INTERNATIONAL DESIGN CONFERENCE - DESIGN 2002
Dubrovnik, May 14 - 17, 2002.

MODELLING THE RELATION NETWORK IN THE INTEGRATED PRODUCT AND DESIGN PROCESS MODEL

N. Pavkovic, N. Bojcetic and D. Marjanovic

Keywords: relations, object-oriented, design process model, product model

1. Introduction

A lot of research efforts in the area of computer-supported design are directed towards development integrated software frameworks (often named designers workbench), which should include and integrate product and design process model. These software frameworks should be the platforms with flexible structure, providing the tools for managing the design process and to integrate the usage of various computer-based design support tools. None of the design theories which are in use today, like e.g. descriptive, prescriptive, general, and axiomatic design theories, is alone capable, to describe and to formalise the whole design process and to make possible a complete computer support of this process [Vajna 1997]. Keeping this in mind, the presented research aims to develop a kernel for the framework that could enable the use of different partial models in an integrated manner. The most important fact in such approach is that design process should no longer be viewed as a static institutionalised structure, but rather as a dynamic network that is constructed in real-time as design goals, needs, priorities, and resources evolve [Wallace 1999].
Engineering data structures are much more complex than those in common business database applications. The associations between structures are numerous, and one data structure can have many different roles. Moreover, the data structures dynamically change during the design process. Some of the structures are being created "from scratch" during the design process in a manner that is hard to predict. All above facts make the task of developing the integrated computer model of product and product design process very complex and difficult. The aim of this paper is to consider criteria for classifying the relations in engineering data structures. Some methods of modelling the network of relationships in the object-oriented environment will be proposed.
According to literature [Raimund, 1992], the object-oriented approach offers better modelling capabilities, especially for complex structures, offering simpler maintenance and upgrades through exploitation period. This research should be an experiment to see if this presumption holds. There will be no system that will cover all aspects of design process, but what we need is the environment that can be built up and adjusted to the particular needs. The authors' opinion is that solving the problems of modelling the complex and dynamic network of relations between objects in product and design process model is one of the crucial steps in developing integrated design environments.
In the proposed approach the network of relations will be considered as established between objects which are instances of classes in the object-oriented design process model. The structure and the entities of the object-oriented design process model are described in [Pavkovic 2001]. Every occurrence, kind of action, set of information, relation and other real-world “things” from the domain of the design process, are attempted to be modelled as objects. In the prototype object model, the

considerations are limited to recognition of basic entities, their attributes and relationships, while the necessary operations (methods) are only denoted. The system is targeted to support the classes of adaptive and variant design. The design process is here viewed as a sequence of transitions from an initial state of data, constraints and goals to a final state, the description of the artefact being designed. These transitions are allocated to individual participants or teams, and individual or integrated computational tools or models.

2. Classification of relations

It should be noticed that criteria for relation classification proposed in this chapter should be viewed just as the starting point and the proposal for further discussion in developing the kernel of the software framework under development.

2.1 Number of the relata comprehended within relation

In set theory, [Lipschutz 1964] the relation is defined as a subset of Cartesian product of two sets:

$$R \subset AxB = \{ (a,b) \in R \ | \ a \in A, \ \ b \in B \ \}$$

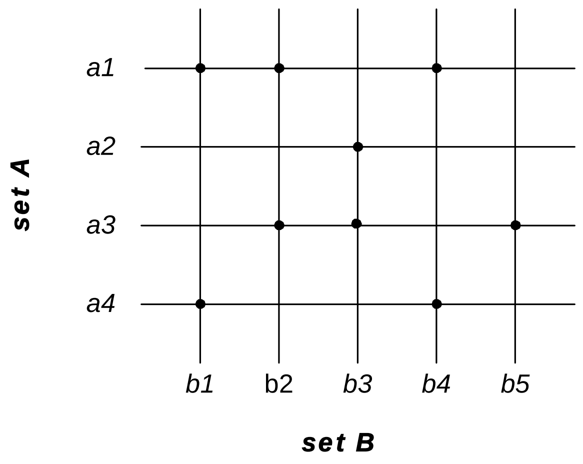


Figure 1. Graphical representation of relation

Relation between two sets is a ***binary*** relation (2 relata). In general, if there are more than 2 relata we have ***n-ary*** relation defined as:

An ***n***-ary relation on sets $A_1, ..., A_n$ is a set of ordered ***n***-tuples <$a_1, ..., a_n$> where a_i is an element of A_i for all ***i, 1£ i £ n*** . Thus an ***n***-ary relation on sets $A_1, ..., A_n$ is a subset of Cartesian product A_1 ´ **…** ´ A_n .

