Revised version to appear in *Automation In Construction*, 1995.

Directions in Modelling Environments

Robert Amor[•], Godfried Augenbroe[*]
John Hosking[•], Wouter Rombouts[*], John Grundy[+]

[•]University of Auckland, Auckland, New Zealand
[*]Technical University of Delft, Delft, The Netherlands
[+]University of Waikato, Hamilton, New Zealand

Schema definition is a vital component in the development of computerised A/E/C projects, but existing tools to manage this task are limited both in terms of the scope of problems they can tackle and their integration with each other. This paper describes a global modelling and development environment for large modelling projects. This environment provides a total solution from initial design of schemas through to validation, manipulation and navigation through final models. A major benefit of the described system is the ability to provide multiple views of evolving schemas (or models) in both graphical and textual forms. This allows modellers to visualise their schemas and instance models either textually or graphically as desired. The system automatically maintains the consistency of the information in these views even when modifications are made in other views. Simple and intuitive view navigation methods allow required information to be rapidly accessed. The environment supports strict checking of model instances and schemas in one of the major ISO-standardised modelling languages now used in product data technology. In this paper we show how such a modelling environment has been constructed for evaluation in the JOULE funded COMBINE project.

Keywords: modelling environment; consistency; multiple views; views; building models; information management; integrated system; product modelling

1. Introduction

The definition of the schema for a particular domain is vital to the success of computerised projects in the A/E/C arena, whether it be the smallest application or the largest ISO standard. Existing tools to help users through this process are limited in the scope of the problem that they can tackle (for example see [1, 2, 3, 4]). Some tools assist in the development of a schema, others check it for consistency, others translate it to an implementation language, and others allow an instance of a model to be perused. However, there is a need to be able to use all of these tools together in an interactive and non-deterministic manner. This has led the authors to design an integrated modelling and development environment which provides many functions to its users in a homogenous environment.

This new environment has been developed for use in projects that deal with computer integrated manufacturing and product data technology in general. The term product data technology (PDT) is used to denote the body of available tools, methodologies and standards in the area of integrated CAD systems, dealing with product data interchange, product models, data modelling, integrated software architectures, etc. Modelling languages have been developed as part of this new technology to specifically capture the semantics of complex artefacts in a design and manufacturing context. The most accepted and standardised language to date is EXPRESS [5]. It is supported by a graphical notation, EXPRESS-G, which covers a subset of the language concerned with the basic entities, their attributes and relationships. This language is used in a number of

projects and standardisation efforts which are faced with large modelling tasks (for example [6, 7, 8]).

In this paper we introduce the COMBINE project giving the reader an understanding of the domain and size of problem that is being tackled by the partners in this project. A prototype implementation of the integrated modelling environment that has been developed for demonstration of concept inside the COMBINE project is then introduced. The structure of this environment is discussed with reference to the utility of its features in a project such as COMBINE, and examples are presented to illustrate the functionality that can be encompassed in such systems.

2. The COMBINE Project

COMBINE (Computer Models for the Building Industry in Europe) is an EC-funded project which started in 1990 [9]. The first phase ended in the fall of 1992 with a seminar and workshop at which the first phase deliverables were demonstrated [10]. The project, which is now in its second phase (1992-1995), spans a total effort of 70 man-years spread over 12 partners from 7 European countries.

The COMBINE project is working towards the practical development of IIBDS's (Intelligent Integrated Building Design Systems) through which energy, services, functional and other performance characteristics in planned buildings can be modelled and integrated. The modelling of information has been one of the greatest concerns in this project, with the first phase of the project concentrating on the integration of data to provide the necessary information for a group of actors. COMBINE's efforts have been concentrated on establishing a data infrastructure and tools for managing the information exchange amongst design actors, ie. members of a collaborative design team. The project thus represents a good example of international research in product data technology.

The COMBINE project is not being developed in a void. Many international projects have a similar aim of integrating the information requirements of a given domain to provide an enhanced working environment to users of that domain. In particular a number of parallel projects in the European ESPRIT-CIME program are engaged in similar research and development to that of COMBINE. In time all of these projects may have input into the ISO-STEP standard which has the stated aim of covering the information requirements for all architecture, engineering and construction domains [6]. In recognition of the importance of STEP, and the practicalities of interchange of results, most projects have chosen the modelling formalisms specified for the STEP standard as the formalisms for their efforts. These include the EXPRESS language, with EXPRESS-G, NIAM, IDEF1X and IDEF0 diagramming techniques, the STEP neutral file format, for exchange of data, and SDAI, the data access specification. This is the case in the COMBINE project where all modelling is done with these formalisms [11].

COMBINE's first phase was concerned mostly with data integration based upon the concept of a set of actors connected to a central common data repository (Figure 1). Its deliverables comprised the first large conceptual integrated data model (IDM) for buildings and a number of interfaced design tools (DT). These results formed the base line technology on which the present second phase builds. This phase is concerned with developing an operational IIBDS according to functional specifications of particular building projects in practice. In doing this, the number of interacting design tools will be expanded to cover existing design applications in the area of costing, HVAC-CAD (Heating, Ventilating and Air Conditioning), architectural CAD, component databases, daylighting and building regulations.

