Features and Constraints in Architectural Design

de Vries, B.; Jessurun, A.J.

Proceedings of DETC'98
1998 ASME Design Engineering Technical Conference
September 13-16, 1998, Atlanta, GA

DETC98/CIE-5537

FEATURES AND CONSTRAINTS IN ARCHITECTURAL DESIGN

BAUKE DE VRIES
Associate Professor

Design Systems
Faculty of Architecture and Building
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, the Netherlands
Tel: +31 40 2472388, Fax: +31 40 2450328
B.d.Vries@bwk.tue.nl

JORAN JESSURUN
Research Associate

Design Systems
Faculty of Architecture and Building
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, the Netherlands
Tel: +31 40 2475096, Fax: +31 40 2450328
A.J.Jessurun@bwk.tue.nl

ABSTRACT

The concepts of the experimental design system that are discussed are feature modeling and geometrical constraints. The main technique for creating the user environment is Virtual Reality. Feature modeling forms the basis for managing the design data. To start with, data storage is implemented in a Relational Data Base Management System. Along with this a (traditional) interface is developed for managing the data. Data management consists of feature type creation and feature type instancing. Features are used to define building elements, their relationships and additional constraints. Apart from the design data, geometrical data are stored. The system contains a library of parameterized geometrical objects which format is in coherence with the geometrical modeling environment (VR). Possible design solutions can be limited using geometrical constraints. Specifying connection types between building elements result in a set of solutions for the position of the bounding boxes of the building elements in space.

Key words: Design Systems, Feature Modeling, Geometrical Constraints, Virtual Reality

INTRODUCTION

Design systems nowadays usually cover only a specific approach to the design problem, whereas designers use several design techniques during a cyclic process to find a design solution. Sometimes systems are connected with each other but only for use in certain stages and in a specific order. To allow for different design techniques in non deterministic way we found that a design system is needed with a flexible and extendible data structure and a range of interfaces, each of them suited for a specific approach to the design problem.
In the experimental design system described in this paper design support is achieved by integrating two theoretical concepts and using VR as a technology for evaluating the results. Since the innovative part of the system is grounded by Feature Base modeling and the use of Geometrical Constraints, these two aspects will be elaborated in general first. Later on the implementation will be described in more detail while explaining the framework of the design system.

FEATURE-BASED MODELING OF ARCHITECTURAL INFORMATION

Feature Based Modeling finds its roots in Mechanical Engineering (Shah and Rogers, 1988). Here Features are (were ?) almost synonym to Form Features. Form Feature descriptions are closely related to manufacturing of these forms. In an almost finished PhD study at the Eindhoven University in the Design Systems Group (Leeuwen et al., 1996) the original concept has been applied to architectural design.

Main purpose was to investigate if the concept could support flexibility and extensibililty.
Flexibility is found to be a key characteristic of the architectural design process. The focus in architectural design shifts between different levels (e.g. urban plan, building, room) and different aspects (e.g. physical, structural). Design decisions are taken in a cyclic process focusing on a part of the complete design task (e.g. the ground floor). Thus, changing a design and its underlying representation is in fact part of the creative process. Moreover design decisions are often dependent on each other. Therefore the design representation must support the linkage of characteristics of design parts (e.g. the color). In that case changing a parameter of one building element is propagated to another (e.g. all internal door share the same color). In Feature-Based Modeling any Feature (characteristic) of a building element can be shared by any other building element. For this purpose three relationship types are introduced: Specification, Decomposition and Association. Specification specifies a certain characteristic of a Feature (e.g. color). Decomposition declares a building element being a part of another building element (e.g. hinge is part of a door). Association is any relation that cannot be expressed by the former two (e.g. a door in wall relation).
The second key characteristic is the fact that a design evolves in time. Descriptions of building elements are often rather abstract and incomplete in the early design stages. Gradually design parts are refined and design alternatives are accepted or rejected. Sometimes the design process is reversed reconsidering previously found solutions. All the time new design decisions must be checked against consistency with other design decisions. The representation must support registration of design solutions of the same building element in different stages. So design solutions should never be destroyed unless the designer explicitly wants to do so. In Feature-Based Modeling Features can be related with each other using the Inheritance relationship. Inheritance specifies a subtype having all the characteristics of the supertype and if necessary one or more extra. Moreover each Feature is time stamped and carries the name of the creator (an identification of the designer).
A third key characteristic of architectural design is the fact that most designs are unique in itself. Therefore describing a generic model of a building appears to be an impossible task. Instead of that in the proposed Feature-Base-Modeling theory, Features can be created on two levels: Feature Type level and Feature level. On Feature level Feature Types are instanced into Features. The designer is offered libraries of predefined Feature Types, but is also allowed to create his/her own Feature Types or Feature Sub Types. In this way standardization can be pursued by the creation of libraries without limiting the designer and refraining him/her of new innovative design solutions.

