Collaborative Product Development in CAD and CAPP

An approach based on communication due to constraint conflicts and user initiative

O.W. Salomons, J.M. Kuipers, J. de Graaff, F. van Slooten, F.J.A.M. van Houten, H.J.J. Kals
University of Twente, Mechanical Engineering Department, Laboratory of Production & Design Engineering, P.O. Box 217, 7500 AE Enschede, The Netherlands, phone: X–31–53–4892532, fax: X–31–53–4335612, e_mail: o.w.salomons@wb.utwente.nl

Abstract

This paper addresses collaborative product development, focusing on computer support of collaboration between users of CAD systems alone, between users of CAPP systems alone and between users of both CAD and CAPP systems. Computer support of collaborative product development may enhance informal information exchange. Apart from formal information exchange between CAD and CAPP systems, informal information exchange is necessary in achieving the goals of Concurrent Engineering. Collaboration can be started in a number of ways; as a result of one or more conflicts in constraints belonging to several users or voluntarily on the initiative of a user. Both these ways of starting collaboration should be supported. Besides collaboration, important related issues are engineering data management, handling of multiple views, design histories and a facility of supporting project management. A prototype implementation restricted to a re–design support oriented CAD system supporting both conflict based and voluntary communication is presented. The re–design support system currently supports several constraint types. The present implementation uses parameter constraints in the support of conflict based communication. A parameter constraint solver based on the use of simulated annealing is employed.

Keywords

CAD, CAPP, collaborative design, concurrent engineering, constraints, features, multi–user systems

1 INTRODUCTION

Due to the growing interest in concurrent engineering as a research topic, a clear trend towards the development of computer support of collaborative product development can be observed.

This paper restricts itself to computer support of collaborative product development in the fields of CAD and CAPP. There are several reasons for considering this area. Bridging island automation is one reason as today's CAD and CAPP systems often still are isolated, thus restricting their users not only in the exchange of formal product model data but also in communicating informally via the system. In the philosophy of Concurrent Engineering, a team approach is essential, so that designers and process planners and other team members should cooperate more. However, not all members of a project team work at the same site or time, which means that close cooperation is difficult. Moreover, due to the fact that design is often performed at one company or department, while process planning is performed at another, computer support of collaborative product development in CAD and CAPP is required in order to make geographically different sites virtually more the same.

It is our philosophy that design and process planning will remain separate disciplines, although a tendency can be observed of merging the two into one. Both specializations are so complex and require so much knowledge and so many skills that it does not seem useful to give all process planning tasks to designers (or vice versa). Of course designers will have a general awareness of manufacturing and process planners of design, but it seems impossible to have complete insight into the other discipline at all levels of detail. In contrast, in Next–Cut, a concurrent design system which aims at integrating detailed component design and CNC process planning, a user is responsible for both design and process planning (Cutkosky et al. 1992). However, this is a situation that should be avoided. Of course, due to a high level of automation in CAPP systems, process planners will have a different task compared to the previous situation. Assuming that they will have more time available in case CAPP systems are employed, they have more time to advise designers with regard to manufacturing issues. To have both process planners and designers perform design as well as make process plans, seems a bit unrealistic.

Cooperation may be formal or informal in two ways. The first way is that there is both formal and informal information exchange between systems. An example may the be the relationship between a contractor (design) and a subcontractor (process planning), where a physical separation between process planners and designers exists. The formal information exchange can be the exchange of a product model in some standardized format. The informal information exchange between systems may be the communication between the users of the systems. The second way of formal/informal cooperation is that communication between users (which is an informal part of the information exchange between systems) can be divided in constraint–based (formal) communication and voluntary (informal) communication. Our aim is to come to both a formal and an informal integration of design and process planning for each discipline individually and between the two disciplines considered.

Collaborative product development is part of concurrent engineering. There are many definitions of concurrent engineering. The following definition is the most practical for our research, as it gives a clear objective: Achieving maximum cooperativeness with a minimum of communication by maximizing the transparency. Transparency itself can be defined as the clarity of processes through which people can see the consequences of their decisions. In the case that clarity is provided, it is likely that people will have a global view of the consequences of these decisions. However, this definition of concurrent engineering does not provide a means to measure its performance or effectiveness. Effectiveness of concurrent engineering can be measured by how long it takes to find and correct a design mistake (Sevenler et al. 1993).

In collaborative product development coordination and integration problems may occur. Crabtree et al. have identified six broad categories of coordination and integration problems in their analysis of coordination problems (Crabtree et al. 1993):

- The information acquisition problem which relates to unavailability of information, mostly design rationale, due to difficulties in acquiring it.
- The information access problem which can be seen as the difficulty of accessing information that is either physically or electronically available.
- The knowledge access problem occurs when expertise of senior people, needed by less experienced designers, is unaccessible due to the fact that the high demand for advise from these senior people makes them hard to contact.
- The decision interdependence problem which relates to how individual decisions can cause severe coordination problems and introduce delay to the program.
- The activity management problem which relates to the non–adherence to schedules for non–technical reasons.
- The agent access problem which arises when key decision makers are unavailable when an important decision is to be made.

The coordination problems may influence each other, but the decision interdependence problem is often quoted as probably the largest and most important problem. As for decision interdependence Crabtree et al. (1993) write: "..decision interdependence requires a method of modelling and managing the inter–dependencies. Luckily there appears to be a solution: constraint networks."

A unified way to represent different levels of objects and the constraints between them, as described in e.g. (Shah et al. 1993b, 1994) and (Salomons et al. 1994b, 1995a), together with a unified constraint satisfaction method seems to be an interesting and powerful approach to combine with the ideas by Crabtree et al. (1993). This approach has been partly implemented in a collaborative product development module for FROOM, a CAD prototype which applies constraints. This paper discusses the general philosophy and the current implementation. At present, no CAPP system has been involved in the implementation effort, although it was considered in the overall philosophy and design. However, the concepts of applying constraints and communication in process planning are similar to design. Therefore, once proven to work within design, the concept will allow to be taken over into CAPP. The CAPP systems that have our special interest are the PART (van Houten 1991) and PART–S (de Vin et al. 1993) systems as these were (respectively are) in–house developments. Formal integration between FROOM and PART–S has previously been considered in (de Vries et al. 1994). In this approach the focus was not on communication between designers and process planners but on a design by least commitment approach in which the role of abstract features and DFM rules was investigated.

By the usage of constraints responsibilities will not only be shared or negotiated but also clearly appointed to a group member. Application of constraint satisfaction will provide rapid feedback. This will lessen the communication needed and improve the clarity of the design process. In the following, the FROOM system is addressed in section 1.1. and the constraints used in FROOM are addressed in section 1.2. Finally, section 1.3 will provide an overview of the remainder of this paper.

1.1 FROOM

FROOM stands for Features and Relations used in Object Oriented Modelling. FROOM is a prototype of a feature based re–design support system allowing the modelling of both components and assemblies. Relations can be expressed between the different objects by means of constraints. The focus has been on the support of re–design tasks because in practice a high percentage of all mechanical engineering design tasks can be considered as re–design tasks.

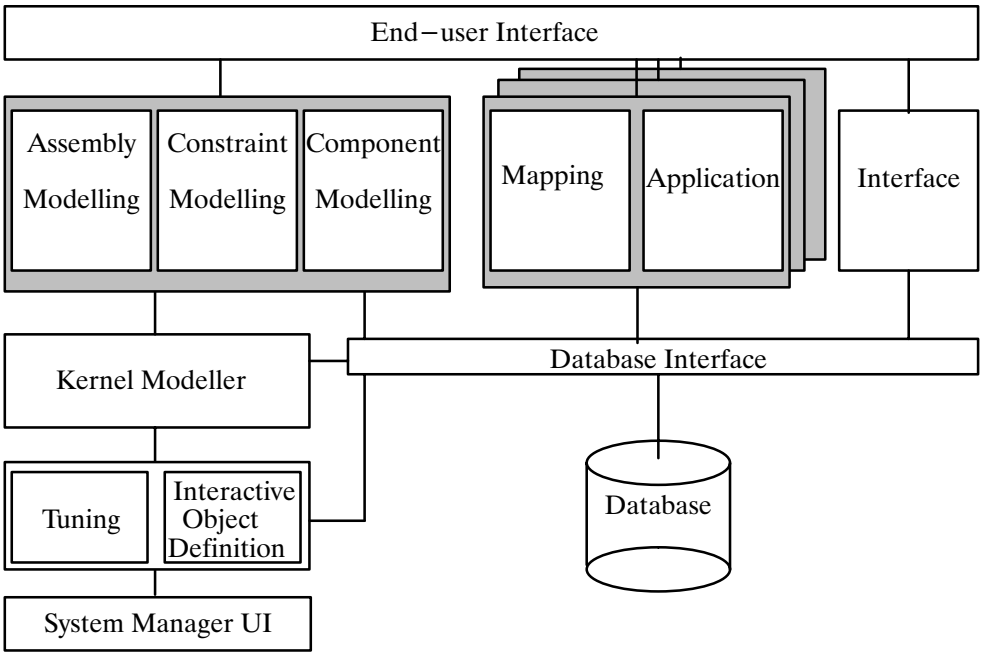

Figure 1 System architecture of the FROOM system.

