

Chapter 11

Flow Handling

A flow handler ties together all the work detailed in the rest of this thesis, providing a tool to manage the actual design process for individual projects. Earlier chapters describe methods of modelling: actors in a project; design tools used; the integrated data model; relationships between data in various schemas; and desired flow of control in a particular project. Chapter 10 describes a system which will move data between model instances of a particular building, and Chapter 8 refers to work on automating the invocation of design tools. The final step is to put the process control tool in place in order to manage the allowable design tasks at any particular time, and to ensure the design is accomplished according to the flows defined by the project manager. This chapter describes a particular implementation of a flow handler which provides managerial control to a project manager and task level control to the actors working on a particular project.

11.1 Requirements for Flow Handling

A flow handling system must provide two distinct categories of functionality. The first is for a project manager who oversees the running of a particular design project and must ensure a timely completion, as well as a final design which meets the initial design criteria. The second is for the actors, who need to see what design functions remain to be completed in their design roles and when they can pass the design on to another actor. The different requirements of the two main classes of user of the integrated design system are detailed in the following subsections.

11.1.1 Project manager requirements

The managerial requirements of a project manager are detailed below:

Project overview: the project manager must be able to determine the current state of the project; which stages of the project have been completed; which stages are still being worked upon; who is currently working on various stages of the project; and what design tasks are still to be completed.

Track design path: in most projects it is imperative to be able to determine who worked on what portion of the design, and to see when particular design tasks were completed.

Status: the project manager must be able to determine the current status of the project window; to see who is working, and on which parts of the design.

Dynamic project flow modification: the project manager must be able to intercede in the flow of a project if it is clear that new resources are needed to complete the project. The project manager must be able to move an actor to a new task; stop a task before its normal completion; stop an actor's design role work; and start a new actor working on a new design role, perhaps concurrently with other actors.

Dynamic flow diagram modification: if the project manager determines that a new path is required in the flow diagram (e.g., to force a particular task to be performed, or to add new functions into the design process, or to add paths requested by actors in the project) then modifications should be able to be made to the flow diagram, and they should be reflected in the running project.

11.1.2 Actor requirements

The requirements of actors are more closely related to their design roles, as detailed below:

Design role overview: actors must be able to manage the design functions required to complete their design roles, and they must be able to determine what is required to complete their design role.

Formal handover between actors: the completion of a design role and handover point between actors needs to be documented, both for the time of handover and the state of the project at the time of the handover.

Determination of allowable functions: the possible design functions for each actor are calculable based on the other actors currently working on the project and the design functions they are working on. Only currently invocable design functions should be allowed to start up, and the status of all currently available, or stalled, design functions should be re-evaluated upon the completion of any design function by any actor in the project.

This provides a level of control suitable for a strict project management regime. Actors have control over the work they perform towards their design roles, but limited to design functions specified in the project window definition. If an actor requires modifications to the project window this must be negotiated with the project manager. This ensures that specified design functions are

performed and there is no way to work around the requirements implicit in the workflow specification.

11.2 The Exchange Executive

The Exchange Executive (ExEx) provides the implementation of the flow of control definitions of Chapter 7 and implements the requirements of a project manager and actors as defined in Section 11.1. The main component of the ExEx is a simulator for the CombiNet specification, determining the design functions which could be next completed in a project. As well as simulating the CombiNet, the ExEx accesses the schema definitions of the various design tools and actors to determine possible restrictions between the design tools when invoked concurrently. Finally, to present the information to the various users there are separate user interfaces tailored for project managers and actors. The working of all of these components is detailed in the following subsections.

11.2.1 Simulation of flow of control

To describe the simulated flow of control in the ExEx, the flow through a net which comprises simple places and transitions will be described (as all CombiNets can be rewritten in this form). In this type of net there are four components: places, transitions, tokens, and connecting lines. The implementational semantics of these four components are described below:

Token: a token represents the current state of an individual workflow. When shown in a place it represents an actor working on the design function represented by that place. As the workflow progresses from design function to design function it may pass across actor boundaries and thereby denote a handover phase between actors. There may be several tokens in a simulated CombiNet representing different concurrent workflows. While a token in a place represents an actor performing a particular design function, an actor may be involved in several tasks at any one time. Each of these tasks would be defined by separate tokens in the CombiNet representing the particular workflow associated with the tasks being undertaken.

Place: a place represents a design function, which may or may not be realised through the use of a design tool. If the place represents a design tool, then the movement of a token into the place signifies the invocation of the design tool with data from the IDM, as described in its input schema definition. The movement of a token out of a place represents the termination of the design tool and the transfer of its resultant data (if any) through to the IDM. Between these two events the actor is free to perform whatever actions they wish with the design tool (as long as the action is within the scope of the area of responsibility of the actor, as defined by the actor's schemas).

Transition: a transition has no implementational semantics in the ExEx. It is purely a representational notation to describe a choice point between several places. Transitions are of most use when accessed by several places, reducing the number of lines in a diagram. The whole CombiNet definition could be redrawn without transitions and retain the same semantics, but many more lines could be required to represent flows between places.

Connecting lines: lines with an arrow at one end provide a path for the flow of control to proceed along. Each line represents a possible pathway, either from a place to a transition, or from a transition to a new place.