GEOMETRIC CONSTRAINTS

Reasoning about geometrical relationships between building elements in general requires a common formal geometry description. In VR systems geometry representations for the surfaces of shapes usually consist of lists of triangles (strips). Computer hardware is optimized for displaying these triangles. Ideally constraints also would use the same triangles as primitives. As a start though, we found that Bounding Boxes containing building elements serve well as far as we deal with architectural spatial relationships. A bounding Box is the operand of the constraint solver and it is determined by its position, length, width and height The theoretical approach we adopted for managing geometrical constraints is based on Allen's temporal interval algebra (Allen, 1983). He recognizes five basic relationships that can exist between a point and an interval: ahead, front-touch, in, back-touch and behind. Because in our design system each of the local axes of the Bounding Boxes representing the building elements are always parallel with one of the global axes, it is appropriate to use interval-interval relationships. This lead to 13 relationship types. (For a more in-depth explanation about 'Qualitive Spatial Relationships', see (Gorti and Sriram, 1995)). Constraints on a Bounding Box can be rewritten to relations between intervals in three sets, one for each axis.
In the design system five constraint types are recognized:

1. **Connection constraints: Touch, Align.** These are constraints that show Bounding Boxes are connedted with each other.
2. **Distance constraints: OppDistance, LatDistance.** These constraints specify the distance between two sides of Bounding Boxes.
3. **Contain constraint: Contains.** This constraint specifies that one Bounding Box should contain another.
4. **Non-intersection constraints: NonIntersect.** This constraint specifies that two Bounding Box should not intersect.
5. **Unary constraints: FixPosition, FixSide, FixDimension, FixOrientation.** These are constraints that put restraints on the position, the dimension and the orientation of the Bounding Box. These constraints are no interval constraints but necessary for the solution process.

For example, a Constains-constraint which specifies that Bounding Box B2 should hold Bounding Box B1 can be rewritten to the following interval constraints:
$I_b^{B1} < I_e^{B1}$ and $I_b^{B2} < I_e^{B2}$ and $I_b^{B1} < I_b^{B2}$ and $I_e^{B2} < I_e^{B1}$ and $I_b^{B1} < I_e^{B2}$ and $I_b^{B2} < I_e^{B1}$.
Here, I_b^{B} and I_e^{B} define an interval belonging to Bounding Box B. The subscripted $_b$ en $_e$ mean the beginning and the end of an interval. For the Contains-constraint the above six constraints are created for all three axes.
Since the purpose of the design system is that constraints can be created between building elements rather intuitively there is

no control mechanism to check whether the geometrical model is under-constrained or over-constrained. In an under-constrained situation the solving process will produce a large number of solutions that is hardly manageable be the designer. In a over-constrained situation the solving process will find no solution at all. In our experimental design system we have chosen a solution method that will generate a design that is 'as close as possible' to the former situation (Kelleners et al., 1997).

FRAMEWORK EXPERIMENTAL DESIGN SYSTEM

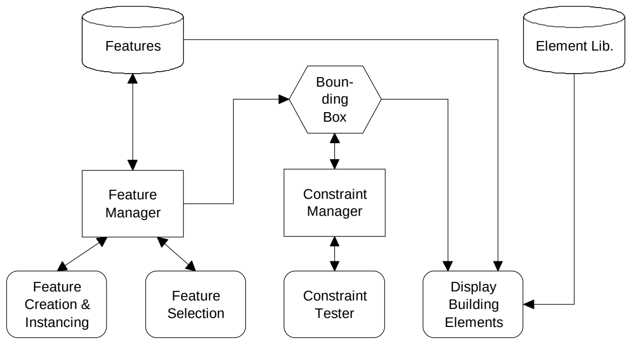


Figure 1. Framework experimental design system

The experimental design system consists of:

- Two permanent stores: Feature-store, Element Library store (magnetic disk symbols)
- Two management systems: Feature Manager, Constraint Manager (square boxes)
- One temporal store: Bounding Boxes (hexagon)
- Four application programs: Feature Creation & Instancing, Feature Selection, Constraint Tester, Display Building Elements (rounded boxes)

The Feature Manager is implemented using a MS SQL Server and so the Feature stores are implemented as tables with records. The Entity-Relationship model reflecting that data structure was constructed from the originally Object-Oriented Feature Model (Leeuwen and Wagter, 1997). The E-R schema is drawn using the IDEF1X notation.

