Further, 2PC by itself may not be sufficient to guarantee consistency but must be accompanied by synchronization of the transactions, another pitfall for autonomy. Consequently, each agent must include a Resource Manager that locally enforces the ACID properties and manages access to the data items, and a Transaction Manager that guarantees, via the 2PC, atomicity for distributed transactions. Such managers are readily available only as part of database management systems.

### 5.3 L3: Transactional Semantics Based on Speech Acts

By shortening the duration of transactions one could mitigate some of the negative effects of the previous approach. Remember that we already relaxed some of the transactional strictness by allowing the developer to define those parts of the agents' states that should not be under transaction control. A more systematic extension of this idea would relate the developer's decisions to a more agent-specific model of transactional semantics.

The existing semantic model for agent communication gives a first clue. Conversations are composed of messages that represent so-called speech acts, and for these formal semantics exist [31]<sup>6</sup>. The underlying formal semantic model defines the impact of the message content on the states of the sender and the receiver for each speech act keyword, like e.g., *confirm* or *propose*.

The meaning of the speech acts is given in the form of a pair of precondition and postcondition, both expressed in a logic of mental attitudes and actions. The logic has been formalized by Sadek [32, 33] in a first-order modal language with identity<sup>7</sup>. The logical model for the *belief*-operator  $B$  is a KD45 possible-worlds-semantics Kripke structure (for a detailed explanation see Fagin et al. [35]). The formal logical representations and models bring with them a large number of axioms and theorems that allow to define some kind of consistency rules for an agent's beliefs, goals and intentions.

In our current work we investigate what impact a commit and a rollback according to the speech act semantics should have. The key question in this context is how to exploit the speech act pre- and postconditions together with the axioms and theorems of the formal semantics in order to derive invariants of the "right" transaction semantics for agent conversations.

Speech acts have been placed on layer L3 of the reference architecture (see Fig. 1). Therefore the technique just described is a layer L3 technique. It still is generic since it can do without specific domain knowledge.

### 5.4 L3/4: Compensation by Belief Management

Clearly, we would be much better off without 2PC, with just the individual agents being transactional but the conversation as a whole not. One could argue

<sup>6</sup> In the referenced FIPA specification speech acts are referred to as *communicative acts*.

<sup>7</sup> The formalization is similar to the better-known work of Cohen and Levesque [34].

that the agents are certain of their own state but uncertain of the state of the conversation. Thus we have a case of fault mitigation on layer L2. To turn mitigation into resilience we must reduce the uncertainty. We can do so only if the world model is known. Hence we should seek part of the solution on layer L3. Clearly, this solution is no longer entirely generic.

Since the BDI framework allows to cope with the non-determinism of the environment where beliefs stand for the uncertain, fuzzy, incomplete knowledge acquired by the agent, beliefs seem a good model to guide compensation [36].

“When changing beliefs in response to new evidence, you should continue to believe as many of the old beliefs as possible” [37, 38]. How this principle is turned into a practical solutions may give rise to different belief models. Our approach uses two non-deterministic transformations [39]. *Revision* adds a new proposition. If the information is incompatible with the previous state then older contradictory information is removed. *Contraction* removes a proposition. Removal may trigger further removals until no incompatible proposition can further be derived. For both there may be more than one way to restore consistency. A third transformation, *expansion*, simply adds the observation as a new belief no matter whether it contradicts existing beliefs or not.

More specifically, we modelled individual beliefs as fuzzy propositions over a discrete set of (not necessarily numerical) values. The fuzziness allows to move from a notion of strict consistency to a gradual consistency degree. An agent can dynamically adapt its consistency degree over time: arriving fresh on a scene, it might want to absorb many, possibly contradicting beliefs, whereas after a longer period of time being more strict contexts have been established. Fuzzified versions of the operators introduced above modify belief sets while considering whether the remaining consistency reaches at least the prescribed degree.

## 5.5 L4/5: Compensation by Distributed Nested Transactions

Belief management involves the individual agents in a compensation. As an alternative, one may try to associate the compensation with the entire conversation and, hence, involve an entire group of agents. As a prerequisite, the conversation must be understood to some detail, and this in turn presupposes some understanding of the agent behavior. Consequently, such an approach is located on layers L4 and L5.

First of all, then, we need a behavioral abstraction for agents that takes the layered architecture into account. A model that reflects that an external event may spawn several interrelated actions on different layers is the nested transaction [40]. In the model a transaction can launch any number of subtransactions, thus forming a transaction tree. Subtransactions may execute sequentially, in parallel or alternatively. A subtransaction (or the entire transaction) commits once all its children have terminated, and makes its results available to all other subtransactions. If one of the children fails, the parent transaction has the choice between several actions: abort the subtransaction, backward recover and retry it, or attempt forward recovery by launching a compensating subtransaction.

