

Chapter 12

Conclusions

The research presented in this thesis examines the type of development framework required to plan and implement an integrated design system. The central premise is that existing stand-alone tools lack the functionality required for large or medium sized integration projects. Research in this thesis also recognises that insufficient aspects of an integrated project are modelled, resulting in stand-alone, or poorly connected tools. Although many projects and many hundreds of person years have been spent developing integrated design systems, their development frameworks all have major shortcomings. These shortcomings range from the limited amount of information modelled through to the nonexistence, or low capability, of tools needed to implement prototype or commercial systems. This thesis analyses and identifies the full range of information which needs to be modelled for an integrated design system and demonstrates a range of tools to support this modelling.

The above analysis shows that modelling and integrating purely data aspects in a project is not enough. To be usable in a development project, an integrated design system must take into account the processes involved in the development and final use of the system. Thus a very wide range of modelling methods covering data, processes, documents, legal aspects, activities, etc. (c.f. the IDEF family of models, Mayer et al. 1994) is needed. Support tools are required to facilitate the creation of an integrated design system, and this thesis demonstrates the benefit offered to a project when they are available. Examples in this thesis show the application of these ideas to an integrated building design system. The need for the tools and integrated systems shown in this thesis is currently growing more urgent. Many small to medium sized integrated design environments are under consideration, or even under development. To ensure that these projects complete successfully, and operate as required by those in the industry, tools and modelling as described in this thesis need to be developed and introduced to the industry.

The major framework advances proposed in this thesis fall into three areas. First, more capable modelling tools are required for the development and testing of the very large models which must be specified for many information types (e.g., product and process). This thesis demonstrates the benefits of such environments. Second, formal mapping languages are required to define the correspondences between information in the many views of a domain found in integrated design systems. The thesis defines one such language and shows that such languages can be supported by automated implementations (rather than using the mapping as a coding specification). Third, formal process specification languages need to be adopted to model the intent and usage of tools inside an integrated design system. This thesis defines a process formalism and demonstrates an implementation which can be used to test and control a running integrated design system.

The remainder of this chapter examines the contributions made by this project and the conclusions that can be drawn from this work, and then proposes further work required in this area.

12.1 The Project Development Environment

This thesis has highlighted the need for development environments to help with the creation of integrated design systems. The tools currently available to developers of such environments, and the model specifiers, are totally inadequate for a well managed project. Inadequacies highlighted in this thesis are: an inability to guarantee the consistency and correctness of a model and the data used to test that model; an inability to manage and present overlapping views which make the models more understandable than canonical forms; and an inability to cope with development teams larger than a single person. This makes existing tools poorly suited for use in development projects where models with hundreds of classes are the norm and where the size of the development team is usually greater than ten people. The development tools presented in this thesis show capabilities which tackle and resolve all of these problems.

The use of several modelling methods to capture different aspects of a project, each supported by a development tool and environment, highlights the need for connections between models. Where several models of information in design tools and central models exist it is imperative to provide a method of mapping between the representations. Where processes and activities are defined they must be linked with the data aspects that support the processes and activities. When information in one model changes there are a range of inter-related models which are in some way dependent upon the changed information and require updating. This requires that all of the development environments communicate between themselves when dependent data in their models is modified. All of the development tools demonstrated in this thesis are built upon a platform which allows related modelling systems to be attached with notification of changes to dependant objects. Even during the schema development stages it is clear that all aspects of the project domain need to be consulted to ensure that the schemas capture the domain sufficiently. The best way to ensure this

sufficiency is to model also the related aspects. This has led to models of data, correspondences between data model, processes, activities, actors, and design tool parameters being incorporated into the demonstrated system. Other related aspects also need to be incorporated, including documents and their associated legal aspects for a project, possible conflicts and their management, though these two aspects are both implicit in the constraints which are seen in the process models.

The development environments created in this thesis all provide a common set of functions much needed in the area. This includes the ability to use multiple overlapping views of components, enabling subset views of a schema to be presented for various design processes, thus highlighting different aspects of a schema and the relationships between entities in the schema. This helps ensure the correctness of schemas by allowing concentration of particular aspects of the schema at any one time and makes the schemas more understandable to audiences with different interests. The other main ability is to be able to work with both textual and graphical representations in the same environment. Again this enables views of varying levels of complexity and completeness to be presented to different audiences for different tasks. Supporting these abilities in existing commercial and research tools would have a marked impact in terms of greater certainty of the correctness of the models being developed, a greater ability for them to be understood, and the reduced time that would be required to develop them.