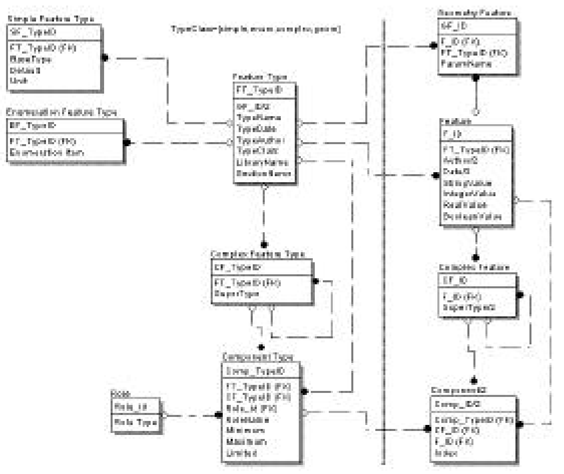


Figure 2. E-R schema of Feature Model

What becomes clear from the schema is the distinction between Feature Types (left part of the schema) en Features (right part of the schema). So we do not deal with a predefined data model for a building, but with a data structure which allows us to create Feature Types and Features during the design process. Another typical aspect of this data structure is that a Feature Type in itself does not contain any attributes other then needed for administration (date, author, etc.). A Feature Type (and so a Feature) is fully defined by its Component Types. In other words a single Feature is more or less meaningless. The attributes of a Feature Type (e.g. a Room) can only be found by searching for all Component Types that are referenced.

Feature creation and instancing is implemented using PowerBuilder to create database forms. Feature Selection is implemented using TransactSQL as a programming language for executing conditional SQL statements. The query result is stored in a temporary table which is accessible for the Constraint Tester. The Constraint Tester was developed using C++ as the programming environment and OpenGL as the graphics library for displaying the Bounding Boxes.

The constraint manager is used by the Constraint Tester to communicate with the underlying constraint solving system. A Bounding Box has two states, the actual state the is manipulated by the Constraint Tester and the previous state. The previous state is initially the state as derived from the Features store. From that subsequent states will follow after each solution process.

The building elements are displayed using WorldUp to retrieve the actual data of the size and location of the Bounding Boxes. The geometry descriptions of the building elements are retrieved from a library of WorldToolKit objects. The WTK objects can be scaled and moved accordingly to the corresponding Bounding Boxes. The material property descriptions are retrieved from the Feature store and are used to find the appropriate material and texture mappings from the WTK library. The geometry descriptions and the material property descriptions together make up the Element Library.

In our experimental design system we want to test and prove the concepts of Feature-Based Modeling and of Geometric Constraints in a VR environment. User interaction in VR is currently limited to (re)locating and (re)scaling building elements. In the near future Feature manipulation will be implemented in the VR environment as well. Then the designer can shift between the Feature Model representation and the building element Geometry representation, depending on the kind of design operations he/she wants to perform.

SAMPLE DESIGN SESSION

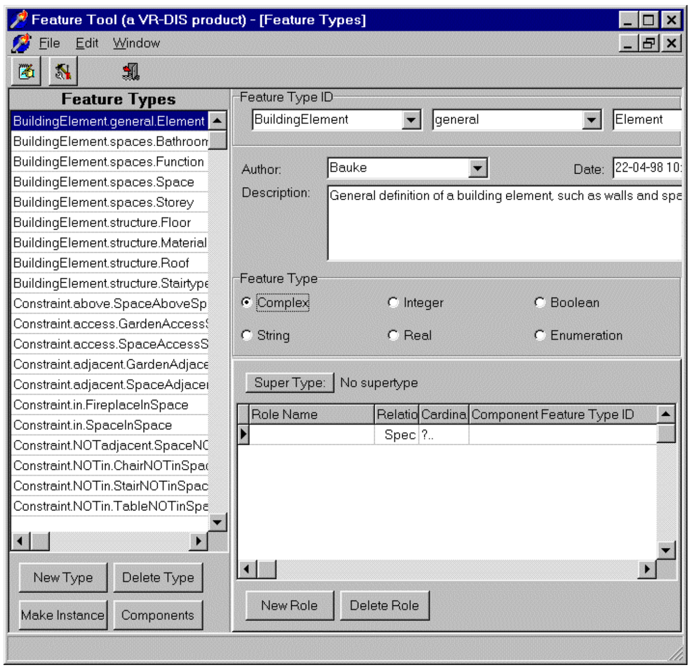
A sample design session is described to illustrate

1. how a simple architectural design is specified using the Feature Model.
2. how constraints support the designer in creating a design in accordance with explicit requirements.
3. how constraints support the 'natural' behavior of the design model.

