PRODUCT MODEL BASED DESIGN OF PRECAST FACADES

SUBMITTED: November 1996
REVISED: March 1997
PUBLISHED: April 1997

Vesa Karhu, Research Scientist
VTT Building Technology, Technical Research Centre, Espoo, Finland
vesa.karhu@vtt.fi

SUMMARY: *In Finland, approximately 80 % of the facades of buildings are manufactured as precast units. Currently one of the obstacles to making the overall design and construction of precast building facades more efficient is the exchange of data about facades between architects, structural engineers and precast element manufacturers. The product model approach seems to offer a new methodology for data exchange and sharing which would solve many of the current problems. This paper presents the results of research carried out at the Technical Research Centre of Finland in which this approach was tested.*

The prevailing way of designing facades was chosen as a reference process model. Based on an analysis of data needs in the different stages of the process a product data model of a facade was developed. The product data model was restricted to facades only and does not include other information about the building. Central data structures in the conceptual schema define how a precast concrete facade consists of precast concrete units, i.e., elements. Structural wall layers that may have openings form the elements.

The conceptual schema was implemented as a prototype which was based on existing software, modified and further developed. The prototype was tested by an architectural design company, a structural design company and a manufacturer. The main conclusion of testing was that the data produced in the architectural design is directly usable in further design. The structural or element design may use the architectural data as such. Also, it is possible to create applications that take into account the architect's preferred design approach.

KEYWORDS*: facade, precast, data exchange, object oriented, architectural design*

1. INTRODUCTION

1.1 Present problems and typical design process

Approximately 80 % of facades are manufactured as precast units in Finland. This has been the prevailing situation in the 1980s and early 1990s. The use of this type of technology has resulted in a design, construction and manufacturing process which differs somewhat from the traditional in-situ construction process. The division of tasks and the exchange of data between the architect, the structural designer, the manufacturer of the elements and the contractor is critical to achieving an efficient process.

The building design process may be divided into stages according to lists of tasks (RT 1995a), (RT 1995b), (RT 1995c), (RT 1995d). The traditional building process in Finland is divided into stages:

- briefing;
- programming;
- global design;
- detail design;
- design during construction; and
- design for usage and maintenance.

The division may differ in other countries. See also Fig. 5.

Exact architectural design data are not transferred to other partners at the global design stage. The global design stage is further divided into three main phases.

- The first phase yields a basic solution in which masses and general site designs are made.
- The second phase yields a proposed solution, which is based on a chosen architectural solution. Feasibility analyses from the structural point of view are carried out before.

- The third phase, scheme design, is used to elaborate designs for the application of a building permit. The building permit can be obtained within one month for a normal building in Finland.

In this project the main focus was on studying the global and the detail design stages and what kind of information (general, detailed) is needed. Exact design data are not transferred to other parts of the building process. The general requirements for a facade are set in briefing and programming. The global design, or the conceptual design, is based on these two stages. The detail design stage is further divided into tender design and detail design. In the tender design stage the documents for the invitation to tender are prepared. Sufficient data accuracy and content are needed in each stage to enable further elaboration of the design solutions. A tender is based on tender designs. Inaccurate designs during the tender stage cause difficulties in determining the costs.

A typical information flow in the traditional building design process goes from the architect to the structural engineer and further to the manufacturer, see Fig. 1.

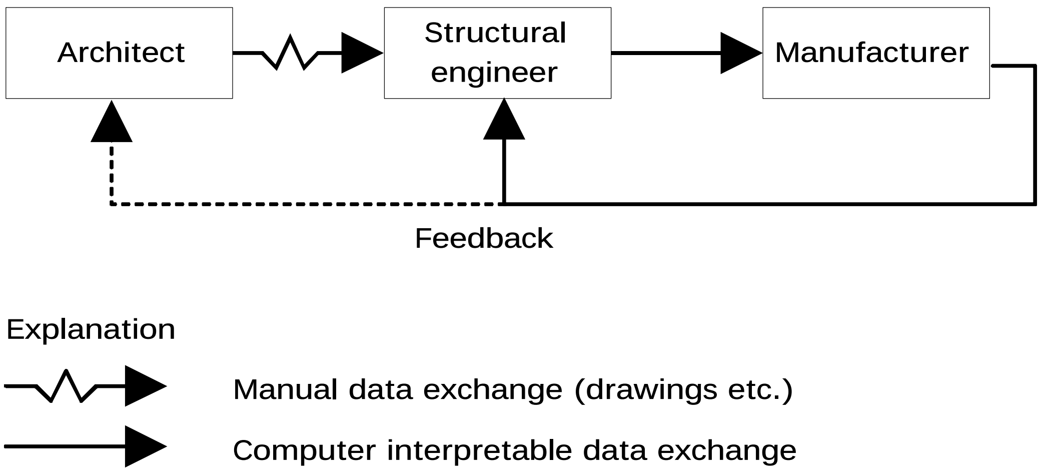

Architect
Structural engineer
Manufacturer
Feedback
Explanation
Manual data exchange (drawings etc.)
Computer interpretable data exchange

FIG. 1: A typical information sequence during a building design process. Data from architectural design are usually produced in drawing format. Structural engineering design uses computer interpretable data exchange internally because most manufacturers are able to receive data in standardised format. Feedback exists (dashed line) but not sufficiently.

The information process described above often leads to reuse of standard design solutions, little no time for new innovations and new technology suggested by the manufacturer (feedback). Feedback from the manufacturer to the architectural design exists but not sufficiently to enable new solutions instead of standard solutions. More data and information exchange are needed.

The architect's main concerns for the facade are repeatable visible surfaces, openings and some details, such as details of windows in the global design stage. The facade is designed as a whole. The facade is divided into elements only later by the structural engineer.

The first architectural designs are passed to the structural engineer who then uses his own software applications to do the appropriate feasibility analyses. The documents produced are drawings and written specifications. These may be unstructured and their data accuracy and reliability is often insufficient. The input of data to other systems is made by manual inspection of the drawings despite the fact that CAD-systems may have been used to produce these drawings. This often leads to misinterpretation of the data.

At the end of this process, it is difficult for the manufacturer to estimate costs. In many cases the cost objectives that were set during programming have to be revised or a lower quality level must be accepted. On the other hand, the data from the structural engineering design may be passed to the manufacturer in a computer interpretable format.

The traditional building design process as such does no longer correspond to the demands of today. If contractual aspects are set aside, one of the main causes of problems has been the lack of standard data definitions for information exchange in various design stages. This leads to a variety in contents and accuracy in the design documents.

1.2 The product model approach to information management

The product model approach seems to offer the tools needed to overcome some current problems. In a product model the data about an artefact are arranged in a systemic way using object oriented data base principles. Traditional CAD-programs primarily model the graphical appearance of the building parts. The product model approach describes building parts directly. This approach has been a prime research subject for some ten years. It is currently receiving increasing attention both in standardisation like the ISO STEP process (ISO 1992b) and the Industry Foundation Classes initiative (IAI 1996) and in the development of commercial software.

A central concept in the following presentation is the conceptual schema. In the theory of conceptual modelling or data base design the conceptual schema denotes the formalised description of the structure of the information stored in a data base (Boman et al. 1991), (ISO 1985).

Conceptual schemas may be used to structure database applications as diverse as census records, banking applications, missile guidance systems or descriptions of aircrafts.

A *product data model* is defined as follows (PM Glossary 1996): A particular type of conceptual schema, which structures the information needed to describe a physical artefact, designed and manufactured by man. The central object classes of product data models describe the functional parts of the artefact and assemblies formed by these, rather than concepts needed for representing these parts in different kinds of documents. A *product model* is a computer-interpretable description of an artefact, structured according to some predefined product data model. For a general discussion of product models and product data models see Björk (Björk 1995).

In Finland, the RATAS model (RATAS-committee 1988) proposed an entity-relationship model which was enhanced with inheritance as a framework for a building product data model. Further work in Finland has resulted in the OOCAD model, which is a generic data model proposal based on a composition of objects with part-of relationships (Serén et al. 1993).

Several foreign authors have proposed generic building product models. The General Architecture, Engineering and Construction Reference Model (GARM) was developed by Wim Gielingh at the Dutch research institute TNO (Gielingh 1988). It organises construction process information at a high level of abstraction. The basic class of the GARM model is called a product definition unit. The AEC Building Systems Model tries to model the functional systems (enclosure, structural, mechanical, etc.) that make up a building (Turner 1990).

Recently there has been a shift from all-encompassing product models to conceptual schemas describing more limited domains. Terms such as aspect models have been used to describe such models. A good example of an aspect model is the Integrated Data Model (IDM) provided by the COMBINE project (Dubois et al. 1992).

For the particular field of structural design some proposals have been made. Hannus (Hannus 1990) discusses CAD-systems based on product modelling and precast concrete structures. Luiten (Luiten 1994) used precast concrete structures to test product models of beams, columns, hollow core slabs and connections. CIMSTEEL (Watson 1995) uses the product model approach for structural steel framing data models. A STEP application protocol is under development for this area (AP 230: Building structural frames: Steelworks).

Attempts have been made in Finland to standardise the data exchange of precast concrete structures. In the 1980s, various software modules were developed for receiving and manipulating graphical files and for the manipulation of tables of data (BEC 1991). In the 1990s, so called object definition cards were developed in Finland. These define a standard data structure for the description of various building components such as windows and doors.

All in all, many of the building product model proposals that have been proposed in the literature are rather theoretical and have not yet reached a stage were they can easily be implemented by software companies. Thus, there is a need to produce more firm proposals, possibly for quite limited application areas, and to test them in real-life design situations. This has been one of the main motivations for the research described in this paper.

1.3 Scope of the research

The aims of the research were to:

- Define a basic activity model of the building design process of precast concrete facades emphasising the architectural design.
- Analyse the problems occurring in a typical design process.
- Define a product data model of a precast concrete facade.
- Define prototype check-lists of the data input and output requirements.

- Develop prototype software based on the product model.
- Test the prototypes in a real project.
- Conclude if the product model based architectural design process enhances the whole building design process.
- Propose guidelines for using the product model based approach in architectural design.

2. METHODOLOGY

2.1 General methods

The following general methods were adopted:

- Gather basic information and data in interviews with clients, architects, structural engineers, contractors and manufacturers.
- Use formal process modelling methods for activity modelling.
- Use formal conceptual modelling methods for product modelling.

2.2 Choice of modelling tools

Systematic data modelling methods were applied in the project. A number of modelling methods for both activity modelling and conceptual modelling are available. Tools that combine these two aspects are relatively few.

In the project, methods included in the STEP standard development process were used. The integrated resources specified in STEP define a generic information model for information about any product (i.e., geometry, topology, product configuration). These resources are by themselves not specific enough to support the information requirements of some particular application area without the addition of specific constraints, relationships and attributes. Application protocols (APs) are developed for this purpose. An AP provides a mapping to show how the interpretation of the integrated resources is used to meet the information requirements of the application (WWW STEP 1996a).

Examples of application protocols currently under development by the building construction committee include

- AP225: Structural Building Elements Using Explicit Shape Information,
- AP228: Building services: Heating, Ventilation and Air Conditioning (HVAC),
- AP 230: Building structural frames: Steelworks.

The first steps in the development of an AP are to define the process in which the data exchange is taking place and a conceptual schema defining the information requirements of this data exchange (a so-called application reference model, ARM).

In this project a basic process model of the design of precast concrete facades and a product data model of a facade were developed. The product data model could be developed further to a STEP application protocol, which would have involved redefining it to reuse the data structures of STEP resource entity definitions. However, this was deemed to be out of scope of the research.