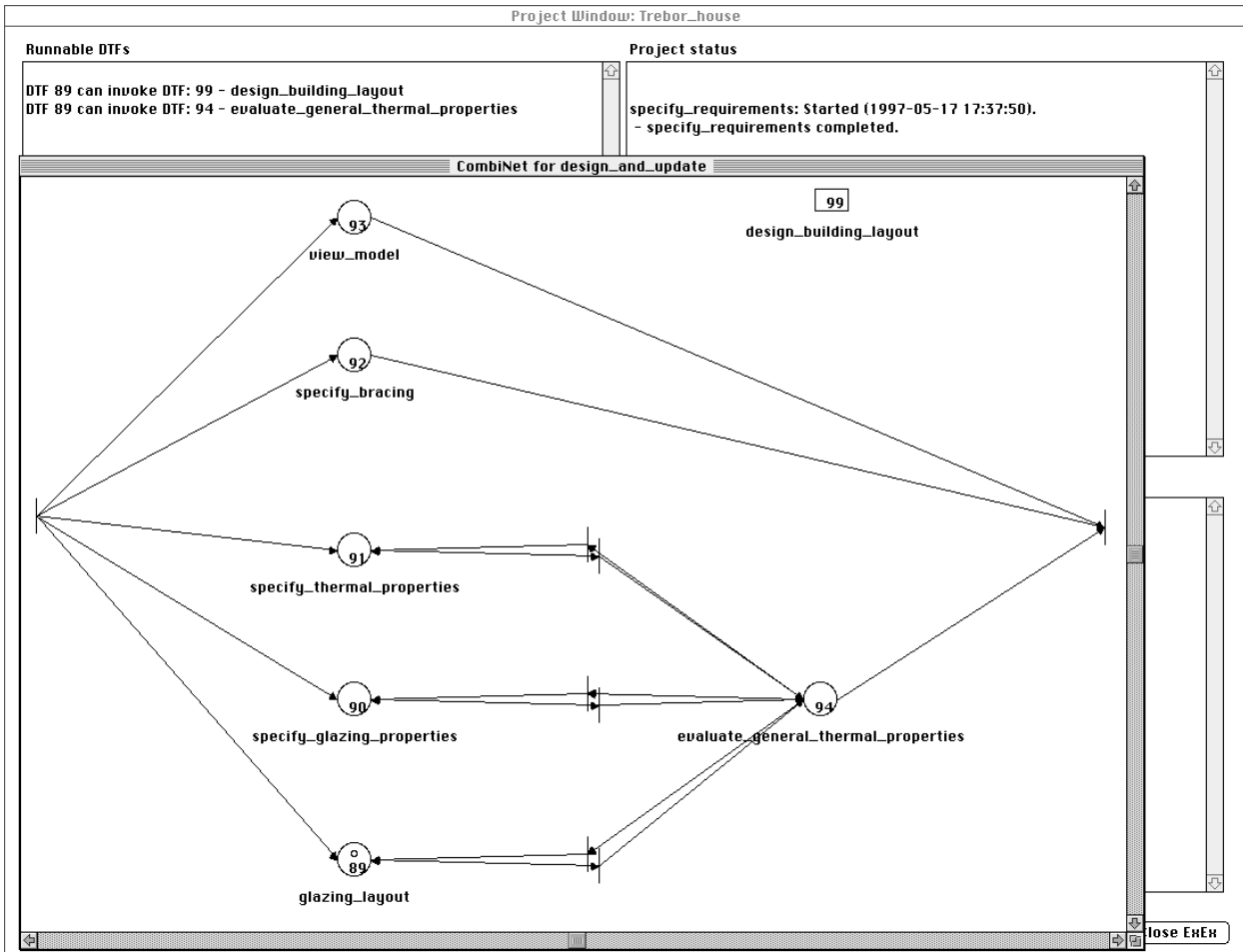


Figure 11.1 Calculating potential design functions from a CombiNet

The simulation of flow in a net comprising the above elements operates in the following manner:

- Identify each token in a place in the CombiNet.
- For each token in a place identify all transitions that can be exited to from that place.
- For each of these transitions identify the set of places which can be invoked from the transition.
- Determine the set of places which are candidates to be invoked from the current place by taking the union of all of the sets of places for each transition .

For example, Figure 11.1 shows a token (the small open circle) in the place representing the *glazing_layout* function. Upon completion of this function there is a single transition which can be exited to. This transition leads to the *evaluate_general_thermal_properties* place, however, there is

also a global place that can be invoked, called *design_building_layout*. The total set of design functions which can be invoked from this point is shown in the Runnable DTFs box of Figure 11.1.

There is a separate set of candidate places maintained by each token, representing possible continuations of the workflow, in the CombiNet. The set of candidates for each token is reduced by taking into account the following conditions:

Stalled place: this is derived from Constraint 1 in Chapter 7: if there is a currently running design function whose output could modify the input of the candidate place then it is labelled as stalled and removed from the list of candidates (e.g., Figure 11.2 shows the DTF *evaluate_general_thermal_properties* of Figure 11.1 running. The output of that DTF impacts on the *evaluate_thermal_code_compliance* DTF's input and for the actor shown in Figure 11.2, forces it to become stalled, as shown in the bottom right sub-window).

Insufficient data: if the data in the IDM is not sufficient to invoke the candidate design function then it is removed from the list of candidates.

Failed start: if there was enough data to start the tool but the tool would not start when invoked (usually through a failed constraint) then it is removed from the list of candidates.

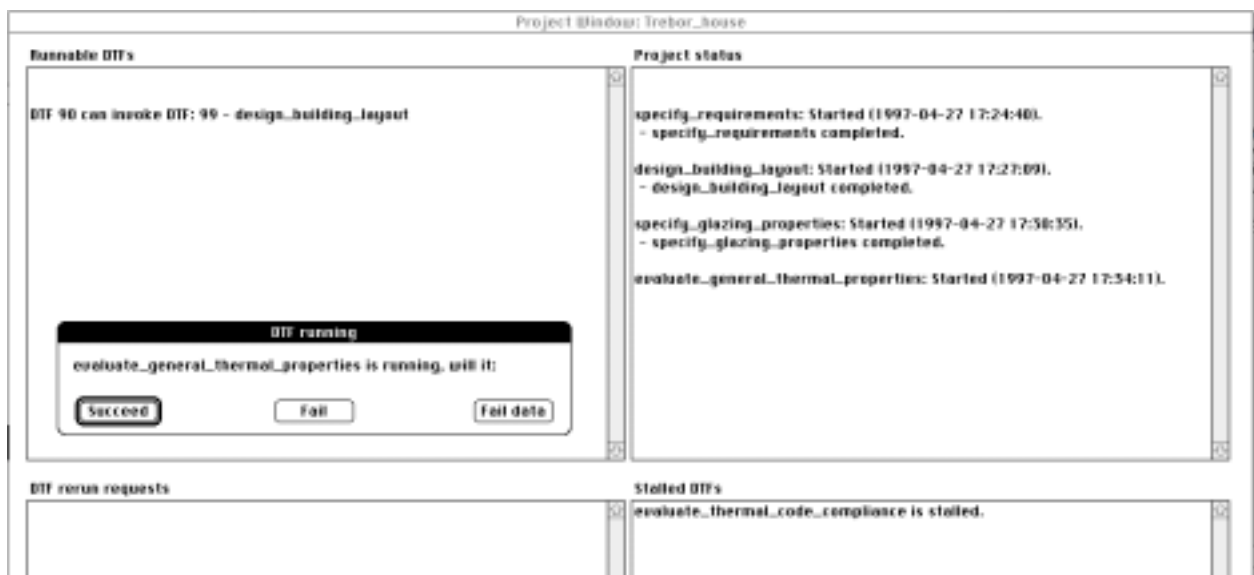


Figure 11.2 Multiple actors causing select design functions to become stalled

The set of candidates that can be run at any time is re-evaluated every time a running design function terminates or when a new design function is invoked. This re-evaluation may: add new places to the list that can be invoked; remove places from the list of candidates; or change the status of places whose status is un-runnable due to the conditions listed above.

