Representation and Management of Systems Architecture at the Enterprise Level

Jun Han *Pin Chen* *Abdel El-Sakka*

Software Systems Group
Peninsula School of Computing
Monash University

Information Architectures Group
DSTO Joint Systems Branch
Department of Defence

{Jun.Han@infotech.monash.edu.au} {Pin.Chen;Abdel.El-Sakka@dsto.defence.gov.au}

Abstract: *System architecture has been traditionally developed in a project context or at a project level. Finding and exercising a workable solution for systems architecture representation and management at the enterprise level will undoubtedly enhance the capability of large organisations in its future development. It is therefore becoming a pressing concern and a serious challenge for large organisations. Although most large organisations are not yet at the stage to formally develop the systems architecture at the enterprise level, their current architecture practices and products form the basis on which such an important organisation asset can be developed. The paper describes a number of management solutions, with its focus centring on initial studies of the architecture continuum-based approach and its associated representation issues.*

1. Introduction

Information has become an essential asset for modern enterprises and organisations. It affects all aspects of an organisation's business activities. The ability of an organisation to effectively manage the quality and flow of its enterprise information is vital to its survival and growth. This reality has made organisations on an unprecedented drive to the adoption of information technology (IT) in all aspects of its business operation. In this context, the IT systems of an organisation should be managed and evolved to facilitate the proper functioning of the business operation. The architecture of an organisation's IT systems as a whole, called the *enterprise system architecture,* is central to the management and evolution of its IT systems and therefore to its business operation. The enterprise system architecture includes both the architectures of individual systems and the relationships between them in the context of an organisation. While there have been a great amount of studies that address how a "to-be" enterprise system architecture should be developed, not much of attention has been paid to addressing how an "as-is" architecture for a large organisation can be reached and managed. We note that the "as-is" enterprise system architecture equates to systems architecture and the two terms will be used interchangeably throughout the paper.

Developing the enterprise system architecture for large organisations is a complicated task and yet to be studied systematically. There are some key factors that are strongly related to the selection of solutions for such development. Such factors include approaches used in individual system development, architectures developed in association with individual systems, features of these systems, main purposes of using the enterprise system architecture and its development process, management and evolution issues. The concept of a scale-oriented architecture continuum is introduced in this paper as the basis for one of the management solutions to organise the enterprise system architecture in terms of the IT systems at different levels of granularity in an organisation. Architecture views and system interface characterisations are used as the primary mechanisms for architecture representation. To facilitate traceability and the architecture development process, relationships between architecture (views) and other system knowledge such as system requirements are also identified and captured. The architecture continuum and the architecture representation techniques provide the foundation for an enterprise system architecture repository that is essential for effective enterprise-wide architecture practice.

The paper starts with a review of some related work. Then it presents an overview of architecture practice and identifies the key requirements for representation and management of the enterprise system architecture. The paper moves on to briefly describe a number of solutions for the enterprise system architecture management and then provides detailed discussion on the representation issues mainly associated with the architecture continuum-based approach. Finally, the paper goes through various applications of the enterprise system architecture in an effort to highlight and illustrate the practical value of architecture representation and management in large organisations.

2. Related Work

2.1. System Design and Design Documents

System architecture is one of key products generated from system analysis and design, which is represented by a set of interrelated views that collectively reflect the concerns and requirements of stakeholder community [El-Sakka99]. The selection of views to constitute the system architecture varies depending on the nature of business it intends to support and on the development approach used. At a project level of application development, a design process usually start with a high-level description of the system to be developed, including views of conceptual design and logical design. To provide detailed guidance for implementation, some master design plans (or views), which should be solution-oriented, may need to be produced, which can include *business view*, *application view, data view, information view, network view, component view, middleware view* and so on depending on the business nature of the system being developed. The result of choosing different views for system architecture in current practice leads to a situation where architecture representation may become undisciplined and lacking in consistency. The consequence of this kind of practice is various difficulties and problems facing both IT and business communities. In addition to the practice lacking disciplines, the diversity of development projects also contributes to the complex situation of system architecture production. For example, the system architecture (such as APIs view, component view, platform view and middleware view) concerned by a team working in integration of two existing systems differs from views generated when the systems were developed.

Serious efforts have been made in developing architectural approaches or frameworks [Bass98], [C4ISR97], [MSF99], [META99], [ODP–RM96] [Zachman96] for the purpose of introducing disciplines to architecture production. Because of the diversity of purposes to use architecture, different approaches or frameworks have been developed to meet different needs in different contexts of architecture production. In a large organisation, therefore, the reality is that there may be quite different sets of architecture products developed for various parts of its information systems that have been developed at different times by different developers. Over the life time these systems, their associated architecture products are likely either not managed properly or even outdated completely.

2.2. Architecture Description Languages (ADLs)

Realising the importance of capturing system architectures, quite a number of Architecture Description Languages have been developed over the years. But they are mostly still at the stage of research prototypes. Representative ADLs include Darwin [Magee95], Rapide [Luckham95], UniCon [Shaw95, Shaw96] and Wright [Allen97, Shaw96]. There have also been a number of surveys aimed at understanding and exploring the features that ADLs provide, including those by Clements [Clements96] and by Medvidovic and Taylor [Medvidovic97].

In general, ADLs are primarily designed for specifying the architectures of individual software systems. A key motivation is to provide formally based architecture representations/specifications rather than those in the traditional box-and-line approach. As such, most ADLs have formal semantics as their language features. The main architectural view that the ADLs support is the modular view showing components and their inter-connections. Support for other views are very limited. Many of the ADLs provide some limited capability for architecture analysis and system simulation, such as architecture consistency check in Wright, deadlock detection in Darwin and Wright, schedulability analysis in UniCon, and behaviour simulation in Rapide. Because of their limited support for other types of architectural views, there is little support for analysis of other quality properties such as performance, reliability and security.