An **ordered *n*-tuple** is a set of ***n*** objects with an order associated with them. If ***n*** objects are represented by $x_1, x_2, ..., x_n$, then we write the ordered ***n***-tuple as <$x_1, x_2, ..., x_n$> **.**

Binary relations can be represented with connection matrices:

Let R be a relation from A = $\{a_1, a_2, \dots a_m\}$ to B = $\{b_1, b_2, \dots b_n\}$.

An *m* x *n* connection matrix M for R is defined by M_{ij} = 1 if $< a_i, b_j > \in R$
= 0 otherwise

While the binary relations are relatively easy for representing and manipulating, a network of n-ary relations that dynamically change during the design process can be very difficult to model and manipulate in computer environment. The ultimate goal of the presented research is to find if the object-oriented environment really offers better modelling capabilities for such problems than traditional database technologies. This issue will be further discussed in following chapters.

2.2 Relation semantics in the context of the general object-oriented information modelling

When analysing different information models one could find some variations in classification of relations. In the proposed model we will rely on classification according to the authors of Unified Modelling Language: *dependency, generalization, association and realization*, [Booch et al., 1999].

Dependencies represent using relationships among classes that states that a change in specification of one thing may affect another thing that uses it, but not necessarily the reverse.
Generalizations connect generalized classes to more specialized ones in what is known as subclass/superclass or child/parent relationships.
Associations are structural relationships among instances. Association specifies that objects of one class are connected to objects of another. Associations can be binary or n-ary.
Realization is a semantic relationship in which one classifier specifies a contract that another classifier guarantees to carry out. Realization is used in the context of interfaces and in the context of collaborations.
Further discussion in this paper will be focused mainly on modelling binary and n-ary associations.

2.3 Relation semantics in the context of the product and design process model

The presented research was up to now focused on generalization and association relations. A prototype of class hierarchy is established, but a lot of work has to be done to refine it, and to develop each subclass. More attention was paid to modelling associations. So far, in the context of design process model, the following semantics of associations have been explored:

- Dependency (algebraic (between design parameters) or information (between design tasks))
- Affiliation (belonging) (product documentation - product components, design parameters - design tasks, etc.)
- Sequence (of execution - between design tasks, design process steps)
- Responsibility (designers, design tasks, design documentation)
- Hierarchy (components - subassembiles - assemblies)
- Constraints and conditions (parameters - requirements)

3. Modelling the network of relations

3.1 Elementary classes and relations of the proposed product and design process model

The proposed model is founded on four loosely coupled sets of entities (classes): elementary classes, classes that models relations, classes that models the design process and the components of system architecture. We will further discuss elementary classes and relations modelled as classes.

<u>Elementary classes are:</u>

> design parameter, design requirement, product documentation object, product physical component, product functional component, design process action, design task, designer, software tool interface

In the presented approach, the relations between objects are also viewed as objects. An instance of such class represents binary or n-ary relation between instances of elementary classes.

<u>Relations between objects (instances of elementary classes) or their attributes, modelled as classes:</u>

