PRODUCT FAMILY MODELLING IN THE CONSTRUCTION INDUSTRY

Kaj A. Jørgensen
Department of Production, Aalborg University
kaj@iprod.auc.dk

SUMMARY

In general, the demand for customised products has increased radically and, as a result, the need for definition and specification of products by configuration has become more important than ever. This development is also related to the construction industry, where there is a clear interest in configuration of products related to building modelling. In this paper, an overview is given about the relationships between building models and product models and a methodology is presented for developing product family models, which are suitable for generation of advanced product configurators. A model of this kind contains definitions of different types of attributes, product modules and product components. The behaviour of the model is defined as different types of constraints and an inference algorithm.

INTRODUCTION

Customised products have primarily been introduced in connection with one-of-a-kind production of individual products or customer order initiated production of relatively small batches of products. However, for some years now, many companies have been aware that the market segments cannot be regarded homogeneous; the needs vary from customer to customer and products have to differ equally. So, many companies have found great potential in better focus on the customers and want to use this as a strategic opportunity to establish a competitive advantage [Sulonen, 1998].

The combination of mass production and the capability to offer customised products is termed *mass customisation* [Pine, 1993]. As shown in figure no. 1, the aim is that mass customisation should take the benefits from both one-of-a-kind production and mass production with respect to customisation and production volume. In order to reach this goal, a considerable readiness has to be developed. The products have to be well designed for this and the production must be preparation to a large degree [Jørgensen, 1998].

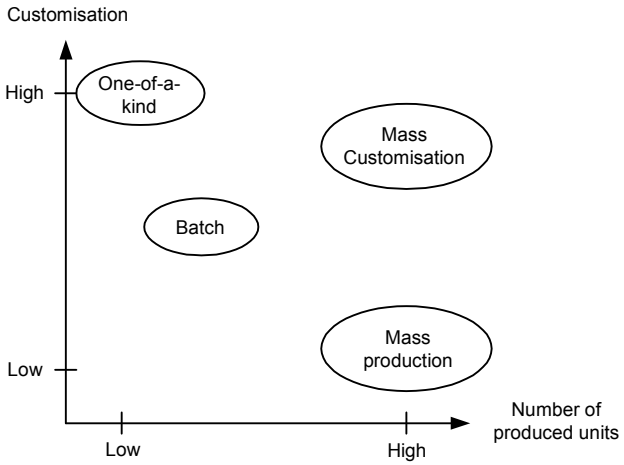

Figure 1 Different production forms positioned regarding customisation and production volume

This development has also become known in building and construction. Manufacturers of building products are responding to the same requirements and seek an optimal balance between customisation and production form. In addition, another development in the industry raises further

requirements and opportunities; new object-oriented building modelling tools provide possibilities to exchange more advanced models of products. When building models are created e.g. by architects, such product models should be available in order to perform flexible design tasks and manufacturers, who are able to offer advanced product models for building models, have obvious advantages. For products, which can be offered in a number of variants, the advantages are even more evident.

Selection of products that fulfil specific requirements is based on the properties of the available products. In general, the optimal product is the product, which properties best match the requirements. This kind of selection is based on fixed properties of products. But products can also have variable properties. A large number of different buildings elements and products can be defined by specification of one or more values, e.g. doors, windows, kitchen elements and furniture. Greater challenges are seen, when more complex products like heating and ventilation installations are considered. In addition, many more "intelligent" products will be available in the future [Bosch, 2000; Jazayeri, 2000], where some of the properties are established by assigning values to parameters in embedded software [Männistö, 2001].

BUILDING MODELS AND PRODUCT FAMILY MODELS

In the past decade, many ideas have been presented about building modelling and representation of building models in data structures [Eastman, 1999]. The currently most dominating data model is Industrial Foundation Classes (IFC), which is developed by International Alliance of Interoperability (IAI) [IAI, 2002]. This data model is partly based on the more general STEP/ISO10303 standard [ISO STEP] but changed and extended towards building modelling. The newest version IFC2x has recently been approved as an ISO standard ISO/PAS 16739 and IFC is now well accepted by leading parties of the industry and the related software vendors. IFC defines a class hierarchy and relationships of a large number of object types for building modelling and for exchange of building models. The available IFC classes provide the foundation for design of buildings by instantiation of building objects and linking them together in data structures. The objects include attributes, which define not only the 3D geometry but also attributes for materials, surfaces, cost, etc. Additional attributes can be defined through property lists. Furthermore, by use of the standard data exchange file format [ISO-21, 1994], building models or selected parts of building models can be exchanged between model design systems that comply with IFC, e.g. ArchiCAD from Graphisoft and AutoCAD from Autodesk. Particularly, individual building elements and products can be extracted from a building model or added to a building model based on IFC.

