Methodologies and Tools for Co-operative Interaction in the Design Process

P. Bottoni
Dip. Scienze dell'Informazione, Università di Roma "La Sapienza"
Via Salaria 113, 00198 ROMA, Italy
tel. +39-6-49918428, fax +39-6-8541842, e-mail: bottoni@dsi.uniroma1.it

M. F. Costabile
Dip. Informatica, Università di Bari, Via Orabona 4, 70125 BARI, Italy
tel. +39-80-5443300, fax 39-80-5443300, e-mail: costabile@di.uniba.it

F. Mandorli
Dip. Meccanica, Università di Ancona
Via Brecce Bianche, I-60131 ANCONA, Italy
tel. +39-71-2204969, fax +39-71-220480, e-mail: ferro@ied.unipr.it

P. Mussio
Dip. Elettronica per l'Automazione, Università di Brescia
Via Branze 38, 25123 BRESCIA, Italy
tel. +39-30-3715450, fax +39-30-380014, e-mail: mussio@bsing.ing.unibs.it

F. Paternò
CNUCE - CNR, Via S. Maria 36, 56100 PISA, Italy
tel. +39-50-593289, fax +39-50-904052, e-mail: f.paterno@cnuce.cnr.it

R. Pizzicannella
IASI - CNR, Viale Manzoni 30, 00185 ROMA, Italy
tel. +39-6-7716433, fax +39-6-7716461, e-mail: pizzican@iasi.rm.cnr.it

G. Santucci
Dip. Informatica e Sistemistica, Università di Roma "La Sapienza"
Via Salaria 113, 00198 ROMA, Italy
tel. +39-6-49918484, fax +39-6-49918331, e-mail: santucci@dis.uniroma1.it

Abstract

Product and process technology are rapidly evolving, and competition is becoming more and more globally based. The increased competition has forced the market to move from mass production to customer-driven production: customers are placing an increasing emphasis on quality and reliability, but at the same time looking for good value. Providing environments which support the concurrent engineering paradigm and the co-operative interaction among the different actors involved in the design process can help manufacturers to reduce design cycle time and improve product value. The aim of our work is to investigate the use of currently available technology for co-operative interaction in the design process and suggest new solutions to improve the application of such a paradigm to the mechanical engineering field.

Keywords

Co-operative environment, visual interaction, interface customisation, design process

1 INTRODUCTION

Increased competition, due to the development of a global market, has forced the industry to move from mass production to customer-driven production. Customer-driven production means that an increased emphasis is placed on quality and reliability, but at the same time on good value and high product flexibility.

Into this dynamic and challenging environment, time to market is crucial for world class manufacturing. Design cycle time can be reduced and product value can be improved by applying the concurrent engineering paradigm and promoting co-operative interaction among the different actors involved in the design process.

Concurrent engineering is based on integrating the design of products with manufacturing and support processes in order to assure a more efficient approach to the manufacturing of the product. Additional considerations, that also need to be integrated with the design of the product, include test and inspection processes, product service and support, logistic and human factors, environment and safety considerations, maintenance, documentation and disposal. In other words, the design of the product needs to be integrated with all aspects of its life cycle. Moreover, a key aspect to successfully obtaining useful and usable products is a full understanding of customer needs. Marketing and program management functions need to be considered during product development and co-operation with the customer is needed to obtain proper feedback.

Technology can play a significant role in this scenario: it can provide a common view of the product; design information can be captured in a manner that more effectively drives downstream manufacturing process; cycle times can be reduced and design enhanced through computer-based analysis and simulation; product design information can more readily create process design data and drive downstream production processes.

Nevertheless, most technologies dedicated to supporting the previously mentioned activities (CAD, CAM, CAPP, PDM) have been developed independently from each other. As a result, we have nowadays islands of automation that are able to cover specific phases of the design/production process, but a real integration among them, that is a basis for the application of concurrent engineering concepts in a co-operative environment, is still missing.

In order to go deep into this research field, a two year co-ordinated project named (CO)^{2}DESIGN (Methodologies and tools for CO-operative interaction in COncurrent engineering) was started by the Italian National Council of Research (CNR) in 1996.

The aim of the project was to investigate the use of currently available technology for co-operative interaction in the design process and suggest new solutions to improve the application of such a paradigm into the mechanical engineering field. Researchers with different expertise have participated to the project and a particular attention has been paid to the definition of a new reference architecture that will be able to integrate different methodologies into a homogeneous environment for co-operative interaction.