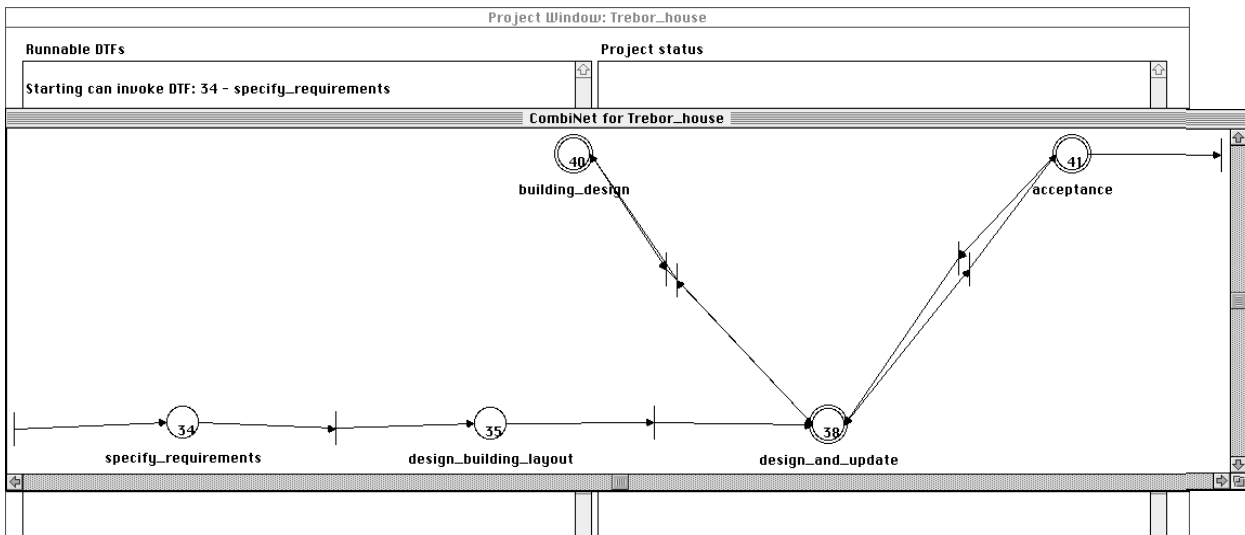


Figure 11.3 Initial CombiNet in a project window

We now extend the simulation semantics to cover several special states, and icons. Their flow conditions are described below:

Start of the project window: the start of the project window is determined by looking at the top level CombiNet (referenced by the user and function modelling diagram, see Chapter 7 and Figure 7.3). The start transition of this CombiNet is identified and the places reachable from it collated to become the initial candidates for the project window (e.g., Figure 11.3 shows the starting point for the examples used in Chapter 7. In this example there is only one place that may be invoked to start the project window, namely *specify_requirements*).

End of the project window: the end of the project window is found by identifying the end transitions in the top level CombiNet. Whenever one of these transitions is visible from a place with a token in it, there is the possibility of terminating the project window. In practice the completion of a project window is when the last token in the project window reaches the end transition (or the last token is terminated by the project manager) (e.g., Figure 11.3 shows a single end transition which can only be reached from the *acceptance* CombiNet).

Actor's design role: when a token passes from one place to another, where the specified actors for the two places is different, this signifies the end of one actor's design role and the start of a new actor's design role. At this point the actor involved in the workflow represented by the token changes. This event is notified to the project manager who is responsible for permitting the new actor to start their design role (or who may terminate the token). At the end of an actor's design role the actor may aggregate the work performed into an aggregate transaction, rather than single transactions for every design function they performed (as previously described in Sections 10.1 and 10.2). If this option is chosen, all their work since starting the current design role is collated under a single label in an aggregate transaction (e.g., Figure 11.3 has an actor changeover between *specify_requirements* and the *design_building_layout* design functions. At this point the *client* has completed the

requirement specification role and the *architect* starts the building design role. The whole requirement specification role could be recorded as a single aggregate transaction). At a hand-over point which offers the possibility of several actors working concurrently, it is up to the project manager to instigate any required concurrency, as is current practice in building design.

