Architectural design-by-features

Paper published as: van Leeuwen, J.P. and H. Wagter (1997). Architectural Design-by-Features. Junge, Richard (ed.) 1997. CAAD futures 1997. Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures held in Munich, Germany, 4-6 August 1997. Kluwer Academic Publishers, Dordrecht, p. 97-115.

ARCHITECTURAL DESIGN-BY-FEATURES

JOS P. VAN LEEUWEN [a]
HARRY WAGTER [a + b]

[a] *Eindhoven University of Technology, The Netherlands*
Faculty of Architecture, Building, and Planning
Building Information Technology
`http://www.calibre.bwk.tue.nl`
email: `j.p.v.leeuwen@bwk.tue.nl`

[b] *Origin International Consultancy*
Eindhoven, The Netherlands.
email: `harry.wagter@nlehvips.origin.nl`

Abstract

Design tasks, in particular architectural design tasks, have been found hard to support by means of computers. The main reason for this is that design is a problem solving process, which requires a dynamic way of handling information involved in the design process. The research presented in this paper focuses on this aspect of CAAD: the support of design tasks with dynamic, flexible information modelling techniques. The basic concepts for the developed approach is taken from the field of Feature-based modelling. We briefly review these concepts and then interpret and transport them to the context of architectural design. In defining types of Features, a distinction is made between domain-specific Features and generic Features for which we propose a classification. A framework for the definition and modelling of Features is discussed as well as a prototype Feature-based Modelling Shell based on this framework.

1. IT Support for Design

Computer support for processes of design has been subject of research for nearly as long as computers have been around. The great amount of today's approaches to supporting architectural design ranges from the development of computer-aided drafting to the application of artificial intelligence in shape-grammars, rule-based design, case-based design, and many more. Common to these approaches is the need for representing and manipulating a multitude of information that is somehow involved in the tasks of design. Modelling information for support of design tasks is subject of research presented in this paper.

Product Modelling (PM) is an approach that concentrates on information modelling for the support of communication between the many different participants

in design processes. It addresses problems such as the distinct views of participants on the product of design and the definition and structure of data involved in these views. This results in the definition of data models that represent the information in various domains of design and engineering. Much of the research in the field of PM focuses on the standardisation of these information models in processes of communication [ISO 1994], also in the Building & Construction (B&C) industry [Tolman and Wix 1995]. However, information models do not merely serve the communication at certain stages in the design and building process. The representation of information during tasks of design, e.g. for support of the analysis, generation, and evaluation of information, is an equally important role for information models. Yet this role of information models imposes fundamentally more complex requirements on the definition and structure of models, mainly due to the dynamic nature of the process of design. Successful research in relation to this subject is found in the work of Eastman et al. [1991, 1995] and Ekholm and Fridqvist [1995, 1996].

1.1 DYNAMIC NATURE OF DESIGN

One of the main problems that need to be addressed when defining information models for architectural design is caused by the dynamic nature of design. During the process of design, information is not treated as static data, but is invariably subject to change. This is due to the problem-solving character of design, which involves the search for information, analysis, structuring, interpretation, and evaluation of information in repeating cycles. Most important, it involves the generation of information during this cyclic process and combining this new information with known data, defining new information-structures that lead to design-solutions. We conclude that the definition and structuring of information during the cyclic process of design is a key issue in supporting design tasks with information models and design information management systems. The dynamic nature of design is a precondition for creative design; its preservation is perhaps the most challenging requirement for the development of design information models and design support systems.

1.2 REQUIREMENTS FOR MODELLING DESIGN-INFORMATION

One of the aspects that form the dynamic nature of design is the flexible manner of handling information. In the problem-solving process of design, the search for solutions, alternatives, and optimisations require that information is constantly analysed, interpreted, generated, and restructured. Information models that are meant to support design tasks will have to show sufficient flexibility, allowing for definition and redefinition, structuring and restructuring of information. Information models will need to evolve in order to reflect the evolution of the design.

Evolution in design does not only occur during the course of a single design project. The fact that designers learn makes them change their approach to solving design problems, finding new techniques, new rules, new concepts. Mitchell [1990] recognises this stylistic evolution as an essential component of creative design which

must be addressed by future CAD systems. For information models, this means that the definition and structure of information should enable adaptation to changes in requirements, insights, methods, and conceptual basis, etc.. On a larger scale in the building industry, there is a similar form of evolution that is manifested in the development of new products, materials, or construction-techniques. This evolution too requires information models to adapt to changing requirements.

1.3 EXTENSIBILITY AND FLEXIBILITY OF INFORMATION MODELS

The requirements for information models in terms of evolution and adaptation along the design process and the development of the design discipline, can be translated into required extensibility and flexibility of information models.

Extensibility Extensibility of information models allows designers to extend the set of definitions that constitute a conceptual information model with definitions that represent specific concepts and notions. In this manner, the domain of an information model can be extended into areas that are specified by the user of the model. This extension may concern, for instance, a specific style of design, style-rules or conventions for a particular building project. Also, new information-definitions will represent new technologies of construction, new products and materials, and their individual characteristics. For construction-systems that are based on industrial production of components, the definition of information will accurately represent the characteristics, requirements, and assembly-procedures, etc. of the components and these systems as a whole.

Flexibility Enabling the definition of entities of information that accommodate specific requirements in design-tasks is one aspect of supporting the dynamic nature of design. A second aspect addresses the flexibility of information models. Flexibility is required when new definitions of information are added to a conceptual information model: extension of the model requires flexibility. Newly defined information entities will define relationships to existing entities, and reversely, existing entities will need to define relationships to new entities. This requires a significant level of flexibility in the definition of information entities, allowing properties or attributes defining relationships to be modified or added.

