A classification of strategies for the development of product configurators

Haug, Anders; Hvam, Lars; Mortensen, Niels Henrik

Published in:
Proceedings of MCPC 2009

Publication date:
2009

Document Version
Publisher's PDF, also known as Version of record

A classification of strategies for the development of product configurators

Anders Haug
University of Southern Denmark
adg@sam.sdu.dk

Lars Hvam
Technical University of Denmark
lhv@man.dtu.dk

Niels Henrik Mortensen
Technical University of Denmark
nhmo@man.dtu.dk

Abstract. Product configurators are a subtype of software-based expert systems with a focus on the creation of product specifications. Product configurators are increasingly being applied by engineering-oriented companies, which has resulted in many positive effects, such as reduced lead times, fewer errors, shorter learning periods for new employees, etc. Unfortunately, also many configuration projects fail because the task of developing the configurator turns out to be much more difficult and time-consuming than anticipated. In order to minimize the chance of project failure, it is crucial to apply the right strategy. However, the literature does not discuss different strategic alternatives in a detailed manner, but only provides generalised recommendations of single strategies. To deal with this issue, this paper defines three main and four additional strategies for the development of product configurators. The strategies are defined based on literature, seven named case studies, and other case experiences of the authors. The paper deduces the advantages and disadvantages of the individual strategies, and gives a general recommendation of which type of strategy to pursue in different types of projects.

Keywords. Product configuration, Product configurator, Configurator development, Configurator strategy

1 Introduction

On the world market today, there is an increasing demand for customer-specific products, while the competition on prices and delivery times is hard (Forza and Salvador, 2006; Aldanondo and Vareilles, 2008). Mass customization is a manufacturing paradigm that focuses on satisfying individual customer requirements, while keeping the manufacturing costs and delivery times close to the ones of mass produced products (Pine, 1993). Mass customization can be seen from two much different perspectives, depending on whether the company that pursues a mass customization strategy is an engineering-to-order company or a mass production company (Haug et al., 2009b). This paper focuses on engineering-to-order companies, in which the use of product configurators in many cases has been an important means of reducing lead times and prices, while maintaining the ability to satisfy individual customer requirements. A product configurator is a subtype of software-based expert systems (or knowledge-based systems) that creates product specifications based on design requirements from users and its rules/constraints for how elements may be combined. More formally, a product configurator can be defined as "a software-based expert system that supports the user in the creation of product specifications by restricting how predefined entities (physical or non-physical) and their properties (fixed or variable) may be combined" (Haug, 2007). In the context of engineering-oriented companies, the use of product configurators has resulted in a range of benefits, such as: shorter lead times, improved quality of product specifications, preservation of knowledge, use of fewer resources for specifying products, optimized products, less routine work, improved certainty of delivery, and less time needed for training new employees (e.g. Ardissono et al., 2003; Forza and Salvador, 2007; Hvam et al., 2008).

Although many engineering-oriented companies achieve some of the mentioned benefits from the use of configurators, many companies also experience great difficulties. Unfortunately, the literature describes almost only successful cases, which does not give a correct picture of the reality. According to the wide experience of the authors from studying numerous configurator projects and through discussions with people from industry and academia, a great part of initiated configurator development projects, aimed at complex products and multiple users, do not result in the creation of a configurator, which is nearly as widely used by the organisation as expected. In fact, often projects are abandoned even before the configurator is completed. On the other hand, it is often seen that although a configurator project fails, later a project with a similar focus but with a different organisation, scope or choice of software succeeds.

There are two basic challenges to overcome in order to achieve success in a configurator project. Firstly, the project has to survive until the configurator is developed to a point where it can be taken into use. If a project gets significantly more costly than anticipated or the project fails to produce prototypes that indicate that the project may succeed, such projects may be abandoned in order to reduce potential losses. Secondly, although a project leads to the development of a configurator, still the configurator has to be accepted and used by the organisation. In this context there are several challenges, such as ensuring that the configurator covers an adequately big part of the products produced, that it can produce adequately extensive outputs, and that it is adequately precise. If such demands are not fulfilled, the use of the configurator may soon be abandoned.

There is a number of different ways of carrying out the project of developing a product configurator. Obviously, the way this is done impacts both costs, development time and the quality of the configurator. However, in the literature a clear definition of the different ways of carrying out the development of a configurator does not exist, but only proposals of specific approaches, without much consideration of alternatives and descriptions of in which contexts the specific approach is suited. Thus, this paper answers two important questions: (1) what are the different strategies for developing product configurators? and (2) what are the advantages and drawbacks of choosing one of these strategies? By clarifying the different strategies in configurator projects and the consequences of choosing them, this paper provides an improved understanding of configurator projects for future research and a better basis for companies engaging in such projects.

The remainder of this paper is structured as follows. Section 2 resumes relevant concepts and literature. Next, section 3 takes a basis in cases previously studied by the authors and defines possible strategies. Section 4 engages in a discussion of the defined strategies concerning the effects of choosing them and of which case characteristics that make some strategies more suited than others. The paper ends with a conclusion in section 5.

2 The development of configurators

In order to define the different strategies for the development of product configurators, some basic concepts need to be defined. This section resumes the concepts of knowledge acquisition and knowledge representation in a configuration context, followed by a description of different perspectives on the configurator development process.

2.1 Knowledge acquisition

In order to obtain the many possible benefits from implementing a configurator in the daily operations of a company, the right information has to be implemented in the configurator. The information that forms the basis for the creation of a configurator is provided by the relevant product experts of a company (i.e. expertise related to design, processes, sales, etc.). The ones responsible for retrieving and formalizing product expert information into conceptual models are called 'knowledge engineers'. The process of retrieving information from product experts and transforming it into representations suitable for implementation in an expert system is referred to as 'knowledge acquisition'. Knowledge acquisition is a process that includes the activities: 1) the knowledge engineer uses of communication techniques to elicit information from relevant experts (knowledge elicitation); 2) the knowledge engineer interprets this information to draw conclusions about what may be the underlying knowledge and reasoning processes of the product experts; and 3) the knowledge engineer uses his/her conclusions to direct the construction of a model (and the implementation of it in an expert system shell or language) (Byrd et al., 1992). Earlier, the development of expert systems was seen as a transfer process, based on the assumption that the required knowledge already existed, for which reason the task of creating an expert system merely was to collect and implement this knowledge. Today, however, there is overall consensus that the process of building an expert system is better perceived as a modelling activity (Clancey, 1993; Studer et al., 1998; Speel et al., 2001). This modelling view on the knowledge acquisition process has according to Studer at al. (1998) the following consequences: 1) models are only approximations of the reality, and principally, the modelling process is infinite since it is an incessant activity with the aim of approximating the intended behaviour; 2) the modelling process has a cyclic course, since new observations may lead to a refinement, modification or completion of the already built-up model; and 3) the modelling process is dependent of the subjective interpretations of the knowledge engineer, for which reason the process is typically faulty, and evaluation of the model with respect to reality is indispensable in order to create an adequate model.