The system architecture is divided into several functional modules, shown in figure 1. The modules exchange information through the database. Two different types of users, also shown in figure 1, are discerned: end–users and system manager users. The end–users are the designers who do the actual (re)design work. They design assemblies, components etc.. One of the philosophies that has driven the development of FROOM is that both system users and end–users should perform as little programming as possible. The system manager user customizes the system to an application domain, to a certain company as well as to certain users. A system manager user may be a project manager as well. For more details on FROOM refer to (Salomons, Kappert et al. 1993) for its concepts and (Salomons, van Slooten et al. 1993) for its methodic design. More recent references can be found in (Salomons et al.1995a,b,c,d).

A typical task for which the system manager user interface can be used is Interactive Object Definition. Objects can be divided into four categories: assemblies, components, features and the basic feature–elements. Interactive object definition can be divided into three sub modules: the Interactive Assembly Definition Module, the Interactive Component Definition Module, and the Interactive Feature Definition Module. The Interactive Feature Definition Module has been described in (Salomons et al. 1994a) and (Geelink et al. 1995). Other responsibilities of the system manager user are for instance the definition of domain dependent knowledge such as catalogues, standards etc..

The FROOM end–user interface, shown in figure 2, consists of two main parts: a graphic–interactive window for directly manipulating geometry similar to those in commercial CAD systems and a (context) window in which a network of objects and relations at various levels of abstraction, is displayed. This network represents the product model and has a bi–directional association with the geometry window. The network is a conceptual graph (Sowa 1984). It is not only used for presentation purposes, but also for the internal representation of objects and their relations and is discussed extensively in (Salomons et al. 1994b). The objects and relations can be detailed further to a lower level of abstraction. The graphic interactive window supports bottom–up design tasks while the context window supports top–down design tasks, thus allowing

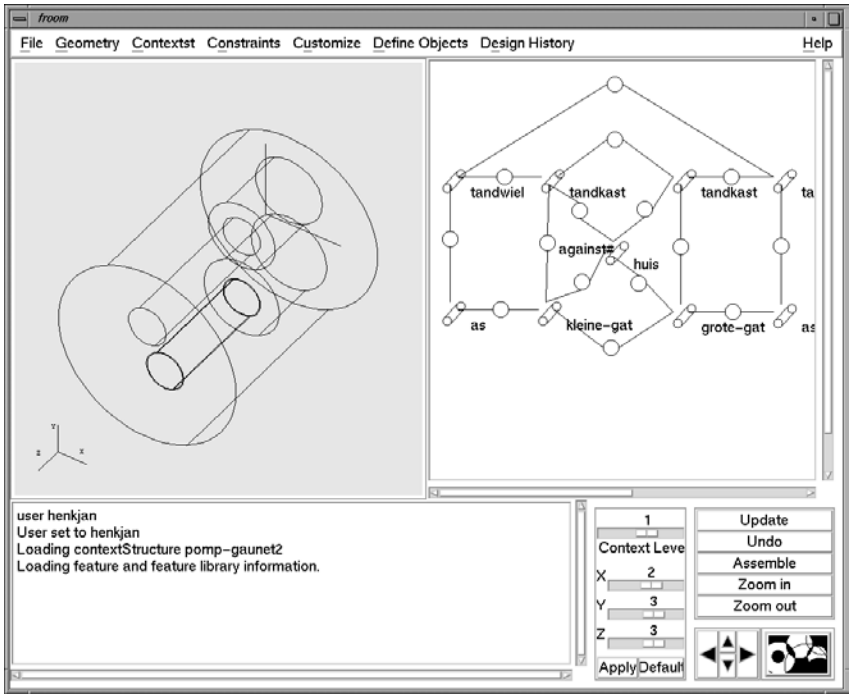

Figure 2 The FROOM user interface, consisting of a geometry window and graph (context) window that have a bi–directional association.

a mixed top–down and bottom–up approach. A design history function is also available in FROOM as well as a functional tolerancing functionality (Salomons et al. 1995a,b,d).

1.2 Constraints in FROOM

There are various definitions of constraints. The following definition is used within the FROOM context: *A constraint is either imposed on a single object or represents a desired relationship among two or more objects.* The constraint classification within FROOM discerns five types of constraints:

- Assembly constraints: these constraints are defined between (faces of) components to enable the modelling of an assembly. In FROOM, the assembly constraints are symbolic constraints like ”against”, ”align/fit”, ”contact”, ”orient”, and ”parallel offset”.
- Feature internal geometric constraints: these are (non variational) constraints within a feature, between the elements that constitute the feature; faces, edges etc.. Feature internal constraints currently applied in FROOM are for example: parallel, perpendicular, adjacent and co–axial.
- Feature external geometric constraints: these are (non variational) constraints between feature elements and non–feature elements within a component. These feature types are similar to the feature internal constraints: parallel, perpendicular, adjacent, etc.. These constraints are important for defining how features are to be instantiated and with respect to their behavior.
- Variational geometric constraints: these constraints can be considered as tolerance constraints and they can be defined between faces, between features or inside features or

even on a single surface. However, tolerances are related to functions and thus, the assembly model is important as well.

- Parameter constraints: these constraints can be defined either by the system manager or the end–user in FROOM. They can be regarded as algebraic equations and affect dimensions, geometry, forces, stresses etc..

The first three types of constraints are related to nominal geometry. A technique for solving geometric constraints called Degrees of Freedom Analysis, as proposed by Kramer (Kramer 1992) has been adopted for solving these constraints. Conventional geometry constraints solving packages use iterative numerical techniques like for instance Newton–Rhapson. Kramer, however, works from the geometry directly instead of using numerical techniques. Therefore, DOF analysis is a novel, more intuitive technique for solving geometric constraint problems and shows to be superior to conventional, iterative numeric techniques. The technique relies on a layered approach to solving a constraint problem. The first step is to reason symbolically about how to assemble a collection of "ghost" parts. The next step is to use the resulting assembly plan to guide the solution of the equations resulting from an algebraic description of the constraint problem. This assembly plan guides the solution of the equations, by implementing actions as simple equations which are solved in stylistic ways (Kramer 1992).

In FROOM, Kramer's DOF analysis approach has been translated from the original kinematics domain into solving geometry related constraints on assembly, component, feature and feature element level. For assemblies, the approach by Liu and Nnaji (1991) was combined with Kramer's approach. The general application of the DOF analysis approach in FROOM is detailed in (Salomons et al. 1994b) and (Salomons 1995a). In feature definition DOF analysis is combined with similar ideas for a 2D sketcher by Arbab and Wang (1989); more details are provided in (Salomons et al. 1994a, 1995a). Similar research on the use of DOF analysis in assembly and feature modelling has been reported in (Anantha et al. 1992), (Shah et al. 1993b, 1994).

The satisfaction of variational geometric constraints is based on the work by Clément et al. (1993, 1994). This work is based on the kinematics of small rotations and translations. For tolerance analysis, this theory has been combined with the kinematics theory by Kramer on large rotations and large translations. For more details refer to (Salomons et al. 1995a,b).

Parameter constraints in FROOM are solved using the simulated annealing approach by Thornton (Thornton 1993) and (Thornton & Johnson 1993). This parameter constraint satisfaction method will be explained further in section 4 and the Appendix as – for the moment – this is the only constraint satisfaction method in FROOM applied to collaborative product development. However, all the five constraint types mentioned earlier can be used as a basis for communication in a collaborative product development module.

1.3 Organization

The organization of the remainder of this paper is as follows. Section 2 briefly summarizes related literature. Section 3 describes the design of the module for collaborative product development in FROOM; collaborative modules for CAPP systems like PART will be similar to this module. Section 4 elaborates on the constraint satisfaction method used for solving parameter constraints that was applied in the collaborative design module as presented here. Section 5 addresses the present implementation of the collaborative product development module in FROOM. Finally, in section 6 conclusions and recommendations are presented.

2. RELATED LITERATURE

Roughly, one can divide multi–user systems into group collaboration architectures that do not use constraints as a basis for cooperation, and those that do use constraints as a basis for cooperation. The non–constraint based systems are addressed in section 2.1 while the constraint based ones are elaborated in section 2.2. In addition, both types of multi–user systems may apply engineering data management (EDM), necessary for managing models and for providing multiple views, which will be treated in section 2.3.

2.1 Multi–user systems without constraint–based communication

There are different multi–user systems that do not use constraints as a basis for cooperation. One such a multi–user system is PACT (Palo Alto Collaborative Testbed), a concurrent engineering structure that encompasses multiple sites, subsystems and disciplines (Cutkosky et al. 1993). PACT included Next–Cut (Cutkosky et al. 1992) as a subsystem and demonstrated how an agent–based approach could be applied to a multi–disciplinary design task. Four independently developed, existing multi–tool systems were encapsulated as agents. Agents are defined as computer programs that communicate with other external programs exclusively via a predefined protocol. These agents communicated through facilitators that translate tool–specific knowledge into and out of a standard knowledge–interchange language. In SHADE (SHAred Dependency Engineering), an extension of PACT, this language is called KIF (Olsen et al. 1994). Ontologies, defined as agreed upon sets of terms and meanings that enable parties to exchange knowledge for some purpose in some domain, are used to build such a language.