The building design process from briefing stage to construction and final approval was modelled using SADT-charts (Marca et al. 1987). Special interest was focused on tasks that are of importance in architectural design.

EXPRESS and its graphical counterpart EXPRESS-G (ISO 1992a) were used for defining the conceptual schema of the facade. The EXPRESS language defines entities that are classes of information defined by common properties. An example of EXPRESS is given below

```
ENTITY facade
SUPERTYPE OF (precast_facade ANDOR cast_in_situ_facade);
   position      : position ;
   id            : id ;
   shape         : shape ;
   balconies     : OPTIONAL SET [1:?] OF balcony ;
   channels      : OPTIONAL SET [1:?] OF channel ;
END_ENTITY ;
```

An entity named facade is defined in this example. The entity facade is a supertype of either precast facade or a cast in situ facade. It can also be both. The facade also has a number of attributes, for instance position.

The graphical representation of an entity is a box, Fig. 2. Relationships are shown with normal lines, optional attributes as dashed lines. Thick lines are used to represent an inheritance relationship, i.e., a subtype and a supertype relationship.

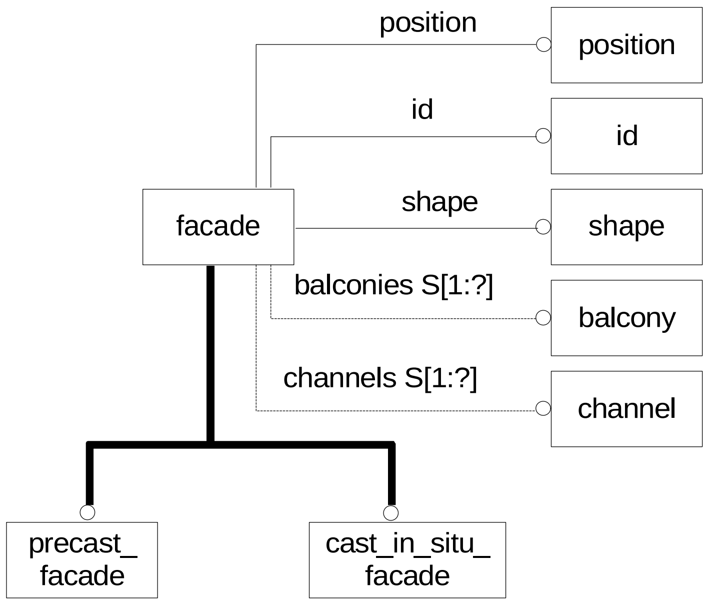


FIG. 2: The graphical representation of an entity and its relations.

A number of both commercial and freeware STEP tools are available (WWW STEP 1996b). Examples of the functionality of the software tools are

- browsers for traversing EXPRESS models;
- compilers that convert EXPRESS into a programming language;
- converters that convert between modelling languages; and
- parsers that check the EXPRESS syntax and possibly semantics.

Also, a number of tools may be used to process the different files.

2.3 Object definition cards

So-called product model object definition cards of the object classes included in the precast facade data model were created. Such cards have recently been adopted in Finland by the RATAS committee (RATAS 1996a) in order to facilitate communicating product model definitions with software developers and end users.

The basic concepts behind the object definition cards are:

- an overall summary data card is used to define a rough hierarchical composition of the facade;
- exact definitions of each object may be found in individual cards; and
- the cards may be updated independently from other cards.

An example of an object definition card is shown in Fig. 3. The object definition cards contain a short explanation in text format, an EXPRESS-G-diagram and an optional picture of the object. The card may also contain an additional explanation in text-format and the EXPRESS-code of the object. The EXPRESS-code is also seen in Fig. 3. Twenty different object definition cards were created in this research and they are found in Karhu et al. (Karhu et al. 1994).

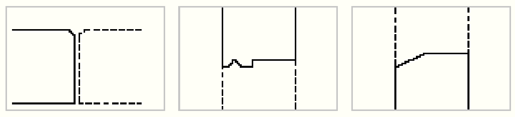
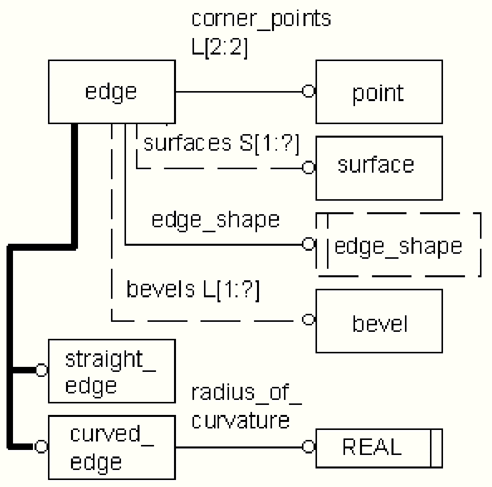
EDGE	*Author:* Vesa Karhu *Organisation:* VTT Building Technology *Date:* 30.12.1996
Decomposition level: Blng | Syst | Subs | **Part/space** | Comp	*Primary actor view:* Cli | **Arc** | Str | HVAC | El/Au | Cntr | **Sup** | Usr | Fmgr
	An edge is defined here as a line between two points. The edge shape is defined as a 2D-shape that is perpendicular to the trajectory between the two end points of the edge. Exampes are on the left.
 corner_points L[2:2] edge point surfaces S[1:?] surface edge_shape edge_shape bevels L[1:?] bevel straight_ edge radius_of_ curvature curved_ edge REAL	**ENTITY edge** SUPERTYPE OF (ONEOF(straight_edge, curved_edge)) ; -- *Properties* -- corner_points : LIST [2:2] OF point ; surfaces : OPTIONAL SET [1:?] OF surface ; edge_shape : edge_shape ; bevels : OPTIONAL LIST [1:?] OF bevel ; position : position ; id : id ; **END_ENTITY;**

FIG. 3: An example of an object definition card (RATAS 1996b).

3. BASIC DESIGN ACTIVITY MODEL

3.1 Traditional design process

The design process was modelled using the SADT notation. The syntax and notations used in SADT-diagrams are shown in Fig. 4. The activities are drawn as boxes that receive input. The activities are controlled by controls such as the design instructions. Mechanisms, that may be persons, machines or computer programs, produce the output of the activity. The diagrams are decomposed hierarchically where more detailed information and activities may be described in deeper levels of the diagrams.

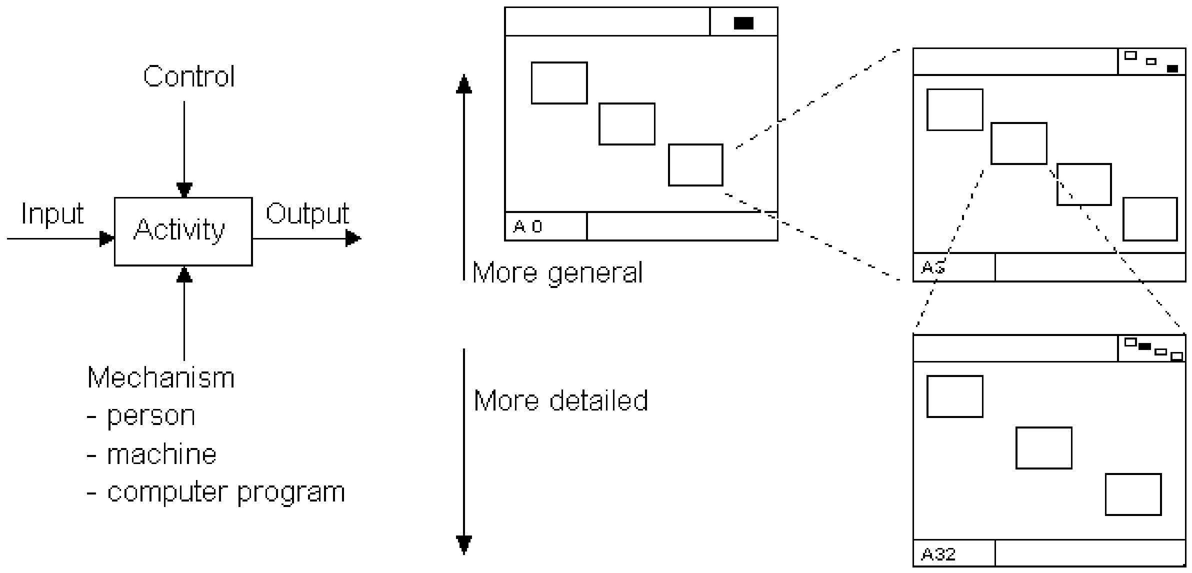


FIG. 4: The syntax and notation of SADT-methodology. The left part shows an activity box with its information flows. The right part the figure shows the hierarchical structure of the diagrams.

This activity model is based on the following premise: all data are accumulated and processed further, and consequently become more accurate as the designs reach the production stage.

Parts of the building design process activity model are shown in Fig. 5 and Fig. 6. Fig. 5 shows the main division of the building design process. The traditional building design includes general controls and methods for design activities. These are not shown as control to the activities. The process starts with a project brief and the programming. The main architectural design process begins in the activity *A3 Make global design*. The architect's tasks during briefing and programming are usually considered as the main designer's tasks.