While some ADLs like Wright support hierarchical architecture specification, the implicit assumption is that the components are to be developed in a uniform manner from scratch. No consideration is given to multi-system integration and interoperation. As such, ADLs do not concern themselves with the assumptions about the technical standards and infrastructure that the system is to operate with. Consequently, the current ADLs ability in dealing with system integration and systems of systems is very limited. Furthermore, most ADLs do not deal with the domain-oriented business/operation views because of their focus on software design. The purpose and operational capability are not addressed by ADLs. Finally, because of their focus on architecture design, the ADLs provide little support for traceability to other artifacts and products of the system development life cycle, such as requirements and test cases.

In general, current ADLs highlight the direction and potential for formal architecture representation and specification, but significant extensions are required for the representation and management of the enterprise system architecture.

2.3. Repository

The increasing demand of using architecture to support and improve IT practice has resulted directly in the increasing value of architecture. In order to better manage and use broadly this organisational asset, a variety of architecture repositories have been developed. Main capabilities of the repository are discussed in the next section. Technically speaking, design solutions for the architecture repository can be based on three different strategies.

I. Document management-based approach.

   - Architecture tool-based repository. This type of the repository is developed as part of the functions provided with architecture tools, such as COOL:Biz, Ptech Framework, Rationale Rose and System Architect.

   - File system-based repository. It is simply developed through using certain file management provided with sharable computing facilities, such Web, Lotus Notes, for effectively managing and publishing. Such repository has the ability of storing system architectures in a tool-independent manner.

II. Synchronized representation. To achieve advanced architecture practice and develop more added functions based on available knowledge of systems architecture, developing a unified or synchronized representation is an important step in allowing a large organisation to make system architecture at the enterprise level available and reachable to whoever needs to use it across the organisation.

III. Combination of I and II.

The main differences between these strategies lie in: 1) Strategy I uses simply the outcomes of systems analysis and design and maintains this sum of knowledge without doing anything apart from provision of better document management and accessibility; 2) Strategy II uses the outcomes captured by the Strategy I as a resource to derive only necessary knowledge, refines and represents it in a unified format no matter which set of views were originally used in design; 3) using Strategy I, one can use mainly the functions provided by the tool and the representation of views developed by individual designers with their specific interests, and there could be many repositories within one organisation if teams use different tools; 4) using Strategy II requires a specific process that is additional but can generate better representation of system knowledge which we note, may become necessary if legacy systems were developed without using a proper tool to generate architecture.

It is thus obvious that the enterprise system architecture can be reached with well-organised representation only through Strategy II. The enterprise system architecture can also be seen as the architecture of an enterprise-wide system. An architecture repository that is simply only a collection or storage of system development documents cannot be considered as the enterprise system architecture or systems architecture.

2.4. Challenges

Despite the difference among various development projects, architecture production at this level is relatively mature and familiar to people. There have been, however, difficulties and confusions in developing the enterprise system architecture for large organisations. In other words, it seems not clear to people when and how the system architecture should be developed at the enterprise level. If it is possible to use an architectural approach to guide the development of the whole enterprise system architecture and there is no need to consider what has been done before, this development principally is not much different from the conventional project level apart from the size.

In current practice of large organisations, the situation is quite different. First, an enterprise or "a bigger system" consists of a number of individual systems or "smaller systems" which usually were or will be developed separately in the context of conventional projects in which a number of architectural approaches can be chosen and used locally to deliver the architecture. Such architecture logically is part of the enterprise architecture. From the time perspective, in other words, we need to think whether the enterprise architecture could or should be developed before individual systems were developed. If developing the enterprise architecture was not done before the development of individual systems, there are questions whether the enterprise system architecture is still needed, how it should be developed and how it is related to the architectures of developed individual systems. To answer these questions, a context needs to be established where those architectures and processes/approaches can be related.

Architecture practice study [Chen99] discussed at a high level the relations among architecture products and processes. As shown in the Figure 1, bringing the architectures associated with individual subsystems back properly to the enterprise (systems) architecture repository is necessary in order to preserve and reuse the knowledge value captured. There are, however, unsolved research issues: 1) what knowledge needs to be derived from the architectures associated with subsystems; 2) how such knowledge should be represented and stored in the repository; 3) how the knowledge can be effectively retrieved for use and reuse purposes.

3. Architecture Practice and Requirements for Architecture Representation and Management

3.1. Systems Architecture in the Context of Architecture Practice

As briefly mentioned earlier, the system architecture production at the project level can be carried out in different ways with different outcomes. The investigation into various architectural approaches and frameworks shows that it is difficult, unsuitable and unnecessary to use a unified set of views for any architecture production. Dealing with different representations of systems architecture is inevitable in enterprise-wide architecture practice (See Figure 1) of large organisations. Studying difference and relations among different representations is vital for improving the current practice and ensuring that the system architecture can be reachable at the enterprise level.

Systems architecture being reached at the enterprise level means: 1) individual system knowledge can be captured to a certain degree; 2) the knowledge can meet most needs of future enterprise-wide development activities; 3) the knowledge can be managed and easily accessed; 4) the knowledge can evolve over time; 5) the knowledge can be used to support architecture analysis and inference across subsystems.

In some situations, the enterprise system architecture could be a blueprint for the organisation if it is reached before individual systems are developed. This architecture may meet some requirements in certain aspects of the five points to certain degrees. Currently, however, successfully developing the enterprise system architecture as a blueprint is a very rare case for large organisations.