A gear pump has been selected as a test case in order to verify developed methodologies and to demonstrate the applicability of our proposed approach. For integrated knowledge handling, computation and interaction, an implemented prototype environment developed in accordance with the proposed reference architecture was used.

2 MOTIVATION, OBJECTIVES, AND APPROACH

2.1 Motivation for co-operative environment

In the current market situation, increased complexity of products and increased competition are driving the need for reduction or elimination of barriers of time and place, better co-ordination of activities, and acceleration of decision-making processes. Multi-user applications provide the right support for this purpose.

However, traditional applications are developed using single-user technologies. As a result, multi-user aspects, such as co-ordination, co-operation and communication, are not supported during the product development nor are there development methodologies that can be practically applied to the development of multi-user applications. There is a lack of systematic task-based approaches for co-operative applications (Paternò, 1997). Current approaches to the design of co-operative environments have mainly two types of limitations. Either they provide low-level toolkits for CSCW (Computer Supported Co-operative Work), which require people with high skill of expertise to use them effectively, or they provide informal approaches which are useful to understand the requirements, but with insufficient support for systematic design.

The support of work in large organisations has so far been based on the following assumption: the total amount of work in the organisation can be divided in atomic tasks which only have input and output relationships with each other; during task execution there is no interaction with other tasks. Technical limitations enforced this unrealistic view on work upon the developers of systems. But time is vaporising these technical constraints.

Some technological innovation in particular contributes to this development:

- Network infrastructure able of supporting CSCW is now available (Internet).
- The improved price/performance of both CSCW hardware and software has made it more available to a larger population.
- Well known vendors such as Microsoft, Lotus, IBM and Digital Equipment Corporation (DEC) are promoting CSCW, thereby increasing awareness in the marketplace.

In $(CO)^2$DESIGN we studied how to allowing more than one user to work on the same task and data, and to enable developers to deliver products that are significantly closer to the current needs in industry.

During the project, the problem of analysing and describing the activities to be performed in co-operative applications for mechanical engineering products has been investigated (Van der Veer, 1996). The purpose was to identify and refine requirements for obtaining co-operative environments which better support the tasks of the classes of identified users. Some of the aspects are already known like communication, co-ordination, co-operation (Calvary, 1997), and awareness (Dourish, 1992). However these aspects still need to be further investigated and precisely defined for the class of application considered in the paper.

In this paper we focus attention on aspects related to the definition of appropriated user interfaces customised on the basis of the specific user class, and to the integration, within a homogeneous framework, of different tools for knowledge representation and manipulation.

2.2 Proposed Approach

In order to improve the effectiveness of the communication among the different types of users, different interfaces will be provided which will be adaptable to the task to perform and to the user knowledge of the application domain.

More specifically, two types of interfaces will be defined, one for the classes of users participating in the design of a product (design phase), the other for customers and salespersons assisting customers during the configuration of a specific product (customisation phase).

The approach we propose is characterised by two aspects: the use of specialised visual languages both in the design and in the customisation phases (Mandorli 1995); and the use of knowledge based technology for the representation of product data (Assogna 1995, Missikoff 1995).

Moreover, we allow the use of the visual components of the specification as a support for the actions during the customisation phase. In this sense, the system is based on the identification of: a) the different typologies of users involved, both in the definition of the product model and in the construction of a specific product instance; b) the different visual languages involved in the needed interactions; c) the relations among the visual languages to be used.

In particular, we see the definition of the interactive visual language for customisation as resulting from the definition of both the product model and the available user actions. To this end, components of the product model, (e.g. in case of a pump model: gears, bolts, shafts, etc.), are associated with (possibly several) pictorial counterparts. Hence, for example a shaft component in the model of a pump can be visualised via an icon, via a three-dimensional solid model, or via an item in a menu. The choice of the preferred type of visualisation can be performed by the designer or by the marketing manager, according to the complexity of the product model and to the user model, or can be left, within predefined limitations, to the user who can customise the interface and choose the preferred visualisation.

From a theoretical point of view, this approach is grounded in a recently proposed theory of visual languages (Bottoni, 1996, 1997a).

3 ARCHITECTURE AND MODULES DESCRIPTION

3.1 Reference Architecture

Co-operative interaction in the design process needs a complete representation of the product as well as the interaction models. This implies the representation of geometrical and topological information as well as of technological, functional and marketing information. A key aspect of the project is therefor the definition of a suitable knowledge base, containing different models, able to represent different knowledge types, integrated in a homogeneous environment. Another relevant aspect has been an understanding of the roles played by the different users, the different tasks performed by them and how they interact each other.