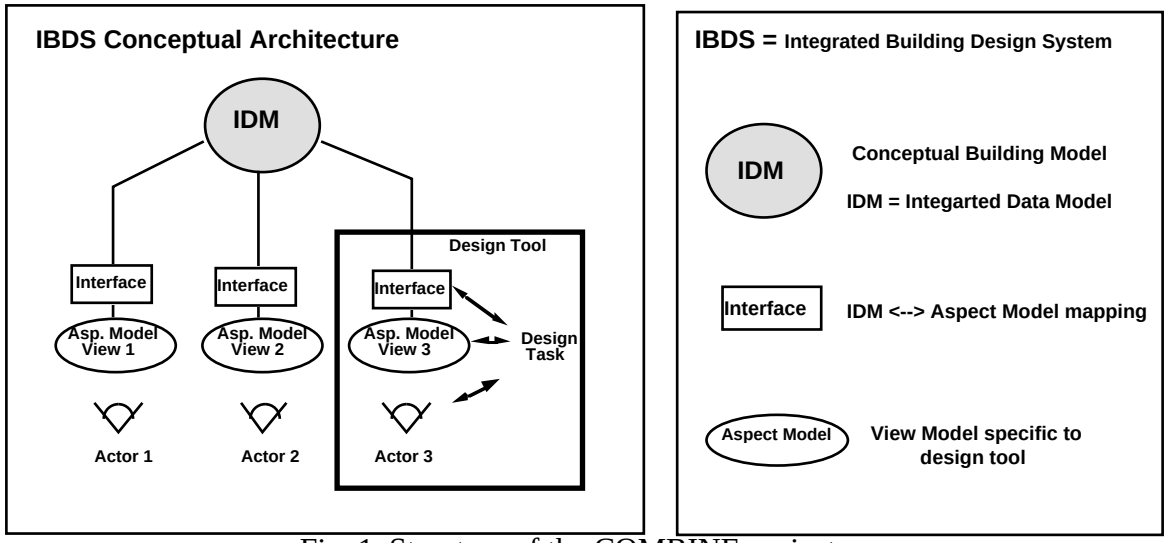

Fig. 1. Structure of the COMBINE project

2.1. Development of a central building model

In the first phase of the COMBINE project the types of actors considered were limited to those in the fields of energy and HVAC performance in the early design stages of building design. However, even limiting the domain of the first phase to this small set of actor types, the IDM developed for this project consisted of some 400 entities and 600 relationships between them. The development of this model was a complicated task. The final model had to incorporate the data requirements of all the design tools which were used in this first phase, some seven design tools, which had very different data requirements to model a building for their simulation needs. The IDM model was developed by first modelling the data schemas of the various design tools, and then, through schema analysis and use of integration techniques, various portions of the models were brought together to help form the final IDM. This was a very time consuming and difficult task. In particular, the developers in the project found it impossible to verify the consistency of the IDM without the aid of computer tools [12].

The second phase of the COMBINE project involves integration of several more design tools, widening the scope of the IDM. This requires a redesign of the IDM to extend the domains supported. Again, the process of development involves comparison of the IDM schema with the data schemas of the various design tools being supported and incorporation of portions of the design tool schemas into the IDM. The resulting iterative process of schema analysis, integration, and prototyping continues until the various design tool support teams are satisfied that the data requirements necessary for their tools can be met by the IDM.

The modelling support presently offered to partners in the COMBINE project comes through the use of the Configurable Graphical Editor (CGE, see [1]). CGE can be configured to support a number of diagramming techniques, among which are those mentioned in the introduction to this section. It also supports a customised version of NIAM called ATLIAM [1] which is capable of exporting an EXPRESS schema of a NIAM diagram and of exporting and importing NIAM diagrams by way of STEP files. Also in CGE is a variation of the PetriNet formalism (called CombiNet [13]) which is used by COMBINE partners to model the project flow constraints with respect to the invocation of design tools in a given project context.

2.2 Modelling requirements in COMBINE

In any large scale multi-partner model development project there are several modelling support issues which need to be dealt with. Discussed in the context of COMBINE, these include:

- *Communication and integration of aspect models:* the development of the building model is based on a cyclic development process where DT teams supply the data

view of their DT in the form of an EXPRESS model (referred to as an aspect model). These aspect models are taken as input by the central IDM development team for constructing a common integrated model. This process employs a combination of top-down structuring and bottom-up expansion of the growing IDM. Resulting drafts of the IDM are subsequently distributed to the DT teams to inspect and refine. This process of iterative refinement may continue for many months and during this time all partners concerned in this process must be kept up to date with the changing schemas.

- *Documentation:* during the model generation process, the communication of models and model constraints is an intensive interaction between various teams in the project. All decisions and justifications for decisions must be permanently documented as they are reached in the project.
- *Mapping definitions:* it is through the definition of the IDM to aspect model mappings that the consistency and adequacy of the IDM is totally checked by its *clients* (those who will use the data in the IDM). The mappings determined by the aspect model teams need to be defined formally, as these mappings describe the required interface between the DT and the IDM.
- *Support for multiple views:* the integration process is very hard to accomplish systematically, let alone automate (the required "meta modelling knowledge" is lacking), so it remains a tedious and non deterministic process to combine multiple views into one coherent whole. To support this integration process the modelling environment should provide methods to enable views to be integrated into a central model and to document the source of entities and structures in the integrated model.

To support these requirements, an ideal modelling support environment (MSE) needs mechanisms for: easy communication of models (in EXPRESS); generation and manipulation of multiple views of a model in a variety of formalisms; annotation of the model with documentation; and flexible update management to support iterative updates of the model. Other important features would be support for the definition of inter-model mappings and for integration of models, although it is recognised that, owing to the strong creative element in these tasks, automated tools are as yet unachievable. Another important issue is the ability to rapidly prototype a resulting operational system. MSEs should provide facilities to allow instantiation of models to be tested in a run time environment, eg. to test mapping specifications and enhance the understanding of the underlying model. In the next two sections of the paper we describe a modelling support environment and tools which have been developed for use in the COMBINE project.

3. EPE: A Modelling Support Environment

To support the modelling process described above EPE (EXPRESS Programming Environment) has been developed to explore the problems of schema evolution and management.

EPE is constructed using the MViews object-oriented framework. This framework provides a set of abstractions for constructing software development environments that support multiple graphical and textual views with in-built consistency between views [14]. MViews has been reused to build an object-oriented software development environment, SPE (Snart Programming Environment) [15, 16]. SPE integrates, in a single environment, tools to assist in systems analysis, design, implementation and maintenance of programs in Snart, an object-oriented logic language [16]. Other environments developed using MViews include an ER (Entity-Relationship) modeller for the database domain, and a graphical forms builder for specifying form layout and semantics for GUI applications [17].

Although developed as a framework for supporting conventional software development, the application of MViews to the development of EPE is a natural one. The most important concepts embodied in the MViews framework and SPE carry over to

other domains, including PDT, where analysis, design, implementation and maintenance of schemas and model instances are an integral part of a project.

One of the basic premises of MViews and SPE is that the user should be able to work with the most appropriate representation of a model at different development phases of a project. In EPE (and, by extension, COMBINE) this equates to graphical and textual views of varying degrees of complexity at different stages of the project. During analysis, simple graphical views which embody high level concepts and relationships between them are mapped out and manipulated. During early design, these simple graphical representations are fleshed out: constraints specified; attributes of entities added; and inheritance hierarchies fully specified. During late design, more detailed information becomes available which is often best manipulated in free form textual representations of portions of the schema. During implementation, the developed schema is compiled, checked for syntax errors, and detailed models have to be loaded and checked for consistency. Throughout the iteration of these stages and during maintenance, modifications can be made at any of the levels described above and must be propagated to all dependent stages. EPE provides integrated support for each of these activities, using the MViews consistency mechanism to provide the required inter-view consistency. In the following, we illustrate EPE using an example from the initial IDM model of COMBINE. We commence with a description of the tools and view types available at each stage of development before describing how views are kept consistent with one another.

3.1. Analysis

Figure 2 shows the type of graphical views used at the analysis stage. In this figure two analysis views showing portions of the inheritance tree for the *technical_system* entity, those entities dealing with ventilation and air conditioning, are specified using EXPRESS-G. Each view is constructed by direct manipulation using tools selected from a tool palette, to the left of the view.