2.2. Knowledge representation

The area of knowledge representation is a subarea of the field of artificial intelligence and focuses on the use of symbol systems to represent (simulate) the relevant knowledge of some domain. Davies et al. (1993) argue that the essence of a knowledge representation best can be understood by the five distinct roles it plays: (1) a surrogate (a substitute for the thing itself, used to enable reasoning about the world rather than taking action in it); (2) a set of ontological commitments (description of how in which terms the world should be perceived); (3) a fragmentary theory of intelligent reasoning (expressed by three components,

a fundamental conception of intelligent reasoning, a set of inferences sanctioned, and a set of inferences recommended); (4) a medium for pragmatically efficient computation (the computational environment in which thinking is accomplished); and (5) a medium of human expression (a language for saying things about the world).

In configurator development projects, two diagramming techniques are frequently used for the capture and representation of product information, namely the product variant master (PVM) technique (sometimes named 'Product Family Master Plan') and class diagrams. PVMs are targeted at representing knowledge about a product assortment. PVMs describe classes, their relations and properties, and constraints that determine how classes and properties may be combined. A PVM consists of two generic sections. The part-of section, placed in the left side of a PVM, defines the classes that a given product family can comprise. The kind-of section, placed in the right side of a PVM, describes the variation of a part, i.e. different types with common characteristics. In the course of time, different definitions of the PVM notation have been proposed, most importantly by Mortensen et al. (2000), Harlou (2006) and Haug (2007). Based on the latter definition, which represents the most extensive and formalized of these definitions, a principal example of the PVM technique is seen in Figure 1. In the example, the guillemets (<<...>>) indicate that the class 'BodyAssembly' is a class of the type 'assembly'.

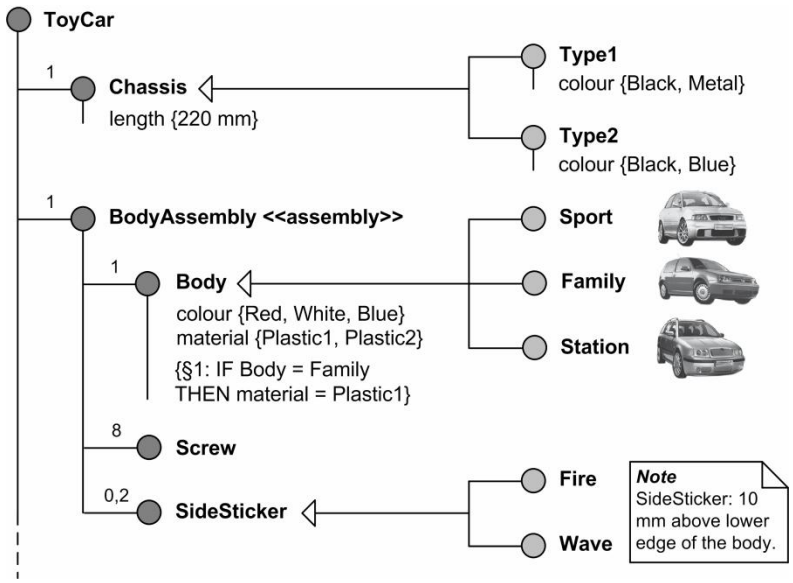

Figure 1. Example of the PVM technique.

In contrast to PVMs, class diagrams have not been created as a language aimed at describing product information, but at general software development. A class diagram describes object classes, their properties and their relationships. Class diagrams are part of the UML (Unified Modelling language), which in version 2.0

consists of thirteen diagram types (OMG, 2005). Figure 2 shows a principal example of a class diagram with some of the most common elements, as if applied in a product analysis context (extended version of the model in Figure 1, without pictures). The relationship types shown are: generalization (generalization-specialization structure); composition (whole-part structure, strong aggregation in the sense that the 'part' is controlled by the 'whole' all its lifetime); association (classes that are related but not part or type of the other class); and dependency (a change in one element affects the other element). For description of constraints on how elements may be combined, UML provides a formal language named OCL (Object Constraint Language). However, the use of OCL is not mandatory in order to comply with normative use of UML, but any formalism for the description of constraints can be used, as long as being within braces ({}).

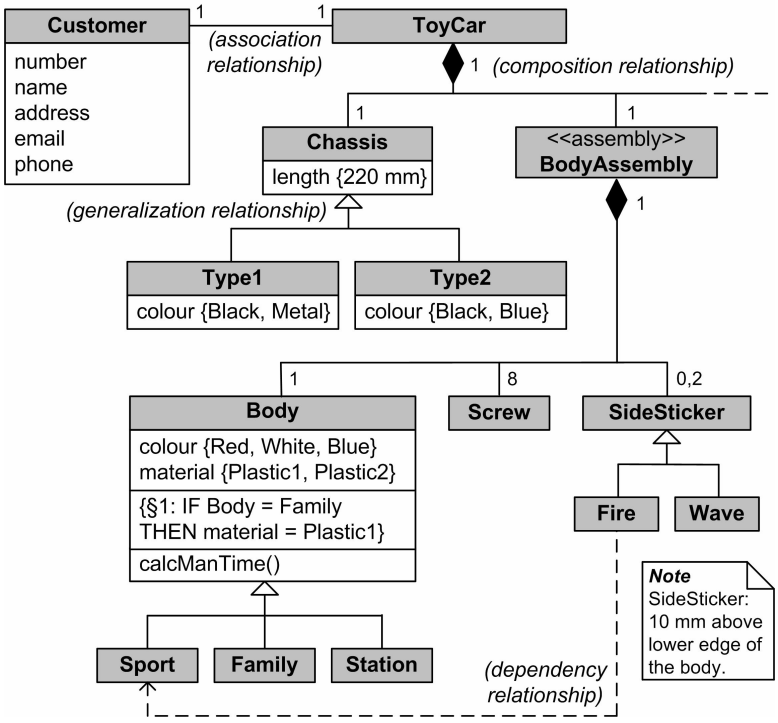

Figure 2. Example of class diagrams.

