NordDesign 2010
August 25-27, 2010
Göteborg, Sweden

Information Modelling for Automotive Configuration

Anna Tidstam **Johan Malmqvist**
Department of Product and Production Development
Chalmers University of Technology
SE-412 96 Göteborg
SWEDEN
tidstam@chalmers.se

Abstract

The topic of the paper is information models that are the basis for product configuration in the automotive industry. The subject is examined from both a theoretical and a practical perspective. In an empirical study, the product configuration information model of an automotive firm is mapped out and issues perceived as challenging by the firm identified. A comparison of the theoretical configuration models and practical models is presented. We identify areas in which the theoretical models offer more expressiveness that the automotive firm currently utilizes. However, we also observe that many of the firm's issues with configuration have much do with the product's design, and processes and methods for working with the information model, not just its information structure.

Keywords: Configuration, information model, automotive, complexity, documentation

Background

The automotive industry deals with mass customization, which increases the product and manufacturing complexities. The complexity is a key cost driver. Maintenance of complex systems is expensive – the annual cost being roughly 10 % of the system cost [1]. Consequently, complexity management is increasingly becoming essential for profitability [2]. According to [3], product complexity has three main elements: (a) the number of product components, (b) the extent of interactions between these components and, (c) the degree of product novelty. This paper studies the second aspect of product complexity, the interactions, with a focus on product configuration. We specifically consider the requirements in the automotive industry, which manages products with a high complexity in terms of number of potential variants on sale to customers. With the vehicles also becoming more complex, it is increasingly challenging to ensure a high quality of the product documentation [4]. Today, many automotive companies argue that commercial Product Data Management (PDM) systems cannot support configuration processes as complex as in the automotive industry [5]. Consequently, they have developed their own systems for configuration management, such as KOLA at Volvo, SPECTRA at Scania and SMARAGD at Mercedes. However, we have noticed signs of significant similarities in the product configuration approaches within the truck and car manufacturers that we collaborate with. Our hypothesis is therefore that there is a potential for creating a generalization of the product configuration methods and models used in the automotive industry. An adjacent task is to pinpoint more precisely in what ways

practically used automotive product configuration models are different from those presented in the literature. Specifically, we investigate:

RQ1: Which elements, relations and rules are included in published theoretical product configuration information models?

RQ2: What product information is used in automotive configuration in practice, considering specifically the feature and item structures? What practical issues can be identified?

RQ3: What are the similarities and differences between the practical and theoretical product configuration information models?

The task is complicated by the circumstance there is no commonly accepted theoretical product configuration information model. Mechanical engineering as well as software engineering researchers have proposed a number of models, but there is no consensus on what constitutes the common ground. To this end, the paper also introduces a framework for comparing product configuration (information) models.

The remainder of this paper is structured as follows: Section 2 describes our research approach. Section 3-5 contains results from the empirical study, from the literature review and from a gap analysis. Conclusions are listed in Section 6. Finally, section 7 outlines our plans for future work.

Research approach

We conducted the research in the following sequence: We first established a basic framework for product configuration information models that identified the key terms and possible variations across these. We then conducted an empirical study in an automotive firm. We mapped out their product configuration model, aiming at validating the basic framework and identifying additional issues in the area. The information sources for the empirical study were (a) the system implementation documentation of the PDM system at the automotive OEM, (b) the PDM learning material at the company and, (c) discussions during work group meetings with both academic and industrial representatives. One researcher was present at the automotive firm site during a longer period to further understand their configuration approach. This was followed by a more comprehensive literature analysis where the constructs of various theoretical models were contrasted against the basic framework and industry issues. We shifted back and forth between theory and empiricism in an iterative process. A table showing all studied product data information models was finally created, where strengths and weaknesses could be pointed out and analyzed.

Framework for product configuration models

In order to provide a terminology for the discussion of the empirical data and the theoretical models, a framework for product configuration with the key terms and their variations is defined. Essentially, the framework is similar to the K- & V-matrix model [15].

The framework consists of two structures: the feature and item structures, and two kinds of rules: configuration rules and item usage rules. The features are what a customer selects when ordering a configurable product, such as engine, colour, and styling package. A feature family is then defined as a class of options, from which the customer selects one, a feature variant. An item is closely related to parts that are manufactured (although it is a more general term, it can also be, eg software or documentation).

Table 1. Example of elements in product configuration framework. Example inspired by [6].