In this paper, we consider the “as-is” enterprise system architecture mainly as a *current picture* of existing systems within the organisation. The five points mentioned above identify initial requirements for systems architecture representation in the context of architecture practice before detailed discussions on the requirements of architecture representation and management are given in the next subsection.

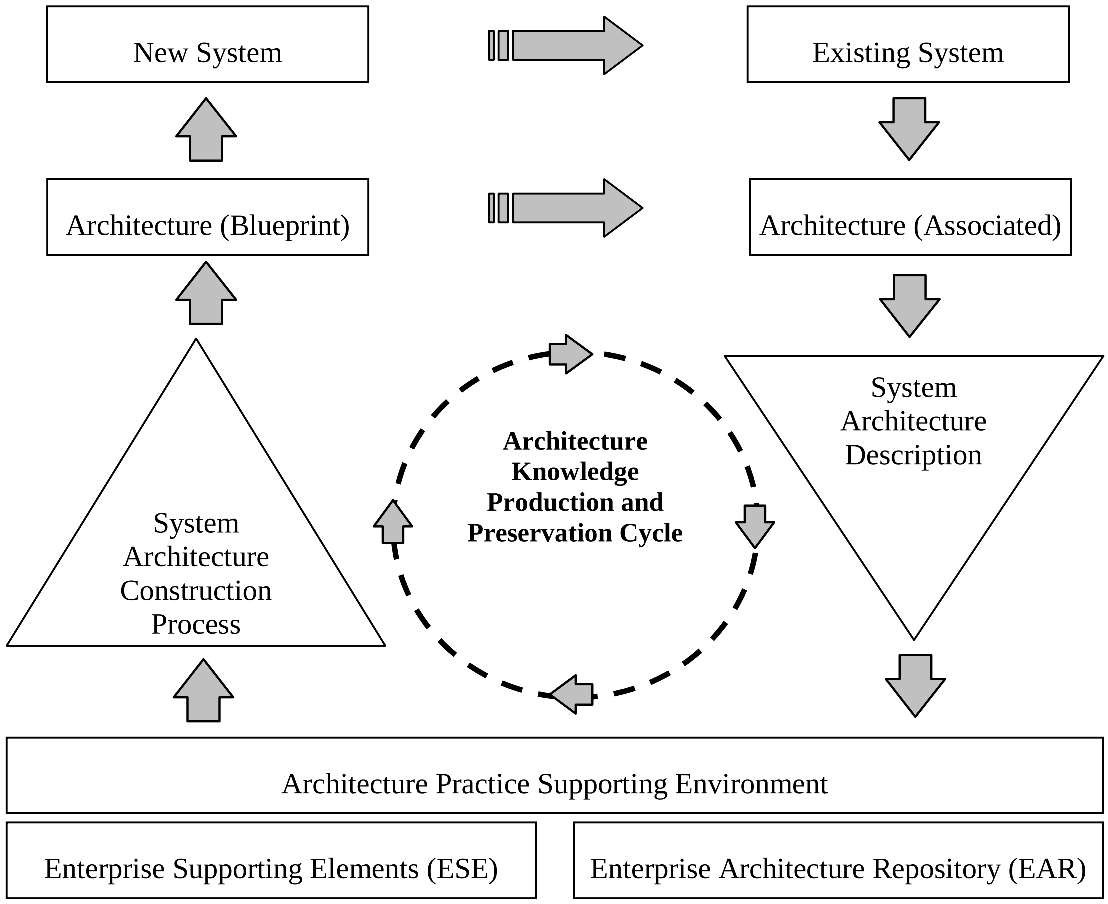

Figure 1. Recommended Architecture Practice Conceptual Model

3.2. Requirements for Architecture Representation and Management

As discussed above, the enterprise system architecture plays a key role in the recommended architecture practice, and needs to be captured and managed through an enterprise architecture repository. In this subsection, we highlight the major requirements for representation and management of the enterprise system architecture.

Management of Scale. According to the recommended architecture practice, an organisation's IT systems as a whole can be seen as a system of systems. The enterprise system architecture should recognise these component IT systems and their relationships. An organisation may have many IT systems in use, and these IT systems may interoperate and be organised into clusters hierarchically. The enterprise system architecture should logically reflect the organisation and interoperation of these component IT systems in an architectural sense. As such, it should readily deal with the scalability of enterprise system architecture and support the concept of “system of systems" at different levels of granularity.

Architectural Interests of Stakeholders. It has been widely recognised that many stakeholders have an interest in the architecture of a system, and the stakeholders' architectural perspectives are often different. For example, the business manager would be interested in how the system architecture (design) would support/facilitate the business operation, the project manager would be interested in how the system is to be subdivided so that different development teams will be allocated to these subsystems, and the IT support team for the organisation would be interested in the technical infrastructure that the system requires. In general, this suggests that the system architecture should be able to capture the different architectural perspectives, and architecture views are a natural mechanism to capture these perspectives.

Diversity of Architecture Specification/Representation. As mentioned earlier, there have been a number of architecture frameworks proposed to guide the development of system architectures. These frameworks have their own emphasis and corresponding architecture specification/representation schemes. In general, it is hard to mandate the use of a specific architecture framework in an organisation, because some systems or projects of the enterprise have characteristics that require a particular architecture framework while other systems require another architecture framework. Considering legacy systems, the scenario gets even more diverse. However, from the architecture practice viewpoint, a common system architecture representation scheme is required to facilitate cross-system analysis and reasoning in the context of the entire enterprise system architecture. Therefore, we adopt an approach with:

- a view-based uniform scheme for architecture representation, and
- a related enterprise architecture construction process with which the common architecture representation is extracted from the system's actual development artifacts/knowledge (see Figure 1).