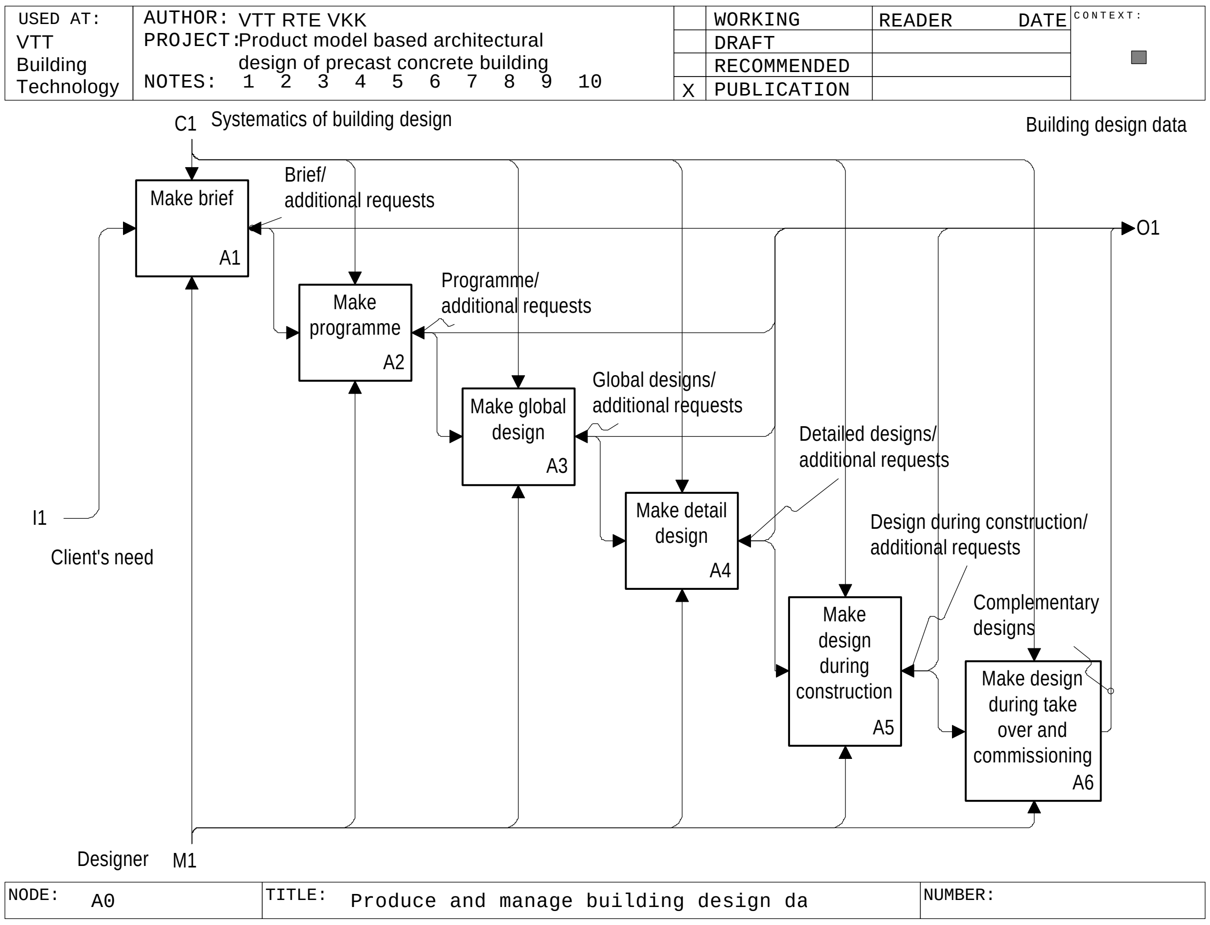

USED AT: VTT Building Technology
AUTHOR: VTT RTE VKK
PROJECT: Product model based architectural design of precast concrete building
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
X PUBLICATION
READER
DATE
CONTEXT:
C1 Systematics of building design
Building design data
Make brief A1
Brief/ additional requests
O1
Make programme A2
Programme/ additional requests
Make global design A3
Global designs/ additional requests
Detailed designs/ additional requests
Make detail design A4
I1
Client's need
Design during construction/ additional requests
Complementary designs
Make design during construction A5
Make design during take over and commissioning A6
Designer M1
NODE: A0
TITLE: Produce and manage building design da
NUMBER:

FIG. 5: The building design process is divided into six main activities.

A detailed diagram of a typical design of a facade is shown in Fig. 6. The process for designing the facade begins with the definition of the general shape of the facade. Openings and surfaces are usually designed next. Tiling may be designed if it is needed. In the next activity, *A4124 Define element division and joints*, it should be noted that the element division may be determined either by the architect or by the structural engineer, depending on the agreement that defines the responsibilities of each participant. The last two activities are used to define some properties of the structural layers and the edge shapes.

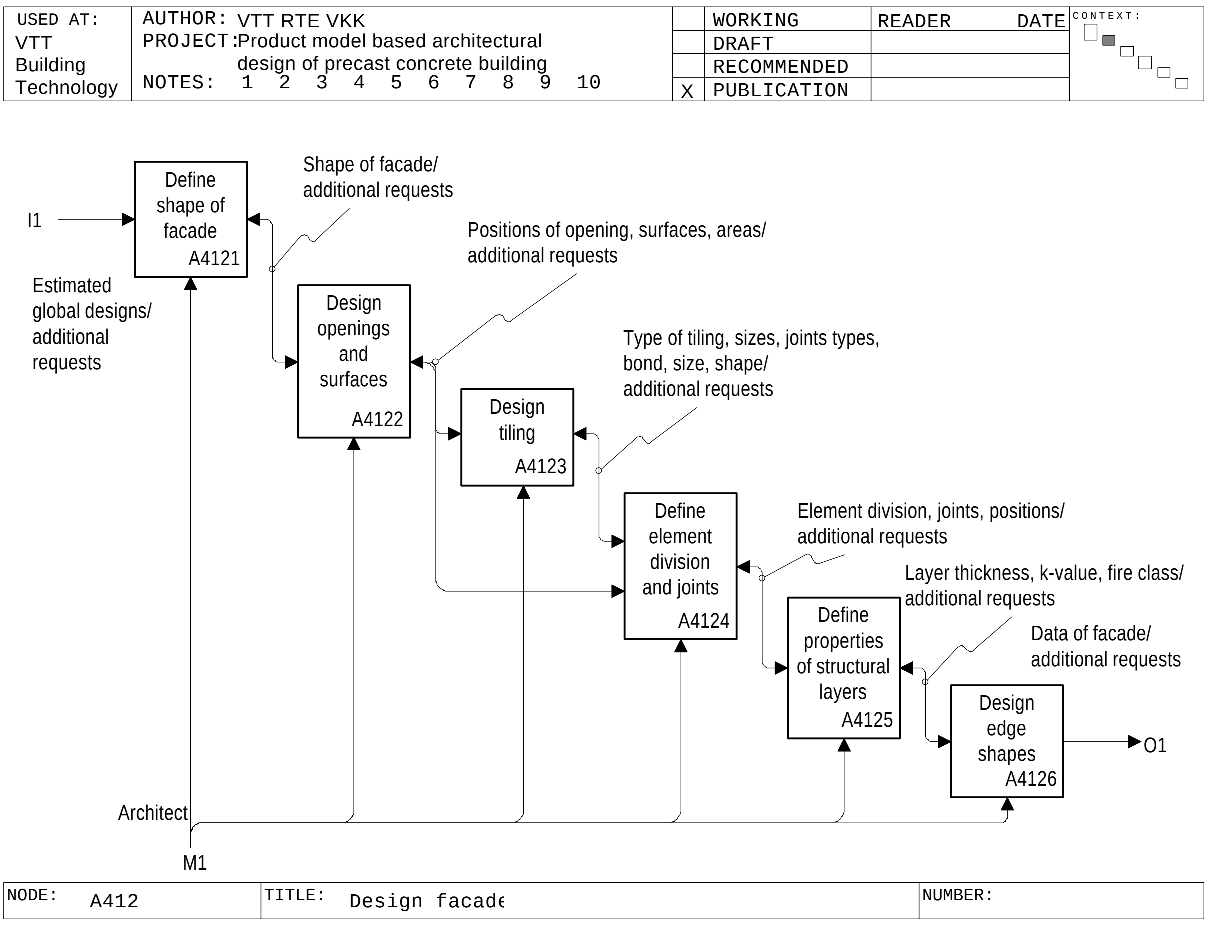

USED AT:
VTT Building Technology
AUTHOR: VTT RTE VKK
PROJECT: Product model based architectural design of precast concrete building
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
X PUBLICATION
READER
DATE
CONTEXT:
I1
Define shape of facade A4121
Estimated global designs/ additional requests
Shape of facade/ additional requests
Design openings and surfaces A4122
Positions of opening, surfaces, areas/ additional requests
Design tiling A4123
Type of tiling, sizes, joints types, bond, size, shape/ additional requests
Define element division and joints A4124
Element division, joints, positions/ additional requests
Define properties of structural layers A4125
Layer thickness, k-value, fire class/ additional requests
Design edge shapes A4126
Data of facade/ additional requests
O1
Architect
M1
NODE: A412
TITLE: Design facade
NUMBER:

FIG. 6: A part of the activity model for the design of a facade.

The complete activity model may be found in reference Karhu et al. (Karhu et al. 1994), in Finnish.

3.2 Analysis of current problems

Information concerning present day problems was gathered in interviews with experts from the companies that collaborated in the research project.

Current problems include:

- inaccurate data definitions in all design stages concerning the architectural design of the facade;
- lack of standard data exchange formats; and
- insufficient exchange of feedback information.

Traditionally, data exchange involving architectural design is done via paper documents, i.e. drawings, though the design data produced is in digital format. The usage of paper format is caused partly by the lack of data exchange standards and partly by the fact that the data contents have not been agreed on beforehand.

Another problem is that the effects of non-standard design solutions on total cost are not clear. Sometimes there is too much detailed information, yet the design as a whole is not clear. At the tender design stage, a tender offer is based on using a typical element as the basis for the cost estimation. The choice of this typical element is in many cases not correct which gives a wrong total cost effect.

Tenders are based on the experience of the person who calculates them. Experience from past projects and solutions is not used sufficiently. The element design is done in a hurry in many cases. This easily leads to lower level of quality.

4. THE PRODUCT DATA MODEL OF A FACADE

In the following section, the product data model of a facade is presented. The most important parts that are designed during the architectural design process (see also Fig. 6) and the model are shown and discussed. These are:

- structural layers;
- openings;
- surfaces; and
- edges.

Finally, the product data model as a whole is presented.

4.1 The decomposition hierarchy of facade

The facade is the front of a building usually given special architectural treatment (Webster 1996).

The hierarchical decomposition of a precast facade into its parts is as follows: a precast concrete facade consists of precast units, i.e., elements. An element in turn consists of structural layers. A structural layer is, for instance, an outer wall panel or an insulation layer. Structural layers and openings have edges, which are a certain 2D-shape. Layers may be arbitrary in shape. A precast facade has also element joints and an entity called element division. Exact definitions may be found in reference Karhu et al. (Karhu et al. 1994).

Going upwards in the decomposition hierarchy we see that a facade is a part of the object called external wall structure, which in turn is a part of the structural system. Highest up we find the building object itself. This part of the product data model is important for positioning the data in the context of the overall building product model, but was not relevant for the testing performed in this project.

It is seen that a facade may be a part of the external wall structure and a building frame. These in turn are parts of the structural system. In practice, a load bearing facade is considered as part of the building frame, a non-bearing facade belongs to the external wall structure. In this research the main focus was on the facade itself.

The EXPRESS-G subschema of the above classes is shown in Fig. 7.