To reflect a conversation the nested transactions of the participating agents must be synchronized. Each transaction is augmented by special synchronization nodes that trigger events in the partner transactions [41]. This mechanism can now also be used to incorporate a conversation-wide error compensation.

The drawback of the approach is that there is no longer a clear separation between agent behavior and conversation. Consequently, conversations are difficult to extract and adapt.

## 6 Conclusions

The paper demonstrates that one can systematically derive from the properties of intelligent software agents a layered reference architecture for agents where the properties can be assigned to specific layers. We showed that this architecture also offers a framework for locating failure occurrences and the ensuing dependability mechanisms to specific layers. Thus the reference architecture allows to include in the design of multiagent systems dependability right from the beginning. Our own work seems to suggest that the approach is indeed viable, although it may not always be possible to confine handling of a particular failure to just one layer. As indicated in Section 5 the detailed dependability techniques themselves still raise intricate issues.

## References

1. Nimis, J., Lockemann, P.C.: Robust multi-agent systems: The transactional conversation approach. In Barley, M., Massacci, F., Mouratidis, H., Scerri, P., eds.: 1st International Workshop on Safety and Security in Multiagent Systems (SaSeMAS2004), New York City, NY, USA (2004)
2. Lockemann, P.C., Nimis, J.: Agent dependability as an architectural issue. In Barley, M., Mouratidis, H., Spears, D., Unruh, A., eds.: 3rd International Workshop on Safety and Security in Multiagent Systems (SaSeMAS2006), Hakodate, Japan (2006)
3. Lockemann, P.C., Nimis, J.: Agent dependability as an architectural issue. In Weiss, G., Stone, P., eds.: Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS2006), New York, NY, USA, ACM Press (2006) 1101–1103
4. Starke, G.: Effective Software Architectures (in German). Hanser, Munich, Germany (2005)
5. Wooldridge, M.J.: An Introduction to MultiAgent Systems. Wiley, Chichester (2002)
6. Luck, M., Ashri, R., d’Inverno, M.: Agent-Based Software Development. Artech House, Inc., Norwood, MA, USA (2004)
7. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture: A System of Patterns. John Wiley & Sons, New York City, NY, USA (1996)
8. Kendall, E.A., Pathak, C.V., Krishna, P.V.M., Suresh, C.B.: The layered agent pattern language. In: Conference on Pattern Languages of Programs (PLoP’97). (1997)

9. Jennings, N.R.: Specification and implementation of a belief-desire-joint-intention architecture for collaborative problem solving. *International Journal of Intelligent and Cooperative Information Systems* **2**(3) (1993) 289–318
10. Stallings, W.: *Data and computer communications*. 7. edn. Pearson Prentice Hall, Upper Saddle River, NJ (2004)
11. Bratman, M.E., Israel, D.J., Pollack, M.E.: Plans and resource-bounded practical reasoning. *Computational Intelligence* **4**(4) (1988) 349–355
12. Braubach, L., Pokahr, A., Lamersdorf, W.: Jadex: A BDI agent system combining middleware and reasoning. In Unland, R., Calisti, M., Klusch, M., eds.: *Software Agent-Based Applications, Platforms and Development Kits*. Birkhaeuser, Basel, Suisse (2005) 143–168
13. Müller, J.P.: *The Design of Intelligent Agents: a Layered Approach*. Springer, Heidelberg, Germany (1996)
14. Lockemann, P.C., Nimis, J., Braubach, L., Pokahr, A., Lamersdorf, W.: Architectural design. In Kirn, S., Herzog, O., Lockemann, P.C., Spaniol, O., eds.: *Multiagent Engineering Theory and Applications in Enterprises*. International Handbooks on Information Systems. Springer Verlag, Heidelberg, Germany (2006) 405–430
15. Laprie, J.C.: Dependable computing and fault tolerance: concepts and terminology. In: 15th IEEE Symposium on Fault Tolerant Computing Systems (FTCS-15). (1985) 2–11
16. Pleisch, S., Schiper, A.: Approaches to fault-tolerant and transactional mobile agent execution—an algorithmic view. *ACM Comput. Surv.* **36**(3) (2004) 219–262
17. Anderson, T., Lee, P.A.: *Fault Tolerance: Principles and Practice*. Prentice/Hall International, Englewood Cliffs, NJ, USA (1981)
18. Halsall, F.: *Data communications, computer networks and open systems*. Addison-Wesley, Harlow, England (1998)
19. Bellifemine, F., Bergenti, F., Caire, G., Poggi, A.: Jade – a java agent development framework. In Bordini, R., Dastani, M., Dix, J., El Fallah-Seghrouchni, A., eds.: *Multi-Agent Programming*. Kluwer, Dordrecht (2005) 125–148
20. Rimassa, G., Calisti, M., Kernland, M.E.: *Living systems technology suite*. Technical report, Whitestein Technologies AG, Zurich, Suisse (2005)
21. Mena, E., Illarramendi, A., Goni, A.: Automatic ontology construction for a multiagent-based software gathering service. In Klusch, M., Kerschberg, L., eds.: *4th International Workshop on Cooperative Information Agents*, Heidelberg, Germany, Springer (2000) 232–243
22. Paurobally, S., Cunningham, J., Jennings, N.R.: Developing agent interaction protocols using graphical and logical methodologies. In Dastani, M., Dix, J., El Fallah Seghrouchni, A., Kinny, D., eds.: *Workshop on Programming Multi-Agent Systems (ProMAS2003)*. (2003) 1–10
23. Nodine, M., Unruh, A.: Constructing robust conversation policies in dynamic agent communities. In Dignum, F., Greaves, M., eds.: *Issues in Agent Communication*. Volume 1916., Heidelberg, Germany, Springer (2000) 205–219
24. Galan, A., Baker, A.: Multi-agent communications in jafmas. In: *Workshop on Specifying and Implementing Conversation Policies*, Seattle, Washington (1999) 67–70
25. Hannebauer, M.: Modeling and verifying agent interaction protocols with algebraic petri nets. In: 6th International Conference on Integrated Design and Process Technology (IDPT-2002), Pasadena, USA (2002)
26. Weikum, G., Vossen, G.: *Transactional information systems: theory, algorithms and the practice of concurrency control and recovery*. Morgan Kaufmann, San Francisco (2002)