The user is free to create as many views as is desired, and may freely lay out and populate each view, either with new information or with information entered into other views. The information in each view is mapped through to the canonical representation of the schema as the view data is entered, and any similarities or conflicts with the data resolved as it is created. This ability to construct multiple views permits both general purpose and specialised views to be constructed. The former may be used to obtain an overview of the system under construction, the latter to focus on more detailed parts of the system. The proliferation of views means that navigation tools are needed to quickly access desired information. EPE provides inter-view navigation using both menu-based search facilities and automatically constructed hypertext links.

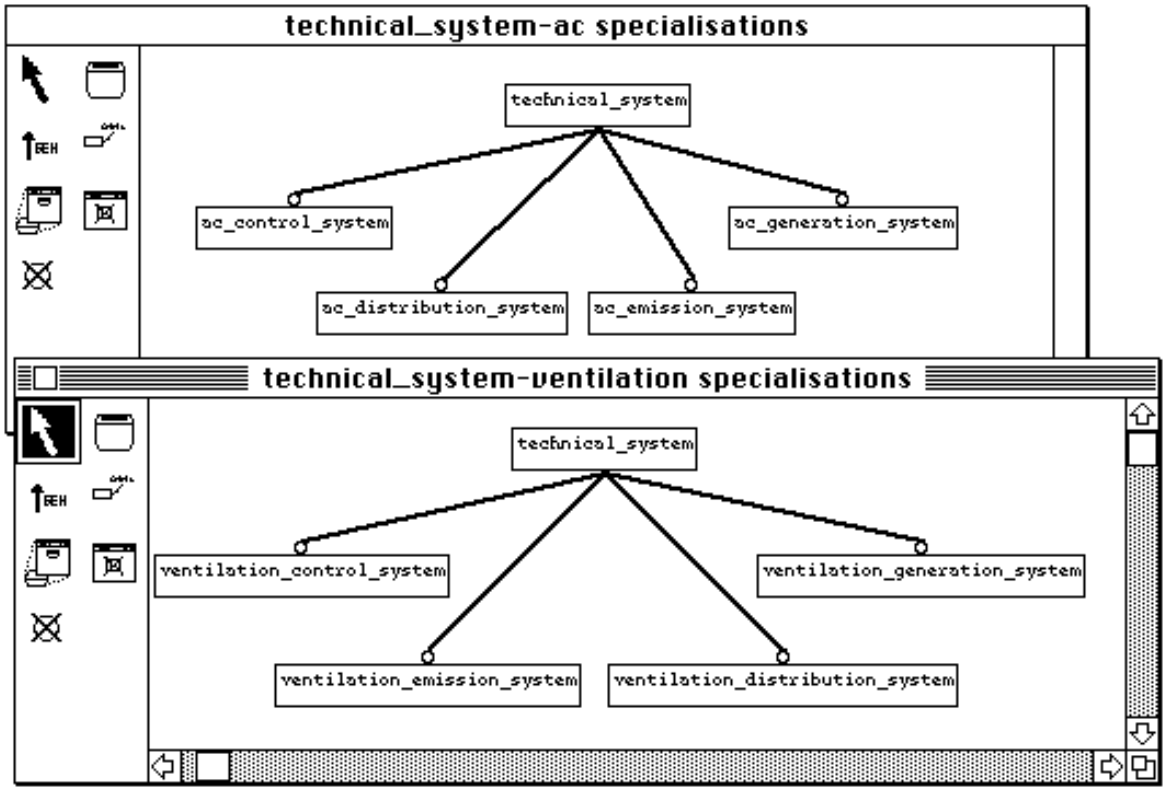

Fig. 2. Two high level inheritance specification for technical systems

3.2. Design

At the design stage attributes are specified, as shown for the technical system example in Figure 3. This can be done, as shown in this figure, with all entities in one graphical view, or by using two or more views. For example, the attributes of basic types may be presented in a separate view to those which define relationships to other entities, thus adding clarity. Again, there is no limit to the number of design views that can be constructed, and the hypertext navigation facilities are available to navigate both between design views, and between analysis and design views. As in the analysis views, all information in the design view of Figure 3 is mapped back to the canonical representation of the schema and all dependent views made consistent with its contents.

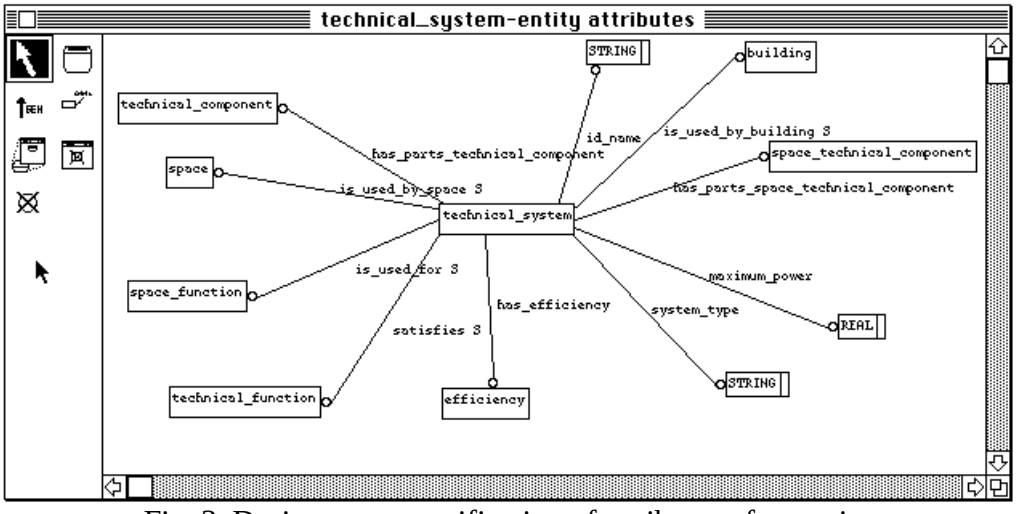

Fig. 3. Design stage specification of attributes of an entity

3.3. Late design

At the late design stage more detailed information needs to be entered. This necessitates a textual representation of the entity in the EXPRESS language as EXPRESS-G represents only a subset of what can be modelled in EXPRESS. For example, UNIQUE clauses, WHERE clauses, rules, and type information have no EXPRESS-G representation. Figure 4 shows an EXPRESS textual view which has been generated from the canonical information on the entity. This view encompasses all information found in all the graphical views which define the *technical_system* entity, such as the information in Figures 2 and 3. This textual view is editable, with the user free to make changes to any parts of the textual description in the view. Modified textual views are parsed and compiled to ensure they represent valid EXPRESS descriptions and their information passed back to the canonical representation.