Flexibility is also necessary to accommodate the design process as a problem solving process, which, apart from generating information, also continually involves interpreting and restructuring available information. Therefore models that are to represent this information, not only at a final stage but also during the course of design, will have to follow this process of restructuring. Again it requires properties or attributes representing relationships to exhibit a sufficient level of flexibility.

Feature-based approach The research presented in this paper addresses the requirements of extensibility and flexibility of information models by applying concepts and techniques from Feature-based Modelling (FBM) in the context of architectural design. Feature-based Modelling is developed in areas of mechanical

engineering and industrial design, concentrating on the description of part-geometry using high-level elements that correspond closely to the terminology of the specific domain of design. The structure of the models that are used in this approach, and the openness offered to modelling systems that employ this approach meet the requirements of flexibility and extensibility, which we have recognised to be important for architectural design systems as well. In order to investigate the applicability of the Feature-based approach in the context of modelling architectural design information, we first analyse the concepts of Feature-based modelling in the area of mechanical engineering. These concepts are then recapitulated in the context of architectural design, leading to the formulation of a Feature-based approach to modelling architectural design information.

2. Concepts of Feature-Based Modelling

2.1 ORIGINS OF FEATURE-BASED MODELLING

In disciplines of mechanical engineering and industrial design, the practice of using computers in the design and production processes has given rise, in the early eighties, to the need for richer information models. Geometric models of a product did not suffice for advanced evaluation of designs, such as manufacturability evaluation, or for integration in for instance process planning tasks. For these purposes it was necessary to develop models with a higher level of information than just geometry. This resulted in the development of high-level information entities which are called Features. Since form is the major aspect of interest in the disciplines mentioned, a strong focus is still today on the development of Form Features.

Many definitions of the term Feature have been given in literature, depending on the context and the purpose for which Features are applied. A definition for Form-Features given by Shah [1991a] seems to cover their common notion: '[Form] Features are generic shapes with which engineers associate certain properties or attributes and knowledge useful in reasoning about a product.' In research and practice of Feature modelling Features generally describe characteristics or parts of a product, mostly concerning the manufacturing of the part, by describing its surface or shape and the technological attributes that are associated to for instance tools and operations.

One quality of Feature modelling that appears to be important for our research is that parts are not represented by a complete, rigidly defined model. The definition of Features is independent of their future context. Therefore the structure of a part in a Feature model is not prescribed, but is composed by creating Features and structuring the Feature-model during design.

Feature modelling technology focuses on the description of the shape of parts, using Form Features. However, there are many other kinds of Features involved in the description of shape, as well as in other aspects of interest in modelling a product. Thus, also more generic definitions of the term Feature are found in literature, such as 'a set of information related to a part's description'. In an

assessment of Feature technology, Shah [1991b] gives the generic definition: 'Features are elements used in generating, analysing, or evaluating designs', and in [Shah 1991a]: 'A Feature is any entity used in reasoning about the design, engineering, or manufacturing of a product.'

There are probably as many classifications of Features as there are definitions of the term Feature, again depending on the context of using Feature technology. Some of the classes of Features generally found are: Form Features, Precision Features, Material Features, Assembly Features, Performance Features, Pattern Features, Connection or Constraint Features, Application Features, etc. An example of some typical Form Features is given in figure 1. Although there are efforts to standardise the classification of Features, it is generally accepted that no collection of Feature definitions can be complete. An important requirement for Feature modelling systems is the level of extensibility of these systems with the definition of new types of Features.

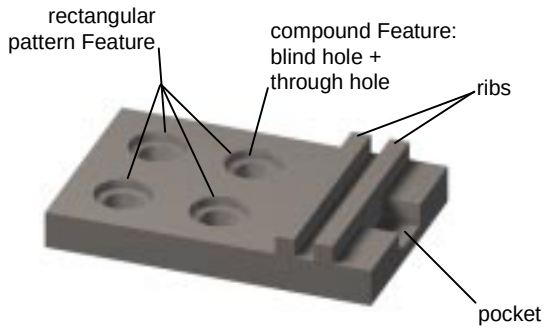

Figure 1 Some typical Form Features.

2.2 FEATURE RECOGNITION AND DESIGN-BY-FEATURES

Feature technology has developed into two main approaches towards high-level modelling of products. One approach is Feature recognition [Henderson and Chang 1988, Laakko and Mäntylä 1993, Meeran and Pratt 1993]. Using geometric models of product-parts, a model of Features is built up after analysis of the geometry. At first this was a human-assisted, interactive process, later Feature recognition was automated. The Feature model could then be used for evaluation and creation of for instance process plans or NC programming.

Among the main problems with the Feature recognition approach are the limited type of information that can be recognised from geometry, and the very complicated procedures that are necessary to extract meaningful Features. Another important drawback is that the geometric model needs to be completed before the recognition process can start. After changes in the geometry, this process needs to be repeated. Moreover, valuable information that is already available during geometric modelling cannot be included in the model and will be lost. An important advantage of Feature recognition is that geometric models created without specific context can

be evaluated and interpreted using knowledge from a particular context, e.g. a specific manufacturing process.

The logical successor of the recognition approach is the design-by-Features approach [Shah and Rogers 1988, van Emmerik 1990, De Martino 1994]. Geometry is no longer the basis for the model that is now built up by creating Features. In Feature models, geometry now forms one aspect of the product-information: also non-geometric information is included from the beginning of the modelling process. Some of the main advantages of the design-by-Features approach are that high-level entities are used to model a design, that these entities correspond (better than geometry) to the terminology of the domain of design, and therefore that the resulting model will more accurately represent the actual design. The error-prone procedures of recognising high-level information from geometry is no longer necessary.