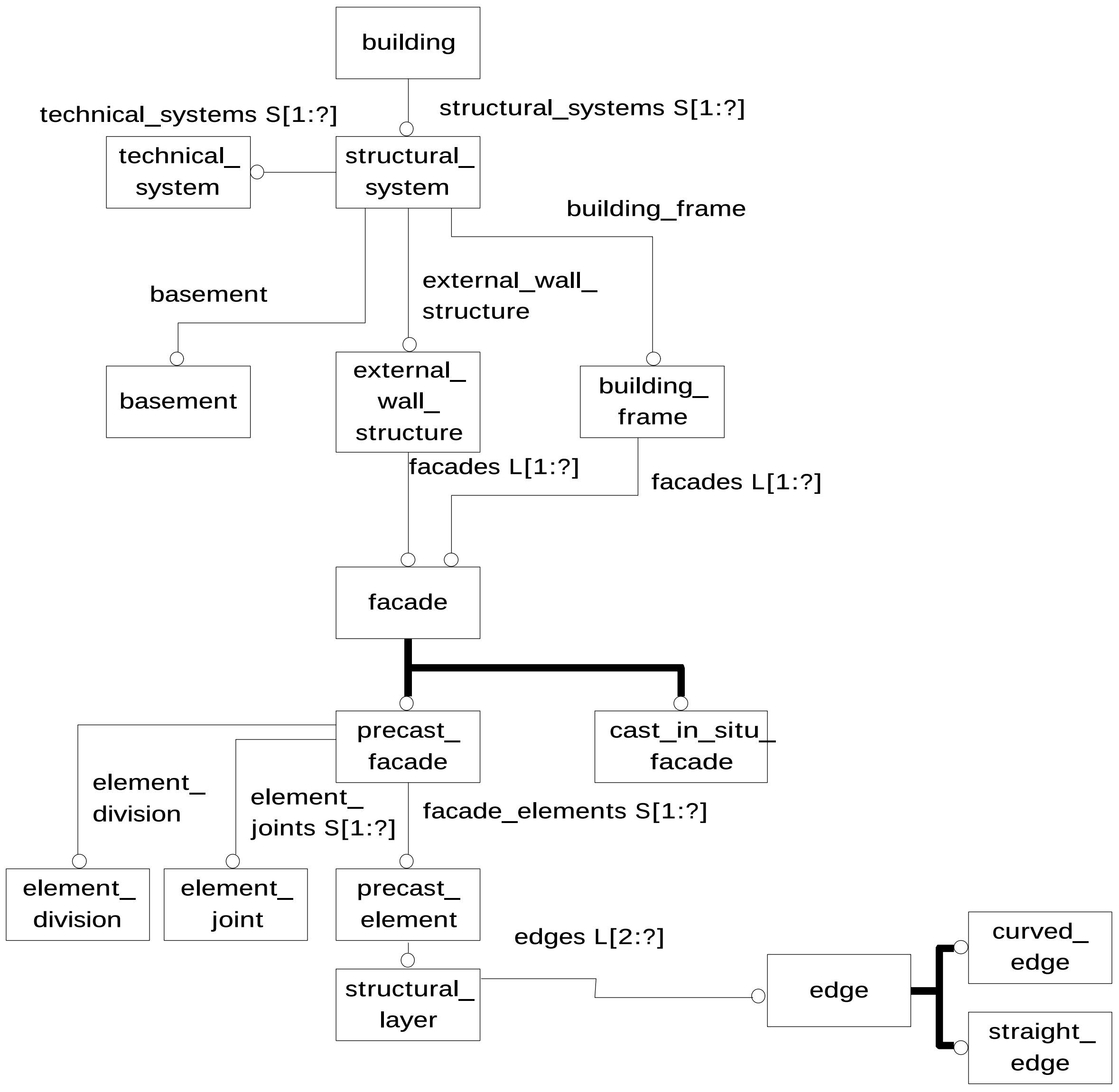


FIG. 7: The EXPRESS diagram of the main decomposition relationships of a facade.

4.2 Structural layer

A structural layer may have openings, see Fig. 8. An opening is in most cases filled by a window, a door or both. A surface may have several configurations such as tiling and it has many attributes (properties) such as colour. Structural layers form the precast units of facade, see Fig. 7. A structural layer has certain material. The layer has surfaces on each side. Windows and doors may be attached to structural layers directly, as well as through openings (see opening).

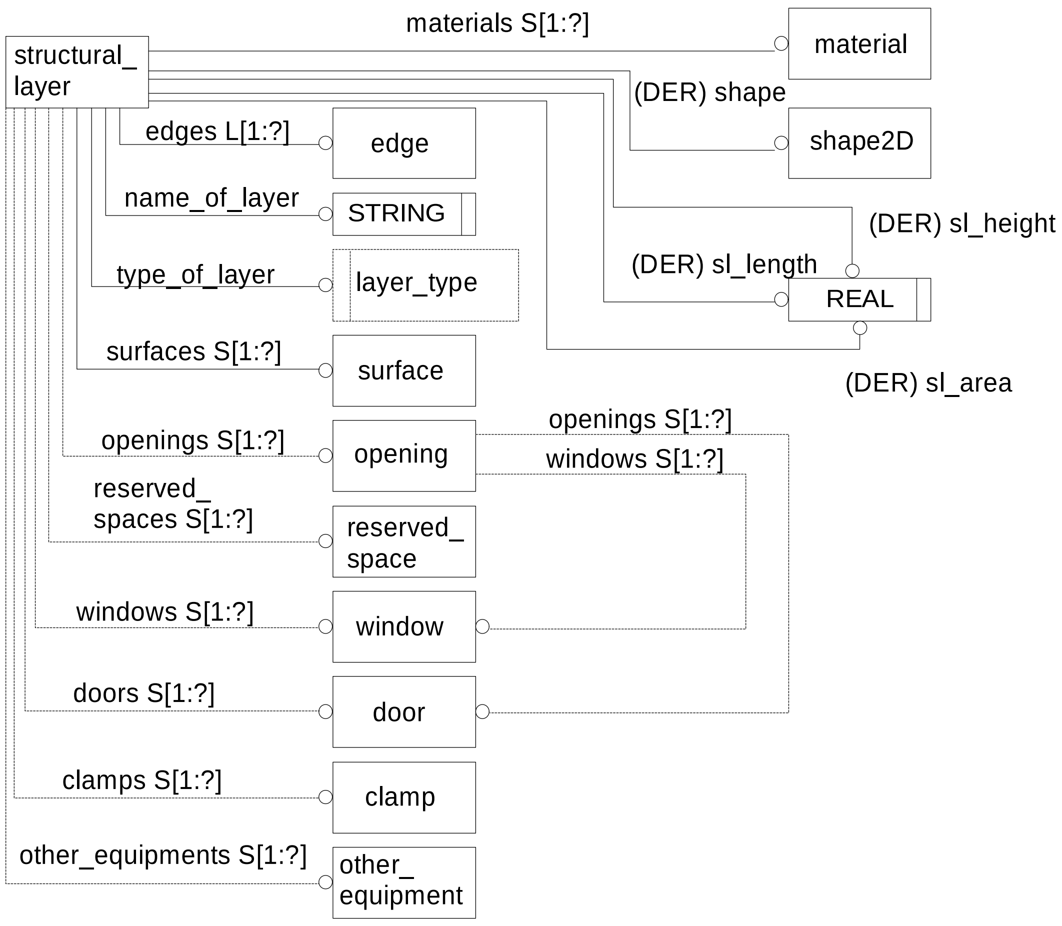


FIG. 8: The information structure for a structural layer.

Layer type is used for more accurate definition. Types include layers with a constant thickness, curved layers or layers where the thickness varies according to a mathematical function. The length, height and area of a layer may be derived from the edges. The reserved spaces (voids) are considered as to be filled with some equipment, usually HVAC equipment. It may be noted that an equipment may also fill an opening, see Fig. 9. Also, the 2D-shape of a layer is derived from the edges.

The definition of a structural layer may be compared to other existing solutions. In the IDM model (Integrated Data Model) of the COMBINE project, layers form a construction type which in turn is an attribute to an element construction (Dubois et al. 1992). In the French GSD (Groupe Structuration de Données) model, layers are defined as parts of composite enclosures. These in turn are a subtype of enclosure (GSD 1991). A layer as proposed by Björk (1992) is a super-type of internal or external layers.

4.3 Opening

The information structure of an opening is shown in Fig. 9. The opening is considered as passing through structural layers. The idea presented here is straightforward. It is also partly based on practical solutions and concepts adopted by designers. An opening is bounded by edges. Windows, doors and other equipment such as pipes are defined as optional attributes for the opening. Thus, an opening may also be empty which means that it may be interpreted as a hole.

Each layer may have an opening of its own. Thus, an opening of the whole element consists of individual openings for each layer. This approach is needed since the shape and size of an opening may vary in each individual layer.

In addition, an opening has an additional attribute in that it serves one or two spaces. This is not discussed here because spaces were not the main interest in this research.

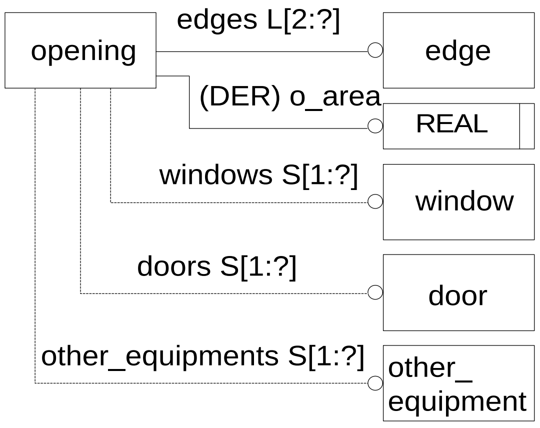


FIG. 9: The information structure for an opening.

A comparison to other models may be made. In the RATAS (RATAS-committee 1988), GSD (GSD 1991) and COMBINE's IDM (Dubois et al. 1992) models the opening is a super-type of a door or a window.

4.4 Edges

Edges may be considered as special parts of a 3D-object. An edge is defined here as a line between two points. The edge shape is defined as a 2D-shape that is perpendicular to the trajectory between the two end points of the edge. The information structure for an edge is shown in Fig. 10.

Edges are of two types: straight and curved. Edges could be defined in another way, i.e. edges are of one type which may have an optional property of radius of curvature. The developed software application uses a menu for selecting different edge shapes, see Fig. 11.

Also, the basic shape of an edge, as shown in Fig. 11, need not define the edge exactly. An edge may have a bevel in the practical structural software application solution. The practical solution for the edge shape is a select-type property where standard or user defined shapes are eligible. An edge may also have a surface. This surface may or may not be seen. For instance, a facade has a main surface that is seen from the outside. The surfaces of the edges may or may not be seen which in many cases depends on the joint width. The surface of an edge is usually considered a separate entity allthough it may be of the same material as the surface of the structural layer. The surface is explained in the next section.

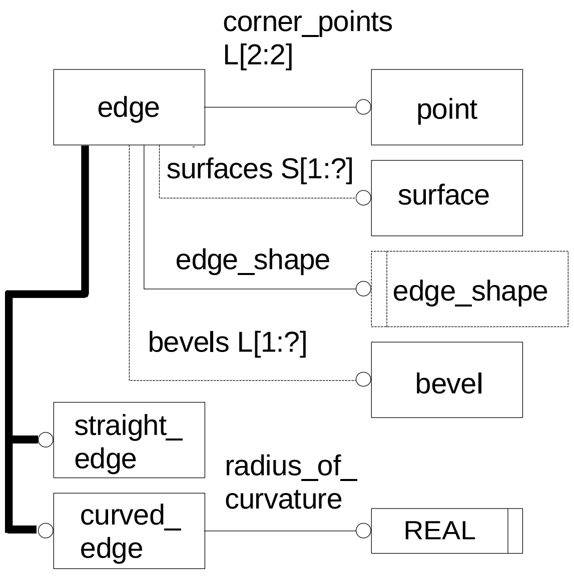


FIG. 10: The information structure for an edge.

An example of the interface for defining shapes of edges is shown in Fig. 11.

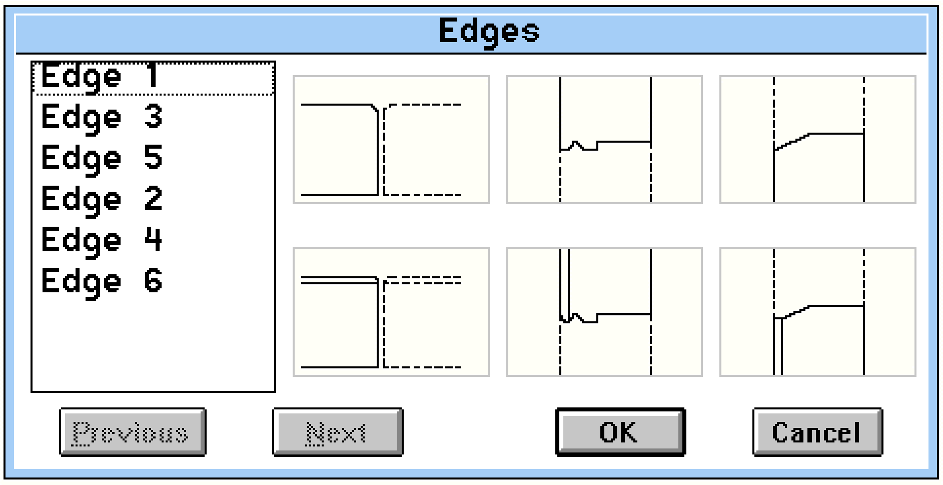


FIG. 11: Architect's edges-menu. Six different edge shapes in 2D are shown.