Though there are only simple ties between the schema development and the testing environments in this project, they show the utility of a fast and automated path between specification and testing. The testing environment tools developed for this thesis allow schemas to be instantiated and then visualised, both in terms of values corresponding to data structures and for renditions of graphical definitions, to perform quick checking of schema sufficiency and validity. The tools developed allow for an early and continuous feedback loop into the model development phase of a project. In this way the sufficiency of the models can be demonstrated and checked from early in the model development. These types of tools and their close linkages with modelling environments need to be more widely used to reduce the number of large schemas which are developed from theoretical foundations without the ability to test against actual working needs until late in the project. It is recommended that instantiations of schemas should be developed almost in a direct parallel with the schema development. This usage of test data alongside the model development does however introduce an additional requirement on the linked tools, that being the ability to move test data sets forward to new schema representations as modifications are made. The use of the VML mapping language is recommended for this phase, especially if tied to the semantics of modification functions in the modelling environments, which allow mappings to be built automatically as schemas are modified.

12.2 The Mapping System

Previous research work into integrated design systems has highlighted the necessity to perform mappings between representations of information in a domain, usually between central models and those utilised by the different actors and design tools used in the integrated system. Existing mapping languages and techniques have, however, proved too restrictive for this task, mainly due to assumptions of commonality of semantics between models being mapped. It is certainly not possible to guarantee this between the models of existing design tools, let alone those employed by the actors from different disciplines in a project. This thesis presents an analysis of the types of mapping required between various models and leads to a set of requirements for a mapping language. The VML language was specified to meet these requirements and is demonstrated through an interpreted implementation.

The VML language provides a mechanism to describe the correspondences between two schemas. One of the main premises of the language is that bidirectional mappings must be supported (as most non-trivial mappings must be performed in both directions). This led to a high-level declarative language with an in-built assumption that any implementation of the language will provide a mechanism to run mappings in the required direction. VML provides a language of far greater range than allowed in RDBMS views, with the obvious drawback that not all described mappings are invertible (which is guaranteed in an updatable RDBMS view). However, the power of automatically invertible declarations is also greater than that offered in RDBMS views, and demonstrated to work through an implementation which must track all modifications and previous mapping connections. To support non-invertible mapping definitions the VML language offers a fall-back position of defining two procedural mappings to enact the mapping in both directions.

A base VML mapping component offers three main functionalities: a specification of constraints to determine when the mapping can be applied; a set of mappings to be applied; and initial values to be instantiated when new objects are created during a mapping. A VML mapping provides unique functionality in allowing methods in object-oriented systems to be referenced and invoked as part of a normal mapping. For example, it enables the semantics of method calls to be mapped between various systems, especially those which have side-effects outside the scope of the data in the model, e.g., screen display calls.

VML allows a mapping to be specified between two schemas. It also describes what type of mapping can be performed, i.e., read only, read-write or an integrated mapping which forces a consistent state to be achieved before application. A network of VML mappings can be specified between a range of design tools. The examples of this project demonstrate a star topology through a central integrated model. However, many other topologies can be specified.

The implementation of VML in this thesis demonstrates the ability to implement complex VML specifications and run them bidirectionally, as illustrated for a set of tools in the building and construction domain. To enable mappings to be made in either direction the mapping system tracks a large amount of meta-data about the mappings which were previously applied. This allows updates to a mapping to be identified and applied with greater efficiency in the implementation. However, it is clear that the amount of meta-data tracked soon greatly outweighs the amount of data in the individual repositories, especially if many small mappings are applied between two data stores. However, this is a minor cost to be paid for the ability to perform complex mappings bidirectionally between two data-stores. It is clear that any system required to perform efficient mappings between two stores will need to track this level of meta-data. This is the only way to avoid problems of trying to match structures in two existing stores through the mapping definitions.

Trying to compare VML to other mapping languages highlighted the need for a methodology to compare mapping languages, and, underlying this, the need for a formal description of the mapping domain. With a description of the mapping domain it would be possible to describe the power of individual mapping languages, their strengths and weaknesses, and their sufficiency for different types of applications. This has been attempted to some extent in the thesis by selecting features of the mapping problem that needed to be supported for the particular domain. However, these features are overlapping in that they provide different views of similar functions and do not necessarily represent a comprehensive set of mapping language requirements.