Recent research has resulted in systems that combine the two approach of Feature recognition and design-by-Features [De Martino 1994].

2.3 ABSTRACTION OF THE FEATURE-BASED MODELLING CONCEPTS

The definition of Features in the original field of development of Feature technology is an important basis for the theory developed in the research presented in this paper. Since the focus in mechanical engineering is on describing shape and geometry, we have to consider the definition of Form-Features as was quoted above. However we anticipate that in architectural contexts the technology of Feature modelling will require a more general definition of Features than one that is restricted to the shape or geometry of physical parts of a building.

Shah warns that too general a definition renders discussions on the subject meaningless [Shah 1991a], he then concentrates solely on the development of Form-Features. We have concluded that architectural information modelling cannot permit itself this luxury and will have to deal with a more general concept of Features. Architectural products cannot be described by an apparent decomposition of physical parts. Architectural design involves many concepts and notions that are not directly, not at all times, or even not at all related to physical parts in a building. Some indicative examples are concepts of space, function, costs, safety, comfort, etc.. Moreover, these concepts apply to both early and later stages of design, and are relevant to different levels of abstraction in looking at a building-design.

In this research, we have abstracted an understanding of Features from the original Feature technology, which allows us to involve both physical and abstract concepts in the definition and modelling of Features. This notion of Features will be applicable to multiple levels of abstraction in architectural design information.

In the next section we introduce a general definition of the term Feature in the context of architecture and propose a generic classification of architectural Features.

Paper published as: van Leeuwen, J.P. and H. Wagter (1997). Architectural Design-by-Features. Junge, Richard (ed.) 1997. CAAD futures 1997. Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures held in Munich, Germany, 4-6 August 1997. Kluwer Academic Publishers, Dordrecht, p. 97-115.

3. A Feature-based approach to modelling architectural design information

3.1 DEFINITION OF ARCHITECTURAL FEATURES

In earlier papers [van Leeuwen et al. 1995, 1996] on this research we have defined a Feature as follows:

> *A Feature is a collection of high-level information defining a set of characteristics or concepts with a semantic meaning to a particular view in the life-cycle of a building.*

Breaking down this definition into three aspects, it can be noted that:

- Features represent semantics of a building (or its design);
  As in other Product Modelling approaches, we attempt to use definitions of information entities that closely relate to the terminology, and therefore semantics, of the domain of application. Important however is how these entities are chosen and how their inter-relationships are defined. This is framed by the second aspect:

- Features are the formal definition of characteristics or concepts;
  In this aspect, the Feature-modelling approach is fundamentally different from the 'traditional' product modelling approaches. In the latter approaches, physical components often form the basic entities in the structure of information models, their properties being the attributes of these entities. In the Feature-based approach, properties or, more general, characteristics and concepts, are entities of information themselves. As a result, the relationships between components and properties of components are much more flexible. This means, for example, that a property that has not been previously assigned to a certain type of component can, at any given moment, be added to the information structure representing the component and its properties. Also, in this manner properties can be shared by different components. It should be noted that characteristics and concepts are not restricted to static data, but may define behaviour or procedural knowledge as well.

- Features are related to particular views in the life-cycle of a building.
  The concept of views is one very common to product modelling approaches. In the Feature-based approach, this leads to the definition of libraries of Feature-types that represent a particular domain of information modelling. In this context we define Generic Feature Types as those types that are part of the domain of B&C as a whole, and Specific Feature Types defining those Features that are specific to a particular view or sub-domain in the B&C industry. This can be the view of one participant, a view related to a particular project, or for instance a domain deriving from a particular industrial construction technology.

3.2 LEVELS OF ABSTRACTION

The definition of a Feature given above does not limit information to a particular level of abstraction or detail. Although not explicitly so stated in Feature-definitions in mechanical engineering, the Feature-technology in that area follows a strict hierarchy of the description of a product, namely the decomposition of a product (or assembly) into parts, and of parts into Features (see figure 2).

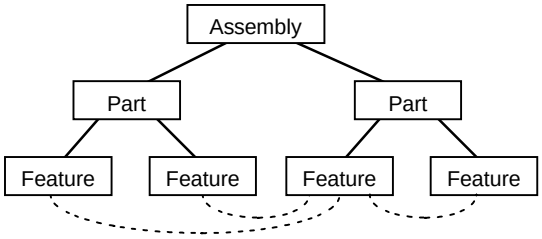

Figure 2 Product hierarchy in mechanical engineering.

Levels of abstraction are often related to different stages in design. In architectural design information is generally involved in several levels of abstraction at the same time, or shifted between these levels. This research does not assume a predefined hierarchy, thus allowing Features to appear at any level of semantic abstraction, and allowing Features to be migrated from one level to another. The levels of abstraction are not intrinsically predefined in our theory, therefore abstraction layers from other approaches, design-methods, or standards can be adopted and incorporated in the structure of information models.

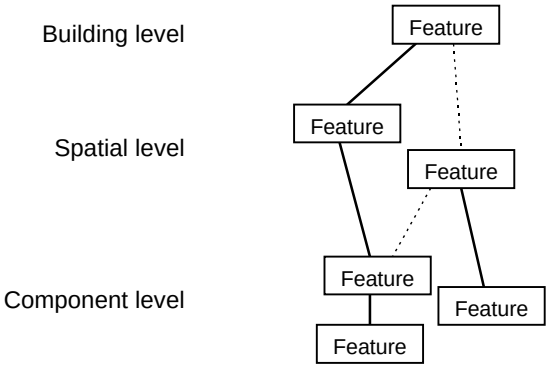

Figure 3 Features at different levels of abstraction.