Objectives of the design session:
Define a room consisting of two long walls, two short walls and a door. Connect the walls with each other. The long walls have the same dimensions, likewise the short walls. Height and thickness of the walls cannot be changed after creation. Change the location of one of the walls a check if the connections are retained by the design system. Also find out what happens with the location of the door.

define features

Feature Type definitions in the experimental design system are established by creation of a new records in a database. First all 'basic' Feature Types need to be created such as Coordinate, Axis, Dimension, Location and Orientation. Secondly the Complex Feature Types such as Volume, Wall, Door and Room are created referencing the basic Feature Types. Referencing consists of attaching a Role Type (Specialization, Generalization, Association), the referenced Feature Type and a specific Role Name.



Feature Types

BuildingElement.general.Element
BuildingElement.spaces.Bathroom
BuildingElement.spaces.Function
BuildingElement.spaces.Space
BuildingElement.spaces.Storey
BuildingElement.structure.Floor
BuildingElement.structure.Material
BuildingElement.structure.Roof
BuildingElement.structure.Stairtype
Constraint.above.SpaceAboveSp
Constraint.access.GardenAccess
Constraint.access.SpaceAccessS
Constraint.adjacent.GardenAdjace
Constraint.adjacent.SpaceAdjace
Constraint.in.FireplaceInSpace
Constraint.in.SpaceInSpace
Constraint.NOTadjacent.SpaceNC
Constraint.NOTin.ChairNOTinSpac
Constraint.NOTin.StairNOTinSpac
Constraint.NOTin.TableNOTinSpa

Feature Type ID: BuildingElement | general | Element

Author: Bauke
Date: 22-04-98 10:
Description: General definition of a building element, such as walls and spa

Feature Type: [✓ Complex] Integer Boolean String Real Enumeration

Super Type: No supertype

Role Name	Relatio	Cardina	Component Feature Type ID
	Spec	?..	

Figure 3. Feature Creation & Instancing Tool

instance features: building elements

Feature Type instancing is at first established for the dimensional parameters of the Walls such as Height, Length and Thickness. From that the Volume Feature Type is instanced for the long wall and one for the short wall referencing the dimensional Features. Then the four Walls are instanced referencing a Volume Feature instance, a Location instance and a Orientation instance. The latter two Features will contain default values as stated at their Feature Type definition. In the same manner as the Walls, the Door is created. Finally the Room is instanced referencing the four Walls and the Door.

instance features: constraints

Before the constraint solve process can be started the geometrical model must have defined position in the coordinate system otherwise there will be no solution. In this example this means that the position of all Bounding Boxes must be known either explicitly by specifying a location vector or implicitly by specifying how the Bounding Boxes are connected with each other.
So, additional Feature Types are necessary for constraint satisfaction. Because the procedure of Feature creation and instancing has been described before it is not worked out in detail here. The constraint types that are used in this example are:
FixPosition: To attach the model at one point to the coordinate system

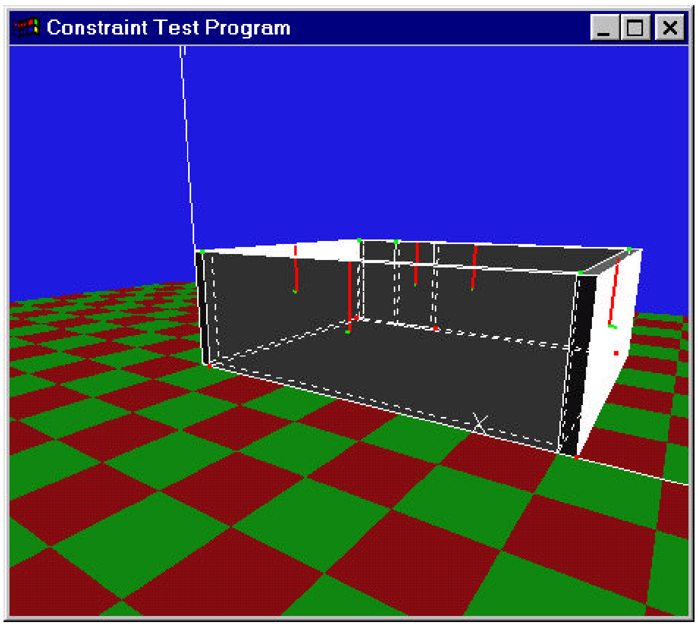
FixDimension: To prohibit the user from changed the Length and Thickness values of the Walls
FixOrientation: To give all walls the initial intended orientation (Left, Right, Front, Back) and to prevent the walls to be turned from inside to outside by the solving process
Contains: To locate the door within one of the long Walls
Touch: To connect the long Walls to the short Walls