As previously noted, it is not possible to apply all declarative mappings bidirectionally. In fact, it is not possible to determine which mappings are possible in a particular integrated environment until the power of the mapping implementation is determined (not all VML specifications need to be implemented in all mapping implementations), and some techniques make more mappings possible, e.g., incorporation of a simultaneous equation solver. So, though a VML specification can describe how a mapping could be performed, it may or may not be calculable in a particular integrated design system. One partial solution to this problem, utilised in this thesis, is to allow constraints to be imposed on particular data sets following a mapping. In this way, though the value for an attribute may not be calculable, it will be possible to determine whether a newly specified value equals the value in the store from which it should have been calculated, or if a new mapping is required to change values in the original store. For example, with $area = height * length$ it is not possible to determine *height* or *length* from *area*, but the two attributes can be constrained to equal the value of *area* when multiplied, or if they do not equal *area* at some later stage it requires *area* to be updated with a new application of the mapping.

12.2.1 Additional applications of the mapping language

The VML mapping language is designed to be domain independent, as is the relational database language SQL. Thus it should be applicable to any situation which requires views of a similar

domain to be kept consistent, and where the system needs to be easily extended without having to compile in new features, functions, or tools. Some areas where VML would be immediately applicable are:

RDBMS Views: VML has already been shown to be at least equivalent to the operators in a RDBMS (see Appendix A.3). Therefore, any existing RDBMS view can be described in VML. However, VML allows more powerful views to be defined as there is no assumption that the view and base are semantically equivalent. Therefore, VML could allow more sophisticated RDBMS views to be defined and updated automatically.

Schema Integration: as with RDBMS views, schema integration techniques for RDBMS assume that the schemas being integrated are semantically consistent before integration. In many real-world applications this is not possible to ensure and hence VML would provide the ability to integrate non-consistent schemas, but still maintain the semantics of the individual schemas. Appendix A.3 shows that VML is at least equivalent to RDBMS schema integration operators.

Object-Oriented Views: some object-oriented languages and distributed object environments allow multiple public interfaces to be defined for an object (e.g., the interface description language (IDL) in CORBA, Otte et al. 1996), equivalent to database views except enhanced with method calls. VML provides an alternative mechanism to specify these views and also allows access to methods to allow methods calls to be tracked or invoked during a mapping.

Data Store Migration: some data repositories need to migrate to new schema definitions over their lifetime. Where simple changes are made, such as adding a new attribute, there is no great problem, but where the schema change involves existing attributes in a new form it is often difficult to create the new version of the data store. VML provides a specification and migration environment which can handle this problem.

Loosely Connected Toolkits: many systems exist as a set of loosely connected components or tools. Where each tool has a well defined set of functions and provides services which may be of use to many other tools not always known in advance. VML can be used in such systems to provide the glue to connect each of these tools as needed using a network approach to defining interfaces between associated tools, or through a mapping to a global and common data store.

12.3 The Flow of Control System

The specification of processes and the flows between them has received scant attention in the development of integrated design systems, mainly due to a focus on data and its transmission. However, as previously stated, it has been recognised that this focus on pure data transmission does not provide a system which can integrate with the processes required in the domain that the integrated systems sit in. Though the more recent European integration projects have considered

process and workflow in their systems, this modelling has often been of processes independent of the rest of the system. In this thesis, the full set of interconnections between various aspects of a project's information was considered. This shows dependencies between a process and data, documents, actors, design tools and the influence on process invocation due to legal constraints, standards and regulations, and the state of the executing system.

The CombiNet process modelling formalism addresses many of these issues by defining a two level specification. At one level this defines the actors in a project, their design tasks, and design functions which are tied to data specifications and design tools. The second level of specification provides a hierarchical specification of design functions and their possible flows of control. Varying levels of constraint in the process flows can be defined using either global processes at different levels of the hierarchy to provide unconstrained process flows, or totally connected processes to force a strict flow of control. This level allows the point of actor handover between processes to be further specified, and allows the user to place constraints on the invocation and completion of processes. One of the major benefits of the flow of control specification is that it allows looping or iteration between processes which is not allowed in the major process modelling formalisms due to the difficulty in simulating processes with loops (unless the number of iterations can be pre-determined). However, in the integrated design systems in which this process model is utilised, it is not important to simulate time taken to complete a project (though that is useful), but it is necessary to define the possible flows between design tasks of the actors to control the project. Though CombiNet ties together many aspects of a project there are still links that could be defined. One of these is a link to documents and documentation which feed into a process, or are required at the termination of a process. This would enhance the ability to specify constraints on processes by allowing legality aspects to be brought into the project (e.g., whether a stage has been signed off properly).