27. The Open Group: Distributed Transaction Processing: Reference Model, Version 3. The Open Group (1996)
28. Vogt, R.: Embedding a transaction-based robustness-service into a fipa-compliant multi-agent framework (in german). Diploma thesis, IPD, Universitaet Karlsruhe (TH), Karlsruhe, Germany (October 2001)
29. Poslad, S., Charlton, P.: Standardizing agent interoperability: The FIPA approach. In Luck, M., Marík, V., Stepánková, O., Trappl, R., eds.: 9th ECCAI Advanced Course ACAI 2001, Selected Tutorial Papers. (2001) 98–117
30. Poslad, S., Buckle, P., Hadingham, R.: FIPA-OS: the FIPA agent Platform available as Open Source. In Bradshaw, J., Arnold, G., eds.: Proceedings of the 5th International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 2000), Manchester, UK, The Practical Application Company Ltd. (April 2000) 355–368
31. FIPA: Communicative act library specification. Specification, Foundation for Intelligent Physical Agents (2002)
32. Sadek, M.D.: A study in the logic of intention. In Nebel, B., Rich, C., Swartout, W., eds.: KR'92. Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference. Morgan Kaufmann, San Mateo, California (1992) 462–473
33. Sadek, M.D.: Attitudes Mentales et Interaction Rationnelle: Vers une Theorie Formelle de la Communication. PhD thesis, Universite de Rennes I, France (1991)
34. Cohen, P.R., Levesque, H.J.: Intention is choice with commitment. *Artificial Intelligence* **42**(2-3) (1990) 213–261
35. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge, MA (1995)
36. Lockemann, P.C., Witte, R.: Agents and databases: Friends or foes? In: 9th International Database Applications and Engineering Symposium (IDEAS05), Montreal, Canada, IEEE Computer Soc. (2005) 137–147
37. Harman, G.: Change in View: Principles of Reasoning. The MIT Press, Cambridge, MA, USA (1986)
38. Katsuno, H., Mendelzon, A.: On the difference between updating a knowledge base and revising it. In Gärdenfors, P., ed.: Belief Revision. Cambridge University Press, Cambridge, MA, USA (1992) 183–203
39. Witte, R.: Architecture of Fuzzy Information Systems. PhD thesis, Universitaet Karlsruhe (TH), Karlsruhe, Germany (2002)
40. Nagi, K.: Scalability of a transactional infrastructure for multi-agent systems. In Wagner, T., Rana, O.F., eds.: 1st International Workshop on Infrastructure for Scalable Multi-Agent Systems, Heidelberg, Germany, Springer (2000) 266–272
41. Nagi, K.: Modeling and simulation of cooperative multi-agents in transactional database environments. In Wagner, T., Rana, O.F., eds.: 2nd International Workshop on Infrastructure for Scalable Multi-Agent Systems, Heidelberg, Germany, Springer (2001)