Relationships between System Architecture and other System Development Artifacts/Products. Architecture design is only part of a system's development process. The system development process produces/involves many other artifacts, such as the system requirements document, program code and system testing cases. Many of these system artifacts are related to the system architecture. For example, the architecture design decisions reflected in the system architecture are to meet certain system requirements, and testing cases are devised to exercise certain architectural features. Furthermore, the system architecture in the repository is often a selected extraction or re-formulation of the actual system design, especially for legacy systems. In general, the relationships between the systems architecture captured in the repository and the other development artifacts/products should be specified and proper links should be created and maintained for traceability and rationale.

Repository capabilities. The enterprise architecture repository is a necessary tool for managing the system architectures in an enterprise. It should provide a range of capabilities to facilitate the development, maintenance and use of the system architectures. Some of these capabilities are:

- management of and navigation around, the enterprise system architecture in terms of the sub-system architectures at various levels;
- management of the system architectures in terms of architecture views and their relationships;
- architecture information retrieval;
- visualisation of architecture views;
- capability for architecture analysis.

In the rest of this paper, we discuss solutions for the representation and management of the enterprise system architecture, in particular at the enterprise level. They address the requirements identified above. As a case study, a scale-oriented continuum of system architectures is introduced in section 4.1 to deal with the granularity and scalability issues; architecture views and their relationships to other system knowledge are discussed in section 5. Section 5 also discusses interface characterisation of (component) systems that supports both management of granularity and scalability and definition of architectural views.

4. Scale Management from Systems to the Enterprise

When considering the issues of systems architecture from the viewpoint of the enterprise (or systems of systems) rather than from a single system/project viewpoint, scale management becomes a critical issue in strategically determining enterprise-wide solutions for the representation and management of systems architecture. Because of difference in the development history, there may be different preference in selecting architecture representation and management solutions. In general, there are a number of factors, which need to be considered when choosing management solutions.

- **Types of Systems**. Information systems can be classified into different classes according to certain attributes, for example, the nature of system distribution (specific local application, generic local application, collectively distributed use and so on);

- **Styles of Information System Architecture**. Systems can be designed in different styles (architectures), such as Monolithic or multi-tier client-server.

- **Use of the Enterprise System Architecture**. Different requirements in using the enterprise system architecture can be met by using different management solutions.

4.1. A Continuum of Systems/Architectures

To be of practical use in managing business operation, the enterprise system architecture of an organisation should not be a monolithic representation of its IT systems' structure. Rather, it should be organised to reflect the organisation's static and dynamic structure. In fact, the enterprise system architecture is a continuum of architectures, corresponding to the organisational entities at the various levels of granularity, including the enterprise, enterprise units, individual information systems, and so on . This is because IT is deployed to facilitate all aspects of the organisation's business activities. At the fine-grained end of this architecture continuum are the various software and information systems facilitating specific business activities. For example, these systems could be the accounting and personnel systems of an organisation. The architectures of these systems serve as the basis for their understanding, operation and evolution in their respective business contexts.

As we have seen over the last few decades, the individual IT systems have been integrated into larger systems supporting larger areas of business operation to facilitate new business needs and challenges. For example, the accounting and personnel systems can be integrated into an organisational administration system to facilitate effective administration by providing direct IT support for the business interaction between the accounting and personnel activities. Similarly, specific production systems can be integrated to allow IT-assisted interaction between the individual systems and therefore deliver increased production capabilities for the organisation. In the context of the larger systems, the smaller systems still maintain their own capability and architectures, while at the same time becoming part of the larger systems. The larger systems formed through integration have their own architectures the smaller systems being their components, which provide the basis for the understanding, management and evolution of the larger systems.

The scale of systems continues to increase by further integrating the larger systems to obtain even greater organisational business advantage, until the point where the entire organisation forms an enterprise-wide system. The retention and management of the system architectures at all levels of granularity is the key to deal with the complexity in understanding, managing and evolving this enterprise-wide system. It is the architectural boundary (or interfaces) between the component systems and their enclosing system (i.e., the container) that enables this modularity (see the next section). The system architectures at the various levels of the enterprise form the *architecture continuum*. They range from architectures for specific software systems and subsystems, through medium-grained section/division systems, to the enterprise-wide system. In fact, this continuum continues if we consider corporate coalitions and the interoperation between enterprises, up to the global system encompassing all the enterprise and organisation systems around the world. However, the most practical and immediate focus would be the architectures of a single enterprise (see Figure 2 below).