The implementation of the CombiNet formalism provides a control system for a running project. This includes project manager type overview functions for controlling the work done on the project. It also provides interfaces for actors to specify the design functions on which they are working, and to notify others of their progress. It also allows actors to identify design functions that they are next able to work on, or to specify a handover to another actor in the project. Due to the connections between the different aspects of a project, the flow of control system can track and manage many functions of a running project. It can determine what design functions can run at any time in the project, providing help to those attempting concurrent engineering. It can also identify which actor in a project is hindering other work, allowing the project manager to smooth the running of a project, or to examine business process re-engineering of the processes. As the implemented flow of control system tracks all processes undertaken, it provides a record of task progression to allow actors to show they completed all design tasks in their programme of work.

12.4 Future Work

The work presented in this thesis has examined requirements for an integrated design environment and provided prototypes of components required to make this environment reality. However, not all of the requirements could be addressed within the scope of this work, and several areas for further work have been identified. A selection of the more important areas for future work are addressed below.

12.4.1 Tighter system integration

The integrated design system framework developed in this thesis provides for communication between several of the modelling and testing tools. However, there is benefit in providing for even closer integration of these tools. One approach envisaged for achieving this integration is to extract the underlying model requirements of each of the tools, and develop an integrated schema incorporating these models. In this way, the development framework could provide services similar to those being created in the integrated design system itself. For example, if the schema modelling and process modelling tools utilise a common data store containing the combined models, then by specifying the model portions in which each tool is interested it would be possible for a tool to be notified when changes are made to information it is referencing. This would provide for greater consistency in the total set of models developed for an integrated design system.

Providing these meta-models for tools would also enable a wider range of modelling paradigms to be more easily integrated into the design framework. If generic model concepts are encapsulated in the developed meta-models, then a variety of modelling methods (e.g., NIAM, EXPRESS and ER) could be used, as long as the tools which model with a given methodology can translate their models into the meta-model, or accept models in the meta-model format.

Tighter integration could also apply between the tools defining models and those manipulating and populating those models. In this way, automatic migration of test data sets between versions would be more easily achieved. Modelling of the functionality available in the different tools, and describing how this maps to the associated data manipulation tool is of considerable benefit. For example, when an attribute's type is modified in a schema development tool, it must identify what function, or set of functions, this maps onto in the data model manipulation tool.

The final outcome of this environment integration would be a set of tools which communicates with all other tools upon which it has an impact. This would apply whether it is a modelling tool or a data manipulation tool. In this environment designers would be sure they are working with the most up-to-date models and that all affected models would be notified of changes made.

12.4.2 Distributed environment

Though all of the tools developed in this system are stand-alone and could be used by a number of concurrent users, multi-user development was not supported in this thesis. The connected tools in this thesis have a very simple view of the information logistics server with which they communicate (i.e., the Macintosh OS). This should, however, be easily scalable to incorporate information logistic servers which handle multi-platform, multi-user and multi-organisational environments. It is envisaged that interfaces to CORBA-like platforms (Otte et al. 1996) would enable the handling of distributed data and distributed functionality for the tools defined in this thesis. The ‘mapping through transaction-based updating’ approach provides a model which would enable multiple users to work on the same underlying model through a distributed environment and guarantee consistency (though at a high level of granularity).

Providing this extension would make the framework developed in this thesis more palatable to the types of teams which are used today to develop integrated building design systems. These teams consist of several modellers and testers in several organisations, usually also in several countries (for the EU funded projects). For these teams, a distributed environment which manages model and data consistency would be in itself of enormous benefit.

12.4.3 Wider incorporation of project aspects

Throughout this thesis it is argued that integrated design systems must enable all aspects of a project to be encompassed in the final system. The system developed for this thesis shows the integration of data and process aspects with some computing environment information also drawn in. However, there are other areas which need to be brought into the system. Documents and documentation are the most important aspect not currently tied into integrated design systems, though there are many commercial systems available (IIC Consulting and Cimtech Limited, 1996). Documents and their legal aspects are closely examined in the ToCEE project (Amor and Clift, 1997) and the following points noted:

- Documents are the main communication mechanism in any construction project.
- All project information is passed through documents and they are the input to all processes, as well as the output of all design processes.
- Documents are the only legally binding information medium in a project.