Figure 3 shows how a network of Features covers multiple levels of abstraction. This network incorporates information concerning a building as a whole, as well as information at the more detailed levels. Different types of relationships between Features establish the structure of information. This approach has also been chosen to allow information modelled at early stages, often not very detailed, to evolve as more details become known during later stages of design.

3.3 GENERIC VERSUS SPECIFIC FEATURES

Although the approach advocated in our research aims at supporting dynamic design-tasks, this does not mean that we oppose any kind of standardisation of data-definition. An important role in standardisation is appointed to so-called core models

which have the function of intermediate model between communicating or data-sharing participants. However, standardisation should not conflict with the requirements of extensibility and flexibility. Therefore, the structure of core models need to meet these same requirements.

Generic Feature Types are those types that form a core model that maintains sufficient flexibility. They are formalised, common concepts and represent terminology accepted and used by the B&C industry. Although typical relationships between Generic Feature Types do occur, they do not form a rigidly defined structure but retain a great deal of independence. The exact and complete context of Features is defined only during the modelling process itself. The next section addresses the classification of Generic Feature Types and includes a proposal for the main categories. However, the task of classifying and defining the Generic Feature Types clearly needs to be left to international standardisation institutes, such as the ISO.

Feature definitions that are not part of the common domain in B&C are called Specific Feature Types. They represent particular views or disciplines, or are related to particular styles, projects, applications, or any other specific context. Standardisation may also play an important role in the definition of Specific Feature Types, e.g. for the standardisation of branch-models. But Specific Feature Types also enable the extension of standardised models with definitions of information-entities that serve certain specific needs.

3.4 CLASSIFICATION OF ARCHITECTURAL FEATURES

We present a proposal for the main categories in a standardisation of Generic Feature Types. Table 1 lists 5 main categories with some examples of sub-categories and a brief indication of the contents of each of the categories. The configuration of the categories is based on an analysis of building- and design-related information and on a survey of existing classification tables, including a proposal by [Woestenenk 1995] for international conversion tables for parts and functions.

This classification is not intended to be complete. Even when standardised, the set of Generic Feature Types cannot be presumed to be complete and must retain a structure that supports the required flexibility for adaptation to future developments. However, the proposed categories are believed to be a valid starting point for standardisation and for the definition of Specific Feature Types.

3.5 A FRAMEWORK FOR FEATURE MODELLING

The notion of Generic Feature Types and Specific Feature Types, and the proposed classification have been the basis for the development of a framework for Feature modelling. From the original technologies of Feature modelling, the design-by-Features approach has been found most appropriate for the problems that challenge us in architectural design. Feature recognition, as in mechanical engineering would soon appear to be too limited, since it excludes valuable information in early stages from the modelling process.

Paper published as: van Leeuwen, J.P. and H. Wagter (1997). Architectural Design-by-Features. Junge, Richard (ed.) 1997. CAAD futures 1997. Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures held in Munich, Germany, 4-6 August 1997. Kluwer Academic Publishers, Dordrecht, p. 97-115.

Table 1 Proposed classification of Generic Feature Types [van Leeuwen et al. 1996].

GENERIC FEATURE TYPES	BRIEF DESCRIPTION
Form Features Morphological Features Topological Features Geometrical Features	Form Features describe the form, shape, or topology of other entities in the building model, which can be physical entities, but abstract as well.
Physical Features Compositional Features Material Features Composition performance Features	Physical Features form the group of Features that describe the physical qualities, performances, and requirements of entities in the building model.
Context Features Design conceptual relation Features Interface Features Performance dependency Features Constraint Features	Context Features define characteristics and concepts that form relationships between entities, such as dependencies, adjacencies, and for instance constraints, such as tolerances.
Procedural Features Planning Features Preparation Features Staging Features Integration Features	Procedural Features include the type of information that somehow describes procedures related to the construction process, from the preparation of the work to the activities on site.
Life-cycle Features Functionality Features Operation Features Quantitative Features Maintenance Features Re-usability Features Security Features	Life-cycle Features are the ones that describe concepts and characteristics that are especially relevant during the complete life-cycle of the building, particularly when it is being used, maintained, revised, renovated, or is given new functions. Also quantitative information such as costs falls within this category.

The design-by-Features approach, employing Features as the fundamentals for representing design information, seems to offer much more potentials in the development of a design-system that corresponds to the terminology of the design-domain. However, it is expected that techniques of Feature recognition, when combined with the design-by-Features approach, will be of significant importance in the support of design-tasks, especially when integrated in design-systems' interfaces to the underlying Feature-models. Our research currently focuses on the design-by-Features approach, recognition techniques will be subject of study at later stages.

