INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 03 STOCKHOLM, AUGUST 19-21, 2003

SOLID MODELLING WITH DIMENSIONAL AND TOPOLOGICAL VARIABILITY

M Cederfeldt and S Sunnersjö

Abstract

With the advent of commercial solid modelling systems some fifteen years ago the opportunity for three-dimensional parametric geometry was opened to industry. Today solid modelling systems are the dominating CAD tool among advanced engineering companies, but despite the time and money saving implications, industry has been slow to exploit the parametric capabilities of these systems [1].

One reason for the slow adoption of parametric modelling is that originally many solid modellers suffered from lack of stability under parametric changes. This situation is now changed and if a model in a modern CAD system collapses, this is usually due to modelling deficiencies rather than numerical failures. Straightforward dimensional variations rarely cause any problems, but to fully exploit the parametric capability for complex features with a variable topology, there is a need for a systematic approach to build stable and purposeful parametric models.

The purpose of this work is to discuss how different modelling approaches relate to the ease of use and robustness of the CAD model in terms of creating variants and product families. We use the term *Design for Variability, DFV,* for a modelling approach that ensures that parametric models are well suited for variation design.

Keywords: Feature-based design, solid modelling, parametric modelling, variability.

1. Introduction

It is a common experience that up to 80 percent of the work carried out in a design office is concerned with redesign of existing products. If some of this work related to mature products could be automated the designers would have more time for creative problem solving associated with refining existing products or designing new ones. Design automation can be done at several levels of complexity ranging from the use of predefined machine elements or family table/template systems [2], to highly sophisticated *Knowledge Based Engineering* systems [3] or *Knowledge Intensive CAD systems* (KIC) [4].

Not least for smaller companies, where budget constraints limit the implementation of large and costly systems, implementation of a complex KBE system is often unrealistic. With the stable solid modellers available on the market today, much of the desired automation of variant design of mature products can be done within the solid modeller itself or in combination with external programs for calculations and optimisation (compare *component technology* [5] and [6]).

For such systems there is always a difficult balance between writing applications in the CAD systems' internal macro-language or constraint system, and using external programs to feed the CAD system with instructions and information. In our work we give high priority to the transparency of the CAD geometry generation and to the design process as a whole by extracting as much information from the CAD system as possible and putting this information into an external, readable document, primarily using the CAD software as a geometry creation tool (Figure 1). The reason for not controlling a CAD kernel directly using specific application programming, which in many ways is an efficient method, is to avoid programming associated with the specific software. This also opens the possibility of changing CAD system vendor, without extensive rewriting and programming. It also gives the designer the opportunity to work with the solid modeller (CAD system), that he/she normally uses, thereby simplifying any manual additions or modifications to the final geometry that might be required.

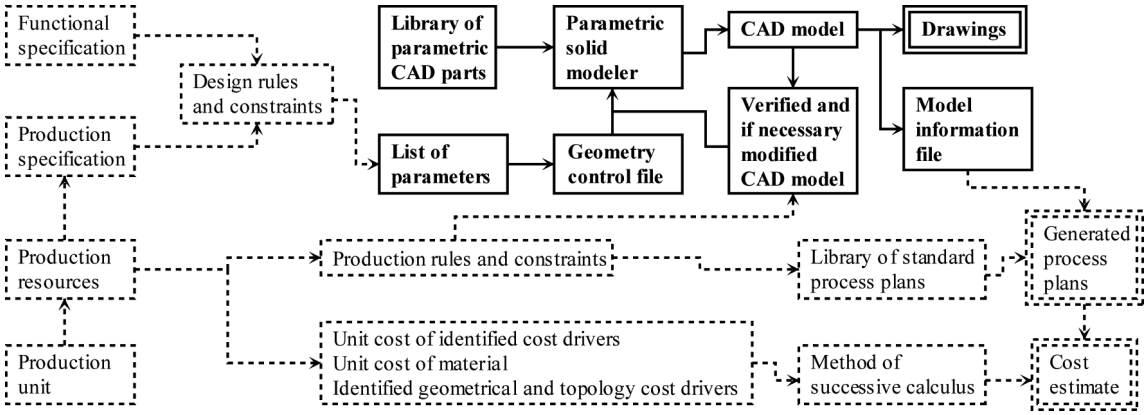

Figure 1. – Information flow in the proposed system assembled from several commercial software packages. The purpose of the system is design documentation (drawings), process plans and a cost estimation of a complete, calculated and optimised product. This paper discusses the solid outlined components of the system.

