PRODUCT MODELLING AT WORK

F.P. Tolman, P. Kuiper and G.T. Luiten
Delft University of Technology, TNO-Building Institute, PO Box 49 2600 AA Delft
The Netherlands. August 1990

Abstract

The paper presents and illustrates a methodology for the development and implementation of product models in the Building and Construction industries.

1 Introduction

In 1985 TNO's Building and Construction Institute in the Netherlands started a research group on Computer Integrated Construction. After a period of heavy discussions, mystifications and study, the main line of research became the development of concepts for product modelling of AEC products (bridges, viaducts, buildings, ...). Product modelling was seen as the application of conceptual modelling techniques [1] to the description of industrial products. A product model is defined as an information model of a product, describing the 'reality' of a product in its different life cycle stages.

The paper will present some of the practical results obtained so far, e.g. the development and implementation of product models for different application area's, and the exchange of product model data in a multi vendor environment. The paper starts with an overview of the methodology for developing product models currently in use and pays attention to our testbed product modeler ProMod and some of the STEP-translators available. Finally the paper will discuss the current fields of research and development.

2 Methodology

A large part of our efforts has been devoted to the development of a methodology for product modeling. One of the results of these efforts is the GARM (General AEC Reference Model) that currently is part of the ISO/STEP Draft proposal [1, 2]. Another result is a formal method to transform conceptually modelled relations between objects to the non-manifold topology scheme [3]. Before embarking on these subjects, we first have to concentrate on an other methodological aspect, e.g. the question of modeling languages.

2.1 Modeling languages

As you probably know the STEP standard will be modelled in Express. Express is an algebraic language that is processable by the computer. We follow the STEP approach and present our detailed models in Express. However the model development process is not very well supported by the Express language. Casual readers find it hard to understand Express models. They look to much like computer programs. We therefore adopted a graphical modeling language called NIAM (Nijssens Information Analysis Method) [4] that we use in the early modeling stages. NIAM is a conceptual modeling language that allows you to formulate binary relations between sets of objects. For instance the NIAM sentence:

![Diagram](image)

Figure 1 A many-to-many relation between sets of objects

Reads from left to right as: "AEC-products contain zero, one or more aspect systems" and from right to left: "Aspect systems are contained in zero, one or more AEC-products". NIAM allows you to make your sentences more precise by adding cardinality symbols. You can for instance add a 'All' symbol to express a fact that holds for each object in the set.
Now the sentence reads from left to right: "AEC-products contain one or more aspect systems". To express the fact 'not more than one', NIAM uses a horizontal bar above the matching relation box. For instance the sentence:

reads from left to right as: "AEC-products always contain one 220 Volt system". Beside these cardinality constraints NIAM also supports the subtype supertype construct and constraints between relations and between subtypes.

The left sentence, or construct says: "Buildings and Bridges are both AEC-products, but a Bridge is not a Building". The right construct says: "Bridges are made of concrete, or steel". With this vocabulary you will be able to read NIAM-models, especially when you forget about the constraints and only concentrate on the sets of objects and their relations.

2.2 GARM

The GARM describes a number of constructs that can be applied on a basic entity, called PDU (Product Definition Unit). Every object, or part of an object is a PDU. PDUs can be decomposed in other (smaller) PDUs. PDUs can be described in different life cycle stages of the product. A PDU in the As Required stage is called a Functional Unit (FU). A PDU in the As Designed stage is called a Technical Solution (TS). And a PDU in the construction stage is called a Built Unit. FU's have Required Characteristics. TS's have expected Characteristics. And Built Units have Realized Characteristics. Expected Characteristics should match the Required Characteristics and Realized Characteristics should match the Expected Characteristics. Other life cycle stages can be considered as well.

In this paper we will limit the discussion of the GARM to only one construct and show how this abstract GARM-construct can be used in the development of product models. The construct of our choice is the so called FU-TS-decomposition. This construct expresses the fact that a top-down design process is ruled by the deive-and-concer principle. Searching a Technical Solution (TS) for any set of requirements collected in a Functional Unit (FU) is done by braking the TS up into lower order FU's. E.g. designing an office building is done by dividing the problem in a number of smaller design problems: designing the spacial system, designing the structural system, etc. These lower order FU's may be related to each other. In NIAM these facts can be described as:
This model contains two supertype/subtype relations: FU's, TS's and Built Unit's are subtypes of PDU, and three specializations of Characteristics. The model also contains the following binary sentences:
- FU's can be satisfied by TS's/TS's can be used to satisfy FU's
- TS's can be decomposed in zero, one or more lower order FU's/etc.
- TS's correspond to Built Units/etc.
- FU's have Required Characteristics/etc.
- TS's have Expected Characteristics/etc.
- BU's have Realized Characteristics/etc.
- Expected Characteristics should match the Required Characteristics/etc.
- Realized Characteristics should match the Expected Characteristics/etc.
- PDU's may have relations with other PDU's

Because subtypes inherit everything of their supertype, FU's, TS's and Built Units may also have relations with other FU's, TS's and BU's. Relations between FU's are functional interfaces, relations between TS's are expected technical interfaces (particular technical solutions for the functionally defined interface requirements) and relations between Built Units are realized (measurable) technical interfaces. Note also that a subtype may have his own relations with other sets of objects, like: FU's have Required Characteristics and also that these other sets of objects can have relations, like: Expected Characteristics should match the Required Characteristics.