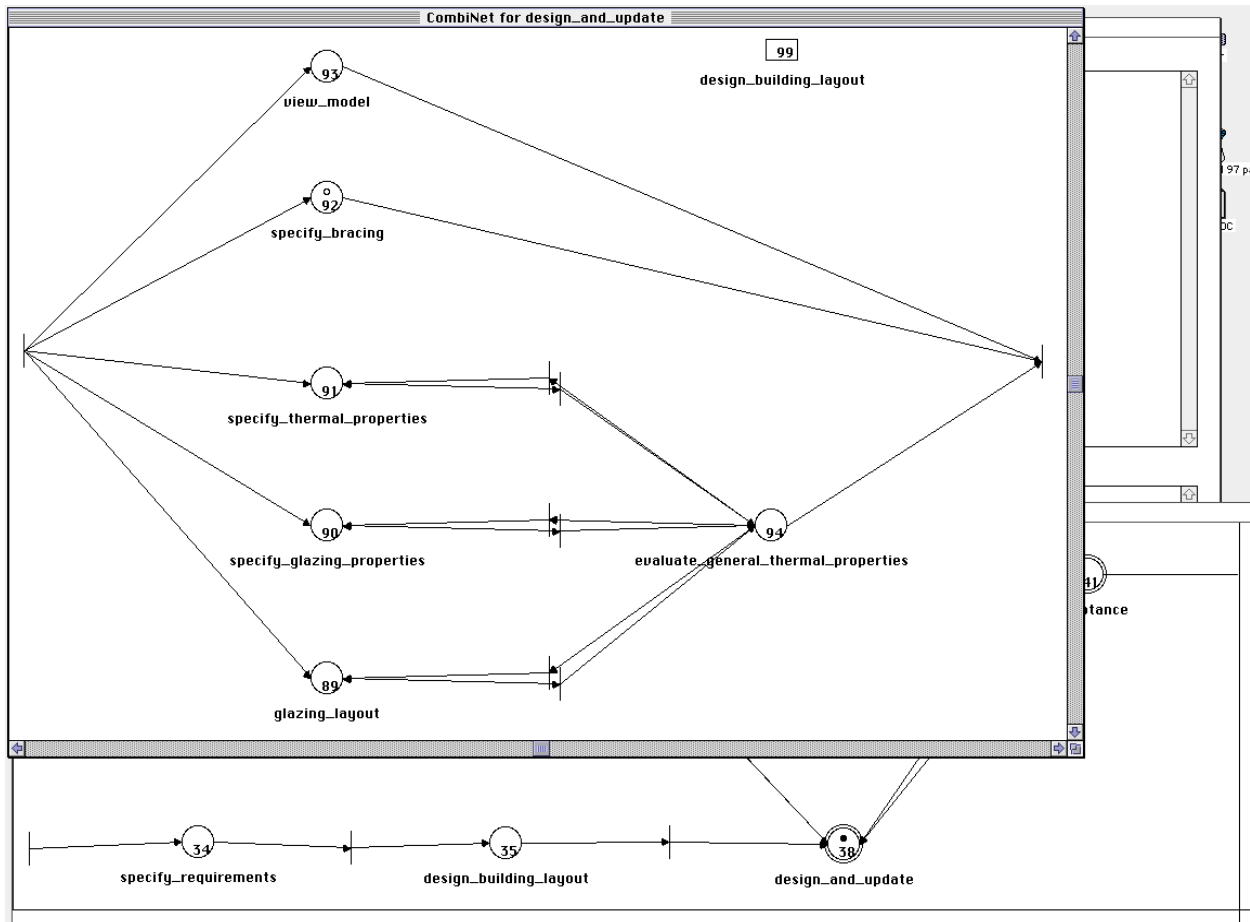


Figure 11.4 Multiple levels of aggregate functions

Aggregate place: an aggregate place represents a whole CombiNet, so when evaluating what can be invoked in an aggregate place, the start transition of the denoted CombiNet is identified, and the places which are reachable from that transition collated (e.g., in Figure 11.4 the *design_and_update* aggregate place, as shown in Figure 11.3, connects directly to the five places reachable from the starting transition, and can also access a global place). This evaluation process can be nested down several levels, as the start of a CombiNet can reference an aggregate place whose start transition would have to be identified, etc. When an end transition is reached in a CombiNet referenced by an aggregate place, it is tied to the output transitions of the aggregate place which referenced the CombiNet (e.g., when the client exits the acceptance CombiNet shown in Figure 11.5 (small CombiNet with two places) the end transition is tied to the output transitions of the aggregate place it was called from, in this case the *acceptance* aggregate place shown in Figure 11.3. In this example there are two output transitions, one is the end transition of the project window and the other leads back to *design_and_update*). Again, this evaluation may be recursive, as the

termination of a CombiNet at one level could lead to the end transition of the CombiNet above it, etc. The path travelled by a token is always maintained dynamically, as many CombiNets could reference the same aggregate place (representing the same CombiNet). As a result, the path taken to a particular design function is not necessarily unique (e.g., *design_and_update* can be accessed from the top level CombiNet shown in Figure 11.3, but is also accessible as a global net at other stages in the project window). These different points of access to a lower level CombiNet can also have different actors assigned to them through the actor overlays described in Section 7.3.2.