For this approach to work in an industrial environment, it requires robust and changeable parametric models. In this work different modelling strategies are discussed and a checklist to be used for evaluating these methods for different types of applications is presented.

2. Application example

As a case of application, order-based design of a submarine bulkhead (Figure 2) and its vertical structural members (Figure 3) has been studied and implemented. The structural members consist of cut, rolled and welded steel plating.

The external program, which generates input to the CAD system, should be chosen according to the specific design problem at hand [7]. The bulkhead design problem can be solved with straightforward procedural programming like C, Visual Basic or similar. Programmable spreadsheets would be another alternative. To improve transparency, an algorithmic program with a text editor, Mathcad (by MathSoft), was selected. By creating an easy-to-read document containing design rules, explanations and calculations of parameters, a set of parameter values, independent of any specific solid modeller, can be generated. These parameter values are in a sense the product geometry described as feature and parameter names and numbers. One could then create a parametric solid model in a specific CAD system governed by this set of parameters.

The structural members are modelled in Pro/Engineer (by PTC) and the parameters are defined in a database. Pro/Engineer was chosen as a program with functionality typical for modern solid modellers and hence providing a suitable test environment.

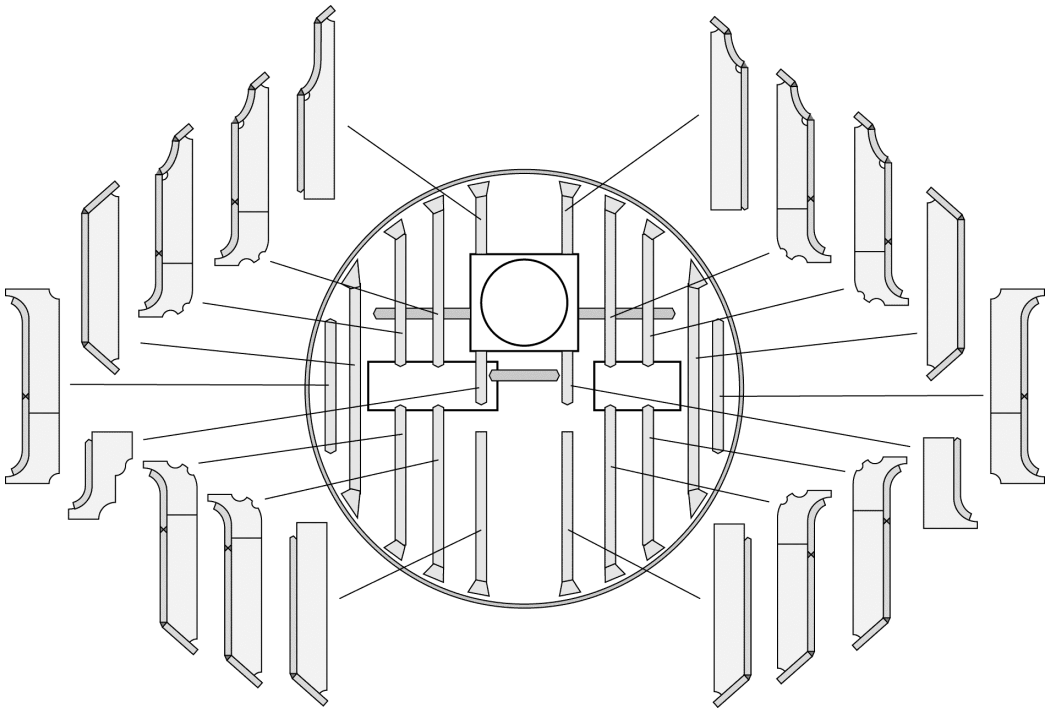

Figure 2. –Variations of structural members of a submarine bulkhead.