In a configurator development context, the two described types of representation techniques can be used more or less for describing the same thing. The main difference is that class diagrams represent a richer and more formal notation than PVMs, while PVMs seem to be easier to learn and review for persons with limited prerequisites concerning software development and conceptual modelling (Haug and Hvam, 2007a; 2007b). Studies of Danish companies engaged in configuration projects indicate that PVMs (or variants of this language) are much more applied than class diagrams in these cases (Edwards et al., 2005; Haug and Hvam, 2007a; Hvam et al., 2008). On the other hand, several reports of the use of class diagrams

in international cases of configuration and product information structuring exist (e.g. Felfernig et al., 2000; Männistö et al., 2001; Eynard et al., 2004; Cerón et al., 2004).

In some configurator projects, PVMs and class diagram are extended by CRC cards (classes, responsibility and collaboration cards) for holding detailed information about classes, in order to avoid loss of the clarity of the diagrammatic representation. Compared to the original CRC-cards, defined by Beck and Cunningham (1989), such CRC-cards can also hold pictures, implementation information, revision information, etc. (Hvam et al, 2008; Haug and Hvam, 2009).

2.3 A knowledge engineering perspective on configurator development

Based on experience retrieved from the numerous projects studied at the Centre for Product Modelling at the Technical University of Denmark (e.g. Edwards et al., 2005; Hvam, 2004; Hvam et al. 2008), at an overall level, the process of creating and maintaining product configurators can be described as consisting of the six main processes illustrated in Figure 3 and subsequently described (Haug, 2007). The figure should only be perceived as an idealized description of the process, which in practice includes variations in the form of skipping, merging or not formally conducting some of the processes. In the figure the dotted line from analysis to design model illustrates that the distinction between analysis and design model is not necessarily explicitly defined. Often in practice, the design models are just more formalized and detailed versions of analysis models. The dotted lines to the documentation system (a system for sharing documentation) symbolize that a documentation system may not included in the different phases.

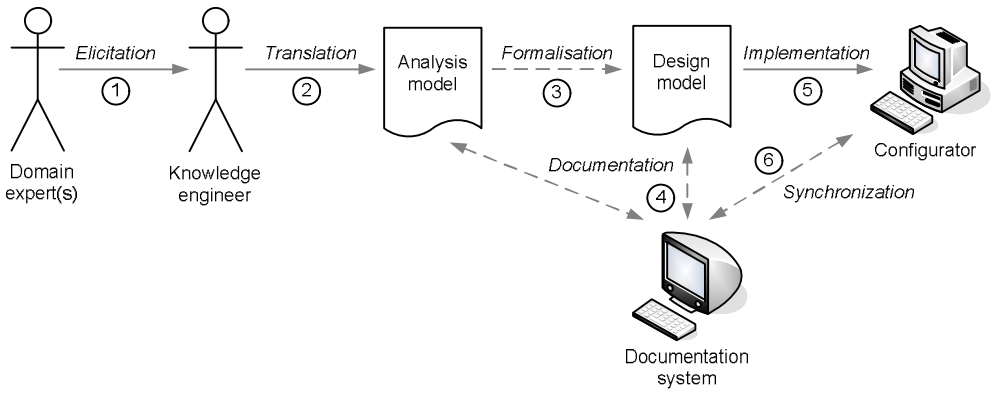

Figure 3. The process of creating product configurators (Haug, 2007).

The first process of Figure 3, 'Elicitation, is the process in which the knowledge engineer retrieves the information related to product in focus. This information can be delivered verbally, by demonstration or in documents of physical or electronic

form. Documented information can be in the form of text, diagrams, tables, formulas, informal rules, sketches, etc.

In the second process, 'Translation', the retrieved information is recorded in analysis models in order to provide a basis for discussing the product information to be included in the configurator. Just as information delivered verbally needs to be translated into the analysis model language, also the information delivered in recorded form, such as texts and diagrams, most often needs to be translated into another format in order to communicate this efficiently with relevant experts. For instance, in many cases it is advantageous to transform textual information into graphical models to provide a basis that is adequately clear and structured to support discussions and evaluation by product experts. Besides retrieved information, new information may need to be created, since some rules may not have been explicitly described before the project. During the translation phase, a product optimization may also take place, i.e. by eliminating or substituting rarely used solution principles and components.

In the third process, formalization, the analysis models are transformed into a format more suitable for implementation. The created analysis models may not be well suited for implementation in a configurator if the constructs of the representation language of the analysis models differ from the ones of the configurator modelling environment. Also, other configurator aspects such as integrations to other systems and user interfaces may need to be defined. As mentioned, in practice, often the transition from analysis to design model will not be strictly divided, but be more a change of focus and a further elaboration of the analysis models.

In the fourth process, documentation, the information acquired and defined during the analysis and design phases of configuration projects has to be documented if others persons than the ones possessing this information are to use or evaluate it. In this context it should be noted that product experts normally do not understand (or have access to) the modelling environment of configurators, which creates the need for external knowledge representations. In the many configuration projects studies at Centre for Product Modelling at the Technical University of Denmark the kinds of software most often used for external documentation are MS Visio, Excel, Word, CAD Programs, Lotus Notes and Product Model Manager (e.g. Hvam, 2004; Edwards et al., 2005; Hvam et al., 2008; Haug and Hvam, 2009).

In the fifth process, implementation, the design models are implemented in the selected configurator software shell or programming language. During the implementation process, changes compared to the design models may occur and new information may be created.

The final process, synchronization, aims at ensuring that the documentation is always up to date. This means that this external information has to be updated when changes in the knowledge base of the configurator occur. If such external

documentation is not created, in many cases it becomes very difficult or even impossible to overview the knowledge implemented (Edwards et al., 2005; Hvam et al., 2005, Hvam et al., 2008).

2.4 Approaches with a wider focus

The provided description of the process of developing a product configurator in section 2.3 only describes a small part of a configuration project. In configuration projects, also issues such as process reengineering, organisational change management, software selection, and configurator maintenance have to be considered. Some existing approaches deal with such issues in varying extent.

