A knowledge-based approach toward computer-supported cooperative design

P. M. Wognum
Department of Computer Science
University of Twente
PO Box 217
7500 AE Enschede
The Netherlands

Abstract

The design of technical products nowadays is a complex process involving many individuals from various disciplines. At suitable time points during the design process, decisions made by these individuals have to be integrated and checked againts a huge number of constraints. Conflicts may be detected at such points.

To resolve the conflicts, members of a design team must understand what caused the conflicts, how serious the conflicts are, and how the elements of a conflict interact. Negotiation between the members of a design team is needed to determine the steps to take to remove the conflicts. To this end, they need a common product model at all stages during the design process. Without computer support, maintaining such a product model will hardly be possible.

In this paper we describe a knowledge-based approach toward computer-supported cooperative design based on a framework, called YMIR, for structuring engineering design knowledge. Product models built from the elements defined in YMIR integrate various aspects of a design. This characteristic enables various decisions to be integrated and allows for the integrated evaluation of a design against related constraints. The product model thus acts as a common model for design team members. It can be used as a basis for presenting conflicts to the design team members involved in the conflicts and as a basis for negotiation to resolve the conflicts.

1 Introduction

In the old days, the design and production of a new product were a one-man's job. Consequently, decisions taken during design and production were inherently integrated and tuned to each other. Current design and production processes, however, involve many individuals from various disciplines. It is a major problem to integrate the decisions made by these individuals, especially if the members of the design team are in different locations or even in different enterprises [Cutkosky et al.].

In addition, as can be observed in practical situations, changes to a design description are often made in isolation by the various members of a design and production team. If those changes are not properly checked against all relevant constraints errors may be detected in follow-up processes. This situation frequently leads to unnecessary iterations, loss of time, reduced quality, high costs, or worse, disasters like the collapse of the skywalks in the Hyatt Regency Hotel in Kansas City, Missouri, involving the death of 113 people [Kaminetzky (1991)].

Nowadays, the need for integration of all life-cycle activities from the start to be able to detect inconsistencies early on has been widely recognized. As a result, multi-disciplinary design teams, involving people from both technical and organisational disciplines like marketing and purchasing, are formed to integrate the related activities. This approach is known as *concurrent engineering*, a word coined in 1986 by the Institute for Defense Analysis (Report R338, IDA):

> Concurrent engineering is a systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support. This approach is intended to cause the developers, from the outset, to consider all elements of the product life-cycle from concept through disposal, including quality, cost, schedule, and user requirements.

From this definition it is clear that during the design of a product decisions are made from various perspectives by design team members. At suitable time points during the design process, these decisions have to be integrated and checked against a huge amount of constraints. These constraints come from various sources like customers, organisation, standards, product norms,

legislation, and various processes involved in the product life-cycle. Conflicts may be detected at such time points.

To resolve the conflicts detected, members of a design team must understand what caused the conflicts, how serious the conflicts are, and how the elements of a conflict interact. Negotiation between the members of a design team is needed to determine the steps to take to remove the conflict. To this end, they need a common product model at all stages during the design process.

The integration of decisions into a common product model, the detection of conflicts, finding their causes, and resolving them, are complex activities. Computer support and especially knowledge-based techniques are needed to manage this complexity.

In this paper a knowledge-based approach is described toward computer-supported cooperative design (CSCD) based on a framework, call YMIR [Alberts (1993)], for structuring engineering design knowledge. Product models built from the elements defined by YMIR integrate various aspects of a design. In addition, the product models represent the design at various stages during the design process. These characteristics enable the integration of the various decisions made and allow for the integrated evaluation of the related constraints. The conflicts detected as well as their causes can be presented to the design team members involved. The product models, hence, act as a common model for the design team members and can be used as a basis for negotiation[1].