Normally, model design systems are equipped with model libraries of standard objects, building elements or building products, which can be used for preliminary modelling, see left hand side of figure no. 2. Usually, additional libraries can be developed and inserted in such systems. Some of these standard models are finalised models of objects or end-products, while other models are parameterised allowing the designer the ability to change the values of the parameters. Typical examples of finalised object or product models are lamps, knobs, handles, furniture, bath equipment, stoves and refrigerators. Parameterised building element models are windows, doors, beams, columns, slabs, walls, etc. Primarily the geometry parameters can be changed but subsequently other parameters can also be adjusted, when more detailed specification is performed. At some point in the life cycle of the building model, each preliminary product model may be substituted by a model of the selected as-built product, i.e. a finalised model delivered by the product supplier. This is necessary for e.g. facility management.

Parameterised models include behaviour, which can respond to the change of parameter values. In the simplest form, the model is recalculated explicitly after a set of change operations is carried out. This approach is used in objects, which are implemented with the Geometric Description Language (GDL) [Graphisoft, 2003; Nicholson, 2002]. Another approach is to define a method for each parameter. Such methods are invoked implicitly when the corresponding parameter is changed. This is the approach in the ISO PLIB standard [PLIB, 1998], where the attributes are divided into three different kinds: 1) part characteristics i.e. invariable attributes 2) context parameters whose values characterises the context in which the part is intended to be placed 3) context dependant characteristics whose values depend on the context parameters. Each context dependant attribute is considered a function, i.e. the value is defined by an algorithm. As shown in the following, other approaches represent the behaviour in different kinds of knowledge bases. Such models are often termed "smart" models or "intelligent" models [Halfaway, 2002; Nicholson, 2002]. It must be realised,

however, that according to the object-oriented paradigm, such models can be enriched with much more semantics and the behaviour can be implemented in more advanced ways.

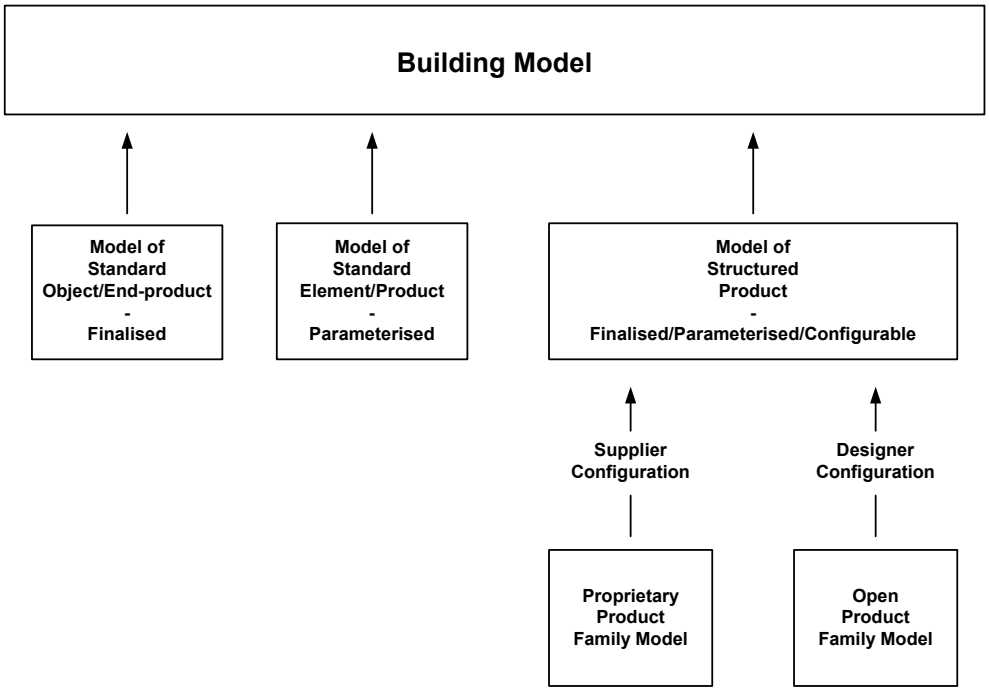

Figure 2 Different kinds of object/element/product models as input to building models

The standard models usually describe objects, building elements or building products as unstructured or else they do not include description of internal components and structure. However, the ability to insert models of structured products is also valuable for designers, see right hand side of figure no. 2. The most advanced models are configurable, i.e. the model can be configured to more simplified models - parameterised or finalised models. In this paper, configuration is considered more advanced because it also covers definition of product components and product structure. In advanced building models, it should be possible to insert open configurable product models and thereby provide the ability for designers and engineers to change configurations if needed. The basis for inserting product models in building models is offered by product suppliers and manufacturers. Typically, such models are proprietary models so configuration is performed by the supplier. However, it should at least be possible to generate parameterised product models.