A significant effort has been put in the definition of appropriate interfaces that will allow different users to add/extract/modify/configure the models stored into the knowledge base. During such an activity, different types of users, performing tasks into two different environments, have been identified: those participating in the product design, viz. the designer engineer, the production engineer, and the marketing engineer, and those performing configurations, viz. the salespersons together with customers themselves. The former users will interact with a suitable environment to feed the knowledge base in order to define both the product model and all the options that are provided to customers in order to customise the product to their needs. The salespersons and customers will use a suitable environment to

have a driven access to the product model in order to be able to investigate different valid product configurations.
The proposed reference architecture and the role of the two identified interfaces with respect to the knowledge base are shown in Figure 1.

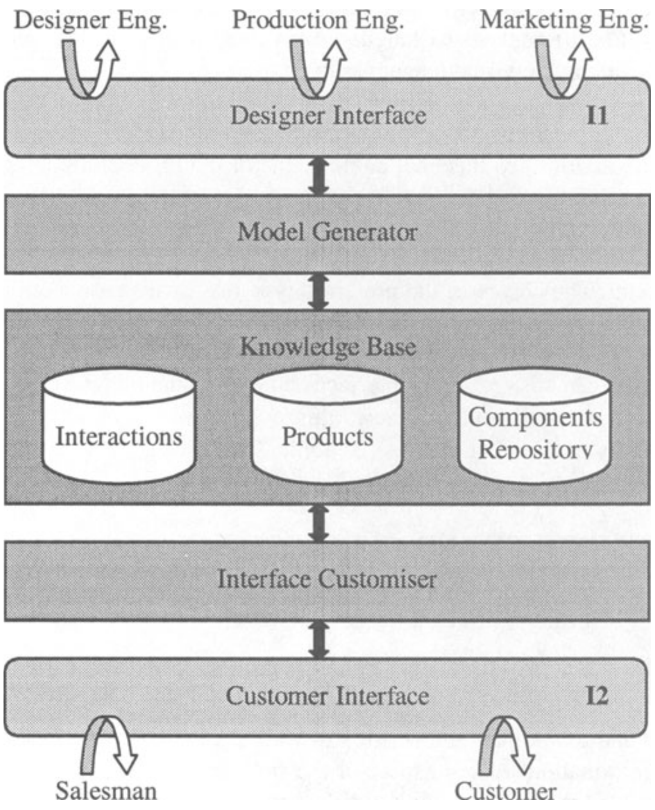


Figure 1 - The Proposed Reference Architecture.

In the following we describe in more detail the different modules of the proposed architecture.

3.2 User Interfaces

In the proposed architecture, users interact with the knowledge base through two separated interfaces: the Designer Interface and the Customer Interface.

The Designer Interface is used by the design engineer, the production engineer and the marketing manager to define product and action models, based on the components present in the Component Repository. A typical actions performed by the designer engineer through this interface is the definition of the structure of a model by listing its components and their relations, usually in the form of a tree whose branches are associated with constraints. The production engineer will place additional constraints deriving from the productive capacity of the factory, and the marketing manager will define the strategies of presentation to the customer, together with constraints in fixing the final price to the customers.

The Customer Interface supports the dialogue between a sales representative and a customer who try, at the customer site, to define a specific instance of a model, based on the user requirements. Such an interface, though sharing the same style with the Designer Interface, is not predefined but is generated by the sales representative, so as to enforce the marketing strategies suitable for the specific customer. In this interface, the two communicants have access to the product model, through the chosen visualisations, and can configure a specific model instance, by setting properties of the components. If necessary, they can also change the forms of visualisations, choosing from the available ones.

From the theoretical point of view, the interfaces for both design and customisation phases adopt a uniform style in which interaction occurs through a set of visual sentences, i.e. a visual language. A visual sentence is a triple <i,d,<int,mat>> where i is an image, d is a set of attributed symbols called description, int is a function mapping the structures of i into the attributed symbols describing them, and mat a function mapping the attributed symbols in d into the structures in i. These visual sentences are organised as pages, where each page allows the execution of logically related activities. The interaction is based on direct manipulation of entities representing model components at different levels of complexity.

3.3 Model Generator

During the design phase, the users interact with the system through the Designer Interface. Actions performed by the users are captured by the interface to produce sentences of the interaction visual language (note that only sequences of actions describing a sentence belonging to the visual language are accepted by the interface). The role of the Model Generator is to map these sentences into rules and data structures defining the product model as well as the action model.