Drawbacks of PACT are its decentral and informal organization. As there is no (seemingly) central database, each agent has to know where it can find the information needed. Changes in (re)design can be made freely, without any responsibility or consideration towards the consequences for other users. In addition, ontologies were informally documented. This implies that a shared knowledge base does not exist within this testbed. Another multi–user system, related to PACT and SHADE, is SHARE (Toye et al. 1993). SHARE uses ontologies as well but these are centrally stored. Constraint solvers are present in SHARE but it is not clear if these are intended to start communication, and if so, how this actually works.

An integrated approach to design data has been made by Shah et al. (1992). Shah et al. created a design environment comprising a number of specialist design systems that cooperatively solve design problems, from conceptual lay–out to detailing.

During collaborative product development, both communication and social behavior play an important role. Multi–user communication systems will have to support social behavior. Studies of social behavior during collaborative product development (e.g. brainstorm sessions) performed by Tang have lead to the following recommendations for multi–user communication systems (Tang 1991):

- Convey gestures, maintaining their relationship to the drawing space
- Convey the process of creating and using drawings, with a minimal delay
- Provide concurrent access to the drawing space
- Allow intermixing among drawing space functions and actions
- Enable all participants to share a common view of the drawing space (WYSIWIS – What You See Is What I See)

One of the first experimental tele conferencing systems trying to apply these recommendations was TeamWorkStation, a desktop real–time shared workspace, developed by Ishii (Ishii 1990,

1991). Another distributed multi–party desktop conferencing system called MERMAID (Multimedia Environment for Remote Multiple Attendee Interactive Decision–making) is aimed to enable real–time conferences by interchanging information through video, voice and multimedia documents (Watabe 1990). Commercial tele conferencing software packages, containing a lot of the above mentioned required functionality, are already available, e.g. InPerson^{TM} (Silicon Graphics 1994). However, InPerson^{TM} lacks some of the required functionality; for instance project management and design history.

2.2 Multi–user systems with constraint–based communication

Wong et al. have developed a collaborative engineering system in which the representation of product information supports communication and coordination (Wong et al. 1993). This product information, encoded in an information model called SHARED, includes not only the geometric and physical properties of the product and its parts but also information about functions, constraints and the design rationale. Taleb–Bendiab et al. present a model of collaborative design embodied in a multi–agent system (SIMAD), using constraints as a basis for communication (Taleb–Bendiab et al. 1992). However, both references are not very clear on the nature and use of their constraint satisfaction methods.

Bowen et al. have shown that constraint–based systems can provide a good basis for collaboration and negotiation by the introduction of a constraint–based programming language, called Galileo3 (Bowen and Bahler 1992, 1993). Galileo3 is able to perform some constraint satisfaction as well as constraint monitoring on parameter constraints. Constraint monitoring, in the Galileo context, means accepting additional parameter and constraint declarations from the user and attempting to derive as many logical consequences of these declarations as possible. However, Galileo3 is unable to handle non–linear equations. The user–interface is very basic and adding and retracting constraints by designers requires programming skills. This is a severe drawback as designers are not programmers in the first place.

Another constraint–based multi–user system is ParMan, a collaborative design agent that is part of the SHADE project (Kuokka et al. 1994). It uses parameter constraint satisfaction based on CLP(R) (Jaffar et al. 1992) in order to support collaborative engineering. CLP(R) is a constraint programming language able to solve (in)equalities, linearities and (some) non–linearities, using different types of iterative numeric techniques. Although a programming language is used, no programming by the designer is required as a (powerful) constraint editor is applied. Apart from the underlying technologies, the user interface shows some similarity with the FROOM system. A drawback of ParMan is that each user has to explicitly make his constraints available to other users. This means that the interdependency problem cannot always be solved due to forgetting to make private constraints public. Therefore, a more fruitful approach can be that constraints are made publicly available by default, while users still would have the possibility to have private constraints but that the private constraints have to be explicitly selected instead of the other way around.

2.3 Engineering Data Management and Multiple Views

For a multi–user system it is necessary to be able to manage different versions of product models as well as to exchange (parts of) the product model during product development. Database technology is essential in both data management and in data exchange. It will sometimes also be necessary to support multiple views when team members from different disciplines are willing to see what their colleagues are doing. In the following, therefore, the issues of databases,

information exchange and multiple views are discussed in relation with collaborative product development.

Databases

Databases in collaborative product development can be of different nature. Roughly a distinction can be made between (virtually) central (but actually heterogeneous) shared databases, and distributed and non–shared databases. PACT shows an example of distributed and non–shared databases (Cutkosky et al. 1993). Communication in this case is supported by ontologies in combination with the agent–facilitator structure. Shah et al. use database technology to achieve integration (Shah et al. 1993a). Shah et al. advocate active database management systems that attempt to provide both modularity and timely response to critical situations. They present a framework that views an engineering design environment as an integrated, heterogeneous database system. This integrated view of data makes use of data exchange standards in PDES/STEP. Active database technology must be used to support a rule–based approach to constraint propagation and change notification (Shah et al. 1993a,b,c,d), (Urban et al. 1992,1993). Design histories and cooperative work have become a topic of attention, especially in combination with the support of collaborative design processes (Shah 1993d).

Information exchange

Formal integration may be achieved by the usage of agreed upon product model exchange formats such as e.g. STEP which is mainly useful when a product model has fully been completed. In order to exchange product model data during the product development process, before the product model has been completed, more advanced exchange formats are necessary. One solution is to allow each system its own representation format and database and to base information exchange on the use of ontologies and agent based systems Cutkosky et al. (1993). This seems a practical approach when a great number of different systems have to be used in a collaborative engineering environment. Another possible solution is the use of a shared representation format containing several types of objects (e.g. assemblies, components, features and feature elements) and a number of standardized constraints that can be imposed on or between the objects, e.g. (Shah et al. 1993b, 1994), (Salomons et al. 1994b). This approach seems to be particularly useful when only a few systems in overlapping domains are to be connected. Together with a unified constraint mechanism, product model data exchange can be performed during the product development process instead of after. Also, it will be easier to get an overall in–depth view of the product model of other team members although they may belong to a different discipline. Central evaluation of the constraints could help detect conflicts in product development early and start communication between team members timely. In case process planning is performed by subcontractors, exchange of constraints would still enable designers and process planners to develop products collaboratively. Informal integration can be achieved with communication systems that support voluntary communication and group sessions.

When considering only CAD and CAPP systems in a collaborative engineering environment, the approach by Shah et al. using heterogeneous databases in combination with a unified approach towards objects and constraints seems to be better suited than the one by Cutkosky et al. (1992). First of all, there will hardly be a need to program software similar to agents, facilitators and ontologies, which can be a lot of work. Second, it will be easier to have a more in depth ”look” at the other side of ”the wall”. Also conflicts will potentially be detected earlier because constraint satisfaction is performed centrally, while in PACT constraint satisfaction is performed only locally when some important changes have been made.

Multiple views

When considering CAD and CAPP only, multiple views can mainly be identified towards the features domain, i.e. design features and process planning features, see e.g. (Salomons, van Houten & Kals 1993) for an overview. Interactive feature definition might lessen the need for multiple views towards features (Salomons et al. 1994a). However, the possibility of having to support multiple views towards features may remain. Therefore, use could be made of the unified object and constraint approach (a unified feature representation scheme) as discussed previously. If this cannot help out, procedural rules for transforming one view (mainly design) to the other (manufacturing) are necessary. One of the possibilities is to apply object oriented approaches like Aksit et al. (1993), another is to use database technology.

3. DESIGN OF A MODULE FOR COLLABORATIVE PRODUCT DEVELOPMENT IN FROOM

The methodic design approach by van den Kroonenberg (1985) has been applied to the design of the module for collaborative product development in FROOM. In fact, this methodology has been applied to the design of FROOM as a whole (Salomons, Kappert et al. 1993, Salomons, van Slooten et al. 1993). This approach provides a structured design approach, allows for discussing the design and helps in identifying weak points. The first aim is to come to a functional black box description of the system with input and output characteristics. This is shown in figure 3. This figure can be explained as follows:

Before a collaborative design session starts, it is possible that some previously designed parts and/or assemblies, including design history, can be re–used. It is the task of the project leader (system manager) to divide the tasks on the basis of a functional, modular structure that he can pre–define. Subsequently he can assign parts of the total structure to individual team members or even teams who then have to work these parts out. The steering control comprises design knowledge of the team, design experience of the team and the manufacturing knowledge of the team. These controls of the process are necessary to come to a (re)designed product, a completed project, and (new) design history. More design and manufacturing knowledge of the team is obtained during the design process and this new knowledge is used as new steering input of the current or future processes.

The second phase encompasses a functional decomposition of the system. In such a functional model only those functionalities that are suitable for application within FROOM, will be present. Figure 4 presents a functional model for collaborative product development for one project and is a refinement of figure 3. The input and output characteristics of the black box work on the entire functional model. These characteristics are not shown in order to keep the model understandable.