Hvam et al. (2006;2008) describe a seven phase procedure for the development and maintenance of product configurators, starting at business process analysis and ending in a configurator maintenance phase. For the description of the existing product information (product analysis), this procedure recommends the use of PVMs. For the creation of design models, the procedure recommends the use of class diagrams. For detailed information of classes in PVM and class diagram models, this procedure recommends the use of special CRC-cards. Besides the tasks of collecting, documenting and implementing product information in a configurator, the procedure also deals with issues such as process reengineering, organisational change management, software selection, and configurator maintenance. Hvam et al. (2008) recognise that the procedure may be used in altered versions, but do not provide an overview of such uses.

Another approach for the development of product configurators is been proposed by Felfernig et al. (2000; 2001). This development process does not have the wide focus as the one by Hvam et al. (2008), as it does not include aspects such as organisational change and process reengineering. Instead, Felfernig et al. (2000) show that UML class diagrams can be used for the construction of a logic-based description of the domain knowledge. This includes introducing classical description concepts for expressing configuration knowledge in class diagrams and automatically translating a class diagram representation into logical sentences, which can be used by a general inference engine for solving a configuration task. In principle, class diagrams function both as analysis and design models in this procedure. The papers (Felfernig et al., 2000;2001) do not engage in discussions of in which contexts the defined process is useful or what the alternative approaches are.

Forza and Salvador (2007, pp. 160) recognise that there is a number of different ways of carrying out a configurator development project. Thus, they settle for a definition of an overall five phase reference process, consisting of a logical sequence of activities needed for the implementation of a configurator. This process starts by a preliminary analysis, which aims at deciding if product customization should be addressed by the company or not, and ends at a phase of

creating, testing and launching a configurator. Forza and Salvador (2007) provide a comprehensive overview of topics related to mass customization and product configuration, but do not go into detail about different representation formalisms and strategies concerning different setups of persons, knowledge representation formalisms and information transfers in the context of developing product configurators.

3 Strategies for the development of configurators

This section firstly describes some basic assumptions made in order to compare different product configurator development processes. Through this optic, seven previously studies cases are analysed, and based on this, the strategies for the development of configurators are defined.

3.1 Basic assumptions

As mentioned, the focus of this paper is on projects that include products of some complexity and in which multiple experts possess the information needed to develop the configurator. To establish characteristics of different strategies for the development of configurators in such projects, to begin with, some basic assumptions are made. As a starting point, the process of developing a configurator is divided into three tasks:

1) Retrieve relevant product information
2) Represent relevant product information
3) Implement defined product information

The first task, retrieve relevant product information, includes identifying which persons, documents or IT systems that hold the information needed in order to develop a configurator. Also it needs to be ensured that the retrieved information is correct and agreed upon, and not just subjective estimates from single persons without proper authority for making such decisions. The next task, represent relevant product information, is most often a necessary step before implementation in the configurator. If relevant product information is not represented in a way that is understandable by relevant parties, then essential discussions and agreements can be difficult to achieve, such as the scope of the configurator, what information to include, definitions of new product information, etc. If the persons who are going to use the configurator do not agree on the information that has been implemented, then there is a great risk that they will not use the system and the project eventually fails. The third task, implement defined product information, varies in complexity, depending on the selected software. In some cases, the selected

software allows persons with only moderate IT competencies to build the knowledge base of the configurator, while in other cases this task requires significant IT expertise.

The next basic assumption is a division of the persons involved in a configuration project into the three following types, which are subsequently described:

1) Product expert
2) Knowledge Representation Expert
3) Configurator Software Expert

Product experts are the persons who possess the relevant information related to the product in focus, i.e. engineers, technical personal, sales persons, managers, etc. This expertise can be in the form of undocumented knowledge/information, which resides in the head of the expert, and documented information in the form of texts, drawings and software systems. The next type of expert, a knowledge representation expert, is a person who has the ability to represent product information in a way that is both adequately formalised and unambiguous to allow implementation in a configurator, while also being adequately easy for product experts to understand, so that they can evaluate the representations. The reason for using the term 'knowledge representation expert', instead of the more frequently encountered 'knowledge engineer', is to avoid confusion about the role. The term 'knowledge engineer' is sometimes used in a broader meaning than just knowledge retrieval and representation, since it sometimes also includes the implementation of product information in the expert system, in this case a configurator. The last type of expert, a configurator software expert, is a person with the adequate skills in developing configurators in the selected configurator software.

3.2 Studies of configuration projects

At the Centre for Product Modelling at the Technical University of Denmark, more than 40 configuration projects have been studied during the last 15 years (Hvam et al., 2008). Many of these cases are described in the literature, and for some of the cases, information about the approach for the development of configurators has been obtained. Seven of the cases that fulfil both of these conditions are:

1) Demex Electric (e.g. Hvam et al., 2002)
2) Fritz Hansen (e.g. Riis, 2003)
3) FLSmidth (e.g. Hvam, 2004)
4) Niro (e.g. Hvam, 2004)
5) APC (e.g. Hvam, 2004)

6) Novenco (Haug et al., 2009a)
7) Grundfos (Oddsson, 2008)

Most of these seven companies have launched several configurator projects and are presently using multiple configurators. However, in the literature mentioned, only some of these development projects have been in focus in the case studies. Focusing on the particular configurator projects and using the optic defined in section 3.1, the seven cases represent the use of three main strategies. Additionally, three other strategies, which were not included in the main development strategies in the investigated cases, were also identified. Finally, based on the six identified strategies, one additional strategy was deduced. Thus, in all seven distinct strategies for the development of product configurators are defined. These strategies are described in the following sections.

3.3 The three main strategies

The three main strategies for the development of product configurators found in the cases studied are shown in Figure 4 and subsequently described.