The model creation can be done in many ways and still result in a parametric model. The challenge is to create a geometry, which best suits the current application, is time and cost efficient, flexible and assures that the geometry is completely parametric. Furthermore, the generated CAD geometry might be used downstream the engineering process, for example in a CAE and/or CAPP system, and contribute parameters and operations order for process planning and cost estimates [8].

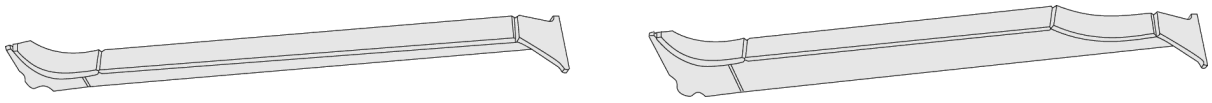

Figure 3. – Two variations of a structural member of a submarine bulkhead.

3. Methods for solid model creation

The discussion below is based on the example with the bulkhead stiffeners, but is expected to be of generic applicability. For modelling other types of products, such as forged, cast and/or machined components, these approaches may need to be revised or further developed. Three generic methods were identified to model the stiffener parts with the goal of creating parametric models for easy reconfiguration, (compare [9] and [1]).

The methods differ in complexity with respect to lead-time (time required to complete modelling and programming of parameter relations), transparency (the ability for the user to understand what the model and its parameter relations do) and programmable internal and external relations (complexity of the relations between geometric variables). These methods

were categorized in order to determine the essential differences between them and for which type of components they are most suitable (Figure 4):

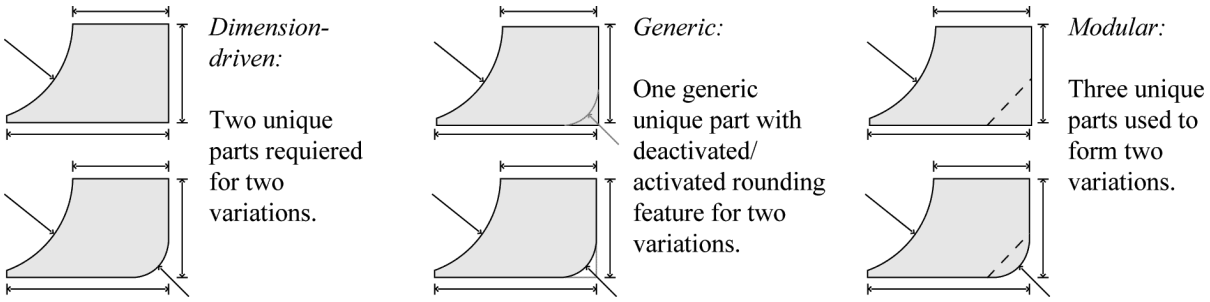

Figure 4. – The three modelling methods applied to a part with two variations.

3.1. Dimension-driven part

This is the simplest way of creating parametric models for automatic regeneration. The designer models the geometry and assigns to it variables that he/she identifies as design parameters (for example length or height). These parameter values can later be altered to generate a different design variation.

This solid modelling method is in this paper named *Dimension-driven* since only the geometry dimension values vary and the geometry topology remains unchanged. With this approach a family table or library of parametric models, which share the same governing parameters, can be created for designs with different topology (i.e. added or subtracted features).

Regarding assigning or programming relations between the geometry parameters, this is a relatively simple method. Since topology remains unchanged within the individual geometry model the relations between the parameters also remain unchanged when regenerating the geometry. In some cases, relations to other parts (models) can be warranted and these relations can be programmed outside the solid modeller or as modeller-specific relations such as assembly relations.

3.2. Generic part with variable topology

A more complex method of parametric modelling is to allow topology changes for variant creation of a generic geometry model. In this paper the method is called *Generic*. It is similar to the *dimension-driven* geometry model in that it allows for variation of parameter values on regeneration. Added to this is the functionality of changing geometry topology by activating or deactivating features (previously modelled and stored in the generic part). The generic geometry model can regenerate into several design variations, and the advantage over family tables is the added transparency gained by extracting the information about topology configuration and controlling it in the same way as the design parameters are controlled and changed.