The communication function from figure 4, fulfilled by a communication mediator, can be divided into two main functions:

- conflict–based communication, initiated or triggered when designers create a conflicting overall product model.
- non conflict–based communication, started whenever necessary. For instance, to make a project planning, to organize a meeting or simply if information has to be requested which cannot be found in the design history or elsewhere.

Communication may be either uni–directional (e.g. e–mail) or multi–directional (e.g. video conferencing). This implies that a communication mediator should allow both. The first communication function, conflict–based communication, needs a tool which can signal the creation of

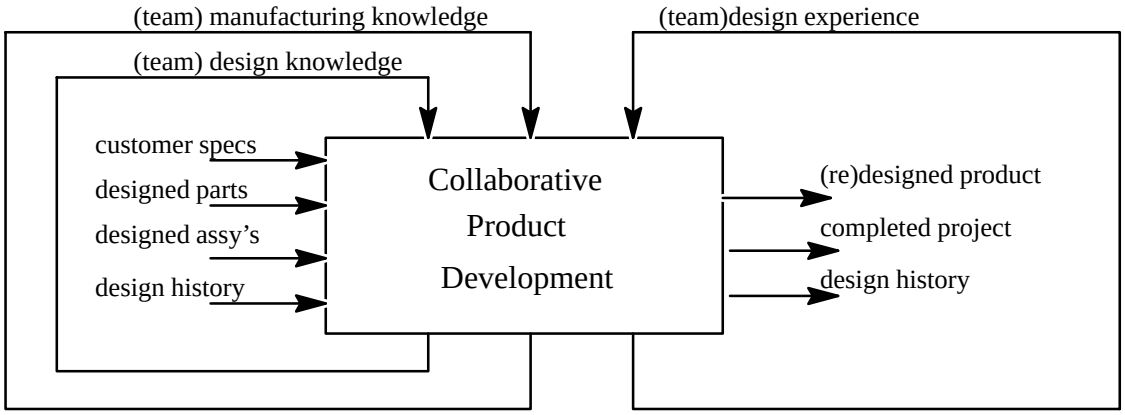

Figure 3 The collaborative product development system as a black box, with its input and output characteristics.

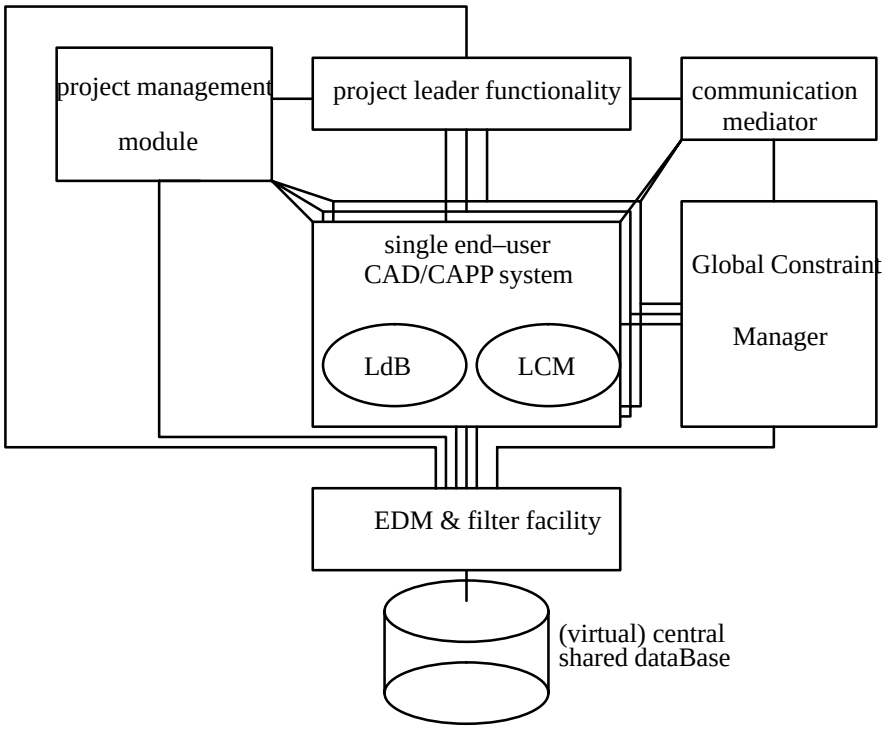

Figure 4 Extended functional model translated into a system architecture for collaborative product development (LdB = Local dataBase, LCM = Local Constraint Manager).

a conflict. Detecting these conflicting constraints requires a constraint manager. Such a constraint manager should be present for each subsystem which then enables each end–user to come to a consistent and well–designed subassembly. This local constraint manager might encompass more than one constraint type. For the time being, we will restrict ourselves to parameter constraints.

The presence of a local constraint manager (LCM) in each sub system provides possible solutions for the design of local (sub)assemblies. It enables the end–user to make a faultless i.e. acceptable design of his own local (sub)assembly. As can be seen in figure 4, each end–user has its own LCM. An acceptable design of a subassembly may not always lead to an acceptable solution when all the subassemblies are linked together in one overall product model. If constraint

conflicts emerge during a global constraint verification, the design will not be acceptable. A global constraint manager is needed in order to signal the unacceptable designs that would be acceptable for the LCM's. Such a global constraint manager couples the subsystems and evaluates the interfacing constraints (on the links) between the subassemblies and other related constraints (if necessary). There is only one global constraint manager (GCM) for the integrated CAD/CAPP platform. If there are conflicting constraints, the global constraint manager triggers a signal to the communication mediator. The communication mediator then starts conflict–based communication between the end–users involved to provide them a view on the conflict. The filter will be set by default to the conflicting constraints because this is where the end– users are most likely interested in. During conflict–based communication it is allowed to adjust the filter in order to obtain a better look or understanding of the problem. This means that the communication mediator offers more than the possibility to communicate efficiently. It also provides transparency by allowing to adjust the filter(s) when necessary.

The different backgrounds of the disciplines (design or process planning or even disciplines within either of the two), result in different views on the same product model. The different views may be integrated by an EDM function to ensure that a seamless shared environment originates. In such an environment each group member can call for the data he needs hence obtaining transparency. The EDM function can be extended by the usage of filters. Filters are necessary in overcoming the different views on the same design between the different domains. This means some kind of hybrid system, where EDM works centrally and the filters work locally.

The fact that constraints can be used very well in collaborative design, was already shown in the research by Bowen & Bahler and by Kuokka et al. However, the approach by Bowen and Bahler did not match very well with the FROOM philosophy, which aims at preventing the designer from programming. Also, the research by Kuokka et al. based on conventional, iterative constraint solvers did not match very well as designers were given the freedom to be as cooperative as they would like to be. Therefore, another approach to the use of parameter constraints and parameter constraint satisfaction was proposed. This parameter constraint satisfaction was based on simulated annealing and has been implemented (section 5). The constraint satisfaction algorithm can be used to detect conflicts.

When the end–users somehow cannot reach a solution during negotiation, they can appeal to the project manager. The project manager then decides what solution is chosen. This may involve interference by the system user in a local constraint manager by setting certain constraints fixed.

Part of collaborative design is (project)planning. The planning function is shown in figure 4 and can be accessed by both system user and end–user. As during a project due–dates or deadlines have to be met, one can speak of time–constraints. A link may be present between the project related time–constraints and the product model related constraints mentioned earlier. However, their mutual influence is beyond the scope of this paper.

For implementation, only one constraint type respectively constraint solver have been considered: parameter constraints and Thornton's simulated annealing approach. This will be treated in the next section.

4. THE PARAMETER CONSTRAINT SATISFACTION METHOD BASED ON SIMULATED ANNEALING

Simulated annealing is an optimization search strategy which was invented in the early eighties by Kirkpatrick et al. (1983). Simulated annealing is based on the behavior of natural systems.

There is an analogy with thermodynamics, with respect to the way that liquids freeze and crystalize or metals cool and anneal. This explains the name 'simulated annealing'.

Thornton uses simulated annealing for the embodiment phase of the design process (Thornton 1993) and (Thornton & Johnson 1993). Her research resulted in a prototype tool called CADET, Computer Aided Design Embodiment Tool. A subsystem of CADET was used for constraint specification and constraint satisfaction in embodiment design. This subsystem is called SANCS, Simulated Annealing Network Constraint Satisfier. When compared with other parameter constraint satisfaction techniques, such as the graph based approach by Serrano (1987), the one by Thornton has the following advantages:

- No numeric instability.
- Support of interval algebra.
- Both equalities and inequalities can be handled.
- Solves under constrained sets of highly coupled, non–linear equations

In the appendix at the end of this paper, a more detailed overview of the simulated annealing approach that was applied is given.