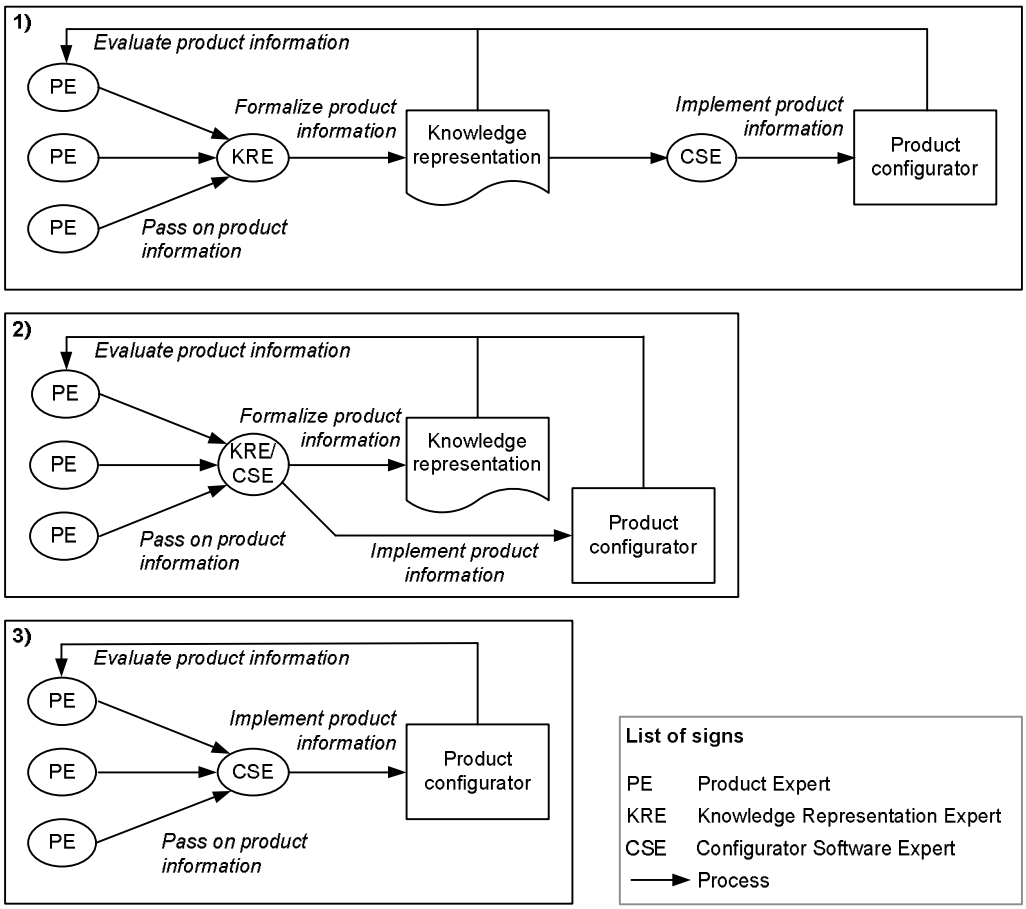

Figure 4. The three main configurator development strategies.

Strategy 1 represents the 'textbook' way of carrying out configurator projects, where each task is carried out by a specialist in one specific area. In this approach, product experts evaluate the models of the relevant product information before it is implemented in the configurator. Strategy 2 represents a case in which the persons that represent the product information also implement it in the configurator. The creation of conceptual models of the acquired product information allows product experts to evaluate this information before and during the implementation in the configurator. Strategy 3 differs in the sense that conceptual models are neither made before nor during the implementation of the product information. Thus, only by testing the product configurator, product experts can evaluate if the product information implemented is correct.

Table 1shows the distribution of the use of the three strategies in the analysed cases. For reasons of confidentiality, the seven companies are named A to G. In two of the cases, two strategies were applied to a great extent.

	Companies							Total
Main strategy applied	**A**	**B**	**C**	**D**	**E**	**F**	**G**	
Strategy 1	X	X				X	X	4
Strategy 2			X	X				2
Strategy 3					X	X	X	3

Table 1. The three main strategies for development of product configurators.

3.4 Four additional strategies

As mentioned, three additional strategies, which cannot be said to have been among the main development strategies in these companies, were identified. These strategies are shown in Figure 5.

The use of strategy 4 was found in two of the cases studied, but as a strategy mainly used in the maintenance phase, rather than a main strategy in the configurator development projects. Thus, strategy 4 does not qualify as being a main development strategy. Strategy 4 is in many ways similar to strategy 3, when considering that the information that product experts hand over to configurator software experts in strategy 3 may be in the form of conceptual models. Strategy 3 differs from strategy 4, since it represents a situation where the product experts provide information spread across multiple models or documents, while in strategy 4 the product experts work on common models. There is also a fine line between strategy 4 and strategy 1. The difference between the two strategies is that in strategy 1 the creation of conceptual models is handled by one or few central persons, while in strategy 4 the product experts document their knowledge themselves.

Strategies 5 and 6 were identified in one of the projects. In this case the software used for the creation of conceptual models could transfer these models to a configurator. At this company, in a single project, a combination of strategies 5 and 6 was applied to create a configurator. This configurator, however, had not been taken into full use at the time of the study, and the other configurator projects at this company had been done in accordance with the three main strategies. Thus, strategies 5 and 6 cannot be considered to be among the main strategies at the company in focus, at least not for the time being.