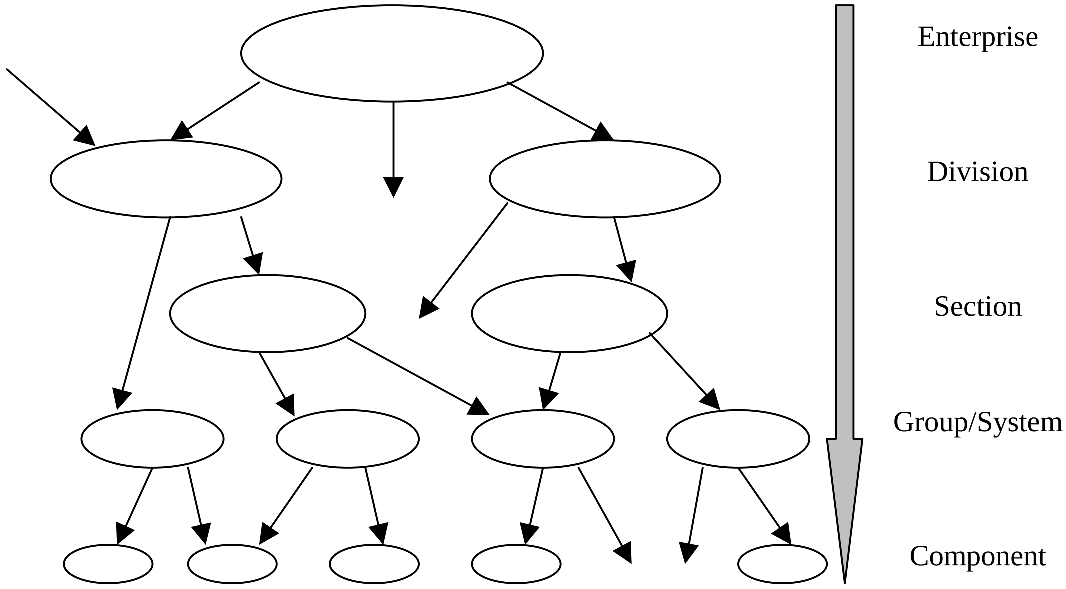

Figure 2. A Continuum of System Architectures

In considering system granularity and interrelationships, it should be emphasized that a system can be (or can be seen as) a component of one or more larger systems, which are even at different levels of the continuum. Furthermore, when considering the architecture of an enterprise, we do not (need to) restrict ourselves to *implemented* IT systems. We may include systems that may still have parts *to be implemented*. As such, we can use the enterprise system architecture as a basis for planning and evolving the enterprise-wide system.

This view of architecture/system continuum directly addresses and naturally facilitates system integration and evolution through the ready availability of system contexts (provided by enclosing systems) and system interfaces (of component systems). When we talk about the integration or interoperation of system A (e.g., the accounting system) and system B (e.g., the personnel system), for example, it is done in the context of an (existing or new) enclosing system C (e.g., the organisational administration system). The business operations of C (in terms of A's and B's business capability), the system interactions between A and B, and the technical interoperation of A and B are considered as architectural issues of C and are described in the architecture (views) of C. This task is made much easier with the clear understanding of A's and B's capabilities that are specified in their interface characterisation. When we talk about the evolution or change of the component system A, it is in the context of its enclosing system C so that the business need, impact (benefit and risk), and degree of change are clearly identifiable for C.

This component based organisation approach to architectures and systems, i.e., *the architecture continuum*, provides the necessary means for structuring, relating and managing the complexity of the enterprise system architecture.

4.2. Federated View-based Management

For certain organisations within which, IT development has been globally managed reasonably well and developing specific federated views (such as data view or application view) is possible, a federated view-based management solution can be considered as depicted in Figure 3. This solution may be more suitable for the organisation that has used certain well-designed methodologies to guide development of some federated views. Developing federated views for large organisations could be a very difficult task if the relations among different views need to be captured and explored in depth. Addressing properly the relations between local views and the federated views is yet to be investigated.

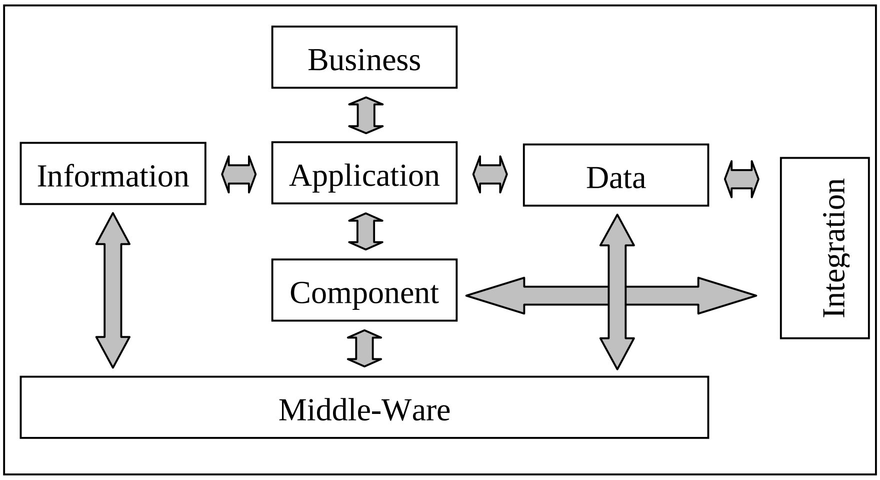

Figure 3. Federated View-based Management

4.3. Meta-model-based Management

The scale management can also be achieved through flexibly using the concept of meta-models. Figure 4 shows how a system architecture is decomposed. Using the meta-model-based management, one can indeed create meta-architecture (models) against views, models or even diagrams according to purposes and interests of management. For example, the C4ISR Core Architecture Data Model (CADM) developed by US DoD is a typical case using the meta-model-based management, which uses the concept in a much broader context. . To a certain degree, the two approaches described earlier are special cases of the meta-model-based management in which two meta-models are created at the system level and the view level respectively.

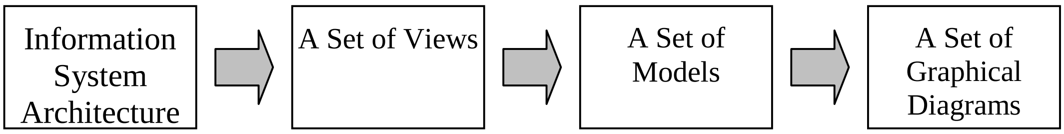

Figure 4. Decomposition of Information System Architecture Representation

In summary, the system architecture management from the systems to the enterprise is a complicated issue and yet to be studied systematically. Like the architecture practice [Chen99], choosing solutions for enterprise system architecture management requires the organisation to understand its specific needs.