4.5 Surface

The information structure of a surface is shown in Fig. 12.

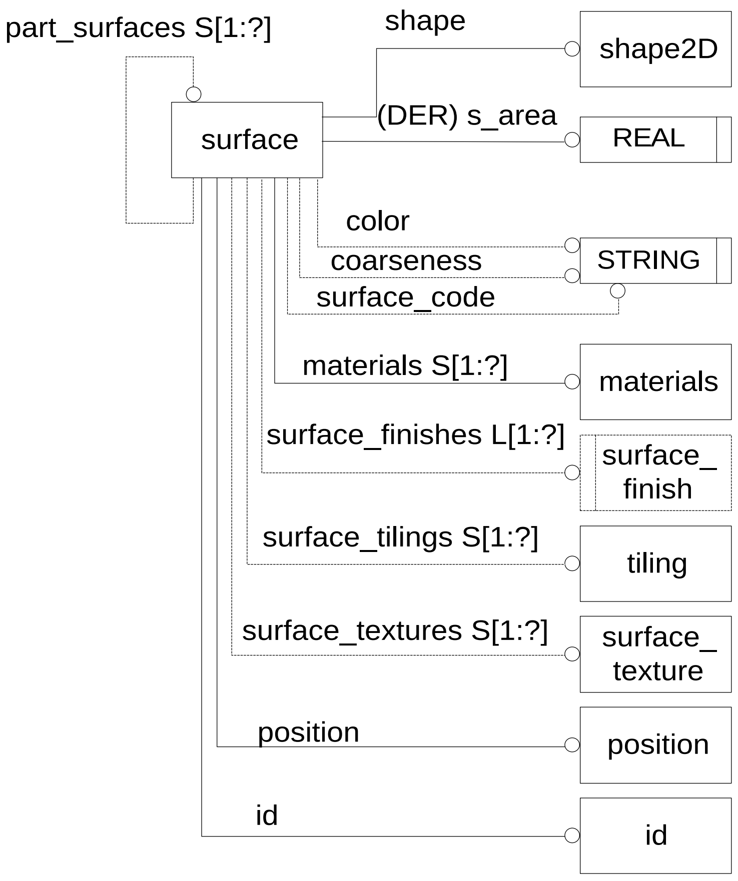


FIG. 12: The information structure of a surface.

In practical applications the 2D-shape is needed to determine areas of certain surface types. A surface may have many material alternatives, it may have a surface finish and may also have different surface patterns. Tiling may be attached to a surface, see also Fig. 16. A manufacturer may calculate costs in terms of areas of different surfaces. A surface finish is chosen from a list.

One may compare the presented solution to other models. The RATAS surface (RATAS-committee 1988) is an attribute of a room. The surface is a supertype of three different surfaces which are floor, ceiling or wall surfaces. A surface is a continuous area on the same wall, where a uniform surface material has been applied. A surface finish may be applied to a surface.

In COMBINE's IDM model, a face is defined as a surface which may have a surface finish (Dubois et al. 1992). It also defines sub-faces. On the element level an enclosure element's geometry has a 2D aspect which is an element surface.

4.6 Overall model

In Fig. 13, the subschemas presented in Fig. 8, 9, 10 and 12 have been integrated and are shown as one schema. For readability reasons some attribute definitions have been omitted from this figure. The full textual EXPRESS code is shown in Appendix A.

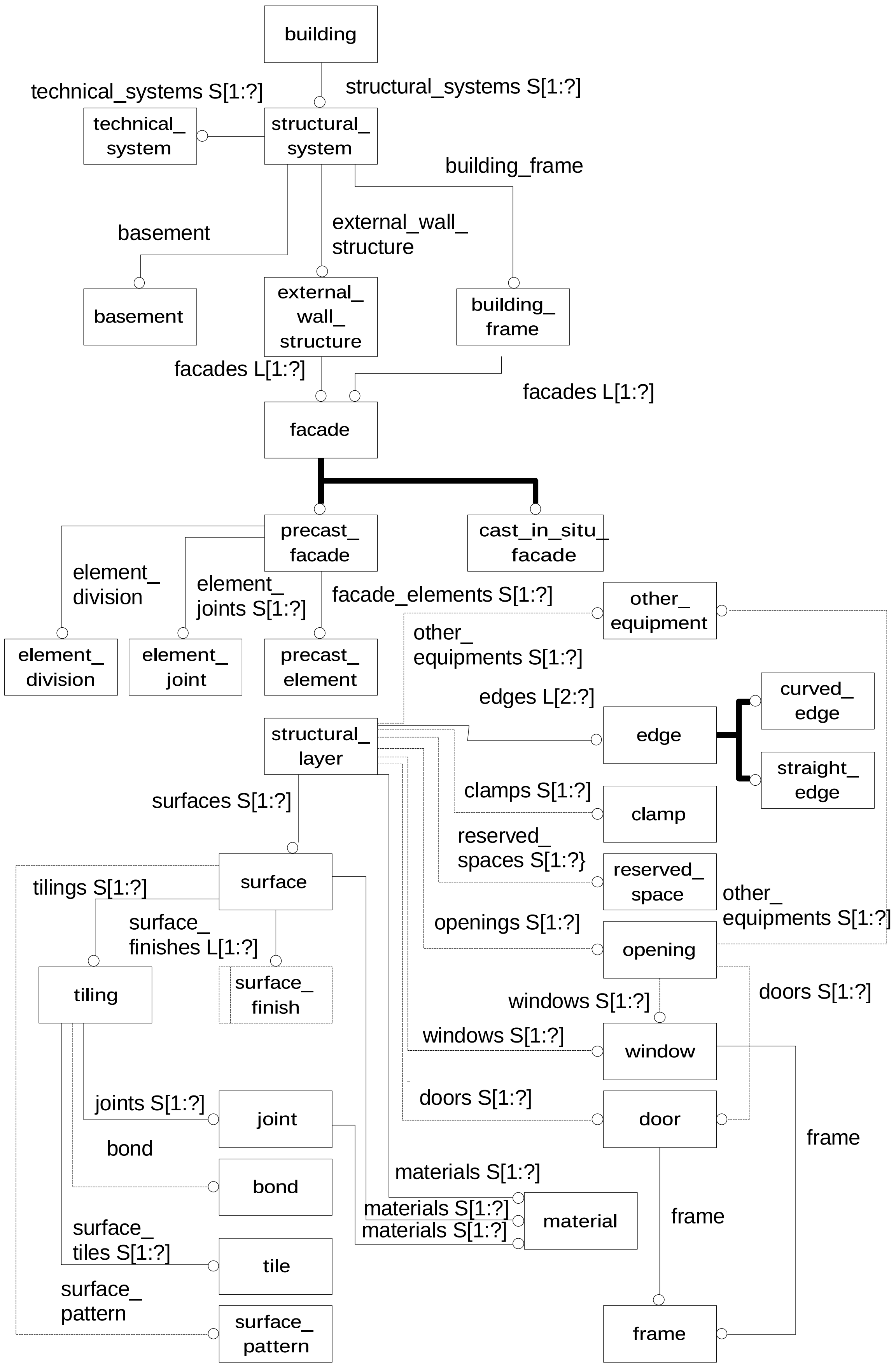


FIG. 13: The overall EXPRESS-G-diagram of the facade.

5. CHECK LISTS

Check lists were used to combine the product data model with the activity model. Such lists are a practical solution that helps in defining the data requirements during the different design stages. The check lists are based on the activity model (see Fig. 6) and the product model (Fig. 13). An example is shown in Fig. 14. For instance, a manufacturer and an architect may agree that the data in the global design stage shall include the shape and position of the facade, the position of the element division and the type of the element joint. The data in the so called tender design stage include more information. The tender design stage is normally the last phase in the global design stage.

This results in a clear definition of the data flow during the whole building design process. The data producer and the data user may be shown in the lists. The check-lists may also be used as supplements to contracts.

Each activity outputs some data of the facade. The check lists are more readable than the product model or the activity model.

CHECK LIST		STAGE		
		global design	tender design	detail design
facade	shape	x	x	x
	position	x	x	x
element division	position	x	x	x
element joint	type	x	x	x
	width		x	x
	length			x
	shape		x	x
	position		x	x
joint material	name		x	x
	colour		x	x
	manufacturer			

FIG. 14: An example of a check list extract. The same check list may be used to define data needs during different design stages as well as for defining limitations. Tender design yields documents for the invitation to tender. It is the first part of the detail design.

The area of different surfaces and element types could also be added to the check lists. This is purely for practical reasons as manufacturers usually calculate costs based on areas. The check lists should be limited in the practical implementations. A large number of entities will yield a large check list.

The check lists that were developed during this research were further included in a recommendation in Finland (RT 1994).

6. DEVELOPING PROTOTYPE SOFTWARE

A prototype computer application was developed to test the product data model. The software platform was AutoCAD. The reasons for this choice were that the architectural company involved in the research uses it and that it has been used for many prototype applications by the research group at VTT Building Technology.

The feature of AutoCAD which was used in the initial phase of the prototype development was the so-called block. A Block is a named assembly of graphical objects and attribute definitions. Graphical objects are lines, polylines, etc.. Attribute definitions contain alphanumeric information. Additionally, a Block may contain an instance of another block.

It turned out that it is difficult to modify blocks dynamically during the design procedure for which reason they were not used in later prototypes. The product model was consequently later implemented using AutoCAD's extended entity data feature (AutoCAD 1992).

The physical data exchange between the different applications was implemented as OXF files. OXF is an object-based data exchange format developed in a research project at VTT (Serén et al. 1993). In scope, it is rather similar to the STEP physical file format (ISO 1994). The main reason to use OXF was the availability of software supporting the generation of OXF files from AutoCAD as well as the technical features of the format. A fuller explanation of these is, however, outside the scope of this paper.

The so called type objects define common properties of one or more instance objects. Instance objects define occurrences of type objects. Type objects may consist of a number of child objects which are instances of other type objects (Serén et al. 1993). The principle of the CAD-implementation between the AutoCAD-entities and the neutral data exchange file is shown in Fig. 15.

FIG. 15: Principle of the CAD-implementation illustrated by mapping between AutoCAD-entities and the neutral data exchange file (Serén et al. 1993).

The programming of the new functions to the prototype was made using AutoCAD's AutoLISP programming language. AutoLISP is a dialect of the LISP (LISt Processing) programming language. The main functions of the prototype were:

- define or modify an attribute set;
- display an attribute set on the screen;
- insert a type object;
- display a type object on the screen;
- edit an instance of a type object, regardless of that the instance may contain instances of other type objects;
- copy instance objects;
- create or show relationships between instances;
- export data to a neutral exchange file; and
- import data from a neutral exchange file.

The prototype was used to design a facade that consists of a few elements. Two designs were made: one for the whole facade, one for a single element. After these a neutral data exchange file was created in OXF-format. An example of how the data were stored in the extended entity data of polylines is shown in Fig. 16. A structural layer and a surface with tiling are shown in the figure.

For instance, the tiling is stored in a polyline called TILING. The polyline has attribute sets named STRUCTURAL_LAYER and SURFACES. The DATA is identified as an area that is denoted by A. Other entities are treated the same.