5. IMPLEMENTATION

Two concepts were implemented: a parameter constraint solver based on simulated annealing similar to SANCS (de Graaff 1994) and a communication tool capable of allowing several different types of communication (Kuipers 1994). The procedure of the use of constraints for conflict–based communication is shown in figure 5 and can be described as follows. The constraints are specified in a constraint editor. However, due to the implementation of the editor not having been completed, tests were run on constraints listed in a file. The constraints are then solved and if any conflicting constraints are detected, the communication mediator is started. This is the so–called constraint–based communication which is oriented towards initiating communication formally. Without any conflicts, the results of constraint satisfaction are presented. Naturally, voluntary, informal communication can be started too. The FROOM environment was used, i.e. C++, X–Windows/Motif, Acis and the Builder Xcessory tool (Integrated Computer Solutions 1994). In the following, the communication tool is elaborated in section 5.1 while the implementation of the constraint solver is detailed in section 5.2.

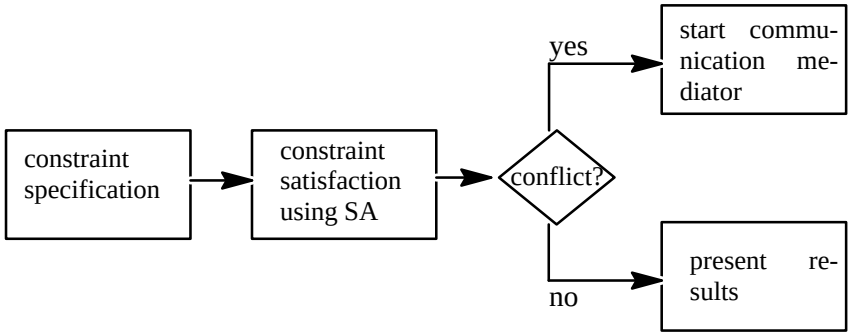

Figure 5 Implemented procedure.

5.1 Communication tool

The communication tool is divided into a uni–directional and multi–directional part, as mentioned earlier in section 3. The window of the communication tools is shown together with the

FROOM window in figure 6. In the left column the buttons for uni–directional communication are placed, and in the right column the buttons for multi–directional communication are located. A possibility of multi–directional communication is video conferencing. The video conferencing tools that are used, are public domain software and can be accessed via the Mbone Homepage (1994). Finally, in a horizontal bar at the bottom of the window, other functions, like the help function and the design history record function, are placed. However, the design history record button does not yet work; the design history functionality as currently present in FROOM has not yet been extended to support collaborative engineering (Salomons et al. 1995d).

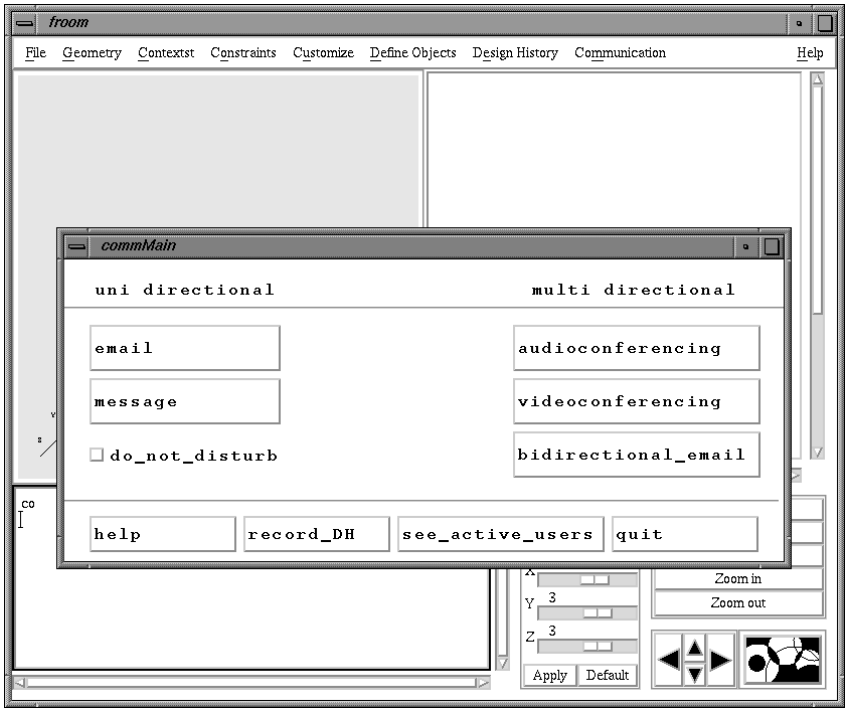

Figure 6 Top level user interface of the communication possibilities offered to the end–user.

5.2 Constraint solver

The simulated annealing approach by Thornton was implemented using the object–oriented C++ language (de Graaff 1994). In the following, only conflict detection is explained in more detail. Generally, three different types of conflicts in constraints can be discerned:

- Conflicts initiated by a wrongly specified constraint. This type of conflict is introduced during constraint specification. This type of conflict has to be detected in the constraint editor before constraint satisfaction is applied.
- Conflicts initiated by a false range of variables. E.g. $x>6$ and $x<5$. Such conflicts are also introduced during constraint specification and are checked before constraint satisfaction is applied.
- Conflicts that occur during constraint satisfaction. These are the most interesting conflicts, which are also detected by SANCS. The conflict detection is performed as follows. After a certain number of iterations, the constraint solver stops calculating and conflict detection is started. This can be performed by the following procedure which uses the normalized error di (see appendix), necessary for the energy calculation, to detect conflicting constraints:

- Take the first constraint
- Calculate the normalized error di of that constraint
- If di>ε and the constraint is an inequality constraint, then warn the user who initiated the conflict (see appendix for di and ε).
- Take the next constraint and execute this entire procedure until the last constraint has been checked.

6. DISCUSSION

Performance of the constraint solver was fair but not in real time as a user would like to have it. Tests were run on constraint sets of approximately 20 constraints with 50 variables. Depending on the range of the variables, it took about 2 – 5 minutes to solve these sets or detect a conflict on Silicon Graphics Indy workstations with R–4000 100MHz MIPS processor.

Up to now, the experiments on simulated annealing were applied only within the FROOM system; no CAPP system was involved.

7. CONCLUSIONS AND RECOMMENDATIONS

A design and a partial implementation of a collaborative product development module for CAD and CAPP systems has been presented. Collaborative product development in this module is based on both the use of constraints and on communication on a voluntary basis. The prototype implementation proved that constraints can indeed be used as a basis for communication in collaborative product development. Constraint conflicts are used to start communication. This will lead to a maximum of cooperativeness with a minimum of communication thereby maximizing the transparency, in accordance with the definition of concurrent engineering from section 1. Therefore, the efficiency of concurrent engineering, which is measured by how long it takes to find and correct a design mistake, is enlarged by the use of constraints as a basis for communication in collaborative design. However, the partial implementation needs to be completed in order to allow more complex and realistic cases of collaboration to be supported. A unified constraint solver, based on other techniques than simulated annealing alone like DOF–analysis, Clément's tolerancing solver should be developed to support extended cases of conflict–based communication. The efficiency of the simulated annealing algorithm could be improved. Also, tests in actual practice still have to be made to verify the preliminary results.

Another future research topic related to collaborative product development is project planning. More research has to be performed into the feasibility of an active, computer supported project planning tool. Also, a link between the Design History tool and the communication mediator is a future topic of research.

Acknowledgements

This research was supported by the Technology Foundation (STW), a Dutch foundation funded by the Dutch government.

REFERENCES

Aksit M., Bergmans L., Bosch J., Yonezawa A., Wakita K. (1993) Abstracting Inter–Object Communications Using Composition–Filters, *Proceedings of the ECOOP'93 Workshop on Object–based Distributed Processing*, R. Guerraoui, O. Nierstrasz, M. Riveill (Eds.), LNCS 791, Springer–Verlag, http://www_trese.cs.utwnete.nl/Tresepapers/tresepapers.html

Anantha R., Kramer G.A. (1992) Crawford R., An architecture to represent over, under and fully constrained assemblies, ASME Winter Annual Meeting, 233–44.

Arbab, F., Wang, B. (1989) A constraint–based design system based on operational transformation planning, *Proceedings of the 4th International Conference on Applications of Artificial Intelligence*, pp. 405–26

Bowen, J., Bahler, D. (1992) Frames, quantification, perspectives, and negotiation in constraint networks for life–cycle engineering, *Artificial Intelligence in Engineering*, vol.7, pp. 199–226

Bowen, J., Bahler, D. (1993) Constraint–Based Software for Concurrent Engineering, *IEEE Computer*, vol. 26, pp. 66–68

Crabtree, R.A., Baid, N.K., Fox, M.S. (1993) An analysis of coordination problems in design engineering. *International Conference on Engineering Design '93*, The Hague, The Netherlands, vol. 1, pp. 285–92

Clément, A., Rivière (1993) A., Tolerancing versus nominal modelling in next generation CAD/CAM system, *CIRP seminar on Computer Aided Tolerancing*, Cachan, pp. 97–113

Clément A., Rivière A., Temmerman M., (1994), "*Cotation tridimensionelle des systèmes mécaniques, théorie & pratique*", PYC Edition, Yvry–Sur–Seine Cedex (ISBN 2–85330–132–X), in French (English version is in progress).