The programming of the parameter relations for these parts is more complex than for the *dimension-driven* parts due to the additional feature dimensions and increased internal relations. Also some geometry parameters may have relations to different parameters depending on the model's current topology. This leads to problems, and possibly an unstable model, if feature activation/deactivation is applied to features with dependent features or parts. The method should therefore be used with care if any feature has children (sub-features) or if other parts (or any of its features) are referenced to the feature in an assembly.

The *generic* parts are controlled/regenerated in the same way as the *dimension-driven* parts, with the addition of parameters for topology control.

3.3. Modular parts

Complex parts, which in real life are considered as a single solid object, like an engine block, may be divided into smaller geometry parts when modelled to make design easier or to distribute the modelling work to several designers. These building blocks are retrieved from a library and parametrically sized and merged as required via Boolean operations.

It is possible to combine both *dimension-driven* and *generic* parts for complex models, but in this paper the approach is limited to combination of simple (non-complex) building blocks, which do not change in feature composition (topology).

This is a modelling method, which results in parts similar to the *generic* parts but without the programming of feature activation/deactivation. Relations between the building blocks can be programmed outside the solid modeller or as modeller-specific relations such as assemble relations. This method is called *Modular*.

4. Assembly combination

In reality almost all geometry models (parts) are assembled into a finished product. This is done by the solid modeller's assembly function and relations for part placement can be described outside the modeller and fed as governing parameters to the individual parts or as internal relations in the assembly file. The *Assembly combination* consists of a combination of the parts created in any of the above described ways, and can be controlled at part level or at assembly level depending on the level of programming that is acceptable for the current application.

The assembly method is not a modelling method in the same sense as the three described above, but any large product design system will need assembled parts and it is of relevance in what way the models function in an assembly and how the assembly is controlled within the system. Therefore the designer must take assembly into consideration when choosing the model approach of the single parts.

One way of adding freedom of choosing modelling method is to build the assembly on a skeleton model, which limits the relations between the individual parts. Another way is the assembly of the individual parts with references only to fixed systems such as datum planes or coordinate systems. This can be done if all positioning information is built in as positioning parameters in the individual parts. This method requires references between the different parts at some higher level. In the proposed system this is handled through references between the Matcad documents governing the individual parts. The assembly can then be created using a *trail*-file, which controls the CAD software. A trail-file is a CAD software-specific text file, which describes the operations in the programming language of the solid modeller.

5. Identifying models, features and parameters

To achieve high transparency in the geometric model, 3D-notes can be added to the model features and parameters and give visual aid to the users [10]. To further extend the transparency, the entities (parts, features, points, datum features and design variables) must be easily identified when working with the model and its relation programming.

The names of the different entities of a solid model govern the way the user is able to access them, and when creating, editing, reusing or programming a solid model the designer must be able to identify particular items within the individual design spaces. Names can either be assigned automatically by the system or be created manually for each new entity [11]. To have the unique names automatically assigned will result in names, which do not describe the properties of the entity and therefore complicate the further use of the solid model. To manually assign the names is more time-consuming, but will result in a model containing meaningful entity names suited for further use of the model, such as programming of relations. For the system developer as well as the user of the models, the transparency will increase if the names are chosen in such a way that they meaningfully describe the entities to which they are assigned. This process can be automated with the help of a database and a rule base for name creation.

6. Comparing modelling methods

The underlying purpose of parametric modelling is to achieve a flexible design at an early design stage and to allow for reuse of existing design solutions with appropriate adaptations to new specifications. The library of parametric models is the company storage of off-the-shelf design solutions and forms a design system for rapid production of customized design solutions. In this respect the design system strongly resembles a manufacturing system and it therefore seems natural to adopt the traditional manufacturing criteria when evaluating different approaches to parametric modelling. This DFM approach will be discussed in detail below.

6.1. A checklist for parametric modelling

Since component complexity varies with model level, the different modelling techniques must be evaluated at several levels, i.e. a modelling approach, which seems to be acceptable at part or component level, may not be acceptable once assembly level is reached. Therefore the modelling criteria can be applied to the following four levels (Figure 5):

1. System level – *Relations between different* assemblies *(subassemblies).*
2. Family level – *Relations between the different* variants *of a specific design.*
3. Assembly level – *Relations between the different* parts *included in an assembly*.
4. Component/Part level – *Relations between part-specific* parameters.