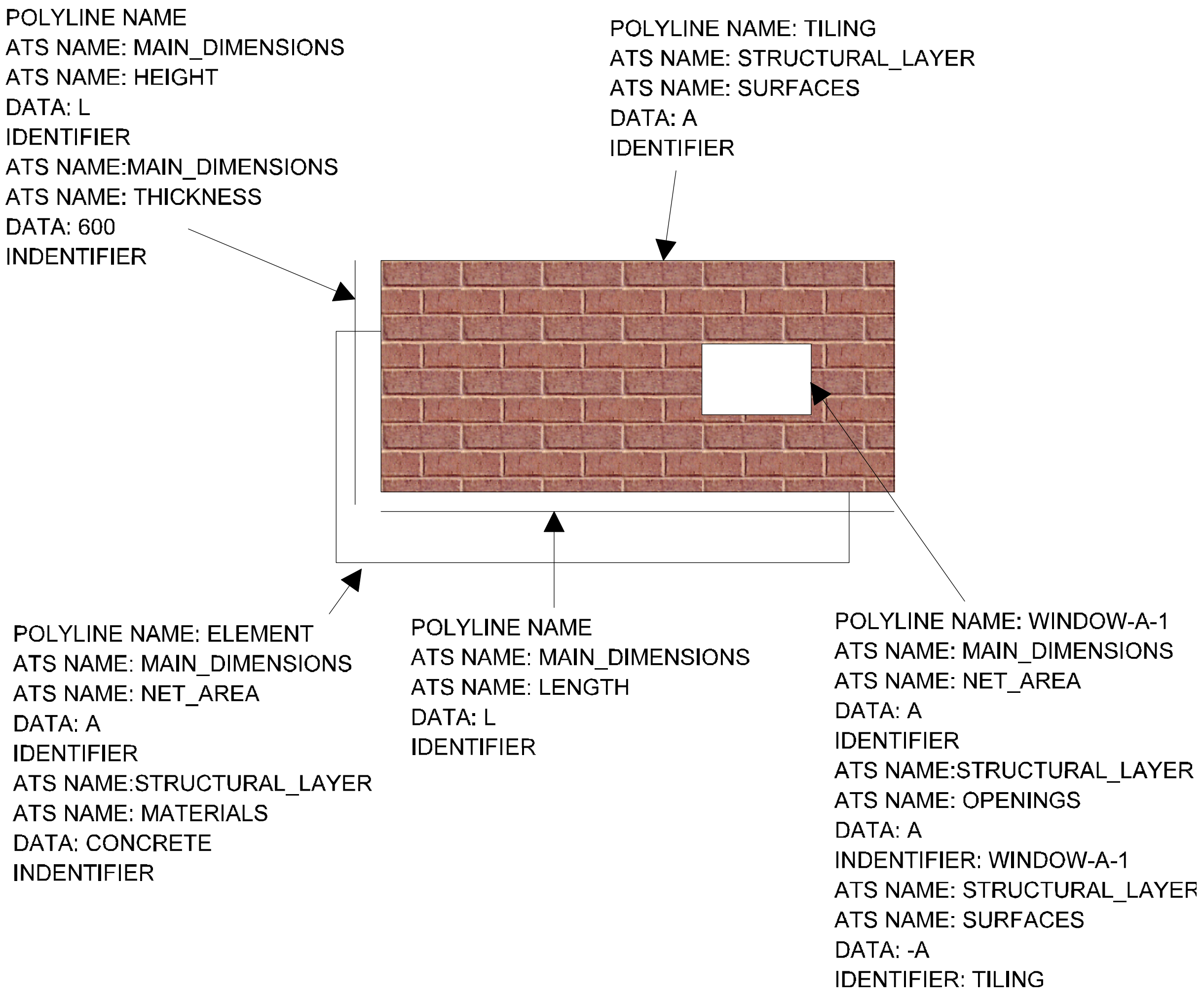


FIG. 16: The dimensional data of an element. The data are stored in AutoCAD's extended entity data.

The final prototype application that was used in a test project (see section 7) was developed in the architectural company. This application also uses AutoCAD's extended entity data for storing the information. The architect's AutoCAD was version 12. The application was developed to enhance the architectural design procedure.

The final prototype application has a simple interface that enables to create an id for an element and attributes that are needed. The functions that were included in this application were:

- design of an element;
- design of an area;
- design of an egde and joints; and
- output to a file.

The prototypes take into account the architect's normal way of design. This means that the produced data are more accurate and detailed in the later design stages (from the global design stage to the detail design stage). Also, as mentioned earlier, the architectural data focuses more on surfaces and openings, not structural details.

7. TESTING THE DEVELOPED METHODS IN PRACTICE

The prototypes were tested with data from a real design project, in collaboration with the firms that participated in the research. The purpose was to test:

- the usefulness of having architectural design and element design data available in a data base format, rather than as ordinary drawings, during the detailed element design and manufacturing;
- if the particular product data model presented above in Chapter 4 has the functionality required (in terms of data content and data accuracy) for supporting the digital exchange of data between the project participants; and
- the effects of using a product model approach on the management of design modification data during the design and manufacturing process.

Testing was carried out as a dry run using information about of a real, recently designed and constructed residential building. The building which was used as a case is a multi-storey apartment house in a suburb of Helsinki (Fig. 17), and is representative of the precast concrete technology currently used for residential buildings in Finland. The total area of the building is 7339 m^2 and it includes 95 apartments.

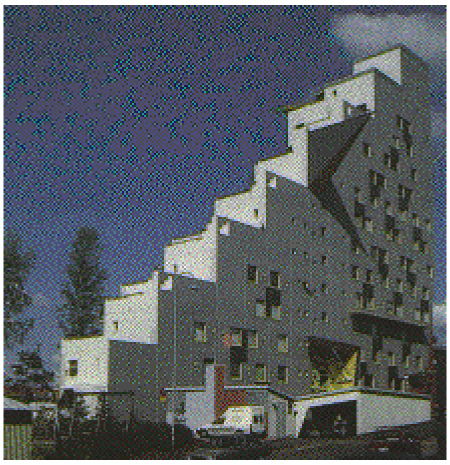


FIG. 17: The test building used in this research.

This particular building was chosen because its design and construction timetable was suited to the schedule of the research project and because the structural designer and the element manufacturer had been involved in the project. Consequently it was possible, at least subjectively, to estimate the effects of using the product model approach, compared to the normal data exchange practice which had been used in the project. In the test, a part of the eastern facade of the building (storeys 13-15) was redesigned and the data were exchanged between the project participants as in the real project.

In addition to the prototypes that were developed as a part of this study commercial applications had to be used in order to complete the whole design and manufacturing ”circle”. The resources available for the research were not sufficient for the development of prototypes that would have covered all the necessary tasks.

The software tools that were used in the testing are listed below:

Architectural design used:

- Prototypes developed in the study on top of AutoCAD (version 12).

Structural or element design used the following modules of ConcreteCAD:

- Gides (whole building, whole facades);
- Exwall (detailed element design).

Element manufacturing used:

- ConcreteCAD;
- In-house CAM-software of Partek.

ConcreteCAD is suite of software modules which have been tailor-made for the design and manufacturing of prefabricated concrete buildings (ConcreteCAD 1992). It is widely used in Finland and has users also in several other countries. The structure of ConcreteCAD is centred on having one and only one model of the building being designed, which is stored in a customised database. Drawings and other output documents are generated from the information in the database. ConcreteCAD was also chosen because the industry practitioners who participated in the testing were already using it.

The manufacturer has a modern facility using computer controlled adjustable element moulds. The software used by the element manufacturer for production control has a relational data base as its platform. In addition the factory uses Concrete CAD for some in-house design applications.

Before the actual design started some initial small-scale tests of the developed AutoCAD prototypes were carried out. Special interest was focused on how data from the architectural design stage may be directly used in the element design. The architectural data that are defined and may be used are modular dimensions, surfaces, openings and edge shapes (see also Fig. 6). In addition, the architectural data consists of details concerning window clamps and fasteners. A number of corrections were made based on the feedback received. The major points from the feedback were:

- The edges of elements were not stored in the architect’s model as it contained only surfaces and shapes of edges.
- The local element coordination in the ConcreteCAD program is calculated from the position of the first edge that is input to the database. This caused that an individual element had an arbitrary position when it was input to the ConcreteCAD program.
- Some openings were not defined, they were just empty areas between surfaces.
- Some edge shapes were missing due to a bug in the architect’s program.
- The DXF conversion did not match exactly.

The following conventions were adopted:

- The origin of the local co-ordinate system of an element will be placed on the lower left corner of the element (seen from the outside of the building). The first edge shall start from the origin.
- All openings shall be designed as real openings because a surface may consist of parts that form the whole surface. The initial tests showed that the architect's "opening" was just an area with no surface.
- The edges of an element shall be designed counter-clockwise.
- An element shall have an attribute which specifies the location of the element in the global co-ordinate system.

After these modifications the actual test design was started. The testing was divided into two main phases:

- phase A included a normal design procedure; and
- phase B included a change in the design solution.

In the phase A the eastern facade of the building for storeys 13-15 was designed. This included the design of the elements by the architect. The architectural designs were used as input to the structural (element) design. The architectural design was started by generating modular dimensions, surfaces, thickness of structural layers and materials. The openings were designed next. The shapes of edges were designed for each individual structural layer including shapes in the openings.

In the phase B the architect made two changes: an element was redimensioned by 300 mm (width) and the position of one opening was changed. These changes resulted in different proportions of surface material. The changes are critical from the manufacturer's point of view.

The main interest in each phase, A and B, was focused partly on the reuse of data from the architect's application by the structural engineer's application, and partly on the exchange of data from the structural engineer's application to the manufacturer's system. In addition, possible errors and lack of data were studied.

The testing was carried out in accordance with the activity model which had been developed in the project. The test design "stage" corresponds to the detail design stage in the activity model (see Fig. 5 and Fig. 6).

The architect used the application which had been developed in the project. This application creates a data base using the conceptual schema presented above, although this ”database” lacks some of the functionality of object-oriented or relational databases, since it has been implemented as part of a sequential CAD file using AutoCAD’s extended entity data. The architect designed the surfaces, openings and some window details. Surfaces consisted of many sub-surfaces. The types of edges, i.e., edge shapes, varied in the openings. The types were chosen using a menu, see Fig. 11. Finally a data exchange file was produced in ASCII format.

In the next stage these data were transferred to the structural engineer’s ConcreteCAD application. Although it would have been possible to write software that could have done this conversion automatically, the conversion was done manually, by interpretation of the human-readable files produced from the architect’s application. The reasons for this were purely due to resource restrictions, since developing such conversion software would have cost too much, in comparison to the overall budget of the project. The data exchange was thus only simulated. The important thing is, however, that the data contained in the architect’s model was transferred to the receiving application, not the speed of the conversion.

After this the element designer started his work using ConcreteCAD's Gides-module. The element division was chosen, based on the modular dimensions of the architectural design. The thickness and material types were designed for each structural layer. Elements were created according to the element division. The element division is decided either by the architect or the structural engineer, depending on what has been agreed. In this example, the architect decided the element division.

The edges were then defined. It showed that the architect had designed edges of the outer wall layer using modular dimensions for which reason they had to be moved inward by 7.5 mm. The insulation layer and the inner structural layers were also treated in the same way. The openings were designed using the same principle. The edges of openings had to be moved by a few millimetres because the architectural data used modular dimensions.

The rest of the structural design was accomplished as a normal design. Reinforcement and other equipment were defined. For further elaboration of the data, the following information for each element was saved in the database of the structural designer:

- ID of the element (precast unit);
- name of each layer, dimensions and relative position;
- surfaces and their materials;
- size and relative position of opening;
- shape of edge of each layer;
- reinforcement rebars of each layer;
- shapes, size and relative position of reserved spaces;
- name, shape, size and relative position of additional equipment.

The position of each structural layer was relative to the position of the element. Elements were positioned in the global co-ordinate system of the facade.