Different model structures and different developing tools can be used to define and maintain the knowledge base; as a consequence, from the implementation point of view, the Model Generator can be seen as a set of translators from visual sentences to sentences of the appropriated knowledge base developing language.

3.4 Interface Customiser

The Interface Customiser integrates the information contained in the product model and in the action model to generate a specialised version of the customer interface, according to rules relating the customer profile to the set of actions which are left available to the user. Moreover, it dynamically generates and updates the representations and the descriptions of the components, as the customer requires the inclusion of specific components or changes their properties. Hence, the Interface Customiser remains active during the interaction with the customer, offering the possibility of further specialisation. In particular, the user is allowed to modify some aspects of the layout, such as window position, or to select some preferred form of visualisation from those available for the interaction with the specification of components.

3.5 Knowledge Base

We consider the Knowledge Base as a repository in which to store the different types of knowledge needed to describe both product and interaction models.

The product model stores information describing geometrical as well as functional and technological aspects of the product. In our activity we investigated the product modeling capabilities of TQL++ (Missikoff, 1995). TQL++ is an Object-Oriented modeling language that provides semantic integrity constraints, particularly suited for modeling constraints that cannot be expressed with traditional O-O languages (Formica, 1995).

Many research works on product model representation have been carried out in the past years, and a large bibliography is available about this topic (Shah 1994, Mandorli 1993).

The interaction model stores information regarding the types of actions allowed in the customer interface and the legal composition of these actions in sequences defining correct interactions. This model is complementary to the product model, in the sense that components of the product can be associated to visualisations on which different actions can be performed (an icon representing a component can support actions of selection, of addition to or removal from a model, of querying on the values of some attributes or of setting some of these values). Moreover, the action model contains rules which states how different actions can be composed (optional components can be available for selection only for particular configurations of the product model instance).

A correct model can be formally specified in terms of vCARW, an extension to the visual case of the conditional attributed rewriting systems exploited to build linear descriptions of images (Mussio, 1988, Mandorli, 1995). Given a vCARW and the definition of the supported actions, an automaton can be defined such that correct sequences of actions are legal paths in these automatons, whereas incorrect actions can be trapped as transitions that would lead into an error state, so that the appropriate messages are issued and recovery actions taken (Bottoni, 1997a). Note that in this way no sentence can be built which would later require deleting some created items to produce a correct visual sentence. Rather, any visual sentence produced by following some path in the automaton can be transformed into a correct one, i.e. one in the desired visual language, by actions which can only add new items or refine the properties of existing ones (Bottomi, 1996).

4 IMPLEMENTATION AND TEST ENVIRONMENT

4.1 The implemented prototype

In order to verify developed methodologies and to demonstrate the applicability of our approach, prototype versions of different modules that are part of the proposed architecture have been developed and tested on a practical example.

The interfaces have been implemented by CVE, a specific tool for co-operative interaction developed by the units of Roma and Brescia (Bianchi, 1993). CVE organises the interactive system in three layers, called execution, observation and surface layer. Each layer is implemented by a network of objects. At the bottom layer, a network of *executor agents* performs the computational activities relative to the task. At the middle layer, *observer agents* capture and organise the executor data to be reported to users according to the chosen visual language. The top layer is a network of *surface agents*, responsible for mapping the results of the activity of one observer in well-determined positions on the screen.

In the design phase, executors build a model in the form of a graph of components with edges labelled by constraints, while in the customisation phase an executor interprets this graph and the variables set by the user (e.g. required components or values of specific properties) to specify an instance of a product, consistent with the stored model. The system has available a set of observers for visualising components, properties and constraints, which can be combined to build the interface with which the customer and the sales person will eventually interact.

A commercially available developing tool has been used build the knowledge base: Selling Point by Concentra (http://www.concentra.com). Selling Point is a development shell based on an Object Oriented language called GSL (Generative Specification Language) that uses a *building block* approach to constructing models. Selling Point Building Blocks consist of objects for logical configuration, functional descriptions, geometric modelling, graphical display, report input and output, and database connectivity. Models are constructed by assembling reusable building blocks from customised libraries.

Selling Point is a standard Windows-based application leveraging current technologies such as structured query language, dynamic link libraries, object linking and embedding and open database connectivity. Selling Point also supplies specific libraries for XcelleNet's Remoteware(R) for electronic software distribution, product data distribution, and order entry transmission.