ITEM USAGE RULES	Red	Blue	Driveable	Not driveable	Turnable	Not turnable		Red	Blue	Driveable	Not driveable	Turnable	Not turnable	CONFIG-URATION RULES
OFFICE CHAIR	x	x							x		x			Rule 1
> UNDER-FRAME			x	x						x			x	Rule 2
>> STAND					x	x								
- - Red, driveable, turnable stand	x		x		x									
- - Red, not driveable, turnable stand	x			x	x									
PARENTS:														**PARENTS:**
OFFICE CHAIR			■	x				■						Red
UNDER-FRAME				■	x				■					Blue
STAND					■					■				Driveable
											■			Not driveable
												■		Turnable
													■	Not turnable
ITEM STRUCTURE			OFFICE CHAIR	UNDER-FRAME	STAND		**CHILDREN :**	Red	Blue	Driveable	Not driveable	Turnable	Not turnable	**FEATURE STRUCTURE**

The framework further identifies two kinds of rules: Configuration rules create constraints for the selection of features (variants). Item usage rules connect features to items in order to provide downstream system with the proper information, such as bill-of-materials. Table 1 provides an example of these structures and rules. The feature structure for an office chair includes the feature families *Colour*, *Driveability* and *Turn ability*, each with two or more feature variants specified. Configuration rules for features state compatibility expressions, e.g. the rule number 2 in Table 1 states that driveable office chairs have to be turnable (expressed as that they may not be not turnable). One example of an item is the assembly for red, driveable and turnable stand. Item usage rules create rules between feature and items are also shown in Figure 1 for the office chair, where e.g. the red, driveable and turnable stand should be used when the features variants red, driveable and turnable are selected by a customer. More objects may be added, for example sales packages that group feature variants.

Obviously, the product configuration information model needs to support the sales-to-order process, but it also needs to support the product modification process. For a configurable product this process typically starts with the proposed introduction of new feature family, e.g. *Upholstery*, which is then introduced in the feature structure. A range of variants for the new feature family is determined (eg Textile, Leather or Plastic). Introduction of new feature families and variants often requires new items to be developed, ie physical parts and assemblies, software, interfaces and documents. This is followed by the creation of configuration rules that define the applicability of the new feature variants and of item usage rules that connect the proper items to the feature variants.

As indicated in the table, item structures as well as feature structure may be complex and contain parent-child relationships. In general, more expressive information models allow more complex relations within and between feature and item structures to be managed. This brings us to an evaluation scheme for product configuration models. The support provided by the product configuration information models is characterized using the ratings *non existence* (□), *low* (○) and *high* (●), according to Table 2.

Table 2 – Evaluation criteria for product information models

ITEM STRUCTURE	ITEM USAGE RULES	FEATURE STRUCTURE	CONFIGURATION RULES FOR FEATURES
□ No items	□ No item usage rules	□ No features	□ No configuration rules
○ Hierarchic structure	○ Personal judgement for good selections of feature variant-item combinations	○ Flat feature structure	○ Personal judgement for good selections of feature variant combination in rule
● Hierarchic structure with views	● Defined practice for selecting suitable feature variant-item combinations	● Hierarchic feature structure	● Defined practice for selecting suitable feature variant combination in rule

Automotive empirical study

The empirical study constructed an industrial example of an automotive product configuration model, which was modelled using Unified Modelling Language [7]. Attributes, cardinalities have been taken away and object class names renamed due to confidentiality reasons, but the model clearly shows that the firm's product configuration model has a fundamental core that is similar to the framework proposed in section 3, Figure 1. It also features some additional elements, to be discussed below.

4.1 The automotive firm's product configuration model

The top-most element of the firm's configuration model is a Product type, a basic classification of high-level product types such as, in the automotive industry cars, trucks, buses. On this level, there is little or no commonality between products and little or no shared product documentation. On the next level, the Product family level (corresponding to a market segment such as *compact size cars* or *luxury cars*) documentation is tied. Product families are then concretized into Product models (corresponding to, eg, a BMW 3 series Coupe) which is the entry point for a customer. In the view of a customer, a product model is defined as a structure of Feature families (Engine, Upholstery, Tyres…), each enabling a choice between several Feature variants (Engine-1.8D, Engine-2.0D...). Choosing feature variants for each feature family results in a vehicle order specification that can be sent to the downstream systems that support the manufacturing and delivery of the vehicle. The product family defines the variance limitations of the product documentation, while the product model is used for packaging the configuration rules within these variance limitations. Item usage rules connect the feature to the items (parts etc) that will be used to build the car.

Let us now classify the information elements according to our framework:

Feature structure (○ = *low*): The feature structure at the firm is essentially flat. One *product family* covers one market segment, such as e.g. *compact size* cars and *luxury* cars. The product model contains a set of features, where none are optional. This can be achieved by creating feature variants also for not including the feature. A limited number of parent-child relationships have been created between features.