PRODUCT FAMILY MODELS

Seen from the customer's point of view, configurable products are products, which can be specified - *configured* - by selecting property values from a given range of options so that each delivered product is individually manufactured according to specific requests from a customer.

Seen from the manufacturer's point of view, a configurable product is more precisely a *product family,* from which each *individual product* can be selected - by configuration. Usually, the product family includes a large number of possible products, which means that it is not feasible to describe all the individual products. Instead, the family must be described as a whole and descriptions of each product are derived as a result of the configuration [Faltings, 1998].

A generic model of a product family is termed a *product family model.* Such a model can serve as a foundation for the configuration process because it has a set of open specifications, which have to be decided in order to determine an individual product in the family. Hence, the product family is the set of possible products, which satisfy the specifications of the product family model. The result of each configuration will be a model of the configured product, *configured product model*. From this model, the physical product can be produced, see figure 3.

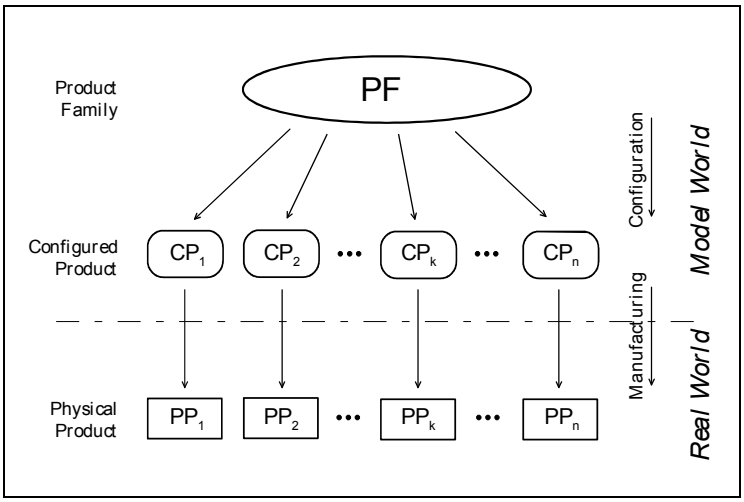

Figure 3 The product family model as the foundation for product configuration

The simplest form of product configuration is selection of appropriate modules. In this case, the product family model contains information about what modules and components are to be assembled and it is supposed unnecessary to change any of the existing modules or construct new modules in connection with sales and manufacturing of the product. In order to secure that only legal configurations are selected, the family model should contain restrictions about what is possible and not possible.

Product structure

The compositional view of a product is that it consists of a number of components, which again can consist of other components, etc. In connection with product configuration, however, it is often advantageous to identify modules on a level above the components. Regular modules are collections of physical components, but modules can also be of logical nature, e.g. language or parameter sets. From a product development point of view, modules are identified in order to implement one or a few functions. This is in contrast to an integral architecture [Ulrich, 2000]. In the product family model, the modules are primarily identified from a configuration point of view whereas components are identified from the product development or manufacturing point of view. Usually, the number of modules is smaller than the number of related components. Thus, in the typical structural model of a product family, products consist of modules and modules consist of components. This decomposition into three levels is shown in figure no. 4.