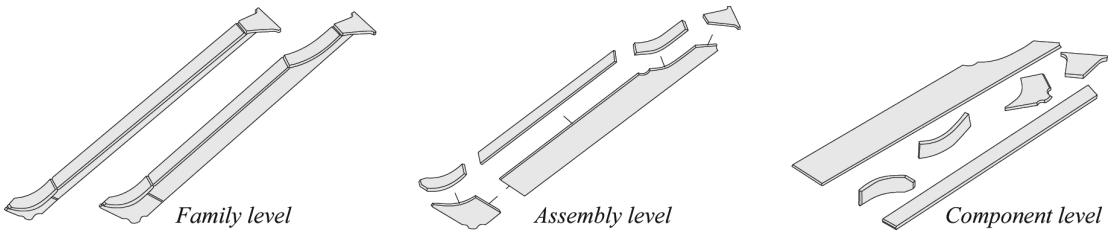

Figure 5. – Different product levels.

In this paper a comparison of modelling methods will be made primarily for the component level, with additional remarks on how the modelling methods differ at assembly, family and system level when there are obvious advantages or disadvantages of the methods.

Fabricius *et al* [12] discuss these levels and define a set of "universal criteria" for "good manufacturability" (DFM), which, with an adaptation for product solid modelling, can be listed and described as the following checklist:

1. Quality – *Completeness:* Does the modelling method allow for a complete geometric model with all desired variations or are there some restrictions (at part, assembly and/or family level)? *Transparency:* What is the level of transparency for the modelling method? Will an end user understand the intent of the model and its variations? *Accuracy of models:* How does the model perform with other models at assembly level and is the level of accuracy sufficient? Are there restrictions on accuracy and are these numerical, reference or programmatic?
2. Flexibility – *Ease of adaptation to new variations:* Does the modelling method allow for easy reconfiguration of the model? Is there a desired variation, which is too complex and requires another approach? Does *variability* affect the model completeness at a higher level (assembly level)? *Scalability:* The model is sufficient and work well in a prototype system, but will it work in a more complex system? Are there any complications that could arise from scaling the system (adding features to a model or adding parts to an assembly)?
3. Risk – *Vendor dependence:* Is the system critically dependent on outside vendors over which the user has no control? Is there a risk that the system will have to be prematurely abandoned or extensively rewritten because of unforeseen changes in vendor software, products being discontinued, vendors closed down or similar? *Unexpected malfunctions of models:* Is the model tested within the desirable range of reconfiguration? Are the parameter relations written in such a way that they allow for all desired *variability* without geometry failure? Will the model perform in the same way both at component level and assembly level?
4. Lead-time – *Time to create parametric models and program relations:* Modelling method, complexity of the geometry and number of variations will effect the time required for creating the geometry and programming the relations for the desired *variability*. The time required to create a basic dimension-driven model with only a few variations is probably short, while a complex generic model take longer to create and program.
5. Resource conservation – *Direct and indirect cost. Use of capital and human resources:* The cost depends on time to create and program the models as well as on the modelling method and system performance. An example of this is a complex generic geometry, which takes a long time to create and program but in return has low system requirements (low memory and CPU usage) and regenerates fast. *Beneficial effects on designers' working conditions:* By creating a system, which performs routine redesign automatically the designer is free to perform more creative work associated with designing new products.

When looking at the individual parts of a larger system, points 1 through 4 are the most interesting, but when evaluating the system as a whole, point 5 becomes essential.

6.2. Comparison between two different parts of an assembly

As an example two individual parts of the vertical structural member, the knee web and the main web (Figure 6), are discussed with respect to points 1 through 4.

To describe the differences between the three main modelling methods previously discussed and to illustrate the direct effects of choice of method, a first comparison can be made by

looking at the number of desired design variations, and the number of parts needed for each method. The *dimension-driven* method for the simplified knee web part (left in Figure 6) requires six parts for six variations. The *generic* method would require one part for creating six variations and the *modular* method would require seven parts for creating six variations.