Element joints may cause problems: module dimension versus real dimension. The architect may design joints exactly by shapes and dimensions but uses modular dimensions on positioning on the element level. Another problem may occur: if the edge of an opening coincides with the edge of a structural layer, i.e., the opening may not be considered as an opening anymore.

In the last stage, the structural engineer passed the designs to the manufacturer electronically using an Ethernet network. Since the manufacturer also used ConcreteCAD up-front no conversion was needed. A macro in ConcreteCAD's Exwall module created an ASCII-file in which the manufacturer added some internal parameters such as product type number and the name of the "lowest" layer during casting. This ASCII file was then used as input information for the production planning software of the manufacturer.

In addition a material table in BEC-format (BEC 1991), (Hannus 1990) was created from the data transferred to the manufacturer. BEC was developed in Finland during the 1980s. The whole BEC-system includes a number of software modules for the receiving and manipulation of graphical BEC-files and for the management of tables. An amount of data to manufacturers of precast concrete structures are exchanged in BEC-format in Finland.

The results of the tests were both quantitative and qualitative. The participating designers were asked to compare the time used when using the product model approach to the way the work had originally been carried out. The results were conclusive to some extent. The total amount of time spent was 20 % longer using the product model approach in phase A. The design changes were 7 % longer. On the other hand this was largely due to the substantially increased workload of the element designer because some data were manually converted from the architect’s application to the element designer’s application. (The data existed in the architect’s designs but there was no conversion program.)

Both the architect and the manufacturer saved time. Compared to the drawing oriented approach the manufacturer may make changes approximately 30 % faster in the product model approach because the data of an element is in computer interpretable format.

More important were the qualitative advantages which the participants testified to in the interviews carried out. The conceptual schema of the facade proved to be sufficient. This led to a better data accuracy and the data content was clearly defined already at an early stage of the building design process. The test showed that the architectural data are usable as such in the element design which means that the normal procedures of architectural design need not be changed. Data in earlier building design stages are considered more general, i.e., facade is designed as a whole where surfaces and opening are placed.

The test also showed that having usable architectural data the element designer may concentrate on essential design matters. Especially, the shapes of edges as defined by the architect reduced the routine design work of the element designer. It is of importance to note that the shapes of edges depend on the manufacturer.

The manufacturer has an advantage: cost estimations may be done earlier than normally. This will also enable more data interchange between the architect and the manufacturer because solutions may be based on the manufacturer’s suggestions. The automation level of the manufacturing process may be improved only through more accurate data.

In the long run such qualitative advantages will no doubt be converted into monetary benefits due to shorted design times, fewer mistakes, etc.

Limitations of the developed model exist. For instance, a window may be included in an opening which also means that an opening is always surrounded by material of the structural layer, the opening is inside the layer. A window is not attached to an opening but to the side of the layer in some cases. It means that the window is no more in an opening of a layer but in an opening of the facade. The problem is illustrated in Fig. 18. The edge of an opening coincides with the edge of the structural layer. It becomes an “abnormal” opening. The presented model is limited to openings in the structural layers only.

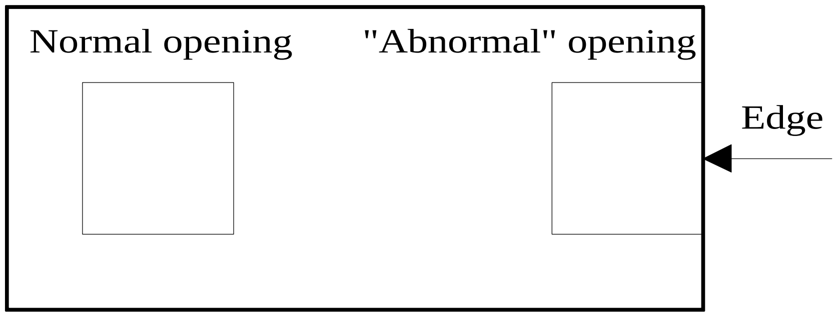


FIG. 18. The problems of an opening: the edge of an opening coincides with the edge of the structural layer.

8. CONCLUSIONS

The architectural data of a facade are complicated. The test verified that the data defined by the architect such as modular dimensions, surfaces, openings and edge shapes, are usable directly in the structural design. Thus, the data are also usable for the manufacturer of the precast facades. It also showed that it is possible to produce the data in computer interpretable format without changing a "normal" design procedure. However, this requires well designed computer applications. The product model based approach forces more accurate data definitions which in the long run results in cost reduction in terms of error and redundant design work reduction.

The basic results obtained in this project may be exploited in all stages of the process of designing and manufacturing prefabricated elements even if the test was focused on the detail design stage. Architects and other participants in a design project may use the check-lists to define their data needs during all stages of the process.

Developers of software applications may utilise the object definition cards and the conceptual schema of a precast facade to develop new software applications. Also, manufacturers may standardise their data needs. A number of criteria need to be set to the software applications. An application shall be able to handle:

- surfaces (material and area);
- openings (dimension and area);
- edge shapes of openings and structural layers;
- the facade as a whole;
- element joints;
- the coordinates of a structural layer.

The coordinate systems for each object should be standardised or agreed at the beginning of each project. Furthermore, an application shall support the normal design procedure of architect.

Extensive exploitation of the ideas and results presented here requires standardising the data exchange structures. ISO/STEP development may be (ISO 1992b) a solution in the form of an application protocol. Also, IAI (International Alliance for Interoperability) and its IFC (Industry Foundation Classes) may give solutions (IAI 1996).

This research was restricted to the traditional design process which has not used the product model approach so far. The SADT method was well suited to the overall description of the activities that occur during the traditional building process. The integration of the activity model to the conceptual model is of importance. Also, the influence of the product model to the process itself is of interest, especially when feedback and experience from past designs and solutions are to be used.

9. ACKNOWLEDGEMENTS

This research was funded by the Technical Research Centre of Finland (VTT), The Finnish Association of Construction Products Industries (RTT) and the Technology Development Centre of Finland (TEKES). The companies that participated in the testing were the architectural company Arkkitehtitoimisto Innovarch Oy, the structural engineering company Tampereen Juva Oy and the manufacturer Partek Betonila Oy.

10. REFERENCES

AutoCAD (1992). AutoCAD Version 12 Manual. Neuchatel, Autodesk BV.

BEC (1991). Manual. Betonielementti-CAD. Helsinki: Finnish Concrete Association.

Björk, B-C. (1992). A conceptual model of spaces, space boundaries and enclosing structures. Automation in Construction. Vol. 1, No. 3, pp. 193-214.

Björk, B-C. (1995). Requirements and information structures for building product data model. VTT Publications 245. Technical research centre of Finland. Espoo.

Boman, M., Bubenko, J. and Johannesson, P. (1991). Conceptual modelling. Department of Computer and System sciences, Stockholm University. Second edition. Stockholm.

ConcreteCAD. (1992). ConcreteCAD Manual. Cadex Oy. Finland.

Dubois, A.-M., Escudié, J.-C. and Laret, L. (1992). COMBINE Integrated Data Model, Volume I - NIAM diagrams, Volume II - Dictionary, V.3.3, 920123, CSTB, Sophia Antipolis, France.

Gielingh, W. (1988). General AEC reference model. ISO TC 184/SC4/WG1 doc. 3.2.2.1, TNO building and construction research, BI-88-150. Delft.

GSD. (1991). Synthese des modeles conceptuels développés dans le la cadre de la recherche batiment en France, Plan Construction et Architecture, Ministre de'l equipement, du logement, des transports et de l'espace, Paris.

Hannus, M. (1990). Computer aided design of component based buildings. Stuttgart. Conference proceedings. p. 6-1 -6-9.

IAI. (1996). Industry Alliance for Interoperability. http://www.interoperability.com/

ISO. (1985). Concepts and terminology for the conceptual schema and the information base. ISO/DTR 9007 (TC 97). Published also as SIS technical report 311, Standardiseringskommissionen i Sverige, Stockholm.

ISO. (1992a). Draft International Standard DIS 10303-11 (STEP). Part 11: The EXPRESS Language Reference Manual.

ISO. (1992b). Draft International Standard DIS 10303 (STEP), Part 1: Overview and fundamental principles. TC 184 SC 4

ISO. (1994). Draft International Standard DIS 10303 (STEP), Part 21: Clear text encoding of the exchange structure.

Karhu, V.; Hannus, M.; Pellosniemi, J. (1994). Elementtirakennuksen tuotemallipohjainen arkkitehti-suunnittelu [Product model based architectural design of prefabricated building]. VTT Research Notes 1568. Technical Research Centre of Finland, Espoo.

Luiten, G. (1994). Computer Aided Design for Construction in the Building Industry. Ph.D. thesis, Delft University of Technology.

Marca, D. A. and McGowan, C. L. (1987). SADT - structured analysis and design technique. New York, McGraw-Hill.

PM Glossary. (1996). http://www.vtt.fi/cic/service/glossary/glossary.html

RATAS-committee. (1988). Rakennuksen tuotemalli. (The building product model), Rakennuskirja Oy, Helsinki, Finland.

RATAS. (1996a). http://www.vtt.fi/cic/projects/ratasimp/

RATAS. (1996b). http://www.vtt.fi/cic/projects/ratasimp/edge.htm

RT. (1994). RT 10-10557. Valmisosarakentamisen tiedonhallinta. Betonielementtirakentaminen. [Building work. Precast concrete construction.]. Helsinki: Rakennustieto Oy. 20 p.

RT. (1995a). RT 10-10575. Rakennuttamisen tehtäväluettelo. [Scope of work client's work in building construction.]. Helsinki: Rakennustieto Oy. 14 p.

RT. (1995b). RT 10-10576. Arkkitehtisuunnittelun tehtäväluettelo. [Scope of work in architectural design.]. Helsinki: Rakennustieto Oy. 12 p.

RT. (1995c). RT 10-10579. Talotekniikan suunnittelun tehtäväluettelo. [Scope of work in building services design.]. Helsinki: Rakennustieto Oy. 12 p.

RT. (1995d). RT 10-10580. Geosuunnittelun tehtäväluettelo. [Scope of work in geotechnical design.]. Helsinki: Rakennustieto Oy. 8 p.

Serén, K.- J., Hannus, M., Karstila, K., Kihlman, M. and Pellosniemi, J. (1993). Object-oriented CAD tool implementations for the construction industry. VTT Research Notes 1460, Laboratory of Structural Engineering, Technical Research Centre of Finland, Espoo

Turner, J. (1990). AEC building systems model. ISO TC184/ SC4/ WG1, Doc. N363. Working paper.

Watson, A. (1995). To product models and beyond. In: Brandon, P. and Betts, M. (eds). Integrated construction information. E&FN Spon, London. pp.159-172.