Therefore, by incorporating documents, it is possible to track and manage legal aspects of a project as they impact on processes and the data being manipulated (e.g., contracts and sign-off points). The development of models and frameworks to handle wider project aspects can be seen in the recent work on the BCCM in STEP (Building Construction Core Model, ISO/TC184, 1996) and the EU-ESPRIT funded ToCEE project (Towards a Concurrent Engineering Environment, Katranuschkov et al. 1996).

Other aspects which could be considered for incorporation include conflict management and negotiation. In past construction projects, this was limited to clash detection (e.g., does a pipe's position clash with a column's placement). This needs to be extended to allow negotiation between project partners when designers fail to agree on the state of overlapping portions of a design. In some instances this can be managed by providing the project manager with tools to arbitrate disputes and apply decisions to the project team, though in others a dialogue between partners must be supported before decisions are enacted.

A final aspect which could be considered, and which certainly has an impact on the whole system, is that of constraints on a project due to standards and regulations. In some cases they determine what processes can be enacted, or in what order. In many cases they impose pre- and post-conditions on invocation of a process as well as on aspects of the data model and documentation which must be produced.

12.4.4 Formal definitions of the mapping domain

In Chapter 5 it is noted that there is no method available to provide a rigorous comparison between different mapping languages. Part of the reason for this is a lack of understanding in the research community of the scope of the mapping problem and few formal methods to help describe mappings (though Ainsworth et al. 1996 seem to provide a way to approach this problem). Further work is required to define the semantics of mappings and mapping languages as the initial step to allowing mappings to be validated and languages to be compared. Though Chapter 5 details the main requirements of a mapping language, these are drawn from analysis of mappings in a single domain (i.e., construction). The scope of these requirements needs to be validated for a wider set of domains and specified in a more formal way. This more formal specification would then allow the power of related techniques to be gauged (e.g., constraint specification systems and the Interface Description Language (IDL) of CORBA, Otte et al. 1996).

A formal specification of the semantics of mapping languages would also help clarify the use of methods and transactions in a mapping. Currently VML allows method calls to be mapped between models, though the time dependency of the method calls is collapsed through the use of transactions to amalgamate multiple model modifications into a single mapping. A formal treatment of methods and their mapping requirements would provide an insight into where transaction boundaries should be drawn for a mapping.

12.4.5 Alternate mapping language implementations

In this thesis an interpreter for VML was constructed. However, VML could be translated into many other forms. An interesting project would be to examine the requirements for translating VML into other languages. This would clarify the trade-off that has been made between the specification language power and implementation complexity. For example, the translation of

VML into a procedural language is likely to be very difficult even when supported by a powerful set of backing libraries to provide many of the mapping functions. Examining the requirements would enable a decision on the technical feasibility and commercial viability of this translation to be made. The translation requirements would also determine the possibility of extending other systems to handle mappings, for example, constraint solving systems.

12.4.6 Incorporation of measure tools

Once all project aspects are incorporated into an integrated design system, there exists a central repository of all information relevant to a particular project and its progress. This would provide an exciting opportunity to data-mine the workings of a project, either to provide feedback for new projects, or as a monitor while a project is running.

A set of tools incorporated into the integrated design system and development environment could provide measures of the state of the running or developing system. These measures could identify aspects of a project which are: outside of best-practice (e.g., an inheritance hierarchy over twenty entities deep); or bottlenecks (e.g., the structural and HVAC engineers are both waiting for the architect to finish the redesign, they have been waiting 2 days, and this is the third time this month); or performance indicators (e.g., how much various parts of the project deviate from schedule).

Tools to examine completed projects would examine common aspects of a series of projects to identify possible improvements. For example, any possibilities for concurrent engineering in similar projects could be identified, or the feasibility of applying business process re-engineering to reduce the development time, and hence the project cost.

12.4.7 Dissemination and exploitation

The integrated design system development environment proposed in this thesis has been demonstrated for the construction domain. However, the system was devised to be generic and applicable to any domain with similar problems. To this extent, the system should be transferable to most engineering domains as well as medical domains and software engineering domains, all of which require the development of similar integrated systems. In many cases it is likely that such a system would make more impact, as these domains are further advanced in terms of data models developed and used, their understanding of process, the legal implications of getting it wrong, and the profits to be made by improvements.

In terms of general software systems there are several existing data management systems which could be improved through the use of a system as described in this thesis. RDBMS specification, development and maintenance could be subsumed by an integrated design system such as demonstrated, providing the same wide range of benefits as well as providing more powerful data

views for the database users. The same approach could be taken for heterogeneous database systems as long as a more sophisticated information logistics system was incorporated. Again this would provide greater power in the data views supported.