This model allows us to develop product models that suite the requirements of the Building and Construction industries. The division between Functional Unit (where the functional requirements are collected) and Technical Solution (where the actual characteristics are collected) reflects the usual thinking. FU's state the WHAT's and TS's one or more alternative HOW's. Building regulations for instance are collections of functional requirements (WHAT's) and libraries of building products collections of characteristics (HOW's). These two should be brought together. In the next section we will give an example how this can be worked out.

Demo Home

With the above construct as the core of the model we have developed product models for different kinds of objects, e.g. a model for Roads, Viaducts, Buildings and Steel structures. As an example we will present a model of a simplified private house, called the DemoHome, as shown below:
We will only concentrate on the first two life cycle stages and forget about the Built Unit. Also we will skip the characteristics from the model. On the first level of decomposition we choose to recognize the FU's: Internal Space, Facade, Roof and Foundation.

There are two types of relations between these FU's: (1) 'is enclosed by/encloses' and 'connects/is connected to'. See figure 9 next page for the resulting NIAM model. For clarity the lower order FU's with their relations are concentrated in a shaded area. As an example we will try to find a TS for the FU Roof. One type of alternative TS's are called Pitched Roofs. We choose the Gable Roof, which may decompose in: Roof Structure (tiles, etc.), Frame Structure, Ridge and Chimney. Between these FU's there are two types of relations: 'supports/is supported by' and 'is located on/locates'.

Note that the TS Gable Roof has many alternatives, like Saddle Roof, etc. Gable Roof decomposed in a number of lower order FU's that may be satisfied by different TS's again. The decomposition of Roof Structure for instance may be will show things like: tiles, asphalt, insulation material, etc.
At this stage it is also easy to understand the entities 'Required Characteristic' and 'Expected Characteristic' in the model of figure 9. Typically one would like to attach some thermal insulation requirement to Roof Structure and while searching for a TS try to find a solution which matches this requirement.

4 From Conceptual Model to Geometric Model

The step from this conceptual model to a geometric representation requires on each level of decomposition: (a) the choice of a Domain representation (a 0, 1, 2 or 3 dimensional element) for each FU and (b) the classification of the adjacency relations between the FU's that have to be represented into: Domain relations ('encloses/is enclosed by') and Boundary relations ('bounds/is bounded by') [2]. Different applications of the model may use different choices of Domain entities and derive an other non-manifold topology model from the same object. As these topology models can support different geometric representations, a variety of shapes can be derived from the same conceptual model.

Figure 9 Informal NIAM model with two levels of decomposition
5 Implementation

Implementation of these models is done in two ways. One way is to translate the NIAM-model into Express. That can either be done manually or automatic [5]. The Express model can then serve as a conceptual schema that can (a) be mapped onto the local database system (internal schema) and (b) support mappings to external application programs (external schema).

Another possibility is to use a real product modeller, like the prototype system ProMod we have been developing at TNO. ProMod supports all the GARM entities and constructs and allows a top-down design process exactly following the model hierarchy. ProMod also allows a bottom-up design strategy, as it allows you to develop and use libraries of TS's.

To illustrate the flexibility of the methodology presented we show two geometric representations of the bathroom of our DemoHome. Figure 10 shows a representation on a certain level of decomposition, where for instance the bath tube and the window just became visible:

![Figure 10 Bathroom of DemoHome](image)

The figure shows a picture of an instance of the model of the DemoHome with a bathe tube located somewhere on the floor, a side wall with window, water pipes and tap. The same instance on a lower decomposition level may look like:

![Figure 11 Bathroom of DemoHome with more detail](image)
6 Other Applications

To show that the model really is a product model we have added also some non geometrical information. One application that can access the product model is a program for energy calculations. Figure 12 shows:

![Energy Calculation Table](image)

**Figure 12 Result of a energy calculation of an instance of the DemoHome**

Another application produces a bill of materials, as shown in figure 13:

![Bill of Materials Table](image)

**Figure 13 Bill of material of an instance of the DemoHome**

Of course there are many other application programs that like to communicate with the product model.
7 Further Research

The methodology presented is currently mainly restricted to the first product life cycle stages. Research into other life cycle stages is under way [6, 7, 8]. An other line of research is the development of product type models.

A product type model is an information model of a class of products. Product type models include general knowledge of the class of products. General knowledge available in different product life cycle stages can be brought into perspective and integrated in a concise product type model.

8 Conclusions

With the methodology presented in this paper it is possible to develop product models for the type of products encountered in the Building and Construction industries. Though we do not claim that the methodology is the final answer, a number of applications have been successful. These successes motivated us submit the GARM to the ISO-STEP community, because real benefits of product modeling can only be expected from an international standard.

References