The outline of this paper is as follows. In Section 2, the framework YMIR for structuring engineering design knowledge is briefly described. In Section 3, the evaluation of a design, possibly leading to the detection of conflicts, is described. In Section 4, the presentation of conflicts as well as their causes is discussed. By indicating the nature of the conflicts, advice about ways to resolve the conflicts can be given to members of a design team. Finally, in Section 5, some concluding remarks are made.

[1]The research described is part of the Stevin project. The Stevin project is aimed at supporting engineering design by means of knowledge-based techniques.

2 YMIR: a basis for composing common product models

Common product models on which the CSCD approach as described in this paper is based, have to be shared by all members of the design team during the design process. These models must, hence, satisfy certain requirements to be suited as a basis for communication and negotiation. These requirements will be discussed below.

The product models must represent the solution at several stages during the design process. At the start, the product model consists of the requirements which the final design must satisfy, At the end, the product model represents a description of the final design ready to be produced. In between, several product models are needed to represent intermediate designs. Consequently, the product models represent design descriptions at several levels of abstraction.

In addition, members of a design team represent specific aspects of a design. Examples of such aspects are performance aspects like the behaviour a design must have to satisfy specific functional requirements and the physical characteristics of a design, which enable the product to behave as required. The members of a design team also represent aspects related to down-stream process in the life-cycle of a product. The product model, hence, must integrate these various aspects.

The members of a design team may use different vocabularies for their part of the design. The vocabulary used in the product model must be understandable to all design team members to be suited as a basis for communication.

To support members of a design team to decompose the design problem into manageable parts and facilitate the systematic construction of product model, a set of predefined building blocks would be useful. To be able to reuse such building blocks for different design problems, the building blocks should be generic and should be based on generally accepted knowledge in the design domain. The building blocks must be instantiated for the particular design problem at hand.

A number of requirements can thus be formulated for common product models:

1. Models must be built from smaller elements, so-called *building blocks.*

2. The building blocks must be formulated at different levels of abstraction to bridge the gap between abstract requirements for the product to be designed and the final design description ready to be produced. Each level must be a true abstraction of the level below by hiding details which are irrelevant at higher levels.

3. The building blocks must represent partial design solutions. They can be specific, representing a small number of actual design solutions. In this case they represent (possibly parameterized) prototypical solutions and are therefore called *prototypes*. They can also be more generic, representin a large class of solutions. The last type of building block is called *generic component*.

4. The building blocks must be based on generally accepted knowledge in engineering domains to be reusable for different types of design problems. Such building blocks also guarantee a common basis for communication between design team members.

5. The building blocks must integrate various aspects of a design which correspond to the disciplines of the members of a design team.

In the Stevin project, a taxonomy of building blocks has been developed, called YMIR [Alberts (1993)] for structuring engineering design knowledge. YMIR satisfies the requirements formulated above.

The concepts defined in YMIR are based on System Theory. We are particularly interested in system models known as network models [MacFarlane (1970)]. These models are based on the conservation of energy in a system. Building blocks as defined in YMIR are based on these network models. They explicitly relate behaviour and form. This property allows for the integration of decisions related to behaviour and form, like decisions related to the geometry of a product and decisions related to the behaviour of a product under various loads. Product models built from the YMIR building blocks therefore inherit this property.

The number of aspects explicitly related to each other in product models based on YMIR can be extended by means of introducing new variables into YMIR building blocks the values of which can be derived from the values of already existing variables. For instance, the length of a product can be derived from the lengths of the constituing building blocks.

3 Design evaluation

The activities involved in the design process are often performed in isolation. For example, a designer may shape the geometry of a product without any notion of the load balance of the design, while the expert responsible for the strengths of the materials may suggest changes in the design based on errors detected during simulation. Also, during construction of a product, decisions are made which may conflict the original design description.

To support design team members, it is necessary to integrate the various decisions concerning a design description and check the resulting description against all relevant constraints. This way, errors can be detected early on thus preventing unnecessary iterations, costly repairs, or disasters like the one mentioned in the introduction.