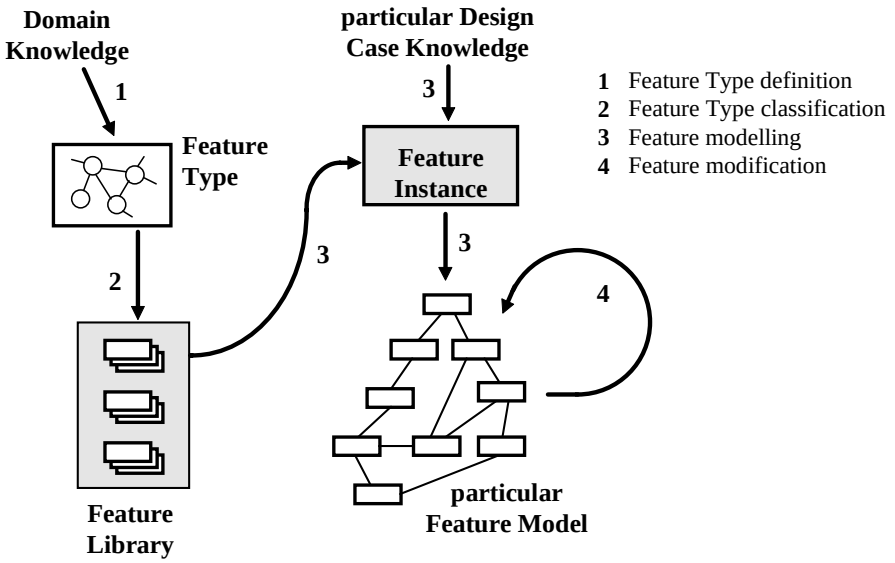

Figure 4 Activities in Feature-based modelling [van Leeuwen et al. 1996]

The framework for Feature modelling describes the activities involved in defining types of Features and creating models of Features. To accommodate these activities, a schema for the definition of Feature Types and their instances has been developed. Figure 4 shows the activities of respectively Feature Type definition (1), Feature Type classification (2), Feature modelling (3), and Feature modification (4).

Feature Type definition is the activity of formalising domain knowledge into the definition of information entities. When a collection of information with relevant coherence has been identified, it can be formally described as a new Feature Type, either from scratch or based on previously defined Feature Types. Feature Types define the structure of their instances, by their state and behaviour. The state of Features is contained in a set of variables or attributes that obtain their values during the modelling activities. Behaviour of a Feature is a set of procedures that implement how a Feature reacts, either to events from outside the model, e.g. user-interactions, or to events from within the model, e.g. modifications to related Features. Relationships to other Feature Types can be expressed in attributes or be included in the behaviour of a Feature. Relationships are to be included in the type definition inasmuch as they are generic relationships, i.e. relevant for every Feature instance that will be based on this new type. Relationships that may occur for certain instances of the new type but that do not have generic relevance, should not be defined as part of the Feature Type, but will be defined during the modelling activities. This distinction between typical and occasional relationships is crucial for the flexibility of the Feature model that will result from Feature Type definitions.

Feature Type classification is the activity of classifying Feature Types into Feature Libraries which have the function of representing a particular domain. Feature

Libraries may be further divided into sections and mainly serve the purpose of organising collections of Feature Types that will often display a certain coherence by means of inter-relationships.

Feature modelling, or instantiation, is the sequence of identifying a relevant and coherent set of information in a particular design case; selecting a Feature Type that is appropriate for the representation of this information (however, it is possible that such a Feature Type does not exist and needs to be defined first); instantiating the Feature Type and its attributes (the new Feature instance may display some kind of behaviour in reaction to this); giving the new Feature its position in the Feature model, in relation to the Features already there.

Feature modification involves a set of activities that modify the state of Features or the structure of the Feature-network: attributes of Features may be given new values; Features may be removed or replaced entirely; relationships between Features may be modified, added or removed. During these actions, Features may react in response to the many different events of modification.

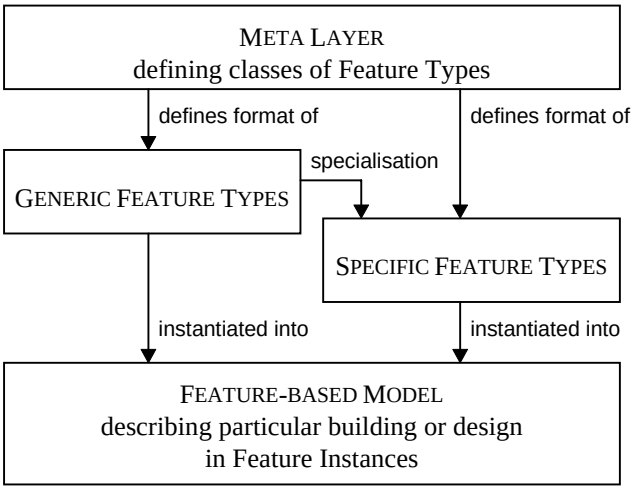

Figure 5 Infrastructure of Feature-based modelling

Schema for the definition of Feature Types and Feature instances The above activities require an infrastructure that provides the formal definition of Features as well as Feature Types. In a Feature modelling system, Features Types define the format for their instances, the actual Features in a particular model. Feature Types themselves are defined by either standardisation organisations or users, e.g. designers. However, the format of the definition of Feature Types needs to be defined independently of their contents. This format establishes the possibilities (and limitations) for formalising domain knowledge into Feature Types.

Figure 5 shows three layers of information-definition. The middle layer defines the Feature Types, both Generic and Specific. This is where the domain knowledge is formalised. The format of these type-definitions is laid out in the upper layer: the Meta Layer. This layer includes a formal description of what may be contained in a Feature Type: it defines the kind of attributes and procedures that make up a Feature Type's state and behaviour respectively. An EXPRESS-G representation of the main components of the Meta Layer is shown in figure 6.

The lower layer in figure 5 forms the level of Feature-based models that represent a particular building or design-case. Feature-based models consist of Features: instances of Feature Types. The domain knowledge which is generically formalised into Feature Types is here particularised with information from a specific case.

The following section discusses a prototype implementation of a so-called Feature-based Modelling Shell. This shell implements the infrastructure for Feature-based modelling and demonstrates the principles of extensibility and flexibility of information models.