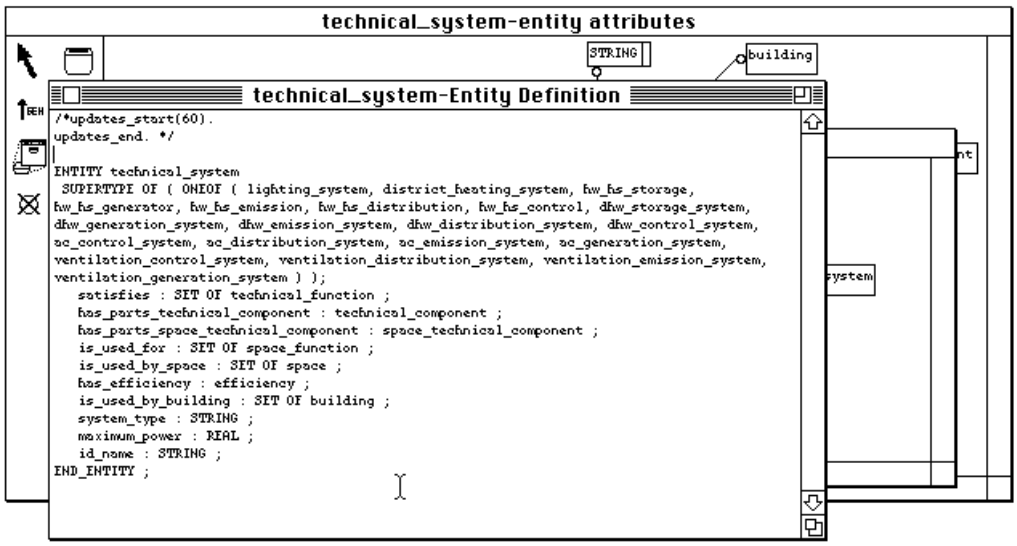

Fig. 4. Textual view derived from graphical views of an entity

3.4. Consistency between views

When changes are made to an EPE view, other views that share information with the updated view may become inconsistent and must be updated to keep the schema consistent across all views. All views affected by a change are notified and, in many cases, are automatically modified to reflect the change. This consistency mechanism works both between views of one phase of development and between views of different development phases.

As an example, Figure 5 shows a modification being made to a graphical design view. The modification tightens a constraint on an entity's value; the lower bound on the number of values in the SET definition is now known to be *1* and is entered in the definition. The information entered in this manner is checked as to whether it is valid EXPRESS syntax before being accepted and allowed to modify the schema being developed. The change is propagated through to the canonical form of the schema which is updated, then all dependant views are identified and notified of the change which has been made.

EPE propagates the change to the other affected views in the form of an *update record*. This record provides a complete description of any single change. How views react upon receipt of an update record depends on both the view type and the nature of the change. In the design view the modification updates the graphical representation of the design view according to the definition of EXPRESS-G syntax, as can be observed in the graphical design view at the rear of Figure 6. The modification is not propagated through to the analysis views as the modified attribute is not seen in these high level views. However, the attribute does appear in the textual late design view and it must be updated to be kept consistent.