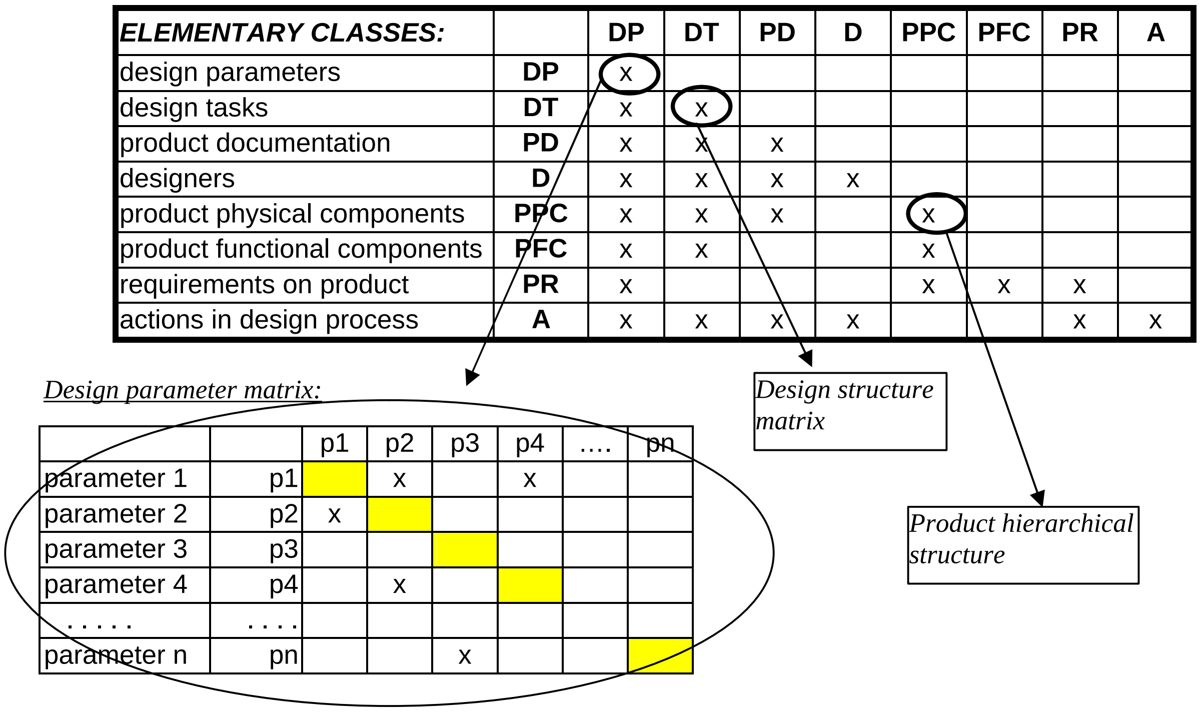
> *dependency relations* between parameters, and between design tasks, *membership (containment) relations* between objects of different classes, *expressions* – relations between object attributes, *design constraints* – relations between design parameters and requirements, *design decision rules* – relations between conditions and design process actions

More complex entities that model the design process will not be considered in detail in this paper. More detailed description of their concept is given in [Pavkovic 2001].

3.2 Modelling relations between sets of instances of elementary classes

The idea further developed in the proposed model is to use the matrix form for representing sets of relations between objects of similar classes, as also between objects of different classes.
In general, a relation can be established between every ordered pair of sets of instances of elementary classes. Upper table in figure 2 shows such set of binary relations. The question is which of all possible relations could be useful to model and manipulate in the proposed system? Some of these combinations were already the targets of thorough research efforts, like e.g. "Design structure matrix" - a relation of information dependency between design tasks. The marks have been put in the cells of table in figure 2 for relations whose exploration and modelling could be useful. Here it must be emphasized that the mark in the cell tells nothing about the semantics of the relation, which should be further explored if the relation is found to be useful.
The table in figure 2 gives us a compact survey of binary relations between large sets of objects, and can be considered as a good starting point for further discussions.

Few examples of relations from table are expanded in more detail in figure 2.



ELEMENTARY CLASSES:		**DP**	**DT**	**PD**	**D**	**PPC**	**PFC**	**PR**	**A**
design parameters	**DP**	x							
design tasks	**DT**	x	x						
product documentation	**PD**	x	x	x					
designers	**D**	x	x	x	x				
product physical components	**PPC**	x	x	x		x			
product functional components	**PFC**	x	x			x			
requirements on product	**PR**	x				x	x	x	
actions in design process	**A**	x	x	x	x			x	x

Design parameter matrix:
Design structure matrix
Product hierarchical structure

		p1	p2	p3	p4	….	pn
parameter 1	p1	🟨	x		x		
parameter 2	p2	x	🟨				
parameter 3	p3			🟨			
parameter 4	p4		x		🟨		
. . . . .	. . . .						
parameter n	pn			x			🟨

Figure 2. A survey of relations between elementary classes

The semantics of "Design parameter matrix" can be twofold - algebraic dependencies between design parameters, and similarly as in Design structure matrix - the information dependencies.