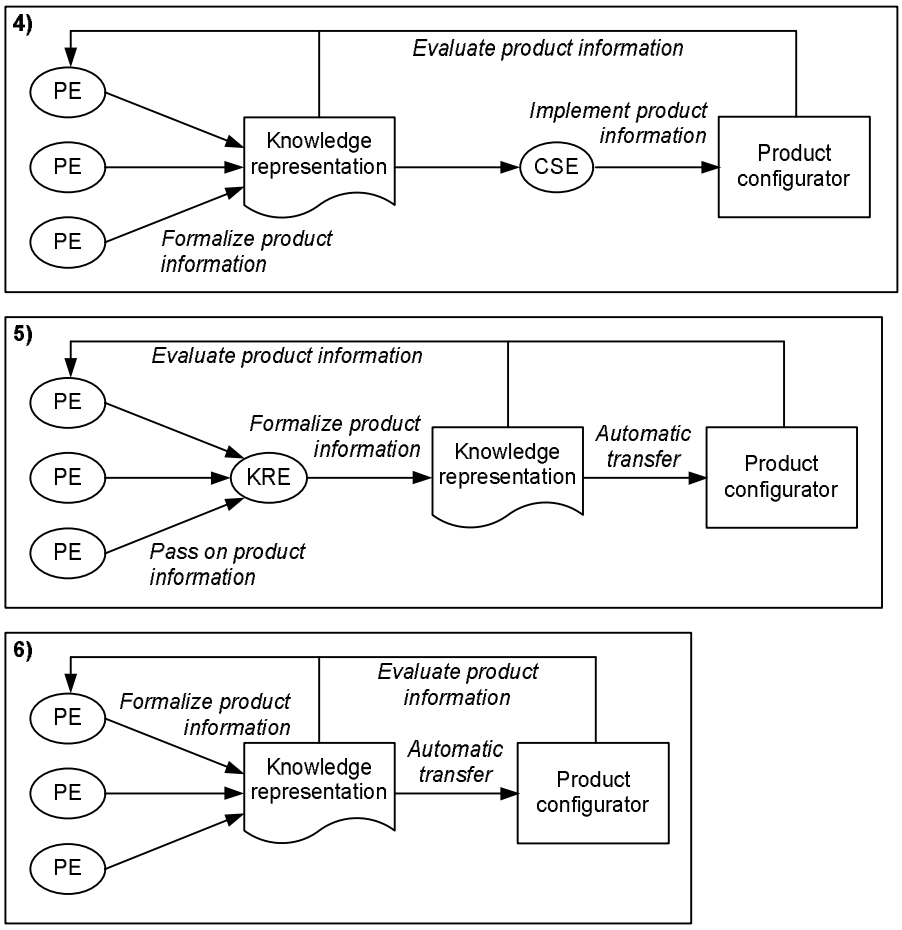

Figure 5. Strategy 4, 5 and 6.

Besides the six defined strategies, a seventh strategy, which was not found in the mentioned cases, can be defined. In this strategy, the product experts themselves implement their knowledge in the configurator software, as shown in Figure 6. Strategy 7, however, in the context of complex products and multiple involved product experts holds some great challenges. Firstly, the strategy requires that multiple product experts are capable of working with the relevant configurator software, which seldom seems to be the case in practice, not least because most configurator software shells are difficult to use for persons without IT

prerequisites. However, configurator software shells may become more user-friendly in the future, and, therefore, this strategy more realistic. Another issue is that multiple product experts need to work on the same model, instead of it being one or a few central experts who are responsible for the configurator development (as in strategy 3). Obviously, this approach could make it very difficult to achieve consistent configurator knowledge bases and avoid errors and disagreements. On the other hand, the approach may work fine if it is only a few central persons who possess the relevant product expertise, and they also have the necessary configurator software skills. However, as mentioned, this simple type of case is not the focus of this paper.

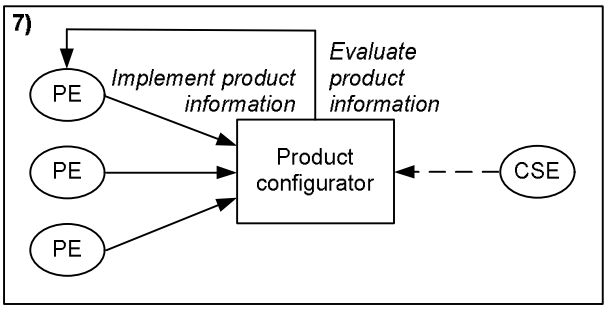

Figure 6. Strategy 7.

4 Discussion of the three main strategies

This section discusses the effects of choosing the individual strategies and in which type of cases each strategy seems to be advantageous. In order to limit the complexity, the focus in this section is limited to comparing only the three main strategies.

4.1 Effects of choosing the individual strategies

The three defined main strategies hold both a set of advantages and drawbacks when compared to each other. Based on literature and the experience of the authors from studying the mentioned cases and others, six dimensions for comparison have been deduced. The dimensions are not complete, but represent some of the most important aspects. Using these dimensions, the three strategies are compared in Table 2. Subsequently the arguments for this evaluation are provided.

	Relative benefits (+) and drawbacks (-)	S1	S2	S3
1	Evaluation of gathered product information before implementation	+	+	-
2	Ease of altering implemented product information	+	+	-
3	Minimal use of resources on documentation work	-	-	+
4	Facilitation of the communication between product and configurator software experts	+	-	-
5	Few handovers of information, i.e. less chance of misunderstandings and errors	-	+	+
6	Instant implementation of gathered information, i.e. faster process	-	+	+

Table 2. Comparison of strategies.

As seen in Table 2, the first dimension, evaluation of gathered product information before implementation, is considered to be an advantage of strategies 1 and 2, while being a drawback of strategy 3. This is because the strategies 1 and 2 in contrast to strategy 3 imply the creation of conceptual models, which can be evaluated before implementation. Obviously, being able to evaluate what should be implemented before it is done, can ensure that many misunderstandings between product experts and configurator software experts are discovered and corrected, and thus the later tests of the configurator become less error-prone. This is often a very important aspect, because if the future users during the tests of a configurator are exposed to numerous errors, this may destroy their trust and their motivation of using the configurator. Another problem of not making conceptual models before implementing information in a configurator is that it easily becomes difficult to overview what information the configurator software experts have been given, especially when dealing with large models. Thus, communication may become inefficient and misunderstandings occur.

In the second dimension, ease of altering implemented product information models, the strategies 1 and 2 have an advantage, while strategy 3 has a drawback. The problems of not having external descriptions of the implemented information emerge when a configurator needs to be changed or updated. Typically, the persons implementing the product information are not experts in the product, for which reason the product experts have to provide the information needed when a configurator does not behave correctly. Having an external model of what is implemented, which is made in a language that the product experts understand, means that the product experts can point out which rules are incorrect and how they should be altered. However, if external documentation does not exist and the product experts do not understand the modelling environment of configurators, product experts cannot point out directly what should be altered, but have to base their comments on tests of the configurator. Thus, the configurator software experts must try to deduce what is incorrect based on such feedback from the product experts. Obviously, this can be a rather troublesome process, for which

reason many of the companies using large configurators use great resources on the creation of external documentation (Edwards, 2005; Hvam et al., 2008).

Strategy 3 holds an advantage over strategies 1 and 2 in the third dimension, minimal use of resources on documentation work. Thus, from a cost-benefit perspective, the choice comes down to an evaluation of the costs of creating external documentation compared to the benefits of having this external documentation.