4. Feature-based Modelling Shell: a prototype

The purpose of the prototype is to demonstrate the concepts of Feature-based modelling in the context of architectural design. The prototype will implement a selection of the contents of the meta layer and the infrastructure that is required for the related activities: definition and classification of Feature Types, and Feature instantiation and modification. We will use the prototype as a test-environment for the definition of Feature Types, especially we will start building a generic Feature Library based on the proposed classification of Feature Types. Flexibility of Feature models will be a major aspect in the demonstration of the Feature modelling processes.

4.1 DESIGN OF THE PROTOTYPE

Within the context of the above objectives for the prototype, the main requirements of the system are the following. The system supports the definition of the classes of Feature Types that are defined in the meta layer (see figure 6): simple Feature types, enumeration types, complex types, specialisation types, geometric types, and constraint types. The system organises Feature Types in Feature libraries. Feature models are created by instantiation of Feature types and their relationships, forming a network of Features. These models can be modified in terms of Feature attributes and by modification of the relationships between Features.

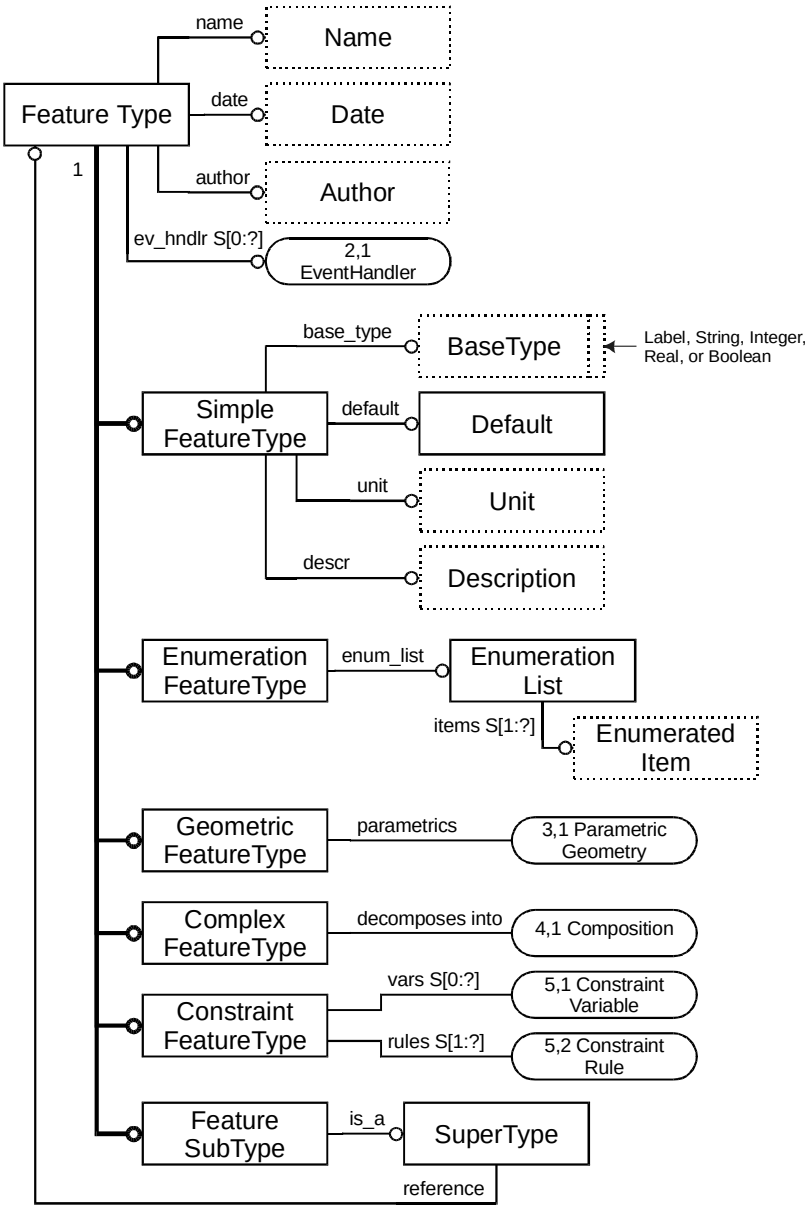

Figure 6 Schema of the Meta Layer representing the format for Feature Type definition (incomplete EXPRESS-G schema)

The system's design is divided into 4 modules which implement the activities of definition, classification, and instantiation, plus some processes required by the incorporation of knowledge into Feature definitions and the Feature modelling process. Some aspects of these modules are described below.

Feature Type definition module The definition of Feature types involves the selection of a class from the meta layer, possibly the selection of a parent Feature Type, and the specification of the attributes and behaviour of the Feature Type. The attributes include some administrative data of the type such as *name*, *author*, *time-stamp*, *documentation*, etc., and the definition of the instances that the type allows. In the case of a simple Feature Type, the attributes include *data-type*, *unit*, and *default*. In the case of complex Feature Type, attributes involve the selection of Feature Types that form the components of the complex Feature Type. Geometric Feature Types contain a parametric description of the geometry.

Interface is important aspect in the definition of all Feature Types. A considerable part of the interface that will be used in the instantiation of new Feature Types, needs to be designed during Feature Type definition. This aspect ranges from the design of dialogue-based interfaces to complex graphic user-interaction for instance with existing geometry.

The behaviour incorporated in Feature Types is yet another aspect of Feature Type definition. Behaviour of Features is defined as their reaction to particular events occurring in the modelling system. In order to allow users to define this behaviour, some kind of programming facility will need to be supplied by the system, for instance an interpreted language such as basic or lisp. This facility needs to have access to the attributes of the Feature Type being defined as well as other existing Feature Types. The complexity of this behaviour-programming will increase with the requested possibilities it should offer, such as access to the user-interface, etc..