5. Architecture Representation

To be able to understand, manage, plan and improve an enterprise's business through its architecture, the system architectures of the enterprise need to be captured and managed. As such, we need to represent individual system architectures and address the relationships between the architectures across the enterprise. While having their own peculiarity and capability, the system architectures at the various levels of granularity share much commonality in terms of their management and organisation. In this paper, our discussion continues mainly along with the approach of architecture continuum.

For a given system, we first need to consider its architecture in terms of its component systems from the viewpoints of business operation, system implementation and technical infrastructure [C4ISR97], *architectural views.* These architectural views are usually formulated based on the system knowledge documented in other sources. For example, the requirement documents for the system will form the basis for the business operation view. To provide the necessary rationale and facilitate traceability, the system knowledge as embodied in various documents need to be linked to the systems architecture (views).

In the context of the enterprise system architecture, the system components are considered in the form of abstraction. That is, only those characteristics of the component systems that are relevant to the enclosing systems' architectures are observable and considered, and the details of the component systems are considered neither relevant at this level and nor observable although these details may be considered as part of the internal architectures of the component systems. Therefore, relative to the enclosing systems' architectures, the component systems should be represented in terms of their observable characteristics, *(external) system interface.* This interface should capture the (observable) operational, system and technical characteristics of the component system concerned for use in defining the enclosing systems' architectures. In general, for a non-atomic system, internally it has an architecture in terms of lower level component systems, and externally it has an interface characterisation for use by possible enclosing systems. An atomic system only has an external interface characterisation without an internal architecture description. With the availability of the interface characterisation for component systems, we can analyse and reason about whether the component systems in a given architecture can achieve the enclosing system's operational objectives, *architecture analysis.*

In the following subsections, we discuss in detail the representation issues of architecture (views), external interface characterisation of systems (components), and architecture-knowledge links. Architecture analysis is discussed in a later section.

5.1. Architectural Views

A system's architecture can be viewed from different perspectives by various stakeholders, and therefore resulting in a set of architectural views. It is this set of architectural views that collectively define the system architecture. The perspectives of the stakeholders can be broad or narrow. Following the C4ISR architecture framework [C4ISR97], for example, we have three broad *architectural views*: operational/business, system, and technical. The operational view is about the enterprise's business perspective. The system view is about the assembly of the system (to support/achieve business functions). The technical view is about the rules, standards and enabling technologies that underlie the system's functioning.

Under these broad perspectives, there are narrower perspectives. For example, there are perspectives about the business process and data flow (under the operational view), and perspectives about the system's interaction and data flow as well as how system services relate to business activities (under the system view). These narrower/finer perspectives result in specific *architectural structures or products* under the relevant broad views.

We believe that there is a *compositional* structure that underlies all the other views and structures in a system's architecture. The compositional structure is about the logical organisation of the system in terms of its component systems. Its primary purpose is to identify the constituent component systems, as described in the architecture continuum, and possibly their logical interactions. The primary units of components in other structures may be different from these compositional units, but they are map-able to them. For example, one or more functional tasks in a data flow structure are performed or hosted by a specific compositional unit.

As expected, the various views and structures of a system's architecture are related to each other, while serving different architectural purposes. These relationships should be captured. Regarding such relationships, there may be further constraints about the entities in the related structures. These constraints should be made explicit so that the consistency between the structures (architecture consistency) is checkable and even enforceable.

Because of the diversity in selecting views when individual systems are developed, whether view standardisation/normalisation is needed becomes an issue when developing the enterprise system architecture. It is necessary only if sets of views used at the system level are different and developing expected functions of architecture analysis and inference requires a consistent representation. This standarisation/normalisation can ensure a unified format to be used in capturing systems architecture across the enterprise.

5.2. Interface Characterisation of (Component) Systems

When we define the architecture views for a system, on the one hand we rely on the capability of the component systems to define the capability of the enclosing system, and on the other hand we do not want to know the unnecessary internal details of the component systems. As argued earlier, this gives rise to the need for defining an external interface for each component system that only contains the necessary *observable* capability of the system. It is through this interface definition mechanism that we can manage the complexity of the enterprise system architecture and deal with system integration, interoperation and evolution.

The interface characterisation of a system should in general deal with the same aspects as the system architecture, but considering only its observable capabilities and properties rather than its internal architectural structures. Therefore, the system interface should include characteristics of business/operation capability, system interaction mechanisms and dependent technical standards and infrastructural support. Within each of these broad aspects, we need to consider the more specific features corresponding to the finer aspects. For example, we need to characterise the observable business functions and their performance measures for the business aspect of a system, the external interaction ports/roles and protocols for its system aspect, and the operating system, network and middleware standards and products it uses for its technical aspect. We plan to extend our current model for software component interface specification [Han98] to meet the needs of system interface characterisation in the context of architecture practice.

5.3. Relationship between Architecture (Views) and System Knowledge

So far in this paper, we have discussed the representation and management of the enterprise system architecture. As discussed earlier, there are different knowledge sources, which an organisation can use to reach the enterprise system architecture. For new systems to be developed, their design and analysis documents are naturally the main source. For existing systems in an enterprise, in most cases there was not an enterprise architecture defined and the existing systems likely do not have architectures explicitly specified. At best, the existing systems have some requirements analysis, system design and implementation documents. As part of introducing architecture practice into an enterprise, a natural and essential step is to find out and capture the architectures of existing systems as part of the enterprise system architecture so that an overall understanding about the enterprise can be achieved before planning for the future.