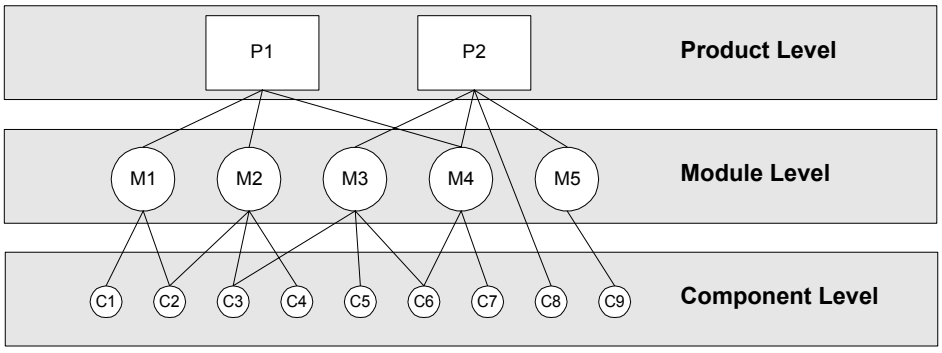

Figure 4 Model of the structure with the three top levels

However, some exceptions are often in order. A module may consist of other modules - sub modules. Furthermore, a component may be a part of the product directly (C8 in figure no. 4), for instance in the case, where the component is included automatically by the configurator. In the opposite case, individual components may be identified as modules (C9 is defined as module M5 in figure 4).

Properties and attributes

Products can be described by a selected set of properties. For each configured product, the resulting properties are, in principle, a function of the selected components and structure of the product. In the product family, algorithms must be developed for estimation of the product properties. Some properties are simply the properties of the components, e.g. the colour of a refrigerator is normally defined as the colour of the front cover. Other properties, however, are determined by functions of the properties of the components [PLIB, 1998]. For example, the weight is simply the sum of the component's weight. Not all resulting properties are so easy to determine. For instance, the resulting performance of a pump is a non-linear function of certain component properties.

In order to develop a product family model all relevant attributes must be specified and their optional values to be selected during configuration tasks must also be defined. In relation to this, it is important to notice that the selectable modules and components can always be substituted by attributes. It is only a matter of convenience. For instance, a room can have air conditioning or it can be equipped with an air condition module. In general, this means that the configuration process can be considered purely as selection of attribute values. In practice, a mixture of specifying attributes and selection of modules is more convenient. Figure no. 5 shows how underlying modules are determined on the basis of the chosen attributes.

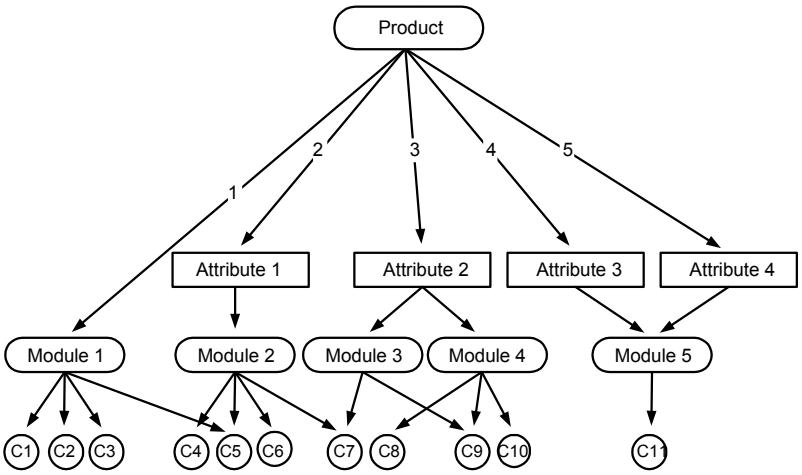

Figure 5 Specification of modules directly or indirectly through attributes

Selection no. 1 is shown as an exception. A specific attribute is not selected because, in this case, it is more natural to choose a module directly - typically add-on modules, e.g. a door handle. At selection no. 2, attribute1 is equal to module 2. For instance, lockable equals a lock. At selection no. 3, attribute 2 results in selection of both modules 3 and 4. An example of this is that several modules may be related to choosing geographic region or a language. Finally, selections no. 4 and 5 show a relatively usual case, where a module is determined by more than one attribute, i.e. the attributes of the module.

DEVELOPMENT OF PRODUCT FAMILY MODELS

As stated, the basic elements of each product family model are 1) the set of possible modules, 2) the total set of attributes of the models and 3) the possible product structures. In the following, a simple and general way of representing product family models is presented. The representation focuses on the product attributes and the relationships among the attributes and between attributes and modules or components. Hence, the representation covers the top level of the product family model. The detailed modelling of each module or component is not covered in this presentation.

The behaviour of the product family model is modelled by constraints. These constraints are divided into domain constraints and relational constraints. While domain constraints define value bindings of each individual attribute, the relational constraints are used for specification of other limitations. The

total set of constraints defines the possible valid selections, i.e. the solution space. The following is a short introduction to the methodology and the examples are rather primitive. However, a number of industry projects have been carried out proving the usefulness of the methodology, e.g. boiler products and electric control panels [Demex, 2003].