Feature Type classification module After the definition process, a new Feature Type will need to be stored in an organised library of Feature Types. The organisation and management of Feature Types is very important throughout the complete system: for Feature Type definition, because some relationships between Feature Types may be included in their definition, which makes them inter-dependent; for Feature modelling, obviously, because the selection and searching for Feature Types is a crucial aspect of the modelling activity.

Feature modelling module Initially, modelling Features involves the selection of an appropriate Feature Type, creating an instance thereof, positioning it in the network of Features that forms the Feature model, and possibly processing its immediate behaviour. Modelling Features and modifying Features or their relationships may invoke more reactive behaviour of Features, for example: a Feature that defines the structural dependency of a beam would react when one of the bearing components, say a column, would be removed from the model.

Constraint handling module A constraint defines a relation that should hold, defined on one or more entities or constraint variables [Dohmen 1995]. We include constraint modelling by implementing a class Constraint Feature Types. In the Feature model, a Constraint Feature defines a relation between other Features which is a condition that needs to be satisfied for the model to be valid. The process of constraint satisfaction is subject of many research efforts, in architectural context [Gross 1990, Yoon 1992], in mathematical context [Saraswat and Van Hentenryck 1995], and in the context of Features technology, as reviewed by Dohmen [1995].

For the satisfaction of constraints, a so-called Constraint Handling Engine (CHE) needs to be integrated into the modelling system. A CHE interprets sets of constraints, using several mechanisms. These mechanisms are aimed at attempting to either satisfy the set of constraints, simplifying constraints or combinations of constraints, or adding new constraints that may help the process of satisfaction (constraint propagation).

The constraint handling module will be a separate module in the Feature-based Modelling Shell, however the concepts of constraints need to be addressed in the definition module as well.

4.2 ASSUMPTIONS AND BASIS FOR IMPLEMENTATION

We will briefly address some of the assumptions and points of departure for the implementation of the prototype. From the requirements for the system we concluded that the system best be based on an object oriented database management system (OODBMS). Geometry, although not addressed in early phases of prototyping, will eventually play an important role in the resulting Feature modelling system, the prototype should allow strong relations to a geometric modelling system, or even be integrated in one. The design-by-Feature approach of our research requires that, in the latter case, the Feature modelling system controls the geometric modelling system. Furthermore, for the scope of the prototype it appeared important to base any form of implementation as much as possible on available experience.

These assumptions have led to the following choices for the implementation environment. The prototype will be integrated in Autocad R13 using ARX technology. This choice implies the choice for Windows NT and MS Visual C++ for development environment. The ARX technology (Autocad Runtime eXtension) allows us:

- to have full control over the geometric modelling functionality of Autocad;
- to have direct access to the OO database that makes an Autocad drawing;
- to define new object classes in the OO database, geometric and non-geometric classes, which are fully integrated in the modeller;
- to use the OODBMS facilities in Autocad (serialisation);
- the full usage of C++ and Windows facilities.

4.3 SOME IMPLEMENTATION ISSUES

Adding new classes to an object oriented system generally requires compilation of the code defining these classes. Since the prototype Feature-based Modelling Shell

has the purpose of allowing users of the system to add new Feature Types, this means that the system either offers very user-friendly capabilities for compiling code and linking the new software to the system, or avoids compilation of code and offers an alternative way of defining Feature Types.

The first approach, compiling new classes and linking them into the system (for instance using parameterised classes or templates), is the most elegant and robust one, however, it requires considerable technical knowledge and effort from the user. The second approach, avoiding compilation of new classes, has led to an approach which implements Feature Types as objects instantiated from classes that represent the meta-layer described in section 3.5. Feature Instances are implemented as objects instantiated from classes that are defined in correspondence to the classes for Feature Types. The instantiation of a Feature from a Feature Type, the activity of Feature modelling, is implemented as instructing a Feature Type to create a new Feature object of the corresponding class. In this approach, the functionality of Features in the model depends on the relationship and joint performance of the two objects 'Feature' and 'Feature Type'.

5. Conclusions and discussion

The concept of Feature modelling is experienced as being refreshing in discussions with experts, when attempting to break the discussion on product models which seems to have stranded on the issue of rigidity of standardised core models. The notion of definable information entities being applied in (early) design phases is attractive to designers who do not find adequate tools for innovative computer-aided design. The customisation of modelling systems for particular purposes, e.g. a specific industrial building system, is known to have great attention. Feature modelling can be expected to give this field of software-development a new stimulus.

The problem of data-explosion will not be solved by applying the Feature-based approach. However, this problem, often incorrectly indicated as a disadvantage of computer support for design, is inherent to the design discipline itself, not a result of using computers. The problem is being given attention since computers slowly start to provide ways of handling the vast amount of data involved in design.

Working with different levels of abstraction requires more detailed research. Many design theories or methodologies employ levels of abstraction or levels of detail in a flexible manner. By means of case-studies the implication and potentials of Feature-based modelling in working with abstraction levels will need to be investigated. First attempts will include an integration with the work of Achten [1996, 1997].

The representation and manipulation of networks of Features, the structure of Feature relationships, is an important subject for further research. Recently we have started research on this subject, investigating the possibilities of using a Virtual Reality (VR) interface to Feature models [Coomans 1997]. In the VR environment, both the abstract entities in the Feature model and the geometric components in the

model will be visualised in an integrated approach. This approach will allow us to represent and manipulate physical and non-physical aspects of design in both modes.