For both new and existing systems, there are various relationships between their system architectures and other system development artifacts such as those about system requirements, system design and implementation, and system validation and verification. These relationships reflect the traceability of the system development process and should be captured. For example, the relationships between system requirements and system architecture show how the requirements are addressed by the architecture design. In the reverse engineering of an existing system's architecture from available system development documentation, in particular, the relationship and traceability between the system architecture and the existing documentation should be captured to provide rationale and context for the extracted system architecture. Ideally, the various system development documents, collectively representing the *system knowledge,* are captured in an appropriate electronic form according to their respective models of representation. Then, the relationships between the system architecture and the system knowledge can be readily represented and captured, even at reasonably fine-grained level and in the same (central or distributed) architecture repository which the enterprise system architecture is hopefully part of. In order to achieve traceability between the architecture and other system knowledge, establishing proper links from the enterprise system architecture to other system knowledge resources is necessary and can be implemented as part of the repository functions.

An Example. Recently, we have developed a practical model for requirements engineering, that includes an information model for capturing requirements, a process to guide the requirements elicitation activities using the information model, and the corresponding tool support [NATS99]. In this model, the requirements for a system are captured in terms of the goals that the system stakeholders want the system to meet. The initial high level goals are refined/elaborated by more concrete low-level goals, and if necessary these low-level goals are further refined recursively into even low-level goals until the goals are concrete enough to be directly operationalised through specific services and qualities-of-service expected of the system. It is those services and qualities-of-service representing the concrete system requirements that provide the natural point of connection for relating to the designed capabilities of the system as captured in the system architecture. Through these relationships between the system requirements and system architecture, we can reason about whether the system (architecture design) meets the requirements for the system. We envisage a representation of system capabilities in terms of services and qualities-of-service in the system's interface characterisation (and the system's architecture should ensure the delivery of such capabilities). In fact, our current model for rich interface specification of software components uses services and qualities-of-service for component interface specification [Han98]. This model will be extended for interface characterisation of general systems in the context of architecture representation.

Another key feature of the requirements engineering model is its clear distinction between *requirements (goals)* for the system to achieve and the *assumptions/constraints* about the system's operating environment that the system has to respect. These assumptions can be about other systems or users that the system interacts with in terms of operational behaviour and interaction specifics, and about the standards that the system conforms to. This also corresponds well to relationship between a component system and its enclosing systems in the architecture continuum. The enclosing systems provide the operating environments for the component system, and their architectural settings should be consistent with the component system's assumptions about its operating environment. When the assumptions of a system are made explicit in its interface characterisation, therefore, they can be checked against its enclosing systems' architectural settings for consistency. From the viewpoint of the relationship between the requirements engineering model and the architecture continuum, there is a natural correspondence between the operating environments of a system and its enclosing systems, between the assumptions about the operation environments and the conditions provided by the enclosing systems.

The above discussion gives an example of how fine-grained linking can be achieved and captured between system architecture and system requirements (part of system knowledge) when an appropriate requirement model is available. It is our intention to closely integrate the requirements engineering model and the architecture continuum so that we can easily check and reason about how the required functionality and qualities can be met by the system architecture design in an enterprise context.

It must be noted that a single architecture view provides system knowledge only in a specific aspect associated with a particular system (which can be either an existing one or "to-be"); the composition of system architecture in reality varies depending on how the concept is used in context. However, any piece of system knowledge is not necessarily an architecture view. The enterprise system architecture is expected to be well-organised and well-represented systems knowledge through a set of selected views, which should cover necessary knowledge of systems across the enterprise and provide effective links to other sources of systems knowledge.

6. Applications

This section explores the possible applications of the enterprise system architecture as represented through the architecture continuum-based approach. These application scenarios illustrate some of the benefits of adopting architecture practice and using the above architecture representation approach.

Architecture Analysis. Two basic aspects of the architecture representation approach are interface characterisation of component systems and architecture views specification of the enclosing system. With the availability of these two aspects, we can reason about the relationship between the enclosing system's capability and the component systems' capability through the use of the architecture. Such architecture analysis is part of other applications described in this section, including business/system understanding, planning, integration and evolution.

The interface of a component system presents its observable characteristics, regarding the various aspects of operation, system and technical standards and infrastructure. The architecture views specification of the enclosing system is about how the component systems are inter-connected with each other, again from the various perspectives. For a given perspective or aspect, we can reason about how a capability of the enclosing system is achieved by a particular architectural connection of the component systems [Bass98]. It is dependent on the capabilities of the component systems and those of the architecture. For example, we may require that the performance of the enclosing system in a particular use scenario be 2 seconds. Through architectural analysis, we can evaluate whether or not this requirement can be met by its architectural design. First, from the architecture, we can work out the services required of the component systems in the given scenarios. Using the performance characteristics of these services in the component systems' interface and the performance characteristics (delays) of the architectural connections, we can calculate the actual performance of the use scenario for the enclosing system and compare it with the required performance. If the requirement is not met, experiments can be carried out to change the architecture or replace some of the component systems. This example demonstrates how valuable it is to be able to perform system analysis at architectural level to aid the system design process. We note that the performance characteristics of the component systems and the architecture may need to be obtained by further compositional analysis or through testing in specified environments.

Business/System Uunderstanding. The enterprise system architecture provides a tangible basis for better understanding of the enterprise's business operation. One consequence of this is better business management. For example, the enterprise business view captured shows how a key business function is actually realised inside the organisation in terms of the functions of its subdivisions and their dependency/relationships. Suppose that due to market opportunity, there is a need to improve the business function for increased productivity. Then the relevant business areas or processes can be examined through the associated architecture description so that the key points or bottleneck are identified. The improvement prospects and the need for additional resources can be analysed and determined using relevant architectural analysis techniques.