If only considering the modelling at part level the main difference in methods is the complexity of the relations between design parameters, where the *generic* method is the most complex since feature activation/deactivation has to be programmed. From an automation point of view and at a higher level (assembly, family and system), the method that requires the least additional programming (a model, which is *complete* and *flexible*) and has the highest *transparency* and *vendor independence,* is to strive for.

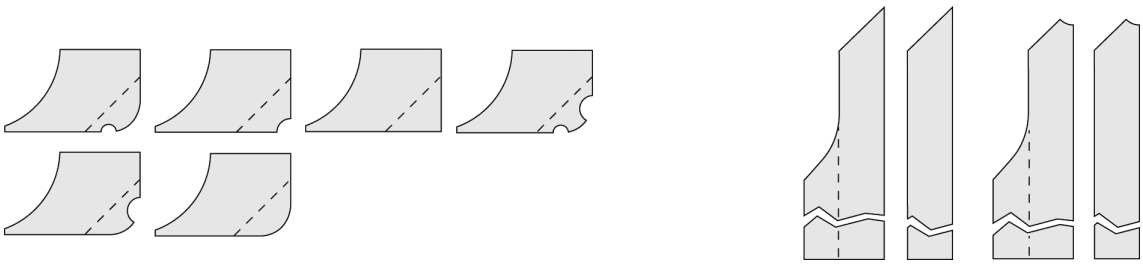

Figure 6. – Different configurations of the knee web (left) and the main web (right). The dotted line symbolizes the dividing line between the modular parts.

For the simplified knee web part the activated/deactivated features have no direct effect on assembly level since none of the features is used as a reference for other assembly parts or requires a change of assembly topology when changing its own topology. Therefore the *generic* method is recommended on the basis that the small variations of the geometry are achievable with relatively easy programming at part level.

For the main web part (right in Figure 6) the *generic* method is also, at part level, the most favourable method, but the activation/deactivation of the transition feature has a substantial effect at assembly level where a change of part topology forces the assembled stiffener to change its topology by adding an additional flange when the transition feature is activated. This adds to uncertainty in model regeneration at assembly level, and *risk* of unexpected malfunctions and loss of *accuracy* of the assembled model. Also considering the additional programming needed at assembly level and the complexity of the relation between parts in the stiffener assembly, the *generic* is unsuitable. The *modular* method suffers from the same problems as the *generic* when consideration is taken to assembly programming and references. The *dimension-driven* method is reliable, but not optimal since it requires a unique part for each variation.

Another approach for this part and in this particular assembly is to combine the *generic* and the *dimension-driven* methods. This approach gives four variations with two parts with the (independent) notch feature activated/deactivated with the *generic* method. At assembly level this method would then require two different family variations of the stiffener assembly.

For the complete bulkhead system we have decided to model the family of stiffeners by assembling parts, which are created with the generic method for cases where dependent features can be avoided, and otherwise created as dimension-driven parts. The complete stiffeners are then assembled to the bulkhead with a short trail file. This approach is a balanced compromise between the criteria discussed above, resulting in a modest amount of programming and vendor dependence and at the same time providing a transparent and reliable model.

7. Conclusion

One of the most important features is the operational life and transparency of the system. A part modelled by the dimension-driven method has the most built-in transparency since the model itself is very simple. For parts modelled by the generic method, the built-in transparency is less than for the dimension-driven models since the changes in topology can be difficult to control. Therefore all essential information of the model and its design parameters are stored in an external file (Mathcad document), which acts as documentation of the design and as a computational tool for creating variations. This gives the models a high transparency at a system level.

A model that works well at part level may not work in an assembly. Reasons for this could be that features, which are used as reference for other parts, may be deactivated or changed in a way that results in model collapse when using a generic model. Although from a system point of view, a generic model is the easiest to handle and control, the added difficulties of programming at assembly level and the risk of model instability for very complex models may warrant the use of a dimension-driven model instead. The problem can also be solved by assigning references so that individual parts are not referenced to each other but to some fixed system like datum planes or coordinate systems.