Specification of attributes

Each attribute is specified with a *name* and a *domain*, which defines the optional values of the attribute. For example, the domain can be a set of discrete numeric or string values. In addition, a *default value* and a *domain constraint* are specified. These constraints are the formal specifications of how the attribute values can be selected. Table 1 shows some examples of domain constraints.

Domain constraint	Description
OneOf	One and only one value from the domain must be chosen.
Optional	The binary choice - a special case of OneOf.
AtMostOne	None or only one value can be chosen.
AnyOf	None, one or more value from the domain can be chosen.

Table 1 Examples of domain constraints for attributes

In a simple form of a declarative statement, the following syntax is used to describe a module type and the belonging attributes:

AttributeName DomainConstraint [Domain] Default [Value]

Examples: Door.Material OneOf [Wood,Plastic,Metal] Default[Wood]
Door.Doorstep Optional [Yes,No] Default[Yes]
Door.Handle AtMostOne [<list of handles>]

Relational constraints

The relational constraints are used for specification of different bindings: 1) between the product attributes, 2) between product attributes and modules, 3) between modules, 4) between product attributes and module attributes, 5) between attributes of each module and 6) between attributes of multiple modules. The relational constraints are divided into *logical constraints* and *arithmetic constraints*.

Logical constraints are used to describe the allowed combinations of the module types and their attributes. This kind of constraints uses operators like AND, OR, XOR, NOT, implication (⇒) and biimplication (⇔). The boolean values True or False might as well be used. A number of operators that uses a list of variables as argument can also be used, for instance AllAlike (the arguments must have the same value) and Impossible (not all arguments can be true).

Some examples to indicate the syntax:

```
Door ⇒ Lock                          // Each door includes a lock
Door ⇒ Door.Screws[8]                // 8 screws belong to each door
Door.Material ⇔ Doorstep.Material    // The material of door and doorstep is equal
```

Arithmetical constraints are used to define the numerical limitations. Operators like addition, subtraction and multiplication are used together with one of the following operators: =, >=, <=, > and <. In addition it is possible to link the arithmetical expression with a boolean variable by means of implication (⇒) or bi-implication (⇔).

```
Hinges >= [2]                        // The number of hinges is at least 2
```

Inference method

Use of constraints in the product family model is an alternative compared to action rules, i.e. if-then statements. Hence, this approach is in contrast to the earlier mentioned implementations in GDL objects and in the ISO PLIB standard. The primary advantages are the following: the constraint representation is more elegant, constraints bases are smaller than rule bases and inference algorithms have a much better performance. In addition, the inference algorithm allow decisions about values of the product attributes to be made in any order, see [Sabin, 1998; Soininen, 2000] for a further comparison. Research results regarding knowledge-based systems have resulted in inference algorithms with very good performance [Møller, 1995].

Implemented in product configurators, such algorithms can be activated each time the user makes a selection [Yu, 1998; Array, 2003]. For large models, the process may take too much time in the beginning, but, after some selections, the results can be presented within reasonable time. Referring to the constraints, this approach has the great advantage that the consequences for related attributes are shown before the next choice has to be made. This implies that, at each point of a configuration process, the already made decisions can be checked for consistency and, furthermore, it is possible to freeze these decisions and transform the configurator to a new configurator with the remaining decisions to be made by others. In this way, a good balance can be maintained between proprietary product family models and open product family models.

User interfaces

The design of product configurator user interfaces is very important and the great potential in using web technologies should not be underestimated. With this in mind, a large number of graphical components should be included where appropriate, i.e. drop-down lists, check boxes and radio buttons. In addition, automatic generation of visual effects in connection with the selectable options would be very helpful. For instance, when different colours are the option, this should be shown graphically and, when different modules can be selected directly or indirectly, they should be presented as graphic 3D images. Ultimately, the complete picture of the configured product should be generated and perhaps presented as a high quality rendered image in 3D stereo.

In order to include such graphical components in the user interface, it is important to include geometrical data in the product family model. This underlines the fact that a product model is more than a CAD model. When a product family model is the origin, it is obvious that geometrical data is only a part of the model. Attributes describing the product geometry may even be included as secondary attributes, where the values are derived through the internal constraints.