Cutkosky, M.R., Tenenbaum, J.M., Bown, D.R. (1992) Working with multiple representations in a concurrent design system, *Journal of Mechanical Design*, Vol. 114, pp. 515–524

Cutkosky, M.R., Engelmore, R.S., Fikes, R.R., Genesereth, M.R., Gruber, T.R., Mark, W.S., Tenenbaum, J.M., and Weber, J.C. (1993) PACT, an Experiment in Integrating Concurrent Enginering Systems, *IEEE Computer*, vol. 26, pp. 28–37, http://www–ksl.stanford.edu.knowledge–sharing

Geelink R., Salomons O.W., Slooten F. van, Houten F.J.A.M. van, Kals H.J.J., (1995) Unified feature definition for feature based design and feature based manufacturing, proceedings ASME Computers in Engineering Conference (CIE'95), Boston.

Goldberg, D.E.(1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, publ. Addison–Wesley

Graaff, J. de (1994) Een Mathematische Constraint Solver voor FROOM, *B.Sc. thesis PT–507*, University of Twente, Enschede, The Netherlands

Houten F.J.A.M. van (1991) PART, a computer aided process planning system, PhD thesis, University of Twente, Enschede.

Integrated Computer Solutions Inc. (1994), The Builder Xcessory user's guide, Unix edition, v. 3.1, Cambridge Ma

Mbone homepage 1994, The Mbone Homepage is accessible under the following internet address: http://www.eit.com/techinfo/mbone/mbone.html

Ishii, H. (1990), TeamWorkStation: Towards a Seamless Shared Workspace, *Proceedings of the Conference on Computer Supported Cooperative Work '90*, Los Angeles, USA, pp. 13–26

Ishii, H. (1991) ClearFace: Translucent Multiuser Interface for TeamWorkStation, *Proceedings of the Second European Conference on Computers Supported Cooperative Work ECSCW '91*, Amsterdam, The Netherlands, pp. 163–74

Jaffar, J., Michaylov, S., Stuckey, P.J., and Yap, R.H.C. (1992) The CLP(R) Language and System, *ACM Transactions on Programming Languages and Systems*, Vol. 14, No. 3, pp. 339–95

Kirkpatrick S., Gelatt C.D., Vecchi M.P. (1983) Optimization by simulated annealing, Science, vol. 220, no. 4598, 671–680.

Kramer, G. (1992) *Solving geometric constraint systems, a case study in kinematics,* The MIT press, Cambridge, Massachussetts, London, England

Kroonenberg, H.H. van den (1985) Design Methodology as a Condition for Computer Aided Design, *VDI–Berichte 565*, VDI–Verlag, Düsseldorf

Kuipers, J.M. (1994) A Collaborative Design Module for FROOM, *M.Sc. thesis PT–505*, University of Twente, Enschede, The Netherlands

Kuokka, D., Livezey, B. (1994) A Collaborative Parametric Design Agent, *Proceedings of the National Conference on Artificial Intelligence '94*, http://hitchhiker.space.lockheed.com/aic/kcd/parman/parman.html

Liu, H–C., Nnaji, B. (1992) Design with spatial relationships, *Journal of Manufacturing Systems*, vol. 10, no. 9, pp 449–63.

Olsen, G.R., Cutkosky, M., Tenenbaum, J.M., and Gruber, T.R. (1994) Collaborative Engineering based on Knowledge Sharing Agreements, *to appear in the Proceedings of the 1994 ASME Database Symposium*, September 11–14, Minneapolis, MN, http://www–ksl.stanford.edu/knowledge–sharing/papers/index.html#olsen–asme–94.

Press W.H., Flannery B.P., Teukolsky S.A., Vetterling W.T. (1992) Numerical Recipes in C: The art of Scientific Computing, Cambridge University Press, Cambridge, second edition.

Salomons, O.W., Kappert, J.H., Slooten, F. van, Houten, F.J.A.M. van, Kals, H.J.J. (1993) Computer support in the (re)design of mechanical products, a new approach in feature based design, focusing on the link with CAPP, *Proceedings of the IFIP TC5/WG5.3/IFAC International Working Conference on Knowledge Based Hybrid Systems in Engineering and Manufacturing*, Budapest, Hungary, 20–22 April, pp. 91–103, http://utwpue.wb.utwente.nl/stw–doc/papers/paper–knowhs.ps

Salomons, O.W., Slooten, F. van, Houten, F.J.A.M. van, Kals, H.J.J. (1993) A computer support tool for redesign, a prototype system resulting from applying a methodic design approach, *IFIP Trans. B–11, International Conference on Engineering Design '93*, The Hague, The Netherlands, vol.3, pp. 1559–1570, http://utwpue.wb.utwente.nl/stw–doc/papers/paper–iced.ps

Salomons O.W., Houten F.J.A.M. van, Kals H.J.J. (1993) Review of research in feature–based design, Journal of Manufacturing Systems, Vol.12, no.2, 113–132.

Salomons, O.W., Slooten, F. van, Jonker, H.G., Houten, F.J.A.M. van, Kals, H.J.J. (1994a) Interactive Feature Definition, *Proceedings of the IFIP international conference on Feature Modelling and Recognition in Advanced CAD/CAM systems*, vol.1, pp. 181–202, http://utwpue.wb.utwente.nl/stw–doc/papers/paper–ifd.ps

Salomons, O.W., Slooten, F. van, Koning, G.W.F. de, Houten, F.J.A.M. van, Kals, H.J.J. (1994b) Conceptual Graphs in CAD, *CIRP Annals*, vol. 43, no. 1, pp. 125–128, http://utwpue.wb.utwente.nl/stw–doc/papers/paper–cirp94.ps

Salomons, O.W., (1995a), "Computer support in the design of mechanical products, constraint specification and satisfaction in feature based design for manufacturing", Ph.D. Thesis, University of Twente, Enschede (NL).

Salomons, O.W., Jonge Poerink, H.J., Slooten, F. van, Houten, F.J.A.M. van, and Kals, H.J.J., (1995b), "A computer aided tolerancing tool based on kinematic analogies", *Proceedings, CIRP seminar on computer aided tolerancing*, Tokyo, pp. 53–72 (and to be published by Chapman & Hall)

Salomons, O.W., Slooten, F. van, Houten, F.J.A.M. van, and Kals, H.J.J., (1995c), "Conceptual graphs in constraint based re–design", *Proceedings, ACM solid modeling symposium.*, Salt Lake City.

Salomons O.W., Slooten F. van, Franken F.J., Houten F.J.A.M. van, Kals H.J.J., (1995d), Design history functionality in FROOM, International Journal of CADCAM and Computer Graphics (Revue Internationale de CFAO et d' Informatique graphique: ISSN0298–0924), Vol 10, no. 1–2, special issue Actes de MICAD'95, Paris, pp. 95–111.

Serrano, D. (1987) Constraints in conceptual design, *Ph.D. thesis*, MIT

Sevenler K., Sherman M.K., Vidal R.W. (1993) Multidisciplinary teamwork in product design: Some requirements for computer systems. *International Conference on Engineering Design '93*, The Hague, The Netherlands, vol. 1, pp. 343–50

Silicon Graphics (1994) InPerson^{TM} Product Guide, Silicon Graphics Inc., Mountain View (Ca) USA

Shah, J.J., Urban, S.D., Raghupathy, S.P. and Rogers, M.T. (1992) Synergetic Design Systems, *Proceedings of the American Society of Mechanical Engineers Computers in Engineering Conference*, vol.1, pp.283–90

Shah, J.J., Rogers, M.T. and Urban, S.D. (1993a) Engineering data management: achieving integration through database technology, *Computing and Control Engineering Journal*, pp. 119–26

Shah J.J., Rogers M.T., (1993b) Assembly modeling as an extension of feature–based design, Research in Engineering Design, v. 5, 218–37.

Shah, Jami J. and Urban, Susan D. (1993c) Interoperability Research Issues for Heterogeneous Engineering Design Data, Proceedings of the National Science Foundation Design and Manufacturing Systems Grantees Workshop, pp. 1843–46

Shah J.J., Urban S.D. (1993d) Functional requirements for capturing design histories, proceedings National Science Foundation Design and Manufacturing Systems grantees workshop, 1831–36.

Shah J.J., Balakrishnan G. Rogers M.T., Urban S.D. (1994) Comparative study of procedural and declarative feature based geometric modelling, proc. IFIP int. Conference on Feature Modeling and Recognition in Advanced CAD/CAM systems, Valenciennes, Vol.2, 647–73

Sowa J.F. (1984) Conceptual Structures, information processing in mind and machine, Addison–Wesley Publishing Company.