solve constraints

Bounding Boxes for constraint management are defined by the a location, a Length, a Width and Depth. In this specific example the Bounding Box parameters have a one to one relationship with the dimensional Features: Height, Length, Thickness of the Walls and the Door. The constraint solving process is triggered by entering the solve command in the text window of the Constraint Tester program. Since the example is rather simple and because the set of constraints is sufficient (see Paragraph Geometric Constraints), the result is almost immediately displayed in the graphical window. Now the Walls are 'decently' connected with each other and the door is somewhere in one of the long Walls. The position of the Door is determined by the fact the Constraint Solver will move the Door is little as possible from its former position to is new one. Originally all location values of the Walls and the Door are equal to the origin of the coordinate system.

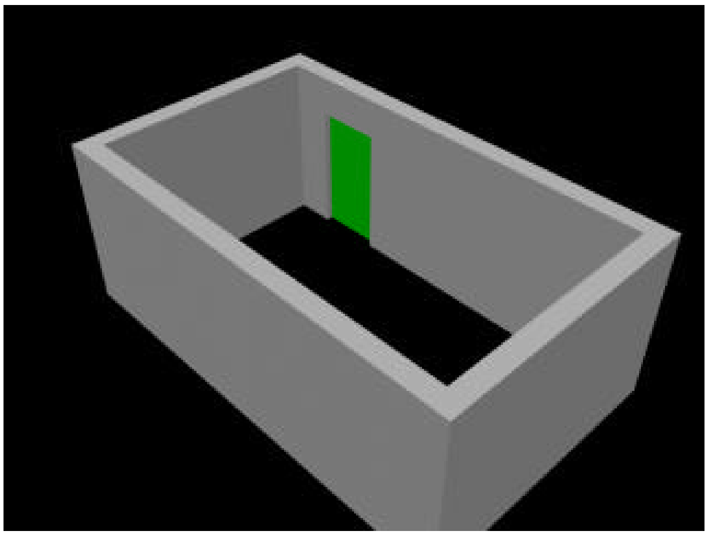
move wall and solve again

Let's assume that the designer wants to change the shape of the room having the long Walls a significantly longer dimension than the short Walls. Therefore the designer relocates one of the short Walls into the desired position. This is achieved by using the BBMove command within the text window of the Constraint Tester. After that the Solve command is fired again. In the graphical window we now see that the size of the long Walls has changed to reestablish the connection between the Walls. The Door stays in place.



display in VR

Finally the Room is rendered in a VR environment using the value for the Material parameter of the Geometry Feature of the Walls and the Door to find the corresponding items in the library of the renderer. The designer navigate inside and outside the room the judge his/her idea.



CONCLUSIONS

From this very simple exercise the following conclusions can be drawn:

- Feature Type creation and Feature instancing takes quite a lot of effort. Working from scratch however is not as we depict our design system. Having a library with Feature

Types available describing commonly used building elements can reduce this problem. While creating such a library there is a danger of trying to standardize building elements. A generic set of Feature Types for building elements could support the design process without restricting it.

- What counts for Feature Types and Features in general counts for Constraints especially. One has to have reasonable knowledge about geometric constraints to able to construct 'just enough' constraints. Underconstained situations may lead to enormous sets of solutions while overconstrained situation may end up in no solution at all. To support the designer the design system should be able to create appropriate constraints itself. Therefore high level design constraints should be available which make the building elements behave like the designer expects (E.g. door *in* wall, living room *adjacent* to bathroom, etc.). Another aspect of generation of design solution is that the outcome should be controllable by the designer. This counts for the number of presented solutions and for the resolution process itself.
- The example does not show the limitations of the Bounding Box approach. Another example that will explain this is a chair that must be located partly under a table. This case cannot be described by Bounding Boxes.
- One could consider elimination of the Constraint Tester and directly display the results of the solution process in the VR environment. Moreover instead of triggering the solution process by the designer the solution process could be triggered by every or certain changes in the design. These aspects which are closely related with user interaction will be studied while doing more realistic experiments with the design system.
- As long as we deal with boxes such as Wall and Doors there is a rather simple relationship with the geometry descriptions of the accompanying building elements in the VR system. Thinking of more complex shapes like curved surfaces it is difficult to find a relationship with the Bounding Box with which it is surrounded by.
- The selection process is implemented using SQL as a scripting language. Instead of ‘manually’ creating selection scripts, the Feature Manager should be equipped with a set of functions that support Feature data selection in a more straight forward way.