Business/System Planning. The enterprise system architecture also provides a valuable tool for planning various business activities, from local improvement to strategic directions. Suppose that the organisation plans to move into a new but related area of operation. A natural question is how the current business practice can be adjusted to fit into this area, in other words, which parts of existing operation can be reused and extended, what the scale and cost of such extension is, what new capabilities and component systems need to be added, and what their costs are. The existing enterprise system architecture can be analysed to help answer these questions in a more realistic and accurate manner. If the analysis shows the need for a dramatic change to the business operation and its resources, maybe a staged plan is more appropriate under a single architectural vision.

System Integration and Restructuring. Due to change in business operations such as mergers and diversification, systems may need to be integrated or restructured. For example, a new division or initiative may be formed within an organisation by drawing on the operations and expertise in a number of existing divisions. This leads to a scenario of integrating existing division architectures to form a new, special purpose division architecture within the context of the overall enterprise architecture, and with or without changing the existing division architectures. This involves analysis of the architectures at the enterprise and division levels, formulation of new architectures and restructuring of some existing architectures with system reuse and sharing. It is the availability of the enterprise system architecture that makes system integration and restructuring a more predictable process. Necessary analysis can be done to assess its impact on the enterprise before its actual implementation.

System Evolution. As business practice improves and technology advances, there is a continuing need to evolve an organisation's enterprise system architecture. This normally involves the replacement of old systems with new systems, addition of new systems, and decommissioning of certain old systems. All these systems are part of the organisation, and their architectures are part of the enterprise system architecture. The availability of the enterprise system architecture and the characterisation of its component systems provide the necessary basis for assessing the impact of such evolutionary changes. For example, we can assess whether a new system will fulfil all the operational capabilities of the old system to be replaced, whether the new system can be readily integrated with the neighbouring systems to achieve the operational capabilities, and whether the technical standards and infrastructure of the new system are compatible and can interoperate with those of existing systems. In addition, we can analyse the system-wide impact of any difference in capability between the new and old systems. This shows that being able to facilitate impact analysis of system evolution is another key benefit from using systems architecture at the enterprise level.

7. Conclusions

Systems architecture representation and management at the enterprise level discussed in this paper is a challenging issue for large organisations. Although most large organisations are not yet at the stage to formally develop the enterprise system architecture, their current practices and products form a basis on which such an important organisation asset can be developed. A number of management solutions were briefly described. The focus has centred on initial studies of the architecture continuum-based approach and its associated representation issues.

8. Acknowledgement

This study is conducted within a joint research project funded by the Defence Science & Technology Organisation, Department of Defence, Australia. The authors would also like to thank Thea Clark for her involvement at the early stage of this study.

9. References

[Allen97] R. Allen and D. Garlan. A formal basis for architecture connection. *ACM Transactions on Software Engineering and Methodology*, 6(3):213-249, July 1997.

[Bass98] L. Bass, P. Clements and R. Kazman. *Software Architecture in Practice*. Addison Wesley, Reading, MA, USA, 1998.

[C4ISR97] The C4ISR Architecture Working Group. C4ISR Architecture Framework (Version 2.0). US Department of Defense, December 1997.

[Chen99] Pin Chen, Abdel el-Sakka and Jennie Clothier. Architecture Practice Study for C4ISR Systems Development, In *Proceedings of the Command & Control Technology & Research Symposium*, 757-772, Naval War College, Newport, RI, USA, June, 1999.

[Clements96] P. Clements. A survey of architecture description languages. In *Proceedings of the 8th International Workshop on Software Specification and Design*, Paderborn, Germany, March 1996.

[El-Sakka *et al.*, 1999] Abdel El-Sakka, Pin Chen and Jennie Clothier, *Knowledge Acquisition Improvement through Enterprise Architecture Practice*, Proceedings of the Software Engineering Australia 1999 (SEA'99), Canberra, April, 1999.

[Han98] J. Han. A comprehensive interface definition framework for software components. In *Proceedings of the 1998 Asia-Pacific Software Engineering Conference*, pages 110--117, Taipei, Taiwan, December 1998.

[Luckham95] D. Luckham, J.J. Kenny, *et al*. Specification and analysis of system architecture using Rapide. *IEEE Transactions on Software Engineering*, 21(4): 336-355, April 1995.

[Magee95] J. Magee, N. Dulay, S. Eisenbach, and J. Kramer. Specifying distributed software architectures. In *Proceedings of the 5th European Software Engineering Conference*, pages 137--153, Barcelona, Spain, September 1995.

[Medvidovic97] N. Medvidovic and R. Taylor. A framework for classifying and comparing architecture description languages. In *Proceedings of the 6th European Software Engineering Conference and 5th ACM SIGSOFT Symposium on the Foundations of Software Engineering (LNCS -1301)*, September 1997, Zurich, Switzerland.

[META99] Meta Group, *Enterprise Architecture Strategies (EAS)*, Meta Delta, 31 March 1999.

[Microsoft99] Microsoft, *Microsoft Solutions Framework*, *http://www.microsoft.com/msf/*, 1999.

[ODP-RM96] Information Standards Organisation (ISO), Open Distributed Processing – Reference Model (ODP-RM), ISO/IEC DIS 10746-2, and ISO/IEC DIS 10746-3, 1996.

[NATS99] The NATS Project Team. A Core Information Model for Requirements Engineering. Department of Computer Science, University College London. June 1999.

[Shaw95] M. Shaw, R. DeLine, *et al*. Abstractions for software architecture and tools to support them. *IEEE Transactions on Software Engineering*, 21(4): 314-335, April 1995.

[Shaw96] M. Shaw and D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline.* Prentice Hall, Upper Saddle River, NJ, USA, 1996.

[Zachman96] J. Zachman, *Enterprise Architecture: The Issue of the Century*, http://www.zifa.com/zifajz01.htm., 1996.