An important issue in the definition of Feature Types is that Feature Types will act as knowledge bases in the modelling system. Designers will formalise generic domain knowledge into the behaviour or constraints defined in Feature Types. The principles and methods in knowledge engineering need to be studied and applied in the design and implementation of Feature-based modelling concepts.

Classification and standardisation of the Generic Feature Library will be the necessary basis for wide-spread application of the concept of Features technology in architecture. Although outside the scope of this paper, the relation with standardisation efforts in ISO 10303 (STEP) [ISO 1994] is evident, and the authors are eager to open the discussion on this subject and collaborate with experts in this field.

Acknowledgements

The research presented in this paper is a PhD. research under the supervision of Prof. Harry Wagter and Prof. Robert M. Oxman who's input to the project has been an invaluable support. Thanks also go to our colleagues Bauke de Vries, Henri Achten, and Marc Coomans for the many useful comments and discussions, and to the members of staff at Building Information Technology for their substantive technical and otherwise support.

References

Achten, H.H., Bax, M.F.T., and Oxman, R.M. (1996) Generic representations and the generic grid: knowledge interfaces, organisation and support of the (early) design process. Proceedings of the 3rd Conference on Design and Decision Support Systems in Architecture and Urban Planning, pp. 1-19. Spa, Belgium, August 18-21, 1996.

Achten, H.H. (1997) Generic representations - typical design without the use of types. These proceedings of the CAAD Futures 1997 Conference, München, Germany, August 3-7, 1997.

Coomans, M.K.D. (1997) Visualisation and manipulation of dynamic data structures in Virtual Reality. PhD-research proposal in preparation, Eindhoven University of Technology.

DeMartino, T., Falcidieno, B., Giannini, F., Hassinger, S., and Ovtcharova, J. (1994) Feature-based modelling by integrating design and recognition approaches. Computer-Aided Design vol. 26(8).

Dohmen, M. (1995) A survey of constraint satisfaction techniques for geometric modelling. Computers & Graphics vol. 19(6): pp. 831-845.

Eastman, C.M., Bond, A.H., and Chase, S.C. (1991) A data model for design databases, in J.S. Gero (ed.), Artificial intelligence in design '91, pp.339-365. Sydney, Butterworth Heinemann.

Eastman, C.M., Assal, H.H., and Jeng, T.S. (1995) Structure of a product database supporting model evolution. Modeling of buildings through their life-cycle (proceedings workshop CIB W78 '95), pp.327-338. Stanford University, CIB W78.

Ekholm, A., and Fridqvist, S. (1995) Object-oriented CAAD, Design object structure, and models for building, user organisation and site. Modeling of buildings through their life-cycle (proceedings workshop CIB W78 '95), pp.553-564. Stanford University, CIB W78.

Ekholm, A. (1996) A conceptual framework for classification of construction works. ITcon, electronic journal at http://www.fagg.uni-lj.si/~itcon/ vol. 1.

van Emmerik, M.J.G.M. (1990) Interactive design of parameterized 3D models by direct manipulation. Ph.D. thesis. Delft, Delft University Press.

Gross, M.D. (1990) Relational modelling: a basis for computer-assisted design, in McCullough, Mitchell, and Purcell (ed.), The electronic design studio, pp.123-136. Cambridge MA, The MIT Press.

Henderson, M.R., and Chang, G.J. (1988) FRAPP: Automated Feature Recognition and Process Planning from solid model data. Computers in Engineering vol. 1: pp.529-536. ASME.

ISO. (1994) ISO TC184/SC4 10303. Industrial automation systems and integration - Product data representation and exchange. ISO TC184/SC4.

Laakko, T., and Mäntylä, M. (1991) Feature modelling by incremental feature recognition. Computer-Aided Design vol. 25(8): pp.479-492.

van Leeuwen, J.P., Wagter, H., and Oxman, R.M. (1995) A Feature based approach to modelling architectural information. Modeling of buildings through their life-cycle (proceedings workshop CIB W78 '95), pp.260-269. Stanford University, CIB W78.

van Leeuwen, J.P., Wagter, H., and Oxman, R.M. (1996) Information modelling for design support - a Feature-based approach. Proceedings of the 3rd Conference on Design and Decision Support Systems in Architecture and Urban Planning, pp. 304-325. Spa, Belgium, August 18-21, 1996.

Meeran, S., and Pratt, M.J. (1993) Automated feature recognition from 2D drawings. Computer-Aided Design vol. 25(1): pp.7-17.

Mitchell, W.J. (1990) A new agenda for computer-aided design, in McCullough, Mitchell, and Purcell (ed.), The electronic design studio, pp.7. Cambridge MA, The MIT Press.

Saraswat, V., and Van Hentenryck, P. (eds.) (1995) Principles and practice of constraint programming - the newport papers. Cambridge, Massachussetts: The MIT Press.

Shah, J.J., and Rogers, M.T. (1988) Functional requirements and conceptual design of the Feature-Based Modelling System. Computer-Aided Engineering Journal vol. 5(1): pp.9-15.

Shah, J.J. (1991a) Conceptual development of form features and feature modelers. Research in Engineering Design vol. 1991(2): pp.93-108. New York, Springer-Verlag.

Shah, J.J. (1991b) Assessment of features technology. Computer-Aided Design vol. 23(5): pp.331-343.
Tolman, F.P., and Wix, J. (1995) Building Construction Core Model, ISO/WD 10303-106. Industrial automation systems and integration - Product data representation and exchange.

Woestenenk, K. (1995) Proposal for international conversion tables for parts and functions. The Netherlands, STABU.

Yoon, K.B. (1992) A constraint model of space planning. Southampton, UK: Computational Mechanics Publications.