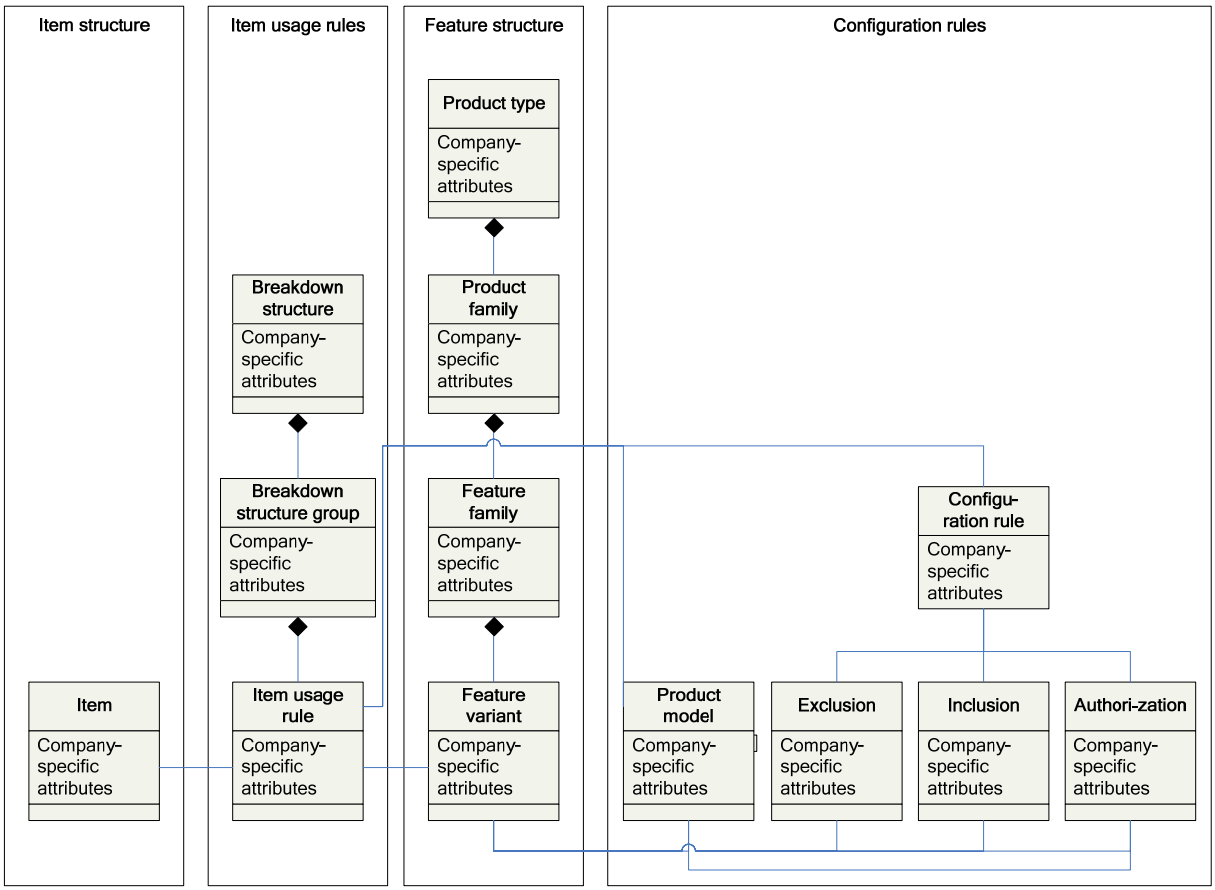

Figure 1. Example of product configuration information model used within the automotive industry (simplified due to confidentiality reasons)

Configuration rules for features (○ = *low*): The configuration rules are of three different types: *exclusions*, *inclusions* and *product model authorizations*.

In line with [8], there are two types of configuration rules that constrain how feature variants may be combined: the *exclusions* and the *inclusions.* The exclusions identify feature variant combinations that are forbidden. One example of an exclusion expression is that *Transmission torque 800 Nm* cannot be combined with *Engine torque 1200 Nm*, since the engine then would be too powerful for the transmission. Inclusions are feature variant pairs with forced values. If one of the feature variants in the pair is selected, the other feature option also has to be selected (IF-THEN). The product model authorizations may be interpreted as a selection of feature variants that may be selected for a specific product model. If the product model is the BMW *3-series cabriolet*, it may be possible to select among *retractable hardtop* and *soft-top* options, while the roof hatch feature variant cannot be selected.