Taleb–Bendiab, A., Oh, V., Sommerville, I., French, M.J. (1992), Collaborative Design: Knowledge–Based Systems for Concurrent Engineering. *ECAI'92, 10th European Conference on Artificial Intelligence, Workshop W14, Concurrent Engineering: Requirements for Knowledge–Based Design Support, Workshop Notes*, Vienna, Austria

Tang, J.C. (1991) Findings from Observational Studies of Collaborative Work, *International Journal of Man–Machine Studies*, Vol.34, pp. 143–60

Thornton, A.C. (1993) Constraint specification and satisfaction in embodiment design, *Ph.D. thesis*, University of Cambridge, England

Thornton, A.C., Johnson, A., (1993) Constraint specification and satisfaction in embodiment design, *International Conference on Engineering Design '93*, The Hague, The Netherlands, vol.3, pp. 1319–26

Toye, G., Cutkosky, M.R., Leifer, L.J., Tenenbaum, J.M., Glicksman, J. (1993) SHARE, A Methodology and Environment for Collaborative Product Development, *Post–Proceedings of the IEEE Infrastructure for Collaborative Enterprises (CDR–TR #19930507)*, http://gummo.stanford.edu/pub/CDR/Publications/Reports/Share.ps

Urban S.D., Shah J.J. (1992) Interoperability research issues for heterogeneous engineering design data, proceedings National Science Foundation Design and Manufacturing Systems grantees workshop, 1843–46.

Urban S.D., Shah J.J., Rogers M.T. (1993) Engineering data management: achieving integration through database technology, Computing & Control Engineering Journal, 119–26.

Vin L.J. de Vries J. de, Streppel A.H., Kals H.J.J. (1993) PART–S, a CAPP system for small batch manufacturing of Sheet metal components, Manufacturing Systems, Vol. 22, No. 2, 133–41.

Vries J. de, Salomons O.W., Streppel A.H., Vin L.J. de, Kals H.J.J. (1994) CAD–CAPP integration for sheet metal products, proceedings SheMet.

Watabe K., Sakata S., Maeno K., Fukuoka H., and Ohmori T. (1990) Distributed Multiparty Desktop Conferencing System: MERMAID. *Proceedings of the Conference on Computer Supported Cooperative Work '90*, Los Angeles, USA, pp. 27–38

Wong, A., Sriram, D. (1993) SHARED: An Information Model for Cooperative Product Development, *Research in Engineering Design*, 1993, vol. 5 no. 1, pp. 21–39

APPENDIX; SOLVING PARAMETER CONSTRAINTS USING SIMULATED ANNEALING

Simulated annealing

In nature, when a metal liquid is heated up to a high temperature, atoms move freely with respect to one another. They are in a so–called high energy state. If the temperature is lowered, the liquid cools and the thermal mobility of the atoms decreases. Finally, the energy state becomes low enough to let the liquid freeze. Then the atoms are often able to line themselves up and form a pure crystal that is completely ordered over a distance up to a billion times the size of an individual atom in all directions (Press et al. 1992). However, if the liquid is cooled too fast, it does not reach the crystalline state but ends up in a polycrystalline state with only meta stable, locally optimal structures. Therefore, the very essence of the process is slow cooling, which gives the energy state of the liquid enough time to reach an equilibrium.

In order to reach that equilibrium, the energy state sometimes increases in order to leave a local minimum. Metropolis et al. introduced a simple algorithm that can be used to provide an efficient simulation of a collection of atoms in an equilibrium at a given temperature (Kirkpatrick et al. 1983). In each step of this algorithm, an atom is given a small perturbation and the resulting change in energy, ΔE, is calculated. If $\Delta E \leq 0$, the change of the configuration is accepted and this changed configuration is used as the starting point for the next step. If $\Delta E > 0$, the probability that the increase in energy of the configuration is accepted, is:

$$P(\Delta E) = \exp\frac{-\Delta E}{kT} \quad . \tag{1}$$

with: k = Boltzmann's constant
T = temperature

Random numbers uniformly distributed in the interval [0,1] are used to implement the random part of the algorithm. One such number is selected and compared with $P(\Delta E)$. If it is less than $P(\Delta E)$, the new configuration is accepted. Otherwise, the original configuration is used to start the next step. It may be obvious that the higher the temperature, the higher the probability that an uphill climb is accepted. This general scheme of always taking a downhill step while sometimes taking an uphill step, is called the Metropolis algorithm (Press et al. 1992) or Metropolis criterion (Kirkpatrick et al. 1983). The process of simulated annealing can be visualized by a ball on a hilly surface. The ball represents the current energy state of the configuration where the surface represents the energy function (Thornton 1993), (Thornton & Johnson 1993). This is shown in figure 7.

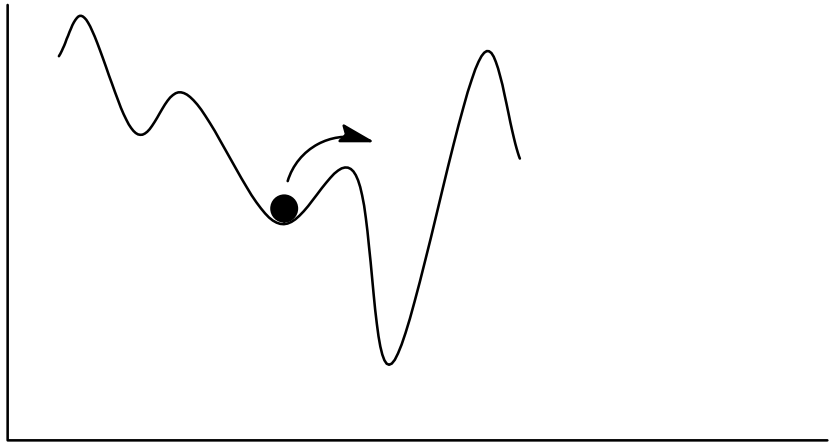

Figure 7 Simulated annealing as an optimizing algorithm capable of finding the absolute minimum where the ball represents the current energy state and the hills represent the energy function (redrawn after (Thornton 1993)).

Simulated Annealing will always find a global minimum, as long as the system is cooled "slowly" enough. This means that the amount of iterations will have to be large enough to reach that global minimum. In practice however, there is a trade–off between finding the global minimum exactly or a point near the global minimum, depending on the time one is prepared to wait until a global solution has been reached. Therefore, it may be more efficient to apply a stopping criterion after a certain number of iterations with only minor changes of the energy function or even without any improvements of the energy function.

Compared with conventional, iterative search techniques, like for instance Newton–Rhapson, simulated annealing provides several advantages. SA is robust and this is a major advantage with respect to conventional, iterative search techniques, as these will need a smooth function without any discontinuities. Even if numerical approximation of derivatives is allowed, this is a severe shortcoming, because it is still not guaranteed that each discontinuity will be solved. Besides, the usage of numerical approximation of derivatives will lead to more calculation time needed. In order to find the local best solution, conventional search techniques will examine the function in the steepest permissible direction (hill climbing). This implies that the method is local in scope. The optimum that is found, is the best in a neighborhood of the current point where the search was started. Once a local extremum is found, further improvement must be found through random restart. This means that random walks and random schemes that search and save the best

solution have to be added to the calculus based techniques. This will have a negative effect on the efficiency of the conventional search techniques. Besides, even if a lot of random iterations are started, it is still not guaranteed that a global extremum (usually a minimum) is found.

Parameter satisfaction using simulated annealing

The energy function of SANCS is based on the errors that appear in the constraints during evaluation and satisfaction. Generally, it can be said that a parameter constraint appears in two forms:

$$F_1(x) < F_2(x) \quad . \tag{2}$$

$$F_3(x) = F_4(x) \quad . \tag{3}$$

Because the magnitudes of the errors may be different, a normalized error measure has to be used (Thornton 1993):

$$d_i = \max\left(0, \frac{F_1(x) - F_2(x)}{\frac{|F_1(x) + F_2(x)|}{2}}\right) \quad . \tag{4}$$

$$d_i = \left|\frac{F_3(x) - F_4(x)}{\left|\frac{F_3(x) + F_4(x)}{2}\right|}\right| \quad . \tag{5}$$

Where equation (4) is the normalized error belonging to equation (2) and equation (5) belongs to equation (3). It is reasonable to assume that simulated annealing will never reach an error exactly equivalent to zero. For after a certain number of iterations SA is stopped (stopping criterion). Therefore, a lower bound ε is proposed for d_i: below this lower bound ε, the constraint is assumed to be solved. Then, d_i is taken to be zero. A value of ε=0.001 is used and gives the appropriate accuracy in the equality constraints.

The energy function can be calculated with the normalized error d_i. The constraint violation function for constraint i is equivalent to the square of the normalized error (Thornton 1993):

$$C_i = d_i^2 \quad . \tag{6}$$

The total constraint violation function, which is the energy function, can be written as:

$$E = \psi = \sum_{i=1}^{n_c} C_i \quad . \tag{7}$$

where n_c is the number of constraints.

SANCS has a constraint network built from variables and constraints where the nodes of the network contain the variables and the links of the network contain the constraints. In SANCS, each link represents one constraint and connects all the variables in that constraint (Thornton 1993). The final constraint network is shown in figure 8.

Before constraints are satisfied, as many equality constraints as possible are removed from the constraint set. Satisfaction is performed by the use of a generation mechanism and is divided into two steps: node selection and node variation. This generation mechanism, the heart of SANCS, is the process by which a neighbor of a current search point is selected (node selection). In order to obtain a transition from one point in the search space to another, the mechanism gives the value of one variable a small perturbation (node variation). The problem is the selection of the variable (node) whose value will be varied. The simplest solution is to make random selections. Another solution, is to weight the node selection such that some nodes are more likely to be chosen than others. This would mean focusing on problem areas i.e. where constraint conflicts occur. Stochastic sampling can provide for this. The node selection algorithm then assigns a weighting to each node based on the constraints related to the node (variable) and has four steps.