Webster. (1996). http://gs213.sp.cs.cmu.edu/prog/webster?facade

WWW STEP. (1996a). http://www.scra.org/uspro/stds/ap_text.html

WWW STEP. (1996b). http://www.nist.gov/sc4/tools/express/

11. APPENDIX A

```
SCHEMA Design_facade ;

ENTITY acid_treatment ;
END_ENTITY ;

ENTITY acoustic_property ;
END_ENTITY ;

ENTITY balcony ;
position        : position ;
id              : id ;
relations       : SET [0:?] OF relation_type ;
END_ENTITY ;

ENTITY basement ;
END_ENTITY ;

ENTITY beam ;
END_ENTITY ;

ENTITY bearing_wall ;
END_ENTITY ;

ENTITY bevel ;
END_ENTITY ;

ENTITY biological_property ;
END_ENTITY ;

ENTITY bond ;
bond_type       : STRING ;
starting_point  : LIST [1:?] OF point ;
END_ENTITY ;

ENTITY brick_tile
SUBTYPE OF (surface_tile);
END_ENTITY ;

ENTITY brush_treatment ;
END_ENTITY ;

ENTITY building ;
structural_systems      : SET [1:?] OF structural_system ;
technical_systems       : SET [1:?] OF technical_system ;
id                      : id ;
position                : position ;
```

```
shape                   : shape ;
address                 : STRING ;
number_of_storeys       : INTEGER ;
area_of_storey          : REAL ;
volume                  : REAL ;
purpose_of_usage        : STRING ;
quality_level           : STRING ;
total_costs             : REAL ;
fire_class              : STRING ;
END_ENTITY ;

ENTITY building_frame ;
columns         : SET [0:?] OF column ;
beams           : SET [0:?] OF beam ;
slabs           : SET [0:?] OF slab ;
bearing_walls   : SET [0:?] OF bearing_wall ;
facades         : LIST [1:?] OF facade ;
END_ENTITY ;

ENTITY cast_in_situ_facade
SUBTYPE OF (facade) ;
structural_layers : LIST [1:?] OF structural_layer ;
END_ENTITY ;

ENTITY channel ;
END_ENTITY ;

ENTITY clamp ;
END_ENTITY ;

ENTITY cleansing;
END_ENTITY ;

ENTITY column ;
END_ENTITY ;

ENTITY curved_edge
SUBTYPE OF (edge) ;
radius_of_curvature     : REAL ;
END_ENTITY ;

ENTITY curved_layer ;
END_ENTITY ;

ENTITY door ;
d_type          : STRING ;
fire_class      : STRING ;
frame           : frame ;
position        : position ;
id              : id ;
END_ENTITY ;

ENTITY durability_property ;
END_ENTITY ;

ENTITY edge
SUPERTYPE OF(ONEOF(straight_edge, curved_edge)) ;
corner_points   : LIST [2:2] OF point ;
surfaces        : OPTIONAL SET [1:?] OF  surface ;
edge_shape      : edge_shape ;
bevels          : OPTIONAL LIST [1:?] OF bevel ;
position        : position ;
id              : id ;
END_ENTITY ;

ENTITY electrical_property ;
END_ENTITY ;

ENTITY element_division ;
END_ENTITY ;
```

```
ENTITY element_joint ;
ej_type          : joint_type ;
shape            : OPTIONAL shape2D ;
joint_material   : SET [1:?] OF material ;
size             : size ;
position         : position ;
id               : id ;
relations        : SET [0:?] OF relation_type ;
END_ENTITY ;

ENTITY even_layer ;
thickness : REAL ;
END_ENTITY ;

ENTITY external_wall_structure ;
facades          : LIST [1:?] OF facade ;
END_ENTITY ;

ENTITY facade
SUPERTYPE OF(precast_facade ANDOR cast_in_situ_facade);
position         : position ;
id               : id ;
shape            : shape ;
balconies        : OPTIONAL SET [1:?] OF balcony ;
channels         : OPTIONAL SET [1:?] OF channel ;
END_ENTITY ;

ENTITY fire_property ;
END_ENTITY ;

ENTITY frame ;
frame_depth             : REAL ;
module_dimensions       : LIST [1:?] OF REAL ;
position                : position ;
id                      : id ;
END_ENTITY ;

ENTITY functional_property ;
END_ENTITY ;

ENTITY grinding ;
END_ENTITY ;

ENTITY hard_burned_brick
SUBTYPE OF (surface_tile);
END_ENTITY ;

ENTITY hewing ;
END_ENTITY ;

ENTITY hidden_joint ;
END_ENTITY ;

ENTITY id ;
END_ENTITY ;

ENTITY joint ;
j_type           : STRING ;
shape            : OPTIONAL shape2D ;
materials        : SET [1:?] OF material ;
size             : size ;
position         : position ;
id               : id ;
END_ENTITY ;

ENTITY material ;
name                    : STRING ;
colour                  : OPTIONAL STRING ;
manufacturer            : OPTIONAL STRING ;
```

```
product_name            : OPTIONAL STRING ;
material_properties     : OPTIONAL SET [1:?] OF material_property ;
acoustic_properties     : OPTIONAL SET [1:?] OF acoustic_property ;
biological_properties   : OPTIONAL SET [1:?] OF biological_property ;
durability_properties   : OPTIONAL SET [1:?] OF durability_property ;
thermal_properties      : OPTIONAL SET [1:?] OF thermal_property ;
optical_properties      : OPTIONAL SET [1:?] OF optical_property ;
fire_properties         : OPTIONAL SET [1:?] OF fire_property ;
structural_properties   : OPTIONAL SET [1:?] OF structural_property ;
electrical_properties   : OPTIONAL SET [1:?] OF electrical_property ;
functional_properties   : OPTIONAL SET [1:?] OF functional_property ;
END_ENTITY ;

ENTITY material_property ;
END_ENTITY ;

ENTITY mathematical_layer ;
END_ENTITY ;

ENTITY mosaic_tile
SUBTYPE OF (surface_tile);
END_ENTITY ;

ENTITY mould ;
END_ENTITY ;

ENTITY natural_stone_tile
SUBTYPE OF (surface_tile);
END_ENTITY ;

ENTITY opening ;
edges                   : LIST [2:?] OF edge ;
doors                   : OPTIONAL SET [1:?] OF door ;
windows                 : OPTIONAL SET [1:?] OF window ;
other_equipments        : OPTIONAL SET [1:?] OF other_equipment ;
position                : position ;
id                      : id ;
DERIVE
o_area                  : REAL ;
END_ENTITY ;

ENTITY optical_property ;
END_ENTITY ;

ENTITY other_edge_shape ;
END_ENTITY ;

ENTITY other_equipment ;
END_ENTITY ;

ENTITY other_tile
SUBTYPE OF (surface_tile);
END_ENTITY ;

ENTITY painting ;
END_ENTITY ;

ENTITY parging ;
END_ENTITY ;

ENTITY point ;
END_ENTITY ;

ENTITY position ;
END_ENTITY ;

ENTITY posture_fixing ;
END_ENTITY ;

ENTITY precast_element ;
```

```
structural_layers               : LIST [1:?] OF structural_layer ;
element_type                    : STRING ;
fireclass                       : OPTIONAL STRING ;
manufacturing_series_number     : OPTIONAL STRING ;
element_series_number           : OPTIONAL STRING ;
position                        : position ;
id                              : id ;
relations                       : SET [0:?] OF relation_type ;
DERIVE
pe_length                       : REAL ;
pe_height                       : REAL ;
pe_area                         : REAL := pe_length*pe_height ;
END_ENTITY ;

ENTITY precast_facade
SUBTYPE OF (facade) ;
precast_elements : SET [1:?] OF precast_element ;
element_division : element_division ;
element_joints   : SET [1:?] OF element_joint ;
END_ENTITY ;

ENTITY relation_type ;
END_ENTITY ;

ENTITY reserved_space ;
END_ENTITY ;

ENTITY roller_application ;
END_ENTITY ;

ENTITY sand_blasting ;
END_ENTITY ;

ENTITY sanding ;
END_ENTITY ;

ENTITY shape ;
END_ENTITY ;

ENTITY shape2D ;
END_ENTITY ;

ENTITY size ;
END_ENTITY ;

ENTITY skewed_layer ;
END_ENTITY ;

ENTITY slab ;
END_ENTITY ;

ENTITY standard_edge_shape ;
END_ENTITY ;

ENTITY straight_edge
SUBTYPE OF (edge) ;
END_ENTITY ;

ENTITY structural_layer ;
material                : SET [1:?] OF material ;
position                : position ;
id                      : id ;
surfaces                : SET [1:?] OF surface ;
openings                : OPTIONAL SET [1:?] OF opening ;
reserved_spaces         : OPTIONAL SET [1:?] OF reserved_space ;
doors                   : OPTIONAL SET [1:?] OF door ;
windows                 : OPTIONAL SET [1:?] OF window ;
edges                   : LIST [2:?] OF edge ;
clamps                  : OPTIONAL SET [1:?] OF clamp ;
other_equipments        : OPTIONAL SET [1:?] OF other_equipment ;
```

```
type_of_layer            : layer_type ;
layer_name               : STRING ;
shape                    : shape2D ;
DERIVE
sl_length                : REAL ;
sl_height                : REAL ;
sl_area                  : REAL :=sl_length*sl_height;
END_ENTITY ;

ENTITY structural_property ;
END_ENTITY ;

ENTITY structural_system ;
basement                 : basement ;
external_wall_structures: SET [1:?] OF external_wall_structure ;
building_frame           : building_frame ;
END_ENTITY ;

ENTITY surface ;
shape                    : shape2D ;
colour                   : OPTIONAL STRING ;
coarseness               : OPTIONAL STRING ;
surface_kode             : OPTIONAL STRING ;
materials                : SET [1:?] OF material ;
surface_finishes         : OPTIONAL LIST [1:?] OF surface_finish ;
surface_pattern          : OPTIONAL SET [1:?] OF surface_pattern ;
part_surfaces            : OPTIONAL SET [1:?] OF surface ;
surface_tilings          : OPTIONAL SET [1:?] OF tiling ;
position                 : position ;
id                       : id ;
DERIVE
s_area                   : REAL ;
END_ENTITY ;

ENTITY surface_pattern ;
END_ENTITY ;

ENTITY surface_tile
SUPERTYPE OF (ONEOF(natural_stone_tile, hard_burned_brick, mosaic_tile, brick_tile,
other_tile)) ;
size             : size ;
colour           : STRING ;
manufacturer     : OPTIONAL STRING ;
product_name     : OPTIONAL STRING ;
quality_class    : OPTIONAL STRING ;
fixing_system    : fixing_method ;
position         : position ;
id               : id ;
END_ENTITY ;

ENTITY technical_system ;
relations        : SET [0:?] OF relation_type ;
END_ENTITY ;

ENTITY thermal_property ;
END_ENTITY ;

ENTITY tiling ;
joints           : SET [1:?] OF joint ;
bond             : bond ;
surface_tiles    : SET [1:?] OF surface_tile ;
position         : position ;
id               : id ;
END_ENTITY ;

ENTITY visible_joint ;
END_ENTITY ;

ENTITY window ;
w_type                   : STRING ;
```

```
frame                   : frame ;
window_ledge_plate      : OPTIONAL window_ledge_plate ;
glasses                 : LIST [1:?] OF window_glass ;
position                : position ;
id                      : id ;
DERIVE
number_of_glasses       : REAL ;
END_ENTITY ;

ENTITY window_glass ;
shape            : shape2D ;
materials        : SET [1:?] OF material ;
thickness        : REAL ;
position         : position ;
id               : id ;
END_ENTITY ;

ENTITY window_ledge_plate ;
END_ENTITY ;

TYPE
joint_type = SELECT (hidden_joint, visible_joint) ;
END_TYPE ;

TYPE edge_shape = SELECT(standard_edge_shape, other_edge_shape) ;
END_TYPE ;

TYPE fixing_method = SELECT (mould, posture_fixing) ;
END_TYPE ;

TYPE layer_type = SELECT (even_layer, skewed_layer, curved_layer, mathematical_layer) ;
END_TYPE ;

TYPE surface_finish = SELECT (
hewing, acid_treatment, brush_treatment, sand_blasting, grinding,
sanding, painting, cleansing, parging, roller_application) ;
END_TYPE ;


END_SCHEMA ;
```