The configuration rules are formulated in a centralized process, with personal judgement of an expert for what is an efficient formulation. Rules can be stated in several ways. When the configuration rules are requested by design engineers, the selection of feature variants to be included is a personal judgement for what is suitable for his/her component. An expert on configuration rules then reformulates the configuration rules, with the aim to write as short and few configuration rules as possible due to efficiency reasons and system limitations. Thus a designer's configuration rules may be reformulated if the expert can find combinations of configuration rules that are simpler and that reach the same end result for the complete

vehicle. This is efficient for the experts and the PDM system, but to the cost that design engineers requesting the configuration rules may have difficulties in finding and verifying them. Another shortcoming is that there is no formalized methodology for how to relate configuration rules to the item usage rules, which would have improved the understanding of the configuration rules for the entire organization. In summary, it is clear that the same information model may result in different data sets depending on how it is used.

Item structure (○ = *low*): There is a flat structure for items. No parent-child relations between items were found in the studied system. Indirectly, the grouping of items can be derived from how the item usage rules are documented. It should be pointed out that the firm has many more enterprise systems that use item information. Some of these utilize systems item hierarchies but these are outside of the scope of the current study.

Item usage rules (○ = *low*): The item usage rules connect feature variants and items. The item usage rules are grouped into vehicle functions or vehicle modules, but this does not provide any formulation support other than looking at what has already been done for item that should be replaced. The number of feature variants included in expressing the item usage rules for a vehicle function, or vehicle module, e.g. a fuel tank, can be very high. One attempt to limit this number of feature variants is to formulate as short item usage rules as possible, again with the cost for the design engineers to have more difficulties when searching for item usage rules in the database. If the user would like to use a feature (variant) not previously used within the vehicle function or module, there is a personal judgement by a senior designer responsible for accepting the selection. Again, the method aspect comes to the front to ensure that the firm gets the "right" data based on its information model.

4.2 Discussion

It thus seems that the automotive firm operates a comprehensive, but not in every dimension maximally expressive information model: hierarchies or views are not utilized for features or items, for example. As a result, the ratings are "low" throughout. We will in the next section see that there are propositions in the literature for more expressive information models. One exception is the item classification structures (vehicle function structure, vehicle modules) for which it is difficult to find theory-based correspondents.

Some challenging issues lie beyond the expressiveness of the information model, and have more to do with how it is used. As exemplified for both configuration and item usage rules, the difficulty lies also in finding the right balance in writing rules that are both easily understandable for non-experts (designers) and efficient for experts and the database system. It can also be argued that the additional complexity in the automotive industry mainly lies in the amount of data in itself. The firm maintains a very large number of features, feature variants, configuration rules and item usage rules. This complexity apparently has not resulted in an information model that is radically different than those proposed by researchers as such, however. In a sense, this makes the problem more challenging as many different kinds of mitigative actions can be envisaged: more expressive information models, better process support, more powerful and scalable database architectures.

Due to confidentiality reasons, the attribute of the information objects are not shown in Figure 1. The firm does operate a number of such attributes. Along the same path as the above discussion, the attributes also need to be coupled to proper methods and processes if the result is to be data of high quality. It would be difficult to construct a theoretical model or a commercial system that predicts all company-specific attributes.

Literature review

Let us now turn back to the literature to review to what extent it suggests information modelling concepts that meets the needs of the automotive firm we studied. The literature review included mechanical engineering as well as software engineering papers. The evaluation of the models was done according to the evaluation criteria presented in Table 2. Due to space limitations, non-existence for an element in the framework is in some cases not presented below.

Generic Bill of Material [6]

Item structure (● = *high*): The office chair in [6] is described with items (generic items) with attributes defined by features (*colour red/blue/grey, with/without stand etc)*. The items have parent-child relations.

Item usage rules (● = *high*): The features for formulating the item usage rules may be found from the inherited features higher up in the item structure, e.g. the feature *Colour* is used on the complete product generic item *Office chair*, which then becomes a feature that can be used on all other generic items.

Feature structure (● = *high*): The feature variants may have indicated parent-child relationship to other features, e.g. the feature variants *With arm rest* is the parent to the feature variant *Arm rest finish*. On the other hand, in most cases, there are no relationships between the feature variants (except of item usage rules and configuration rules). McKay *et al*. provide a disclaimer in observing that the example of an office chair is too small to show more parent-child relations for features.