First, the error associated with each constraint is calculated (d_i) and with it the constraint violation function (C_i). Second, the constraint violation function is divided by the number of

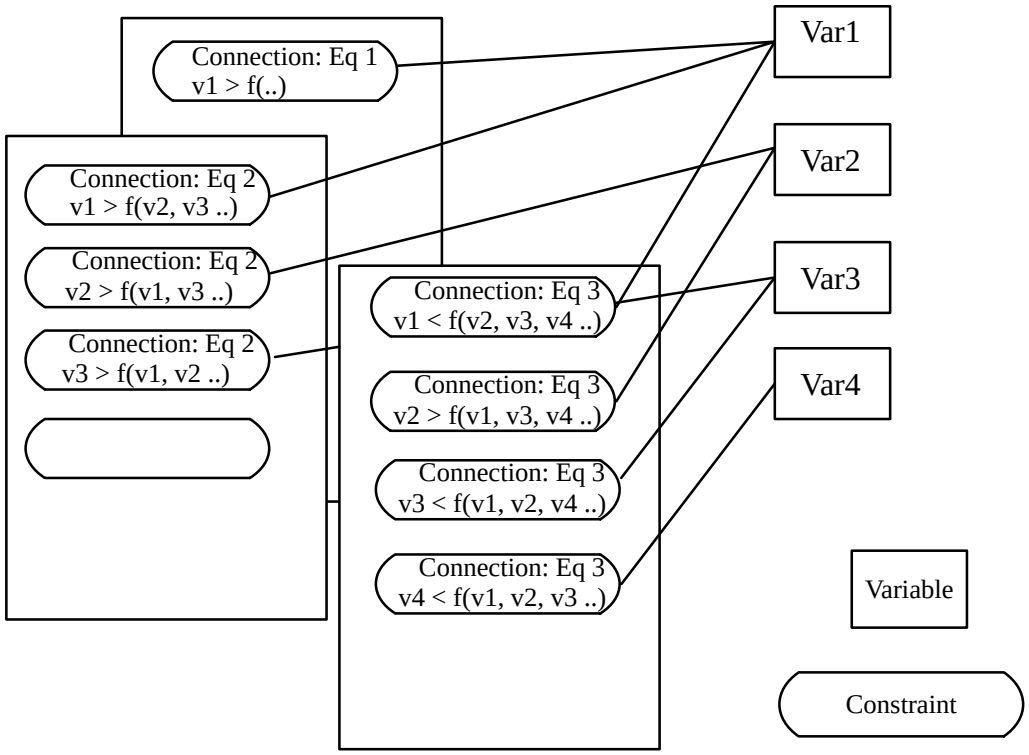

Figure 8 The constraint network (redrawn after (Thornton 1993)).

variables in the constraint (n) and each variable in that constraint is assigned a proportion of it ($C_{var} = C_i/n$). Third, the errors assigned to each variable are summed ($C_{vartot} = \Sigma C_{var}$). Fourth, the roulette wheel selection algorithm (Goldberg 1989) is used to select the next node. The energy of the constraint network is calculated by adding all the total errors assigned to each individual variable ($E = \Sigma C_{vartot}$)

After selection of the variable's new value, the change in energy, due to the change in the variable's value, is calculated. Then the decision whether to keep this change is made. For this decision, a slightly altered Metropolis criterion is used. The probability $P(\Delta E)$ of the new value being kept is calculated by (Thornton 1993):

$$\Delta E > 0, P(\Delta E) = 1. \quad (8)$$

$$\Delta E < 0, P(\Delta E) = \frac{2}{1 + \exp^{-\frac{\Delta E}{T}}}. \quad (9)$$

ΔE is calculated here such that it has a negative value if the energy in the new point is greater than the old. Obviously, the value of $\Delta E/T$ is not scaled by the Boltzmann constant. This scaling is not needed if an appropriate starting temperature is chosen. The combination (kT) of equation (1) determines the probability whether the increase of the energy is accepted. The higher this combination, the higher the probability that an increase of the energy of the configuration will be accepted. Choosing an appropriate value for T will remove the need for keeping Boltzmann's constant. Thornton therefore uses Boltzmann's constant with a value equal to 1. The annealing schedule can be broken into four parts: the initial value of the temperature, the final value of the temperature, the Markov chain length i.e. the number of iterations at the same value of the temperature, and the method of decreasing the temperature.

The initial temperature T_o determines the number of iterations. In order to avoid preliminary convergence to a local minimum, the initial temperature should be high enough. However, if T_o

would be chosen too high, the search might be unnecessarily long. An algorithm is provided by SANCS in order to choose a good starting temperature (Thornton 1993).

Because the aim is at acceptable design, the algorithm is always stopped when the constraints are satisfied, i.e. constraint conflicts are solved for a certain set of parameters. However, the SANCS algorithm will not always find a solution either because none exists or because it converges to a local minimum which means that an alternative stopping criterion is required for these two cases. Two criteria are implemented in the SANCS algorithm. The first criterion is to stop SANCS when a given number of iterations occur without an improvement in the current solution. For this, the Markov Chain Length is used. The other criterion is to stop when the temperature becomes too low, i.e. below T_{min}. Finally, the rule for the decrement of temperature is given by the following equation:

$$T_{k+1} = \alpha T_k, k = 0, 1, 2, .., \quad (10)$$

where α is less than but close to 1.

Discussion of implementation of SA–based constraint solver in FROOM

During implementation of the parameter constraint solver some problems occurred. Thornton gives rules for the rearrangement of constraints in order to build a constraint network as shown in figure 8. However, these rules are not applicable in every situation (Thornton 1993). These exceptions were not implemented and evaluated in the experiments by Thornton. An example of such an exception is the following equation, which was already identified by Thornton:

$$C < (A + D) * (A + B).$$

Suppose we want to rearrange this equation to an equation of the term A > ... It can easily be seen that A cannot be solved. This means that for such situations an alternative has to be provided. This can be done by the introduction of a new variable. The procedure for this introduction is to first replace (A+B) by Q. The equation can now be rearranged to: A > (C/Q) – D. Finally, add a new constraint: A = (Q–B).

A sophisticated constraint solving algorithm should provide for this. However, it may be difficult to program such an algorithm that recognizes the situation mentioned above and provides solutions for it at the same time. For the time being, this task may be directed to the end–user who then could use the constraint editor in order to overcome such an incapability of the constraint solver. At the moment the introduction of new variables is done by hand.

Rearrangement may lead to another additional, preventive measure in the source code. Take the following example: $A = (D^2+K)$. Solving for D gives:

$$D = \sqrt{(A - K)}.$$

This means that an additional requirement has to be solved: A>K. However, it was previously stated that during the satisfaction of the constraints, normalized aberrations (di) are allowed. This means that parameter K can obtain (temporarily) a higher value than parameter A. The equation is not solvable as square–roots of negative values are not allowed. This problem has been eliminated by the use of the following procedure:

- Assign to the constraint the maximum of the normalized error, i.e. di=2.
- The sum of normalized errors of each variable of the constraint will be increased with 2/m, with m the number of variables of that constraint.

For this equation it would mean that the sum of normalized errors for both A and K would be increased with one.

BIOGRAPHY

O.W. Salomons obtained a PhD degree on his work on the FROOM system at the University of Twente. He obtained his MS degree in Mechanical Engineering in 1990 at the same university. Presently, as an assistant professor, he is performing research at the laboratory in the field of design support systems as well as their link with process planning systems.

J.M Kuipers recently obtained his MS degree in mechanical engineering at the University of Twente. His MS work was on the collaborative product development module of the FROOM system. Currently he is fulfilling his civil service.

J. de Graaff recently obtained his BS degree in Computer Science. His BS work was on the parameter constraint solver of the FROOM system. Currently he is fulfilling his civil service.

F. van Slooten holds a BS degree in Computer Science, which he obtained in 1990. Since 1991 he works as a system analyst/ technical software developer at the laboratory of Production and Design Engineering of the University of Twente.

F.J.A.M. van Houten is associate professor of the laboratory of Production and Design Engineering at the University of Twente. Professor Van Houten obtained his MS degree in Mechanical Engineering at the Technical University of Eindhoven in 1977. He has been working at the laboratory of Production Engineering at University of Twente since 1978. Van Houten has worked in the field of CAD and CAPP; he has been closely involved with the development of several CAPP systems. In 1991 he obtained a PhD degree on his work on the PART system. He is member of CIRP and IFIP WG 5.3.

H.J.J. Kals is professor and head of the laboratory of Production Engineering at the University of Twente. He is also part–time professor at the Technical University of Delft. Professor Kals obtained his MS degree in Mechanical Engineering in 1969 at the Technical University of Eindhoven. In 1972 he obtained his PhD degree. In 1977 he became professor of the Laboratory of Production & Design Engineering at the University of Twente. Professor Kals is a member of CIRP. He is active in the fields of CAD, CAPP, CAM, workshop– and work station control. Currently one of his main scientific interests is concurrent engineering.