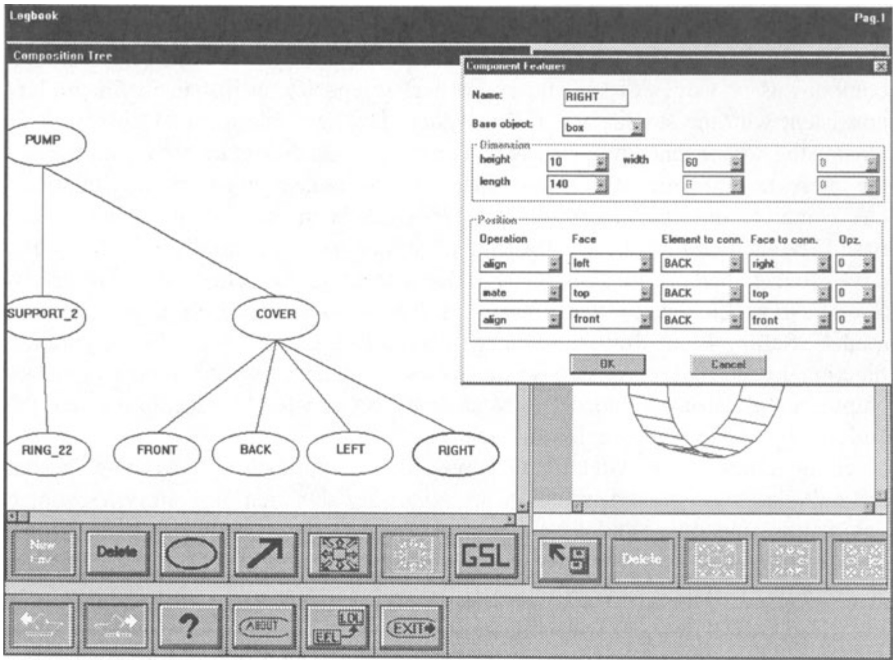
A Model Generator has been implemented in order to translate the CVE model into the GSL code that can be interpreted by Selling Point in order to generate the product model.

4.2 Application example: gear pump

A gear pump has been selected as application example. During the design phase, the gear pump is interactively defined by the user: components are selected from the components repository and added to the model by placing nodes and arcs in the CVE composition tree window (see Figure 2, left part). The model is then build as a tree where leaves represent basic components while nodes represent sub-assemblies.

When a new node is added to the model, all of its defining properties need to be filled up with a value, a reference to another property value, or a formula. If shape properties are used to define the part, a particular set of properties need to be defined in order to specify the shape positioning in respect with the rest of the model (see pop-up window in Figure 2).

The geometrical aspect of the generated product model can be visualized in the CVE graphical window (see Figure 3, note the GSL button allowing the activation of the Model Generator that translate the CVE model into GSL code).



Name: RIGHT
Base object: box
Dimension
height 10 width 60
length 140

Position

Operation	Face	Element to conn.	Face to conn.	Opz.
align	left	BACK	right	0
mate	top	BACK	top	0
align	front	BACK	front	0

Figure 2 - Example of node definition in the Designer Interface

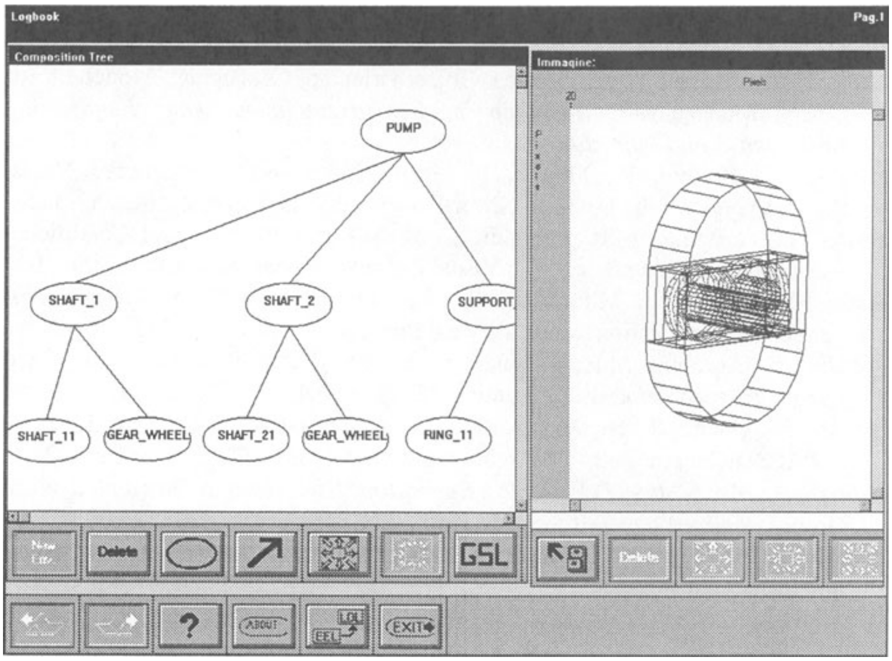


Figure 3 - Visualisation of the product model structure and shape.