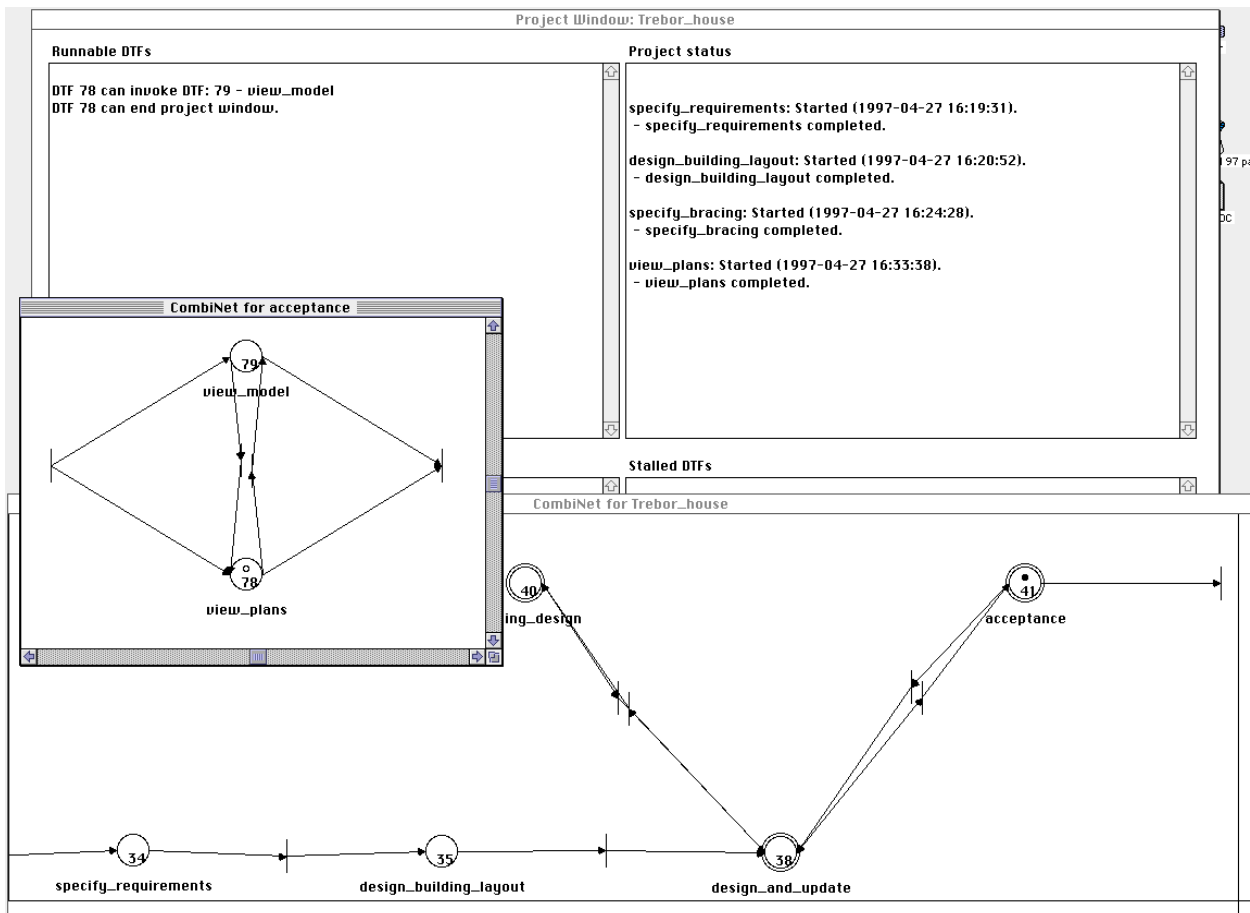


Figure 11.5 Exiting from a CombiNet representing an aggregate place

Global places: a global place is reachable from any place in the CombiNet in which it is defined, and from any CombiNet invoked through an aggregate place or global net (i.e., any descendant CombiNet). When exiting a global place, the token returns to the place from which the global place was invoked (e.g., the *design_building_layout* global place in Figure 11.4 can be reached from all the other places shown in that window. After completing the *design_building_layout* function the available design functions would be recalculated from the same place as the *design_building_layout* function was entered from). A global place is also accessible by all descendant CombiNets reachable from the CombiNet in which the global place resides.

Global net: a global net has the same functionality as a global place, except that, as it represents a whole CombiNet, its candidate places are calculated in the same manner as for an aggregate

place. As noted previously, when inside a global net it is not possible to reinvoke the same global net (it is removed from the list of candidate places).

Double transition: this shorthand notation is replaced with its two-transition equivalent in the ExEx, and handled in the same manner as other transitions (e.g., the transitions between *building_design* and *design_and_update* in Figure 11.3 are the expansion of a double transition).

When determining the set of candidate places from a given place it is possible to encounter the same design function along different paths. All occurrences are displayed for the actor to choose between, as each occurrence is part of a different workflow through the project window. Currently actors must navigate the CombiNets to ascertain the actual path traversed to particular instances of a design function.

11.2.2 Representation of design tool invocation

The ExEx, described in Section 11.2.1, simulates actors working in a particular project window. However, to perform this simulation, it requires information about the status of the design functions which are being performed. Though this ExEx implementation does not support design tool invocation (for the reasons described in Section 8.4), the point at which this would occur is simulated through a starting and ending dialogue for each design function. These two dialogues are described further below.



Figure 11.6 Design tool start up dialogue

At the time an actor decides to perform a particular design function which requires the use of a design tool a dialogue is initiated (see Figure 11.6). This dialogue is used to ensure that the actor performs a mapping from the IDM through to the design tool model, using the mapping manager described in Chapter 10. Dependant upon the outcome of the mapping process, and the ensuing design tool start up, there are two messages that can be passed back to the ExEx by selecting one of the buttons in the dialogue. These are:

Succeed: all of the required data existed in the IDM and was able to be mapped through to the design tool model. The data in the model was successfully translated into the form required by the design tool, and the design tool started without problems.

Fail: some initial constraints in the design tool model were invalidated, or it was not possible to start the design function due to the failure of the design tool to start up (e.g., due to design tool constraints not specified in the design tool's data model).



Figure 11.7 Design tool termination dialogue

After the design tool start up dialogue is completed, a new dialogue is presented (see Figure 11.7). This dialogue must be completed when the actor has finished work with the design tool and has mapped resultant data back to the IDM, using the mapping manager described in Chapter 10. Dependent upon the outcome of the mapping process there are three messages which can be sent back to the ExEx by selecting one of the buttons in the dialogue. These are:

Succeed: the design tool completed normally, and all data in the resultant data-files of the design tool were mapped back to the IDM through the design tool data model.

Fail: the design tool terminated abnormally or incorrectly.

Fail Data: the mapping system was not able to map the resultant data through to the IDM. This would be due to a violation of constraints specified in the IDM schema, which may be more constrained than the design tool's output model.

If the ExEx is informed of any of the three failure modes described above, it will roll the token back to the place at which the choice was made to invoke the design tool which just failed. The choices at that point are re-evaluated. The new list of available functions is presented to the actor and a new design function is chosen to continue work on that actor's design role. This list may still include the design function which previously failed if the failure was due to its operation rather than data constraints.

It is conceivable that failures of design functions could lead to a state where an actor has no further choices available. There are two possible methods of resolving this situation. First, if other actors are working on the project window, their completion of design functions will force the re-evaluation of stalled and failed tasks. Second, the project manager is informed of any actor who is stalled in this manner. The project manager has two options at this point, either: terminate the actor's design role (removing the token); or move the actor to a new point in the design process.

11.2.3 Project manager interface

To support his/her project management role, the project manager is provided with a comprehensive query, control, and browsing interface to the ExEx, allowing arbitrary movement around the flow definitions, and with the ability to easily modify the status of actors working in the project window. The main interface provided for the project manager is shown in Figure 11.8. This shows a list of all actor workflows currently executing in the project window, and the path which

brought them to their current position. The path for a particular actor workflow starts at their current position and steps back through all design functions invoked in their design role. Where an aggregate place or global net has been entered a bullet is placed before the place name to denote the point at which the lower level CombiNet was entered. For example, in Figure 11.8 the structural consultant is working on *define_wall_structures*, the path taken to this design function for the particular design role and workflow is from the *structural_work* aggregate place and before that the *building_design* aggregate place. Any actors who are currently stalled (i.e., having no possible design functions available at the current time) are denoted with a ‘•’ before the actor name. Actor-specific functions supplied by the buttons in the interface become available when an actor is selected. The actor-specific functions offered by the buttons are:

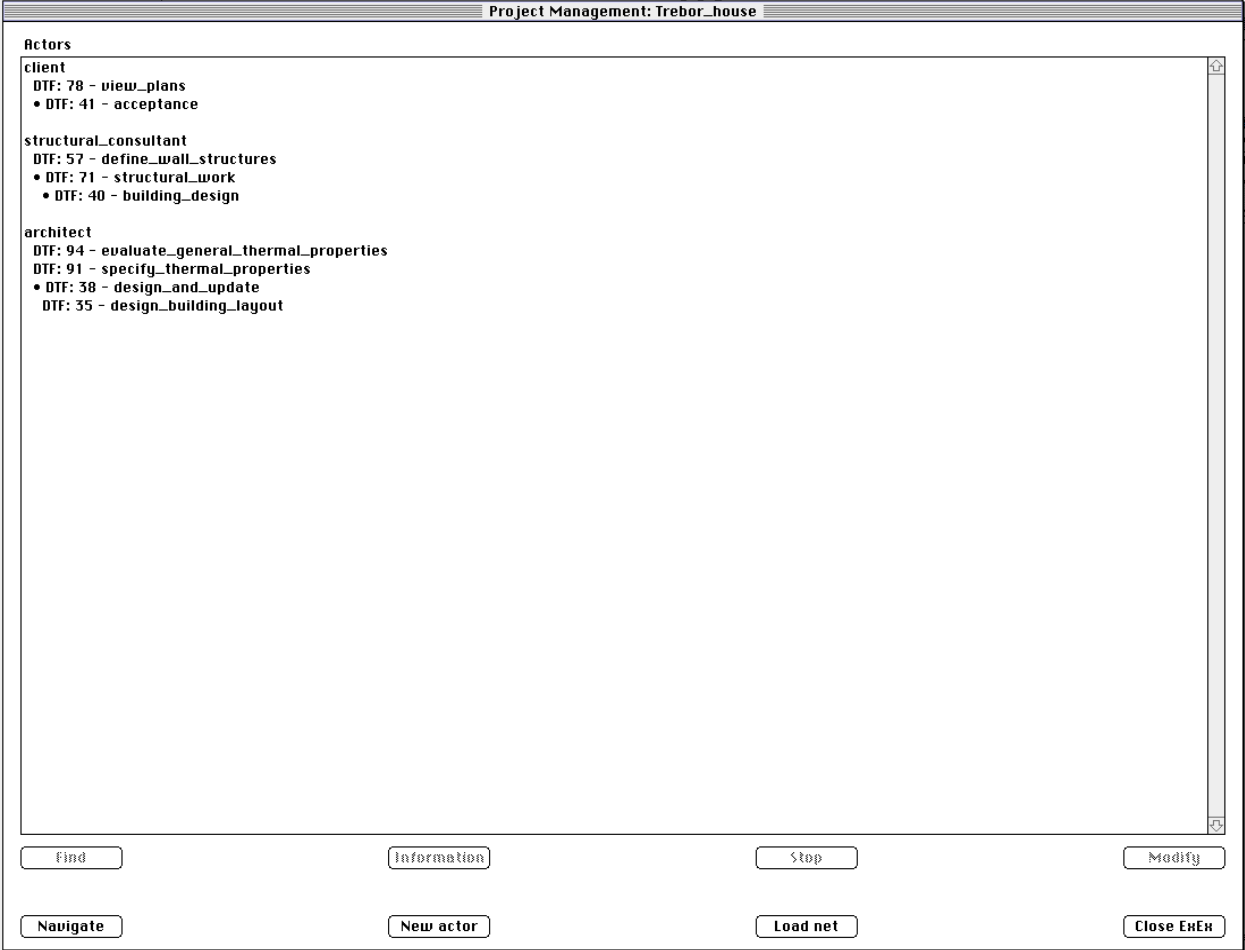


Figure 11.8 Project manager user interface

Find: opens up all CombiNets that were entered by the highlighted actor to reach the position at which the actor is currently working. The windows are opened in order, from the top level CombiNet through to the CombiNet in which the actor is currently working. Any aggregate places or global nets that an actor passed through to reach the current position are denoted with a filled token in the higher level diagram. A token is shown in these CombiNets for any other actor working in a CombiNet passed through to reach the selected actor.

Information: details the design functions available to the selected actor, shows any stalled design functions, and lists the work performed by the selected actor, using the interface provided to individual actors (see Figure 11.10 for an actor's user interface).

Stop: terminates the design function and workflow in which the selected actor is involved. The actor is removed from the project window at this point and must negotiate with the project manager if they wish to be re-involved in the current project window.

Modify: the selected workflow of an actor is modified to work in a new place in the project window. This is used to help an actor workflow which has no available design functions, or to make more resources available in a certain portion of the project window. The selected actor workflow is removed from the current location, and the project manager navigates through the CombiNets (as explained for the *Navigate* function below) to the place where the workflow is to continue from. When the project manager finally double clicks on a place, or global place, this is defined as the current location of the actor in that workflow. After placing the actor, the project manager must specify the place, or global place, from which it is assumed the actor reached the current place. This is in case the design function fails and the actor has to roll back one place.