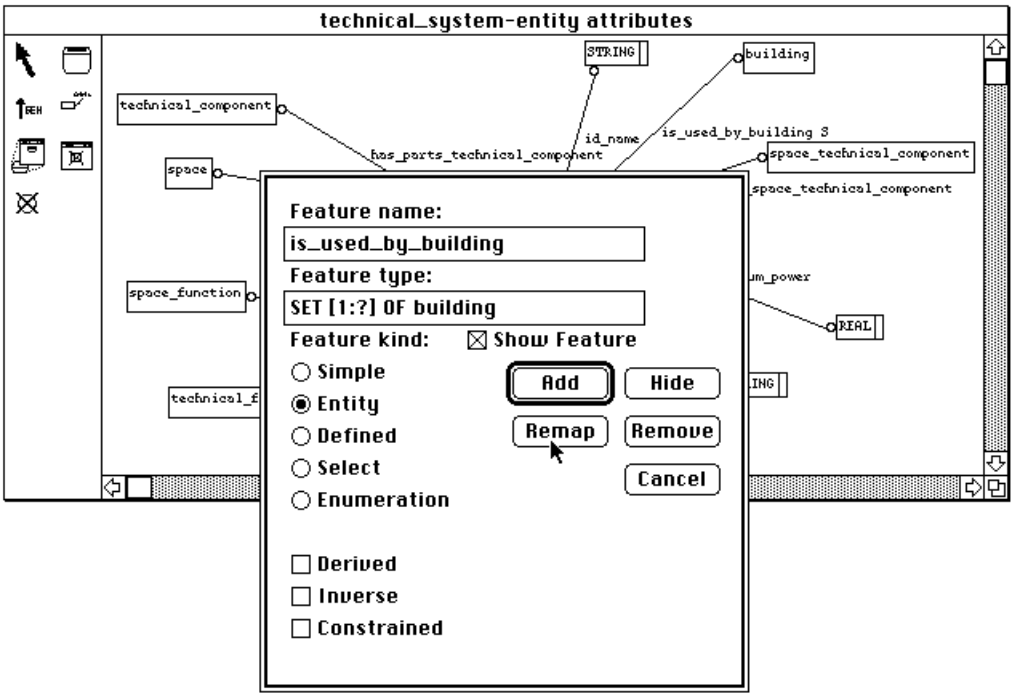

Fig. 5. Constraining the cardinality of an attribute at late design stage

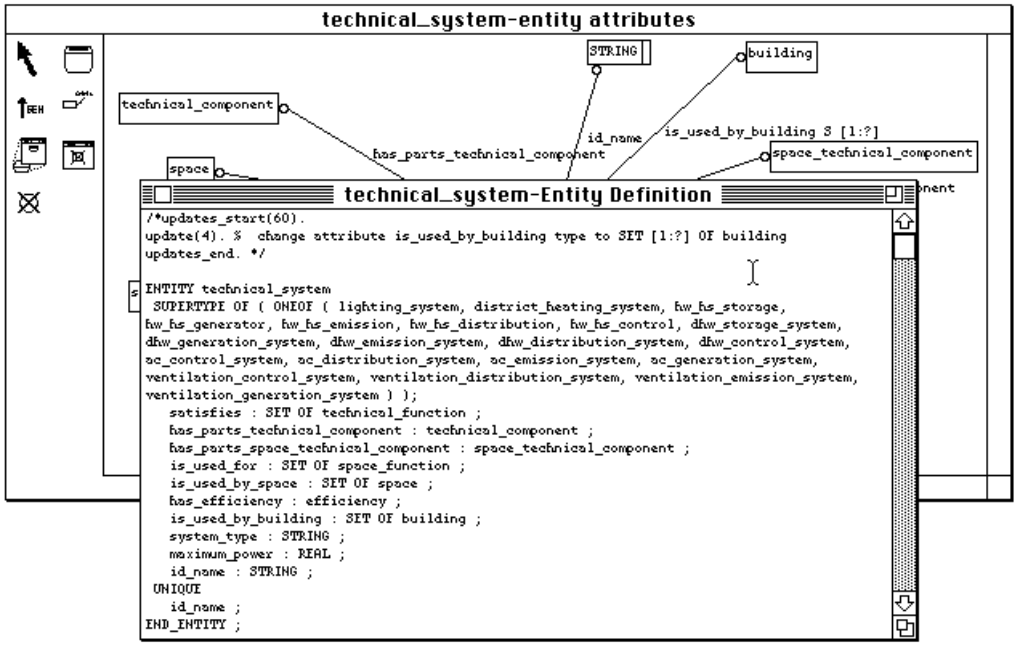

Fig. 6. The propagation of an update record to a dependant textual view

In the EPE system update records propagated to textual views are not applied automatically[1], though many of them could be. Instead the update records are displayed in the view and the user has control over which updates are applied at which time. As can be seen in Figure 6, the graphical update to the *technical system* entity generates an update record in the entity's textual view. If the user instructs EPE to apply the update the resulting view of Figure 7 is generated, where the notification of outstanding updates on this view has been removed, and the attribute definition has been automatically rewritten.

[1] This is a design decision for EPE and reflects a similar one made for SPE, as discussed in [19].

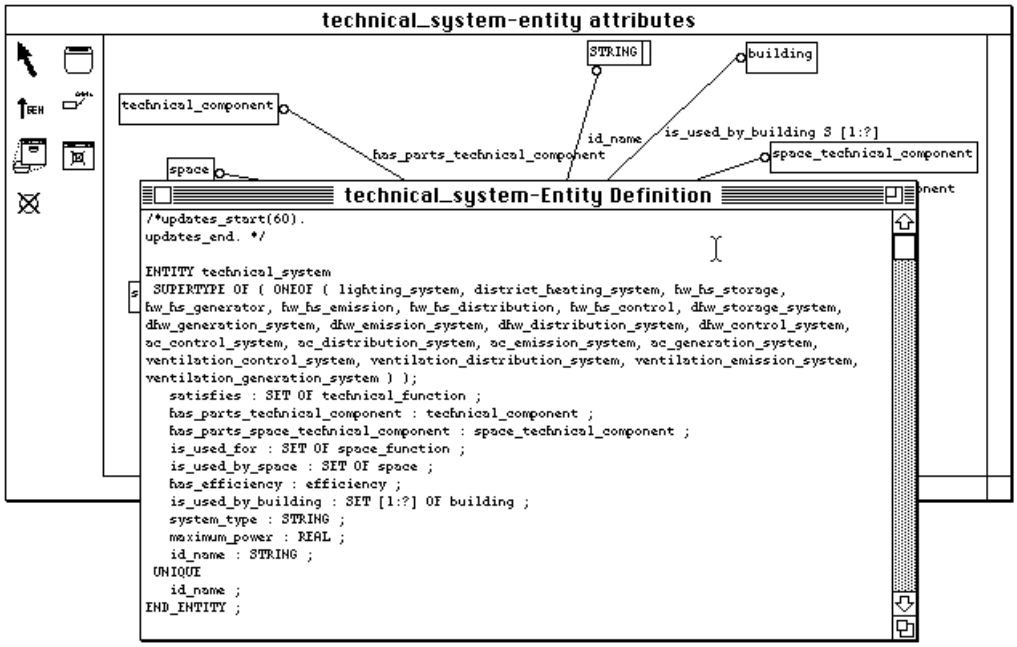

Fig. 7. The automatic application of an update in a textual view

This strategy of allowing the user to apply updates to textual views exists to handle the problems caused by the specification of a constraint on an attribute or entity in EXPRESS-G. For example, when the user specifies that an attribute is constrained (see the attribute dialogue box in Figure 5) they are denoting that the attribute takes part in either a *unique* clause or in a *where* rule. However, there is no way in EXPRESS-G to specify which one the attribute takes part in, or in what form. Therefore, an update of this form can not be automatically applied to a textual view and must be implemented by the user.

3.5. Documentation

In addition to providing a consistency mechanism between views, update records are retained in a persistent form in the EPE system. An update record browser and editor gives the user the ability to browse the changes that have occurred to an entity in the evolution of the schema and add further documentation to each update record. In this manner a portion of the documentation of the history of development of the system is automatically built up as work progresses. Having this update history on-line also allows system developers to trace back through previous design decisions while entities are further refined.

Figure 8 shows the update record browser displaying a list of changes that have been made to the *technical_system* entity based on the update records generated by the changes in various views. They include a renaming of the entity, changing the entity from an abstract supertype to a normal entity and the cardinality constraint imposed on the SET declaration of an attribute. The full details of the modification to the attribute is displayed in the top window highlighting the comment field that can be filled in by the system developer.

Other documentation support in EPE includes the ability to create textual documentation views (accessible via the hypertext navigation facilities) for entities. In such views, the various experts working on a model can document the reasoning behind decisions made and other information relevant to a particular entity and its attributes in a central and managed fashion. A useful feature of documentation views is that update records relating to the entity are automatically added as textual comments to the view as the entity changes.