Model accuracy is important at assembly level where the different parts have to fit together without the risk of model failure due to mismatching parts. Because of this the individual model accuracy is important, and depends on how the model is created and how the design parameters are assigned. The numerical accuracy of the models depends on the CAD system, but also, and to a greater extent, on the accuracy of the externally calculated design variables. It is also important that the relations between parts are kept accurate throughout the range of variations.

The checklist questions can be answered if the model is tested in the desired range of reconfiguration and by having good knowledge of the actual product, the solid modeller and by knowing the consequences of changing design parameters.

The conclusion is that with consideration only to one level, there is a risk of choosing an unsuitable modelling approach. It is therefore important, when using the proposed modelling methods and checklist for comparison, that consideration is given to the complete product.

Acknowledgement

This work was conducted within the Vinnova program for “Information Technology in the Manufacturing Industry” and financial support is gratefully acknowledged. The authors would also like to express their gratitude to Kockums AB for information on application cases and helpful discussions.

References

[1] Amen R. and Sunnersjö S., "Solidhandboken" (in Swedish), Sveriges Verkstadsindustrier, Faktarapport Produktion Konstruktion, Vol. 4, 1996. ISSN 1103-7067.

[2] Siddique Z. and Yanjiang Z., "Automatic generation of product family member CAD models supported by a platform using a template approach" Proceedings of DETC2002: ASME Design Engineering Technical Conferences, September 29-October 2 2002, Montreal, Canada. DETC2002/CIE-34407.

[3] Sriram R. D., "Intelligent systems for engineering: a knowledge-based approach", Springer-Verlag Berlin Heidelberg New York, 1997. ISBN 3-540-76128-4.

[4] Finger S. et al, "Knowledge Intensive Computer Aided Design" Proceedings of IFIP TC WG5.2 Third Workshop on Knowledge Intensive CAD, December 1-4 1998, Tokyo, Japan. ISBN 0-7923-8694-9.

[5] Flores R., Jensen C. G. and Shelley J., "A web enabled process for accessing customized parametric designs", Proceedings of DETC2002: ASME Design Engineering Technical Conferences, September 10-13 2002, Baltimore, Maryland. DETC2000/DAC-14275.

[6] Tucker S. S. et al, "Parametric Engineering Design Tools and Applications", Proceedings of DETC2000: ASME Design Engineering Technical Conferences, September 10-13 2000, Sacramento, California. DETC1997/EIM-3718.

[7] Amen R., Rask R. and Sunnersjö S., "Matching design tasks to knowledge-based software tools – When intuition does not suffice", Proceedings of DETC99: ASME Design Engineering Technical Conferences, September 12-15 1999, Las Vegas, Nevada. DETC99/DAC-8688.

[8] Elgh F., Sunnersjö S., "An automated cost estimating system for variant design based on the method of successive calculus", Proceedings of ICED 03: International Conference on Engineering Design, August 19-21, Stockholm, Sweden.

[9] Shah J. and Mantyla M., "Parametric and Feature-Based CAD/CAM", John Wiley, New York, 1995. ISBN 0-471-00214-3.

[10] Grote K.-H. and Schumann S., "Methods of 3D-modelling with parametric cad-systems of the newest generation", Proceedings of DETC97: ASME Design Engineering Technical Conferences, September 14-17 1997, Sacramento, California. DETC1997/EIM-3718.

[11] Krause F.-L. et al, "Implementation of Technical Rules in a Feature Based Modeller", Proceedings of Second Eurographics Workshop on Intelligent CAD Systems – Implementational Issues, Apri19-23 1988, Veldhoven, Nederlands. Springer-Verlag Berlin Heidelberg New York. ISBN 3-540-50914-3.

[12] Fabricius F. et al, "Design for Manufacture – DFM." Institute for Product Development 1994, Lyngby, Denmark.

Mikael Cederfeldt
Jönköping University, School of Engineering, Department of Mechanical Engineering.
P.O. Box 1026, SE-551 11 Jönköping, Sweden.
Tel: Int +46 36 15 77 00 E-mail: mikael.cederfeldt@ing.hj.se