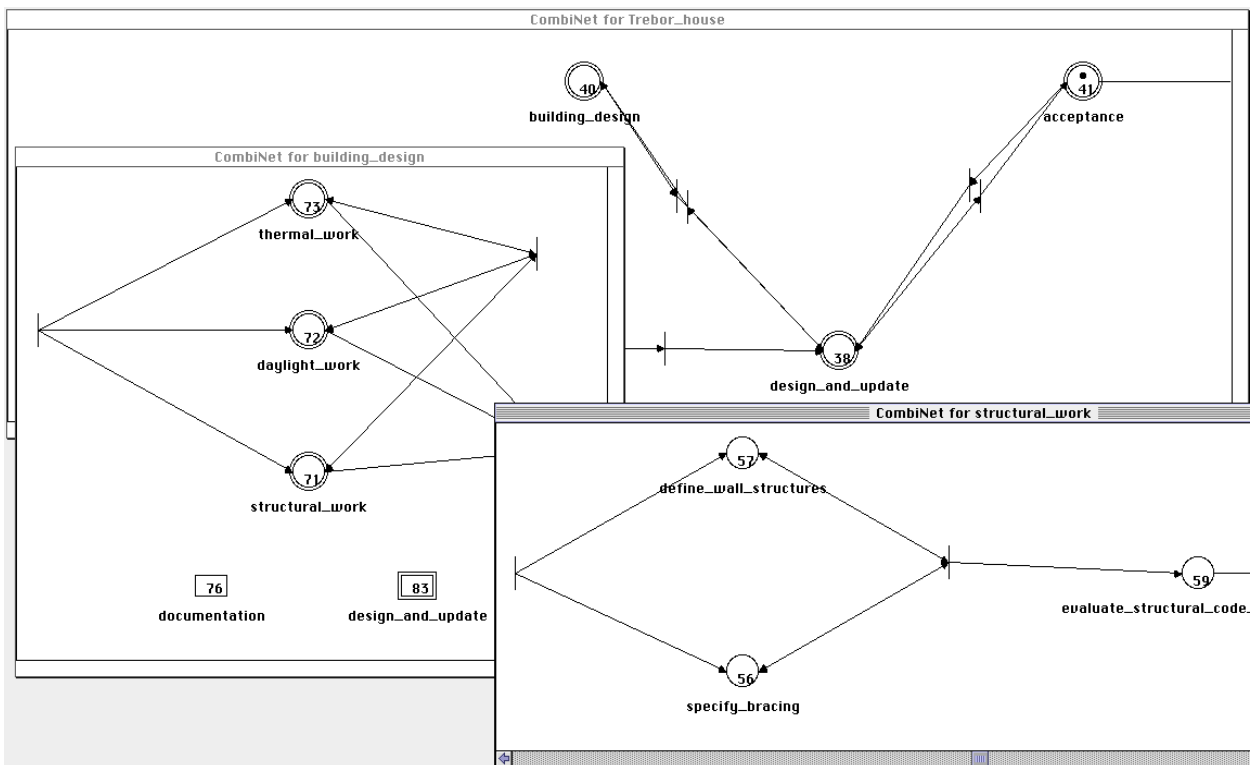


Figure 11.9 Project manager navigation through CombiNets

The project manager also has three function buttons available which are always invocable. These allow the following management functions to be performed:

Navigate: closes all currently visible CombiNets and displays the top level CombiNet. The project manager can double click on aggregate places, or global nets, to expand a level in the project window, displaying the new CombiNet. The project manager can backtrack to the previous CombiNet by closing windows (see Figure 11.9 for a navigated view from a

project manager interface).

New Actor: allows the placement of a new workflow for an actor in the project window. The project manager starts navigating from the top level CombiNet as explained in the *Navigate* function above. The aggregate places and global nets passed through during the navigation provide the path of the new actor. When the project manager finally double clicks on a place, or global place, this is defined as the current location of the new actor. After placing the new actor, the project manager must specify the place, or global place, from which it is assumed the new actor reached the current place. This is in case the design function fails and the new actor has to roll back one place.

Load Net: replaces the currently loaded project window with a modified version. The paths of the actors in the current project are used to try to place the actors in the new project window definition. If any actor can not be placed in the new project window (for example, if a place they passed through does not exist any more) then they must be placed in the new project window by the project manager in the manner described for the *Modify* function.