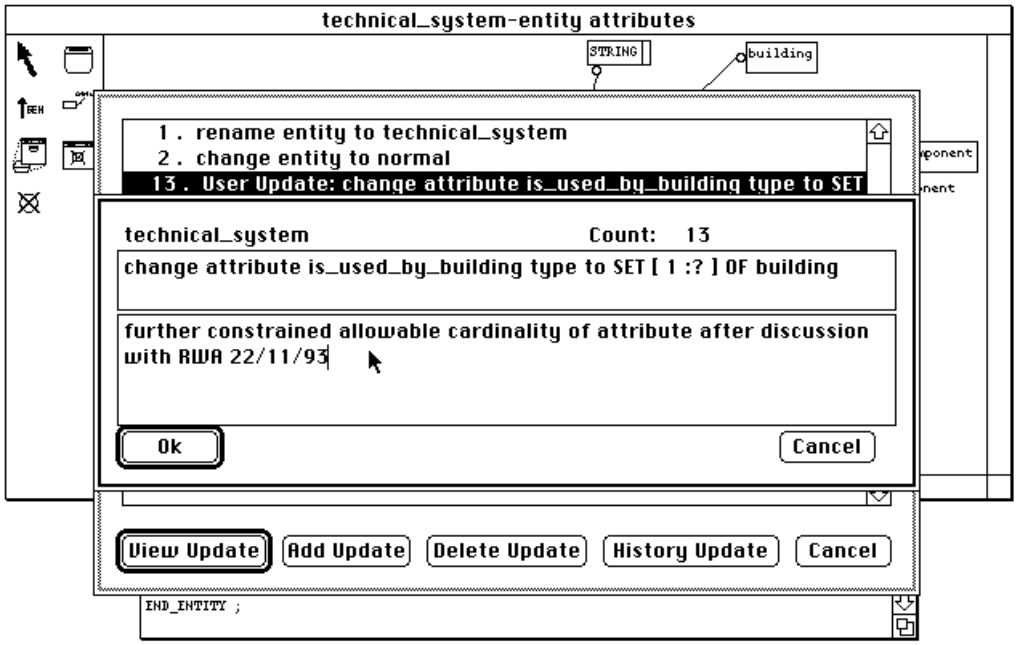

Fig. 8. The persistent update record viewer with documentation facility

3.6. View navigation

As can be seen from the preceding example, the EPE modelling environment captures a large amount of information about a model. This includes many graphical and textual views specifying various properties of the schema, entities, and documentation of changes made to the schema. In large systems this can lead to problems in finding a particular view or detailed information that has been entered. In the EPE environment the views and entities themselves act as a navigation and search facility for the project. Mouse clicks on graphical view components allow rapid access to other views containing that component. Figure 9 shows an example of this process for the *technical_system* entity. After clicking on the *technical_system* icon in the graphical view a list of all the graphical and textual views that index the *technical_system* is displayed and the user can navigate to these views in a hypertext like fashion.

4. Implementation Support

Multiple graphical and textual views are not only useful at the schema development stage. Similar methods for navigation, perusal and checking of the instances in a model are essential when dealing with a model as complicated as a building. To this extent the ability to view graphically the physical components of a building, and to navigate through the building structure at either a textual or a graphical level with the ability to switch to the opposite representation of a selected instance is particularly useful. Tools offering support for viewing and manipulating instances of the model are used by both the IDM development team and the aspect model teams to check instances of models, determining whether the structure is as they imagined when detailing their models and whether it accurately represents the information requirements of their models.

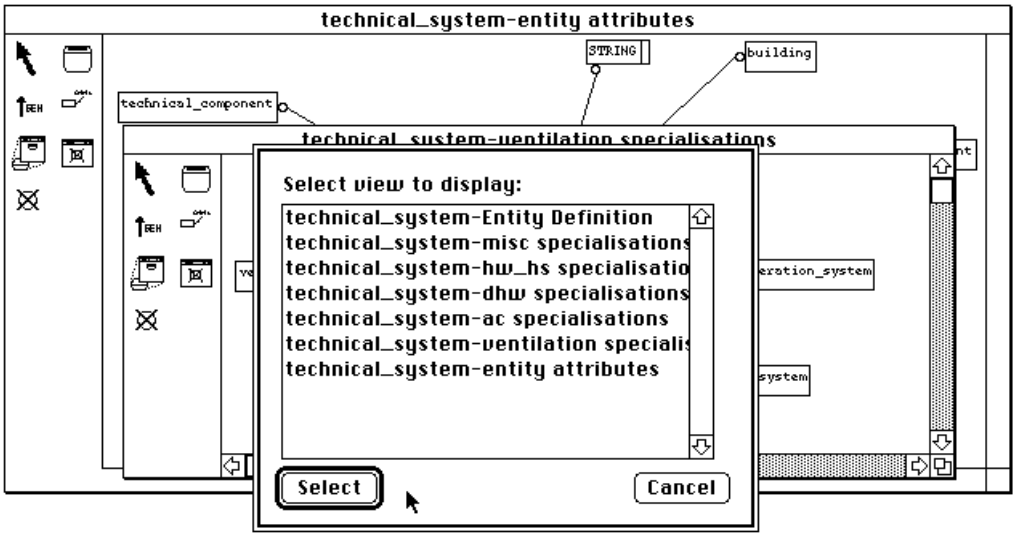

Fig. 9. The view navigator invoked for the *technical_system* entity

In this section we describe two tools developed in the COMBINE project for handling instances of a model. The first, an extension of EPE, provides all the benefits of EPE at the instance level as distinct from the schema level. The second is configured to recognise entities which have a graphical representation of a particular type. Entities with a known graphical representation can be viewed and navigated in their graphical form as well as in a textual representation.

4.1. EPE: an instance construction and browsing system

In addition to analysis and design, the EPE system also supports implementation and maintenance. The schema developed in EPE can, at any time, be compiled to an underlying object-oriented form and instances of the entities created and manipulated. Whole instance models, in the form of STEP data transfer format files, may be loaded into and transferred out of the system. Figure 10 shows an instance view of the *technical_system* entity modelled in previous figures.

Instance views consist of visual renditions of entity instances listing their attributes, the attribute's value, and the links between entity instances. Navigation through the instance model is via the entity instance links, which are displayed as small boxes in an instance view. These links can be expanded to view the data contained in the linked instances. The amount of data seen for each entity instance can be controlled by the user so as to present views which show, for example, all instantiated attributes, or all attributes which are references, or all attributes with a basic type, or some combination of types. In Figure 10 all attributes of an instance of a *ventilation_generation_system* are shown, including links which are not instantiated (shown as greyed boxes). This figure also shows the single reference to a building contained in the *is_used_by_building* attribute, and all the information in that instance of a *building.*

The user also has control over the way in which the information in entity views is laid out. This is defined through the use of a special purpose display language. This display language allows for displays which present sets of information in a variety of useful forms, such as bar graphs and tables, and allows templates of a given layout to be created. These templates can be applied to instances of the same type and will result in a visual display of the data in the same layout as was seen in the template.

In a similar fashion to the analysis and design views in the schema modelling section, multiple instance views with overlapping information can be created, and the elements (eg. entity attributes) in instance views are editable, allowing changes to be made to the underlying model instance. EPE's instance editor is a specialisation of CernoII, a runtime debugger and visualisation system developed as a companion for SPE [18, 19].