The fourth dimension, facilitation of the communication between product experts and configurator software experts, is considered to be an advantage of strategy 1, while being a drawback of strategies 2 and 3. Compared to strategies 2 and 3, strategy 1 implies the inclusion of a person responsible for collecting and structuring the information from the product experts before this is handed over to the configurator software experts. The advantages of having a person between the product experts and the configurator software experts are that the configurator software experts can focus only on software implementation, there is a clear demarcation of responsibilities, and the persons that represent product expert expertise (i.e. the knowledge representation expert) can be chosen based on their product understanding rather than IT skills. Concerning the latter advantage, i.e. having separate experts for handling the representation of product expert expertise and for implementing these representations in a configurator, yet an issue should be considered. In many of the cases studied at the Centre for Product Modelling at the Technical University of Denmark, the project of representing product expert expertise was linked with a task of optimising the product assortment, before implementation of product information in the configurator. Thus, the knowledge representation expert also needs an understanding of how to optimise engineered product designs, which typically is not a type of expertise that a software expert possesses.

The fifth and sixth dimension, few handovers of information and instant implementation of gathered information, are considered advantages of strategies 2 and 3 compared to strategy 1. These advantages both relate to the fact that the persons implementing the product information are also the persons who communicate with the product experts, for which reason a handover of information can be avoided and the implementation be initiated sooner.

4.2 Case characteristics promoting different strategies

As mentioned, strategy 1 has a main advantage over the two other main strategies in the facilitation of the communication between product experts and configurator software experts. In general, this advantage point becomes greater, the more complex the case gets, since product complexity calls for persons that: are skilled at understanding the products in focus, can make product experts agree on the product information to implement, and can represent product information in a

manner that is understandable to product experts. As mentioned, it may be difficult to find persons with all these skills, while also being an expert in the configurator software. However, this is not to say that it is not possible, in fact this was to a great extent the case in the companies in which strategy 2 had been applied. The choice of strategy in these cases may partly be explained by the fact that the tasks of creating conceptual models and implementing these in the configurator software were handled by internal personnel instead of consultants, i.e. much time had been given for getting acquainted with the products in focus and the selected configurator software. On the other hand, the companies that used strategy 3, when their configurators had grown in complexity, all of them had experienced problems related to getting an overview of what had been implemented in the configurator and in the communication with product experts. Although strategy 3 represents a way of minimising the use of resources and shortening the duration of the configurator project, much indicates that such benefits are not achieved if the product is of some complexity. In fact the opposite may be the case, since much time must be used on continuously modifying the configurator to correct misunderstandings and adapt the configurator to a behaviour on which the product experts can agree (Edwards et al., 2005; Hvam et al., 2008; Haug and Hvam, 2007c). In worst case such problems may lead to projects being abandoned. Thus, based on the cases studied, it seems that strategy 3 is best suited in projects of little product complexity.

In summation, in cases of simple products, the use of strategy 3 may shorten the project duration and minimise the resource consumption, but it may turn out to be a dangerous path if the complexity of the product information is high. In such cases, much indicates that strategies 1 and 2 are more appropriate. The choice between strategies 1 and 2 comes down to: (1) if single persons with the adequate product understanding, knowledge representation skills and software expertise can be found; (2) if a clear demarcation between the persons responsible for conceptual modelling and configurator development is desired; and (3) if less information handovers and shorter projects is found to be important.

Another issue to consider is that it may be beneficial to combine different strategies during a project. One pattern seen in some of the cases in focus and in other cases studied is that projects starts with strategy 1, and after testing the first real model changes to strategy 2 or 3 to get the last minor changes in place faster. During the maintenance phase, all the strategies 1, 2, 3 or 4 have been applied in the cases in focus and in other cases studied. However, as mentioned, it seems that the companies with the products of the highest complexity are dependent of external documentation in order to be able to further develop their configurators, for which reason in particular strategy 1 or 4 is chosen, sometimes in combination with strategy 2.

5 Conclusions

This paper defined seven distinct strategies for the development of product configurators. The strategies were defined based on a discussion of literature, earlier documented case studies, and the experience of the authors. The strategies were defined by using some basic characteristics that defines configurator development projects. The task of developing product configurators was divided into: retrieval of relevant product information, representation of relevant product information, and implementation of defined product information. The relevant experts were divided into: product experts, knowledge representation experts, and configurator software experts. In the seven named cases, three strategies for the development of product configurators had mainly been applied. The remaining four strategies are characterised by: being applied mostly in the maintenance phase, being at a somewhat experimental stage at the moment, or being merely hypothetical.

The three main strategies combine the defined tasks and experts in different ways. Strategy 1 includes two separate experts for representing the expertise of the product experts in conceptual models and for implementing the conceptual models in the configurator software. In strategy 2 the same persons make the conceptual models and implement these in the configurator. In strategy 3 the information from the product experts are implemented directly without external documentation. The three strategies had each been the main applied strategy in two to four of the named cases (in two cases there had been two main strategies). Thus, there is empirical evidence that these strategies are actually being applied in practice.

The three main strategies were compared concerning strengths and weaknesses and discussed in relation to selecting the right strategy for a specific case. It is argued that strategy 1 is the most suitable for cases of high complexity, while strategy 3 could be problematic in such cases, because it does not include conceptual models that allow product experts to discuss what should be implemented in the configurator. On the other hand, strategy 3 may reduce time and resources needed for developing the configurator, although there is a great risk that it ends up being more costly if problems emerge. Strategy 2 may also be applicable in cases of high product complexity, but may pose a challenge in finding persons that are adequately competent in product design, knowledge representation and configurator software at the same time.

In summation, this paper has provided an important contribution to product configuration literature by clarifying what are the different development strategies and which advantages and drawbacks they hold.

References

Aldanondo, M. and Vareilles, E. (2008). "Configuration for mass customization: how to extend product configuration towards requirements and process configuration", *Journal of Intelligent Manufacturing*, vol. 19, no. 5, 521-535.

Ardissono, L., Felfernig, A., Friedrich, G., Goy, A., Jannach, D., Petrone, G., Schäfer, R. and Zanker, M. (2003). "A framework for the development of personalized, distributed web-based configuration systems", *AI Magazine*, vol. 24, no. 3, 93-108.

Beck, K. and Cunningham, W.A. (1989). "A laboratory for teaching object-oriented thinking", *Proceedings of OOPSLA '89* and *ACM SIGPLAN Notices*, vol. 24, no. 10, 1-6.

Byrd, T.A., Cossick, K.L. and Zmud, R.W. (1992). "A synthesis of research on requirements analysis and knowledge acquisition techniques", *MIS Quarterly*, vol. 16, no. 1, 117-38.