Product models based on YMIR are suitable for evaluating decisions in an integrated way. First of all, the nature of generic components allows for the integrated evaluation of constraints related to form (material and shape) and behaviour (the behaviour of a product as a consequence of its form). In addition, the range of aspects and related contraints can be enlarged by extending generic components and product models based on YMIR with the relevant variables. For example, the costs of a product can be estimated by adding cost variables to instantiated versions of generic components.

The evaluation of a design has to be performed at suitable stages during the design process. The choice of these stages may coincide with finishing a design at a particular abstraction level as defined in YMIR. At each such evaluation point different subsets of constraints must be met. In [Dikker & Wognum (1993)] a method is described to select the constraints which must be checked at a certain stage.

The evaluation of the design and its related constraints can be performed by means of techniques developed in research on constraint-satisfaction problems (CSPs) [Bowen et al. (1990)] and reason-maintenance [deKleer (1986), Doyle (1979)]. These techniques can be used to detect conflicts which must be solved before the design process can continue [Dikker et al. (1993)]. In the next section, we describe some problems involved in presenting conflicts to the relevant team members and offering advice to solve the conflicts.

4 Presenting and solving conflicts

The constraints that must be checked during evaluation come from different sources, like customers, organisation, standards, production norms, suppliers, or legislation. By marking the constraints with their source, the source of the constraints in a conflict can easily be detected [Dikker & Wognum (1993)]. These constraints must be presented to the relevant design team members in such a way that they understand the conflict and know how to change their part of the design to resolve the conflict.

Presenting a conflict to design team members may involve the translation of constraints which together form the conflict, into the languages used by the respective disciplines. In this way team members may use additional tools to solve their part of the conflict.

A conflict must be solved before the design process can continue. Conflict resolution is an important aspect of CSCD and has been a major topic of discussion at a MIT-JSME Workshop in 1989 [Sriram et al. (1991)]. Several levels of complexity for solving conflicts have been recognized. In our approach, we are able to detect the level of complexity of a conflict by determining which elements form the conflict.

A conflict always consists of inconsistencies between the product model and constraints which must be satisfied. Constraints may be hard or soft. The last category of constraints may be relaxed to resolve the conflict. Techniques developed in model-based diagnosis research [Bakker et al. (1993)] can be used to detect which constraints may be relaxed to make the problem solvable. Relaxing constraints, however, may involve changing the original functional requirements and, hence, the original design problem.

If the constraints involved in the conflict are hard constraints, alternative solutions must be considered. To this end alternatives to (a combination of) building blocks have to be checked. Reconsidering alternatives may involve reconsidering decisions made in previous design stages. The simplest case consists of changing a particular instantiation of (a combination of) building blocks. More difficult is the case where building blocks have to be replaced by others. In this case additional constraints are added due to the selection of such new building blocks. The selection of a new generic component may also involve the change of a complete substructure of the design if the conflict involves a component introduced in previous stages.

Finally, if alternative solutions have been exhausted, which means that

the problem cannot be suitably solved with the current set of generic components, a total change of the design structure may be needed. Such a change may involve the selection of alternative sets of generic components from different engineering domains. This selection may be supported by using a method as has been developed by Altshuller [Altshuller (1984), Arciszewski (1987)]. This method has been implemented in the Invention Machine, a program developed in the Inventive Machine Laboratory in Minsk, Belarus. This program supports engineers in finding inventive solutions for difficult design problems [Dikker et al. (1992), Mars et al. (1993)].

5 Concluding remarks

We have described a knowledge-based approach for CSCD, computer-supported cooperative design. The approach is based on a common product model which acts as a basis for communication and negotiation between members of a design team. This product model is built from building blocks defined in YMIR, a framework for structuring engineering design knowledge. Such a product model integrates various aspects of a design and enables the integrated evaluation of constraints related to these aspects.