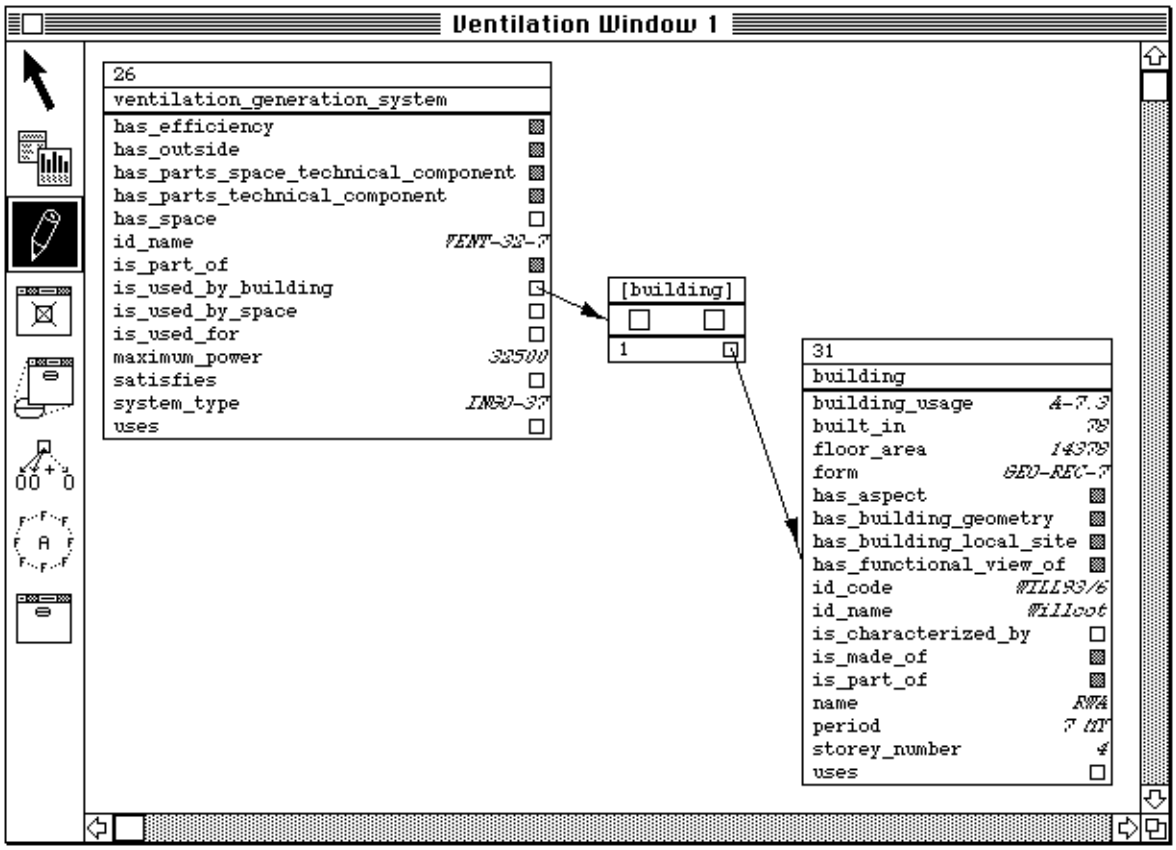

Fig. 10. Instance viewing and navigation

4.2. InSTEP: a graphical instance browser

During the first phase of the COMBINE project several tools were developed to enable participants in the project to manage the developed schemas and instance models. InSTEP is a tool that enables textual and graphical browsing, navigating and editing of a STEP file according to a specific EXPRESS schema. While this is not by itself remarkable, the InSTEP tool is unique in that it provides a graphical view of any model which is based on a schema which uses graphical entities of known types. Figure 11 shows both the graphical and textual view of a portion of a building. In this case the tool recognises entities that have a predefined geometrical representation [10, 20] and will render them in a graphical view which gives the user the ability to navigate through the instance model using either textual links or graphical links.

The textual browsing and editing of the model is performed on an object by object basis, though the user may follow relationships between objects in a hypertext like fashion. The graphical view displays a selected portion of the model and the user can set the level of navigation and selection at which the tool will operate at. In Figure 11 the user is navigating through the graphical model at a wall level, though other object types are displayed for reference.

In COMBINE this tool provides feedback to the conceptual task. Once a conceptual schema has reached a more or less stable version, DT teams and anyone who has a direct interest in the IDM are encouraged to browse instances of the schema and suggest modifications.

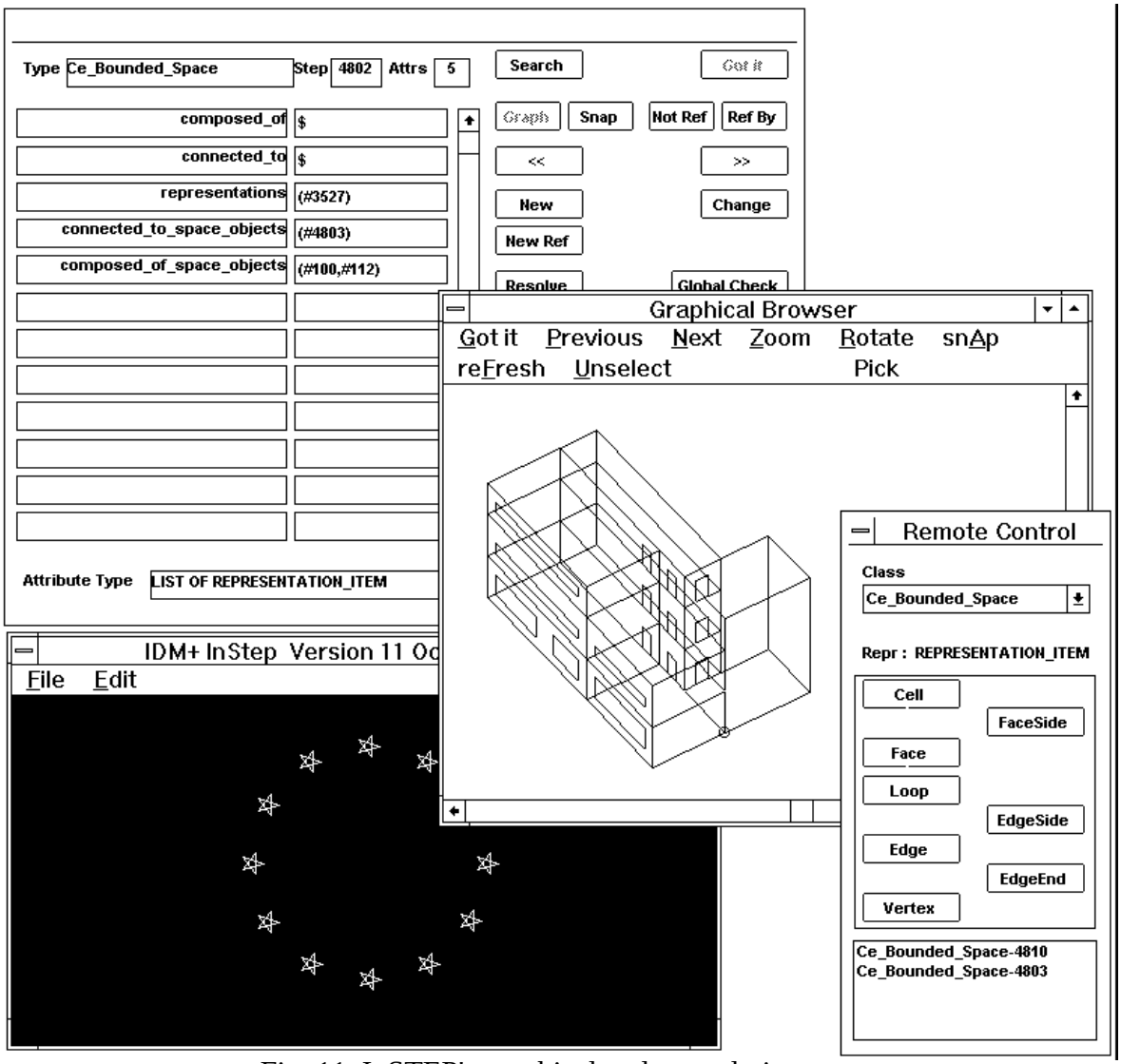

Fig. 11. InSTEP's graphical and textual views

5. Conclusion

This paper described the requirements for a totally integrated modelling and development environment in a project which has a major modelling component. To demonstrate what such an environment would look like and the benefits it can give in operation, a proof of concept system has been developed which provides an environment for modelling support from design phase through to implementation and maintenance of a project. This system demonstrates what would be required in a project which uses the ISO standard modelling paradigms of EXPRESS and EXPRESS-G for modelling and the STEPfile protocols for data exchange as is the standard developed for STEP, and used in many projects concerned with integration such as the COMBINE project in which this prototype was developed.