3.3 The prototype implementation in object database

In the current phase of research, some of the relations shown in figure 2 are modelled in the environment of object database. They are implemented as classes whose instances represent the rows of the relation connection matrix. Let us consider a connection matrix for relation from set A to set B (figure 1). Both sets are considered as instances of classes in the proposed model. Such a matrix can be reduced to three columns (table 1). The second column in such view contains a subset of elements of set B that are related to particular element of set A. The third column records the relation semantics, if there is a common one for that particular matrix row. Now we can model the matrix as a class (set of instances) in which every instance represents one particular row. Every instance of matrix class has one pointer to the instance of class "A", and a set of pointers to related instances of class "B". The proposed model gives a more compact representation of relation, instead an approach where each instance of class "A" has a set of pointers to related instances of class "B". In the second mentioned approach, every class should have several sets of pointers (the one for each relation in which that class participate). Moreover, having a class that represents the relation, the procedures of searching, retrieving and updating are easier to implement and maintain.

Table 1. Reducing a connection matrix to three columns

	b_1	b_2	b_3	b_4	b_5	b_6	….	b_n
a_1				x				
a_2			x			x		
a_3		x			x			x
….								
a_n					x	x		

®

	Pointers to related instances of class B	Relation semantics
a_1	b_4	$a_1 = (b_4)^2$
a_2	b_3 b_6	$a_2 = b_3 + b_6$
a_3	b_2 b_5 b_n	…..
….		
a_n	b_5 b_6	

The left part of figure 3 is an UML notation for the example of the class that models a set of algebraic dependencies between design parameters (a relation between instances of the same class). The "parameter dependency matrix" class has two associations with "design parameter" class. The "left" line is "one to one" association modelled with one pointer, while the "right" line is "one to many" association modelled with set of pointers.

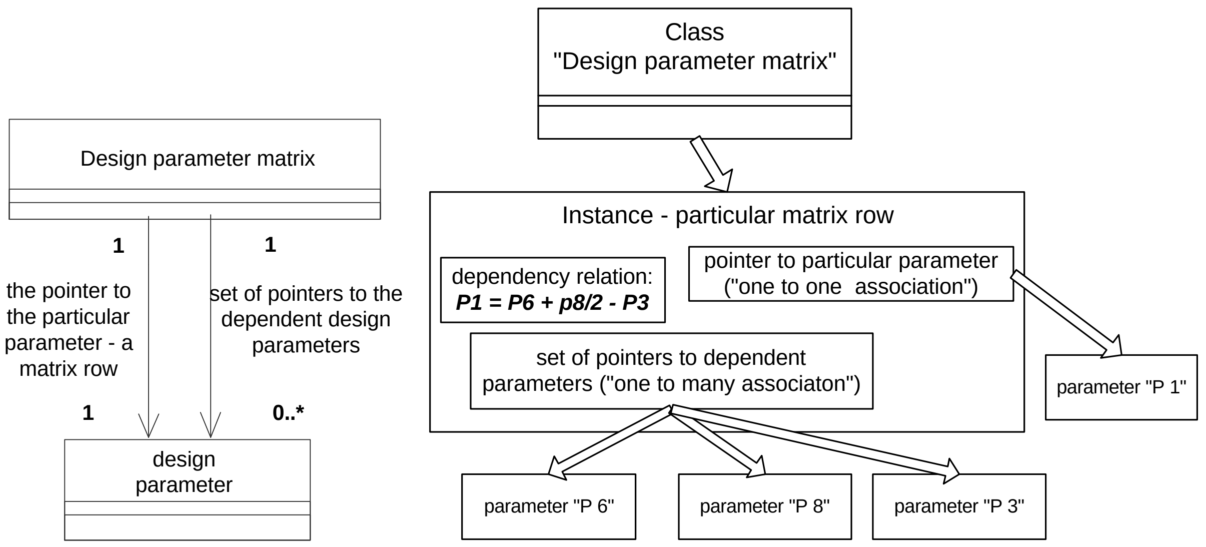

Design parameter matrix
1
1
the pointer to the particular parameter - a matrix row
set of pointers to the dependent design parameters
1
0..*
design parameter
Class "Design parameter matrix"
Instance - particular matrix row
dependency relation: P1 = P6 + p8/2 - P3
pointer to particular parameter ("one to one association")
set of pointers to dependent parameters ("one to many associaton")
parameter "P 1"
parameter "P 6"
parameter "P 8"
parameter "P 3"

Figure 3. "Design parameter matrix" class

The right part of figure 4 is an example of one instance of "design parameter matrix class. The scheme shows pointers to parameter objects that are related. The very similar model is developed for "Design structure matrix" class in which every instance has a pointer to one particular design task and a set of pointers to other design tasks that must contribute information to that task for proper completion of the design.