CONCLUSIONS

The increasing demands from customers for customisation have lead to the introduction of mass customisation and product configuration. In response to this situation, suppliers and manufacturers in the construction industry must realise the importance of developing different kinds of models of their products. It is also important that these models can be incorporated in building models in different forms - in the most advanced form as configurable models. The origin of such models is product family models, which can serve as the basis for configuration tasks. A product family model is a generic and synthetic model of a configurable product, i.e. of the set of possible products in a defined product family. A product family model includes description of attributes, modules, components and constraints. The constraints are used to describe limitations regarding attribute values and relationships between attributes and modules or components. The presented development methodology results in an elegant description of the model and provide a basis for implementing efficient and effective inference algorithms in product configurators. Such models are also the basis for introduction of more open product family models.

REFERENCES

[Array, 2003] *Array Technology*, www.array.dk, 2003.

[Bosch, 2000] J. Bosch: *Design and use of software architectures - adopting and evolving a product-line approach.* Addison-Wesley, 2000.

[Demex, 2003] *Demex Electric*, http://www.demex-electric.dk, 2003.

[Eastman, 1999] Eastman, C. M.: *Building Product Models*. CRC Press, Boca Raton FL, 1999.

[Faltings, 1998] Boi Faltings and Eugene C. Freuder (Ed.): *Configuration - Getting it right.* Special issue of IEEE Intelligent Systems. Vol.13, No. 4, July/August 1998

[Graphisoft, 2003] *Graphisoft*, Budapest Hungary, http://www.graphisoft.com, 2003.

[GDL, 2003] *GDL Technology*, http://www.gdltechnology.com, 2003.

[Halfawy, 2002] Halfawy, Mahmoud R. and Froese, Thomas: *Modelling and Implementation of Smart AEC Objects: An IFC Perspective*. In: Proceedings of CIB W78 Conference 2002. Århus, 2002.

[IAI, 2002] *International Alliance of Interoperability*, http://www.iai-international.org, 2002.

[ISO STEP] ISO10303: *STEP - Standard for the Exchange of Product Model Data* (series of standards). ISO, Geneva.

[ISO-21, 1994] ISO10303-21: *Product Data Representation and Exchange, Implementation methods: clear text coding of the exchange structure.* ISO, Geneva, 1994.

[Jazayeri, 2000] M. Jazayeri, A. Ran and F. van den Linden: *Software architecture for product families: Principles and practice.* Addison-Wesley, 2000.

[Jørgensen, 1998] Kaj A. Jørgensen and Thomas Raunsbæk: *Design of product configuration management systems.* In: Proceedings of 2nd Int. Conf. on Engineering and Design, Hawaii. Integrated Technology Systems, Inc., 1998.

[Männistö, 2001] T. Männistö, T. Soininen and R. Sulonen: *Product Configuration View to Software Product Families.* In: Proceedings of Software Configuration Management Workshop (SCM-10). Toronto, 2001.

[Møller, 1995] Møller, G.: *On the Technology of Array Based Logic.* Ph.D. Thesis, Technical University of Denmark, 1995.

[Nicholson, 2002] David Nicholson-Cole: *The GDL Cookbook.* ISBN 0 9535216 0 5. Marmalade Graphics, Nottingham, 2002

[Pine, 1993] B. Joseph Pine: *Mass Customization - The New Frontier in Business Competition.* Harvard Business School Press, Boston Massachusetts, 1993.

[PLIB, 1998] ISO 13584-42: *Industrian automation systems and integration - Parts Library - Methodology for Structuring Parts Families.* ISO, Geneva, 1998.

[Sabin, 1998] D. Sabin and R. Weigel: *Product Configuration Frameworks - A survey.* In IEEE intelligent systems & their appplications, 13(4):42-49, 1998.

[Soininen, 2000] T. Soininen: *An approach to knowledge representation and reasoning for product configuration tasks.* Doctoral thesis. Helsinki University of Technology. 2000.

[Sulonen, 1998] Reijo Sulonen, Juha Tiihonen, Timo Soininen, Tomi Männistö: *Configurable Products - Lessons learned from the Finish Industry.* In: Proceedings of 2nd International Conference on Engineering Design and Automation, Hawaii, Integrated Technology Systems, Inc., 1998.

[Ulrich, 2000] Ulrich, Karl T. and Eppinger, Steven D.: *Product Design and Development*, 2. int. edition. ISBN 0 07 229647X. McCraw Hill, 2000.

[Yu, 1998] Yu, Bei and Skovgaard, Hans Jørgen: *A Configuration Tool to Increase Product Competitiveness.* In IEEE Intelligent Systems, vol. 13, no. 4, 1998.