5 CONCLUSIONS

This work is part of the CNR project (CO)^{2}DESIGN. Key aspects of the research activity have been the definition of a suitable knowledge base, containing different models, able to represent different knowledge types, integrated in a homogeneous environment, and the definition of appropriate interfaces that will allow different users to add/extract/modify the information stored into the knowledge base.

A reference architecture has been proposed that integrates commercially available tools for model representation and two types of interfaces: one for the design phase, the other for the customisation phase.

A gear pump has been selected as a test case in order to verify developed methodologies and to demonstrate the applicability of the approach.

6 AKNOWLEDGMENTS

The support of CNR for the project (CO)^{2}DESIGN, grant n° 97.02947.CT07, is acknowledged. We also thank Domenico Navarra for implementing the prototype used for experimental work.

7 REFERENCES

Assogna, P., Missikoff, M., (1995) Object-Oriented Conceptual Modeling for Design Management, *Workshop on Concurrent/Simultaneous Engineering Frameworks and Applications*, 389-400.

Bianchi, N., Bottoni, P., Mussio, P., Protti, M., (1993) Cooperative Visual Environments for the Design of Effective Visual Systems,*JVLC*,4(4),357-382.

Bottoni, P., Costabile, M.F., Levialdi, S., Mussio, P., (1996) Visual Conditional Attributed Rewriting Systems in Visual Language Specification,*VL96*,156 163

Bottoni, P., Costabile, M.F., Levialdi, S., Mussio, P., (1997a) From Visual Language Specification to Legal Visual Interaction, *VL97*, 238-245.

Bottoni, P., Costabile, M.F., Levialdi, S., Mussio, P., (1997b) Defining Visual Languages for Interactive Computing, *IEEE SMC-A*, 27, 773-783.

Calvary, G., Coutaz, J., Nigay, L., (1997) From Single-User Architectural Design to PAC*:a Generic Software Architecture Model for CSCW, *CHI97*, 242-249.

Dourish, P., Bly, S., (1992) Portholes: Supporting Awareness in Distributed Work Groups, Proceedings CHI'92, ACM Prees, pp.541-547.

Formica, A., Missikoff, M., Terenzi, R., (1995) Constraint Satisfiability in Object-Oriented Databases, *Workshop in Computing*, 48-60.

Kariko-Buhwezi, B., Cugini, U., (1995) A Knowledge Based Computer Programming for the Automatic Design of a Gear Pump, *Preprints of the First IFIP WG 5.2 Workshop, Knowledge Intensive CAD-1 (KIC '95)*, T. Tomiyama, M. Mantyla and S. Finger eds.., 421-434.

Mandorli, F., Bottoni, P., Mussio, P., Cugini, U., (1995) Interactive Knowledge Elicitation for Application Dependent Feature Evaluation Supported by Conditional Attributed Rewriting Systems, *Revue Internationale de CFAO et d'informatique graphique*, 10, (1-2), 71-93.

Mandorli, F., Otto, H., E., Kimura, F., (1993) A reference kernel model for feature-based CAD systems supported by conditional attributed rewrite systems, in: *Second ACM/IEEE symposium on solid modeling and applications - Solid Modeling '93*, 343-354.

Missikoff, M., Pizzicannella, R., (1995) A Knowledge Based Approach for Product data Modeling, *Workshop on Standards and Information Technology for Concurrent Engineering.*

Mussio, P., Padula, M., Protti, M., (1988) Attributed conditional L-systems: a tool for image description, *9th International Conference on Pattern Recognition*, 607-609.

Paterno', F., Mancini, C., Meniconi, S., (1997) ConcurTaskTrees: a Diagrammatic Notation to Specifying Task Models, *INTERACT'97*, 362-369.

Shah J.J., Mantyla M., Nau, D., (1994) Introduction to Feature Based Manufacturing, *Advances in feature based manufacturing*, Elsevier Science.

Van der Veer, G., Lenting, B., Bergevoet, B., (1966) GTA: Groupware task analysis - Modelling Complexity, *ACTA Psychologica, 91*, 297-322.