The prototypes of the proposed classes are realised in C++, in an object database environment. The object-oriented database maps objects that has been created in C++ or Java onto objects in the database. This kind of direct mapping enables the database schema to be generated automatically. Even in the case of highly complex object models, there is no difficulty writing them to the database. The database "knows" what an object is and also recognizes the relationships (references, pointers) between objects. These are simply stored together with the objects themselves, and are therefore reproduced easily at any time the data is required.

In the current research phase other associations between classes are under development, as well as other classes that models relationships (see figure 2). Future research efforts will be directed towards exploring modelling of n-ary relations between various classes. A common representation of n-ary relation is a table in a relational database. The structure of tables and relationships in relational databases is relatively static, and not easy to update – usually requires a lot of programmers work. It is expected that modelling of n-ary relations in object environment will give more flexible structures. In the design process many of the data structures and relations are being created for the first time, in the ways that are often difficult to predict.

4. Conclusion

The presented research tries to develop a kernel of object-oriented product development process model. According to the literature, in the design of object-oriented systems, the most interesting part are not the objects themselves, but the relationships among objects, and these relationships constitute a large proportion of the semantics of most systems. Therefore in the first phase of the research, the emphasis has been put on modelling the relations. The results are promising, there are many advantages comparing to traditional relational database technologies. Some of the proposed ideas offer more flexible data structures that are easier to create, update and maintain. Also, the object model is closer to the reality, therefore it should be easier to understand and learn for users.

The research projects in the area of computer-based modelling of product development process were mainly focused on issues that are islands in the area (sometimes isolated). The attempts of development of general integrated frameworks were numerous, but it seems that there are still no good results, i.e. widely acceptable and usable systems. It is obvious that such a task is very huge and complex. It is difficult to believe that isolated research teams could come to widely acceptable solution. The presented work is also only a small portion of the job that has to be done – just "the top of the iceberg". One of the basic conditions for the success of such projects is the development (and the existence) of widely accepted product development process ontology.

References

Booch G., Rumbaugh J., Jacobson I., "Unified Modeling Language User Guide", Reading, Addison Wesley, 1999.

Lipschutz S., "Theory and problems of set theory and related topics", McGraw-Hill, 1964.

Pavkovic N., "Object-oriented approach to design process modelling", Ph.D. thesis, University of Zagreb, Faculty of mech. engineering & naval arch., 2000. (in Croatian)

Pavkovic N., Marjanovic D., "Considering an object-oriented approach to the design process planning", International Journal of Technology Management, Vol. 21, No. 3/4, 2001.

Pavkovic N., Marjanovic D., "Structuring a Designers Workbench with Object-Oriented Design Plans", Proceedings of ICED 99, Vol. 3, pp. 1401-1406, WDK, 1999.

Pavkovic N., Marjanovic D., Dekovic D., "Object-Oriented Modelling of the Design Process", Proceedings of ICED 2001, Vol. Design research–Theories, methodologies and Product Modelling, pp. 275-282, WDK, 2001.

Raimund K. E., "Object-Oriented Programmnig with C++", Boston, AP Professional, 1992.

Vajna S., Wegner B., "Optimization of the Product Development Process with Evolution Algorithms and Simultaneous Enginerring", Proceedings of ICED 97, Vol. 3, pp. 3/67-3/70, WDK, Heurista, 1997.

Wallace D. R., Abrahamson S. M., Borland N. P., “Design process Elicitation Through the Evaluation of Integrated Model Structures”, Proceedings of DETC 99, ASME Design Engineering Technical Conf., 1999.

Neven Pavkovic, PhD
University of Zagreb, Faculty of Mechanical Engineering & Naval Architecture, Design Department,
Ivana Lucica 5, 10000 Zagreb, CROATIA
Tel: + 385 1 6168 545
Fax:+385 1 6156 940
Email: neven.pavkovic@fsb.hr