CURRENT RESEARCH & DEVELOPMENT

Currently a PhD research is going on to develop a tool in the VR design environment that allows the designer to create and manipulate Features. The kind of tool that we are thinking of is a Feature Box which constraints the complete Feature structure. One side of the Feature Box is transparent and allows the designer to pull some Features in front so that he/she can inspect them. Another way to check the Feature's values is to make them visible as soon as the designer reaches or touches a building element. Such kind of tools will eliminate the need for database forms to create and instance Features. It is our aim to make all design data available in a single environment through VR.

Another PhD research that will soon be started is to develop design techniques and tools for the manipulation of geometries in VR. As mentioned in the introduction we want to create a really interactive design environment. Whereas Feature Modeling forms the theoretical basis for design datamanagement we also need the tools in VR that can support the designer in his/her creative design process. VR offers the opportunity to develop a set of tools that are far more 'natural' than was possible until now in the traditional CAD environment. It is our intention to take advantage of that.

REFERENCES

Allen, J.F. (1983), “Maintaining knowle4dge about temporal intervals,” Commun. ACM Vol 26 No 11, 1993, pp 832-843.

Bridges, A. and Charitos, D.,1997, “The architectural design of virual environments,” Proceedings of the 7th Conference on Computer Aided Architectural Design Futures. Munich, Germany, Kluwer Academic Publishers, pp.719-732.

Coomans, M. K. D. and Oxman, R. M., 1996, “Prototyping of Designs in Virtual Reality,” Proceedings of the 3rd Design and Decision Support Systems in Architecture & Urban Planning. Spa, Belgium, Eindhoven University of Technology, pp.20-34.

Coomans, M. K. D. and Timmermans, H. J. P., 1997, “Towards a Taxonomy of Virtual Reality User Interfaces,” Proceedings of the International Conference on Information Visualisation (IV97). London, pp.27-29.

Donikian, S. and Hegron, H., 1995, “Constraint Management in a Declarative Design Method for 3D Scene Sketch Modeling,” In: Saraswat, V. and Hentenryck, P. ,van. v. 1, (22):p. 427-444. Principles and Practices of Constraint Programming. The MIT Press, Cambridge, Massachusetts.

Earnshaw, R. A., Gigante, M. A., and Jones, H., 1993, “Virtual Reality Systems,” London, Academic Press.

Gorti, S. R. and Sriram, R. D., 1997, “From symbol to form: a framework for conceptual design,” Computer-Aided Design, 28, p. 853-870.

[Kelleners, Veltkamp, Blake 1997]
Kelleners, R. H. M. C., Veltkamp, R. C., and Blake, E. H., 1997, “Constraints on Objects: a Conceptual Model and an

Implementation,” Proceedings of the 6-th Eurographics Workshop on Programming Paradigms in Graphics. Budapest, pp.67-78.

Leeuwen, J. P.,van and Wagter, H., 1997, “Architectural Design-by-Features,” Proceedings of the 7-th International Conference on Computer Aided Architectural Design Futures, Dordrecht, Kluwer Academic Publishers, pp.97-115.

Leeuwen, J. P.,van, Wagter, H., and Oxman, R. M., 1996, “Information Modelling for Design Support - A Featured-Based Approach,” Proceedings of the 3rd conference on Design and Decison Support Systems in Architecture & Urban Planning. Spa, Belgium, Eindhoven University of Technology, pp.304-325.

Shah,J,J. and Rogers, M.T., 1988, “Functional requirments and conceptual design of the Feature-Based Modeling System,” Computer-Aided Engineering Journal, vol.5, no. 1, pp. 9-15.

Vries, B. ,de and Leeuwen, J. P.,van, 1997, “Design Studio of the Future,” Proceedings of the CIB W78 conference "Information Technology Support for Construction Reengineering". Cairns, Australia, James Cook University, pp.139-148.