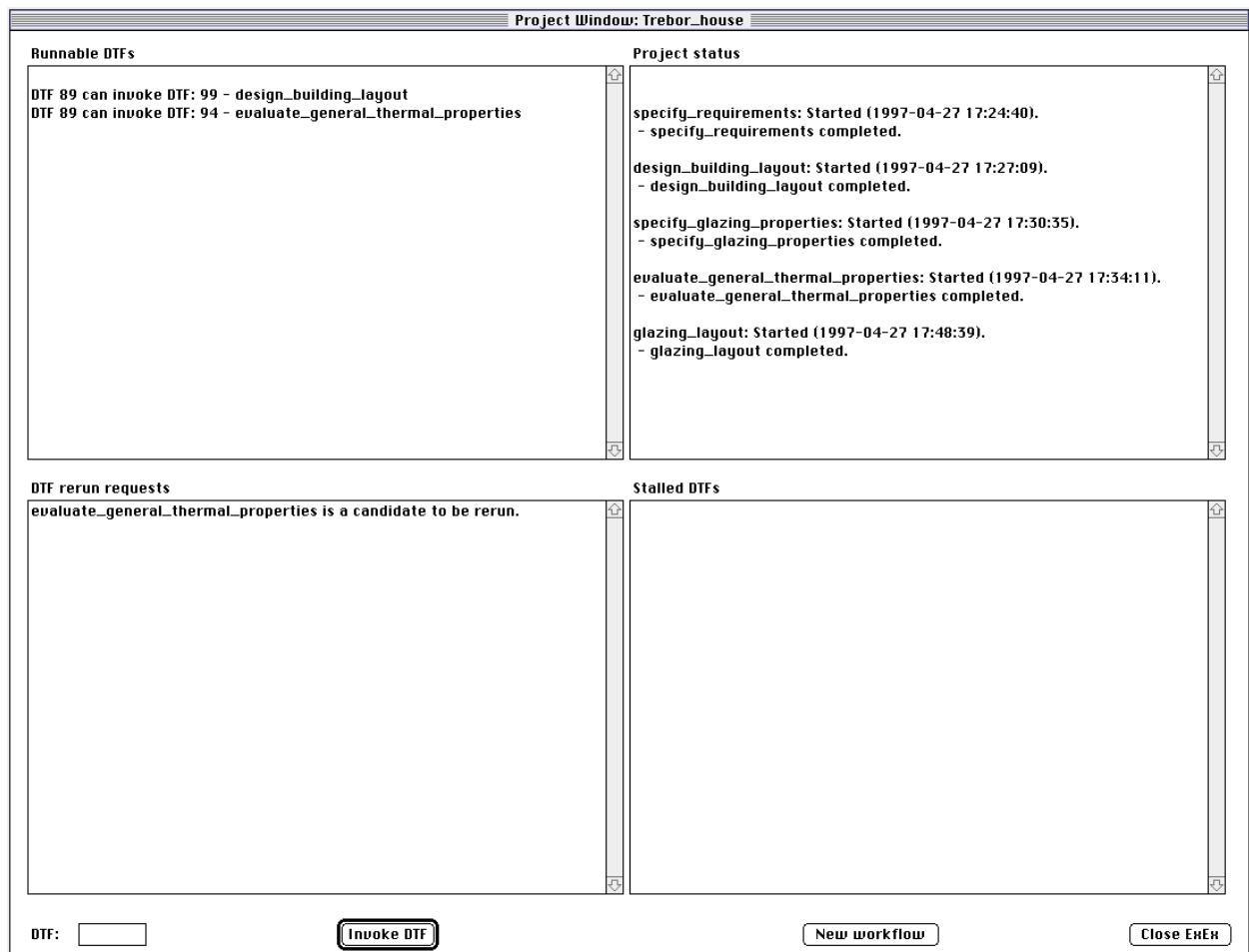


Figure 11.10 Actor's user interface

11.2.4 Actor interface

The actors are each provided with a task level interface to the ExEx, allowing specification of the design function they wish to work upon and determination of the status of the candidate design functions from their current position. This interface concentrates upon an actor's design role and

provides very little information about the status of other actors in the project window. However, any actors working in the same CombiNet (having followed the same path) will be displayed in the actor's interface. An actor will also see any filled tokens denoting an actor working down an aggregate place, or global net, which is represented in the actor's current CombiNet.

The main interface for an actor is shown in Figure 11.10. There are four main sections to this interface providing information about the actor's current state, as described below:

Runnable DTFs: provides a list of design tool functions which are currently invocable. This list is recalculated for the current actor every time an actor in the project window completes or starts a design function. This list is also calculated over the time that an actor is performing a design function, providing a list of design functions that the actor could perform concurrently (e.g., a documentation function at the same time as a simulation function). Apart from the design functions that an actor can perform there are two special states which may be shown in this window, they are:

end: denotes the end of the project window, usually only reachable by one or two actors in the project window. By specifying *end* the actor terminates the project window (assuming that no other tokens exist in the current project window).

hand-over: denotes the hand-over point from one actor to another. The hand-over specifies the actor who takes up the new design role and the design function they will perform to start that role. When choosing a *hand-over* the actor's design role is completed, and the token now represents the actor to whom the design has been passed. When an actor completes their design role they may choose to collect their work into an aggregate transaction representing all the work performed in that design role.

Status: details the work performed by the current actor in their design role. This list details when design functions were started, when design functions completed, if any design functions failed to start, or terminate, normally.

Stalled DTFs: provides a list of candidate design functions which are not currently invocable. This can be due to the functions being performed by other actors, or to the failure of design functions to start or terminate correctly when previously tried.

Rerun requests: lists all design functions previously performed by the actor whose input has changed due to the performance of subsequent design functions. This list is purely informative, and carries no obligation for the actor to re-perform a design function whose input has changed.

As well as being provided with the textual interface to current design functions and the design status, an actor is also presented with a graphical window showing the CombiNet they are currently working in, see Figure 11.5. These views can be navigated in the same way as described for a project manager. As the actor chooses new design tasks to perform, the token representing their position in the CombiNet moves through the graphical view. As they move up or down

between levels in the project window, new graphical views are provided showing their current location.

11.3 Appraisal of Flow Handling

The ExEx provides an implementation of the flow of control specification from Chapter 7, with enough checking and control to manage project windows of the small size demonstrated in the figures in this chapter, through to large project windows (containing several hundred places) as demonstrated in the COMBINE project (Flynn 1994). The interfaces to the ExEx provided for its users provide the required access to the state of a project window and the control required to manage the running of the project window. The project manager has full control over who works in the project window and what functions they perform, but, assuming that the actors work well together and the design progresses without problems, the project manager may not have to do anything to aid the completion of the project window. The actors only have control over their design functions, but can manage all tasks pertaining to their design role with the guarantee that they are not impinging on other actors working in the system.

There are a few aspects of the ExEx which do not provide the level of support or control which was envisaged at the start of the project. These points are elaborated below.

The project manager's ability to dynamically modify the running project window is limited. What is required is a close link between the project specification tool and the ExEx, so that the project manager may make modifications to the project window definition and have them appear in the running project window, without the need to reload the whole project specification as is the case currently.

There is a gap between the set of available design functions displayed to an actor in their control interface and what they can determine from the CombiNet display of their current place. This gap is due to not being able to see the path to all the design functions which are available at any one time. However, there seems to be no easy way to resolve this problem, as available design functions could have been inherited from several CombiNets above the current CombiNet as well as from several CombiNets below through aggregate places and global nets. Place navigation aids are available to provide navigation similar to that offered to the project manager, but these still do not provide a good view of the relationship between the current design function and all the available options.

The mechanism used to calculate stalled design functions provides an overly conservative determination of what can be performed concurrently. This is due to some portions of the IDM being generic (e.g., the geometric model). Almost all design tools draw data from the geometric

definition, so any design function which outputs geometric data will stall almost every other design function in the project window. In some cases this may be sensible, but in the majority it is not necessary. For example, consider a design function which defines the paving around a building. When using this design function, all design functions which require only internal aspects of the building will be stalled as they access the geometry portion of the IDM. One solution would be to mark portions of the IDM as being generic (e.g., geometry, documentation) and not consider these portions when determining stalled design functions. However, this may be problematic for design functions where the use of generic portions of a model does impinge on their operation (e.g., a visualisation tool whose input is almost purely geometry). Another solution is to consider the actual objects used by running design functions when determining what is stalled in collaboration with the intersection of models. However, it is impossible to know whether a design function will produce output which is going to affect a design function which wishes to start up. The method used in the ExEx does guarantee consistency of the model, with the trade-off that design functions may stall unnecessarily.

In a similar vein, the current model of intersection checking does not consider two design functions whose output may partially overwrite each other, for the reasons described in Chapter 7. However, assuming concurrent design in a project window, the current ExEx does not stop a design function writing its data back to the central store, even if it may be overwritten by a design function which does not have permission to overwrite it (e.g., a running design function which should be completed before the design function which wishes to write the data). Stalling design functions whose outputs overlap with currently running design functions would provide a conservative strategy (for the same reasons as mentioned above), but would guarantee consistency of the central model. This problem and the one above indicate that further work is required in determining strategies to manage concurrency and consistency in project windows.