Ceron, R., Arciniegas, J.L., Ruiz, J.L., Duenas, J.C., Bermejo, J. and Capilla, R. (2004). "Architectural modelling in product family context", *Lecture Notes in Computer Science*, vol. 3047, 25-42.

Clancey, W.J. (1993). "The knowledge level reinterpreted: modeling socio-technical systems", *International Journal of Intelligent Systems*, vol. 8, no. 1, 33-49.

Davis, R., Shrobe, H. and Szolovits, P. (1993). "What is a Knowledge Representation?", *AI Magazine*, vol. 14, no. 1, 17-33.

Edwards, K., Hvam, L., Pedersen, J.L., Møldrup, M. and Møller, N. (2005). *Udvikling og implementering af konfigureringssystemer: økonomi, teknologi og organisation*. Final report from PETO research project. Department of Manufacturing Engineering and Management, Technical University of Denmark, Lyngby, Denmark.

Eynard, B., Gallet, T., Nowak, P. and Roucoules, L. (2004). "UML based specifications of PDM product structure and workflow", *Computers in Industry*, vol. 55, no. 3, 301-316.

Felfernig, A., Friedrich, G.E. and Jannach, D. (2000). "UML as domain specific language for the construction of knowledge-based configuration systems", *International Journal of Software Engineering and Knowledge Engineering*, vol. 10, no. 4, 449-469.

Felfernig, A., Friedrich, G. and Jannach, D. (2001). "Conceptual modeling for configuration of mass-customizable products", *Artificial Intelligence in Engineering*, vol. 15, no. 2, 165-176.

Forza, C. and Salvador, F. (2007). *Product information management for mass customization*. Palgrave MacMillan, New York.

Forza, C. and Salvador, F. (2006). "Application support to product variety management", *International Journal of Production Research*, vol. 46, no. 3, 817-836.

Harlou U. (2006). *Developing product families based on architectures - Contribution to a theory of product families*. PhD thesis. Department of Mechanical Engineering, Technical University of Denmark, Lyngby, Denmark.

Haug, A. (2007). *Representation of Industrial knowledge - as a basis for developing and maintaining product configurators.* PhD thesis. Department of Industrial Management and Engineering, Technical University of Denmark, Lyngby, Denmark.

Haug, A. and Hvam, L. (2007a). "A comparative study of two graphical notations for the development of product configuration systems", *International Journal of Industrial Engineering*, vol. 14, no. 2, 107-116.

Haug, A. and Hvam, L. (2007b). "Product Structured Class Diagrams to support the development of Product Configuration Systems", in W.J. Mitchell, F.T. Piller, M. Tseng, R. Chin, B.L. McClanahan (eds.) *Proceedings of the MCPC 2007*, MIT, Boston, MA.

Haug, A. and Hvam, L. (2007c). "The modelling techniques of a documentation system that supports the development and maintenance of product configuration systems", *International Journal of Mass Customisation*, vol. 2, no. 1/2, 1-18.

Haug, A. and Hvam, L. (2009). "CRC-cards to support development and maintenance of product configuration systems", *International Journal of Mass Customisation*, vol. 3, no. 1, 38-57.

Haug, A., Hvam, L., Mortensen, N.H., Lundvald, S. and Holt, P. (2009a). "Implementation of conceptual product models into configurators: From months to minutes", *Proceedings of MCPC 200*9, Aalto University, Helsinki, Finland.

Haug, A., Ladeby, K., and Edwards, K. (2009b), "From Engineer-To-Order to Mass Customization", to appear in *Management Research News*, vol. 32, no. 7.

Hvam L., Pape S. and Nielsen M.K. (2006), "Improving the quotation process with product configuration", *Computers in Industry*, vol. 57, no. 7, 607-621.

Hvam L., Pape S., Jensen K.L. and Riis, J. (2005). "Development and maintenance of product configuration systems: Requirements for a documentation tool", *International Journal of Industrial Engineering - Theory Applications and Practice*, vol. 12, no. 1, 79-88.

Hvam, L. (2004). "A multi-perspective approach for the design of Product Configuration Systems – an evaluation of industry applications", *Proceedings of the International Conference of Economic, Technical and Organizational aspects of Product Configuration Systems*, pp. 13-25, Technical University of Denmark, Lyngby, Denmark.

Hvam, L., Mortensen, N.H. and Riis, J. (2008). *Product customization*. Springer-Verlag, Berlin and Heidelberg.

Hvam, L., Riis, J. and Malis, M. (2002). "A multi-perspective approach for the design of configuration systems", presented at *15th European Conference on Artificial Intelligence*, Lyon, France, July 21-26 2002.

Mortensen, N.H., Yu, B., Skovgaard, H. and Harlou, U. (2000). "Conceptual modeling of product families in configuration projects", *Papers from the Workshop at ECAI , 14th European Conference on Artificial Intelligence*, pp. 68-73, Humboldt University, Berlin, Germany.

Männistö, T., Peltonen, H., Soininen, T. and Sulonen, R. (2001). "Multiple abstraction levels in modelling product structures", *Data & Knowledge Engineering*, vol. 36, no. 1, 55-78.

Oddsson, G.V. (2008). *Indlejret produktkonfiguration hos Grundfos Strukturering af produktviden*. PhD dissertation. Department of Management Engineering, Technical University of Denmark, Lyngby, Denmark.

OMG (2005). *Unified Modeling Language: Superstructure: Version 2.0 (formal/05-07-04)*. OMG, Nedham, MA.

Pine, B.J., Victor, B., and Boynton, A.C. (1993). "Making Mass Customization Work", *Harvard Business Review*, vol. 71, no. 5, 108-119.

Riis, J. (2003). *Fremgangsmåde for opbygning, implementering og vedligeholdelse af produktmodeller*. PhD dissertation. Department of Manufacturing Engineering and Management, Technical University of Denmark, Lyngby, Denmark.

Speel, P.-H., Schreiber, A., Joolingen, W. and Beijer, G. (2001). "Conceptual models for knowledge-based systems", in *Encyclopaedia of Computer Science and Technology*, Marcel Dekker Inc., New York.

Studer, R., Benjamins, V.R. and Fensel, D. (1998). "Knowledge Engineering: principles and methods", *Data & Knowledge Engineering*, vol. 25, no. 1-2, 161-197.