It is clear that the EPE system and InSTEP create an environment which supports the development, communication and integration of aspect models as well as the documentation requirements of an integrated modelling environment, as detailed in section 2. This support can be seen in the following properties of the environment: allowing the development of a model of a particular domain with total consistency from the earliest stages of the project right through to the implementation; allowing the user to develop multiple graphical and textual representations of a domain during analysis and allowing these to be expanded at the detailed design stage with new or updated views, many of them with overlapping information, yet still maintaining one canonical,

consistent representation of the model; giving support to the modeller by tracking all updates made to the model and propagating them to dependant views; providing on-line documentation with the developing model; providing easy navigation facilities around the various views in the model; at the implementation stage allowing the user to view instances of the developed models; and where they have a graphical component to view and navigate around the instances through the graphical representation.

However, there is still further support that could be offered in this integrated environment. In section 2 it was stated that an integrated modelling environment needs to support the modelling of multiple views with support for deriving the resultant integrated model, as well as the definition of mappings between aspect models and the integrated model. This can be managed to some extent (see [21] for an example of aspect model integration using SPE) as multiple views of a particular artefact, coming from any number of aspect models, can be detailed in EPE and the integrated model derived from information in those views. The harder problem is the ability to define and support mapping definitions between various aspect models in EPE, to support the mapping of data from any aspect model through to the integrated model and vice versa. This problem and the ability to manage concurrent model development in the MViews and EPE environment are areas of continued research by the authors of this paper [22, 23]. With the fruition of this current research it is envisaged that we can provide a truly integrated modelling environment suitable for large model development in projects of the scope of the COMBINE project.

References

[1] T. Vogel, *Configurable Graphical Editor: Users Guide*, TNO Institute for Applied Computer Science, Delft, The Netherlands, 91-ITI-382, February, (1991).

[2] B. Luijten, *A collection of PMShell papers*, Report B1-92-087, TNO Building and Construction Research, (1992).

[3] P. Poyet, A-M. Dubois and B. Delcambre, Artificial Intelligence Software Engineering in Building Engineering, *Microcomputers in Civil Engineering*, 5, pp 167-205, (1990).

[4] A. Boyle and A. Watson, *STEP Tools Review: Phase 2*, Computer-Aided Engineering Group, Department of Civil Engineering, University of Leeds, England, 26pp, February, (1993).

[5] ISO/TC184, *Part 11: The EXPRESS Language Reference Manual in Industrial automation systems and integration - Product data representation and exchange*, Draft International Standard, ISO-IEC, Geneva, Switzerland, ISO DIS 10303-11, August, (1992).

[6] ISO/TC184, *Part 1: Overview and fundamental principles in Industrial automation systems and integration - Product data representation and exchange*, Draft International Standard, ISO-IEC, Geneva, Switzerland, ISO DIS 10303-1, (1993).

[7] W. Gielingh and A. Suhm, *IMPPACT Reference Model: An approach for integrated product and process modelling of discrete parts manufacturing*, Springer Verlag, (1992).

[8] ATLAS, *ATLAS: Architecture, methodology and Tools for computer integrated LArge Scale engineering*, ESPRIT 7280, 37pp, February, (1993).

[9] G. Augenbroe and L. Laret, *COMBINE pilot study report*, CEC-JOULE Report, (1989).

[10] G. Augenbroe (ed.), *COMBINE Final Report*, CEC-DGXII, (1993).

[11] G. Augenbroe, Integrated Building Design Systems in the Context of Product Data Technology, *ASCE Journal of Computing in Civil Engineering*, 8(4), pp 420-535, October, (1994).

[12] A.M. Dubois, *COMBINE IDM conceptual development task*, COMBINE-Report, CSTB, (1993).

[13] TU Delft COMBINE Team and R. Amor, *COMBINE Project Windows Modelling Approach*, COMBINE 2 Working Document, (1994).

[14] J.C. Grundy and J.G. Hosking, Constructing multi-view editing environments using MViews, *Proc. 1993 IEEE Symposium on Visual Languages*, IEEE Computer Society Press, Los Alamitos, CA, pp 220-224, (1993).
[15] J.C. Grundy and J.G. Hosking, Integrated software development in SPE, *Proc. 13th New Zealand Computer Society Conference*, New Zealand Computer Society, August, (1993).
[16] J.C. Grundy, *Multiple Textual and Graphical Views for Interactive Software Development Environments*, PhD thesis, University of Auckland, Auckland, NZ, (1993).
[17] J.C. Grundy and J.G. Hosking, *Constructing integrated software development environments with dependency graphs*, University of Waikato, Department of Computer Science, Report No 94/4, (1994).
[18] S. Fenwick, J.G. Hosking and W.B. Mugridge, *Cerno-II: A program visualisation system*, University of Auckland, Department of Computer Science, Report No 87, February, (1994).
[19] J.C. Grundy, J.G. Hosking, S. Fenwick and W.B. Mugridge, *Connecting the pieces: integrated development of object-oriented systems using multiple views*, Information Engineering Report No 93/4, Information Engineering Section, Department of Electrical Engineering, Imperial College of Science, Technology and Medicine, London, November, 16pp, (1993).
[20] W. Rombouts and P. de Vries, *InSTEP Manual*, COMBINE 2 Working Document, (1994).
[21] W.B. Mugridge and J.G. Hosking, Toward a Lazy, Evolutionary Common Building Model, *accepted for publication in Building and Environment*, (1995).
[22] R.W. Amor and J.G. Hosking, Multi-Disciplinary Views for Integrated and Concurrent Design, *The Management of Information Technology for Construction*, First International Conference, Eds K.S. Mathur, M.P. Betts and K.W. Tham, Selected (refereed) papers, Singapore, 17-20 August, pp 255-268, (1993).
[23] R. Amor and J. Hosking, Mappings: The Glue in an Integrated System, *The First European Conference on Product and Process Modelling in the Building Industry*, Dresden, Germany, 5-7 October, (1994).