We have described the evaluation of a design as a constraint-satisfaction problem. If a conflict is found, it can be determined which members of the design team are involved in the conflict. The conflict must be presented to these members in a suitable format. By determining the nature of a conflict engineers can be supported in finding ways to solving the conflict.

We have described an approach in which the actors in the design process are human beings. The approach can be extended to incorporate computer systems which perform parts of the design process.

Acknowledgement

The author thanks her colleagues in the Stevin project, L.K. Alberts, F. Dikker, F. Tempelman, and N.J.I. Mars, for their valuable contributions to the research described in this paper.

References

[Alberts (1993)] L.K. Alberts, "YMIR: a domain-independent ontology for engineering design", Ph.D.-thesis (forthcoming), University of Twente, Enschede, The Netherlands, 1993.

[Altshuller (1984)] G.S. Altshuller "Creativity as an exact science. The theory of the solution of inventive problems", Gordon and Breach Science Publishers, New York, 1984.

[Arciszewski (1987)] T. Arciszewski "Ariz77: an innovative design method", in: Proceedings of the International Conference on Engineering Design, W.E. Eder (ed.), Boston, 1987.

[Bakker et al. (1993)]
R.R. Bakker, F. Dikker, F. Tempelman & P.M. Wognum "Diagnosing and solving over-determined constraint-satisfaction problems", accepted for IJCAI'93, Chambery, France, 1993.

[Bowen et al. (1990)] J. Bowen, P. O'Grady & L. Smith "A constraint programming language for Life-Cycle Engineering", Artificial Intelligence in Engineering 5, 1990, p. 206-220.

[Cutkosky et al.] M.R. Cutkosky, R.S. Engelmore, R.E. Fikes, M.R. Genesereth, T.R. Gruber, W. Mark, T.M. Tenenbaum & J.C. Weber "PACT: an experiment in integrating concurrent engineering systems", IEEE Computer 26, 1993, p. 28-38.

[Dikker et al. (1992)] F. Dikker, G.B. Leemeijer, P.M. Wognum & N.J.I. Mars "Towards computer-supported creative design based on Altshuller's theory of inventive problem solving", Memorandum Informatica 92-49, University of Twente, Enschede, The Netherlands, 1992.

[Dikker et al. (1993)] F. Dikker, L.K. Alberts & P.M. Wognum " A constraint-based approach to design critiquing", Accepted for publication in the Proceedings of IEA/AIE-93, Edinburgh, June 1-4, 1993.

[Dikker & Wognum (1993)] F. Dikker & P.M. Wognum "Constraint-based design critiquing: a knowledge-based approach to design for quality", accepted for ICED'93, August 17-19, The Hague, The Netherlands, 1993.

[Doyle (1979)] J. Doyle "A truth maintenance system", Artificial Intelligence 12, 1979, p. 231-272.

[Kaminetzky (1991)] D. Kaminetzky "Design and construction failures: lessons learned from forensic investigations", Mc Graw-Hill Inc., 1991.

[deKleer (1986)] J. de Kleer "An assumption-based TMS", Artificial Intelligence 28, 1986, p.127-162.

[MacFarlane (1970)] A.G.J. MacFarlane "Dynamic system models", George G. Harrap & Co. Ltd, London, 1970.

[Mars et al. (1993)] N.J.I. Mars, V.V. Sushkov & P.M. Wognum "AI and creative engineering design", in: Proceedings of the AAAI93 Spring Symposium on AI and Creativity, Stanford University, March 1993.

[Neches et al. (1991)] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator & W.R. Swartout "Enabling technology for knowledge sharing", AI Magazine 12, 1991, p. 36-56.

[Sriram et al. (1991)] D. Sriram, R. Logcher & S. Fukuda (eds.) "Computer-aided cooperative product development", Proceedings of the MIT-JSME Workshop, MIT, Cambridge, USA, November 1998, Lecture Notes in Computer Science, G. Goos & J. Hartmanis (eds.), Springer-Verlag, 1991.