Configuration rules for features (○ = *low*): During the formulation of configuration rules for feature variants, there is no support from the product information model showing the features as a list. However, since McKay *et al*. claims that the feature variants may contain parent-child relationships it may be assumed that there is some support for formulating the configuration rules.

Feature models [9]

Feature structure (● = *high*): In [9] there is an example showing the *Interior control* of a car where there are several parent-child relations between features which shows the intention of the model.

Configuration rules for feature (● = *high*): The feature structure with parent-child relations supports the selection of features when formulating of configuration rules.

Probabilistic feature models [10]

Feature structure (● = *high*): In [10] there is a description for how to create a *probabilistic* feature model by using the information that configuration rules are carrying concerning the allowed variant combinations. *Soft constraints* are introduced to express constraints between features that are satisfied by most configurations (do not need to be satisfied for all configurations). For example, it may not be true that no *Manual gear boxes* can be built in *North America*, but according to the probabilistic configuration rules the combination is highly unlikely. The highest probability is that the *Gearbox* will be *Automatic* in *North America*, which creates a strong relation between this market and this feature variant.

Configuration rules for features (○ = *low*): The probabilistic feature models shows which features that have strong relations, which may be considered as giving a rough representation of the configuration rules set.

Product Configuration view to Software Product Families [11]
Feature structure (● = *high*): In [11] there is another model presented, which has several levels of hierarchy in the variant structure, but also the features of *ports* and *resources*.
Configuration rules for features (○ = *low*): The configuration rules are stored in a separate database. Nothing stated about how to formulate configuration rules.

Configurable components [12]
Feature structure (○ = *low*): A *configurable component* [12] is intended to be regarded as a system construct containing items, with a feature interface that makes it possible to specify the configuration of the component. Nothing is mentioned concerning the relationships between features (excluding item usage rules and configuration rules).
Item usage rules (○ = *low*): There are two methods to create the instantiation of the product structure, either by recursive instantiation of component instances that interact, or by complete variant specification. The feature interface controls which feature variants that may be used in the item usage rules, but there is no explanation how this interface is created.
Item structure (○ = *low*): There are no relations stated between items.
Configuration rules for features (● = *high*): There are no global configuration rules; instead the configuration rules are stored locally within the configurable components. The locally stored configuration rules have to be able to describe all possible feature combinations for the configurable component. As with the item usage rules, it is the feature interface that controls which features that may be used for the configuration rules.

System-based product modelling [13]
Item structure (● = *high*): In system-based product modelling [13], products are represented by systems, which may contain parent/child decompositions to single items. There is further a method for representing functions.
Item usage rules (○ = *low*): It is not explained how the selection of features is done to formulate the item usage rules.
Feature structure (○ = *low*): The features are managed by the interface control objects. It is not described what these interface control objects contains. Feature variants have identical interfaces within the feature family. Each interface may link to multiple systems or modules, and any system or module may be linked to any interface.
Configuration rules for feature (□ = *non-existence*): The configuration rules are substituted by interface control objects. The aim is to create a configuration model by *networks of interactions.*

Product Family Master Plan [14]
Item structure (● = *high*): The *Product Family Master Plan* [14] has an item structure based on parent-child relations, and supports multiple levels of abstraction, ie function, organ and part.
Item usage rules (○ = *low*): Items within the item structure may be interchanged due to the selection of features. Harlou does not present any instructions on how to formulate item usage rules.
Feature structure (● = *high*): The feature structure may contain parent-child relations between features.
Configuration rules for feature (○ = *low*): The configuration rules are located within a specific class, even if it is using features between several classes. There is no information about if the item structure use inheritance for suggesting features.

The K- & V-matrix [15]
Item structure (● = *high*): The V-matrix [15] is similar to a Design Structure Matrix, capable of both showing modular structures and incompatibilities.
Item usage rules (○ = *low*): The K-matrix represents the item usage rules but does not explain why the particular features (variants) have been selected for the relation.
Feature structure (○ = *low*): With the targeted compatibility application of the V-matrix, it does not present any parent-child relationships between features.
Configuration rules for features (□ = *non-existence*): The (in)compatibility between features presented in the V-matrix is not sufficient to formulate more complicated configuration rules than incompatibility between feature variant pairs, which makes it too simple for automotive applications.

QFD method [16]
Item structure (● = *high*): The items in the QFD [16] may be related to each other using the roof in the House of Quality.

Complexity Manager [17]
Item structure (● = *high*): In the software *Complexity Manager* [17], the item structure is derived from the assembly sequence and contains all buildable assemblies.
Item usage rules (□ = *non-existence*): Nothing mentioned concerning item usage rules.
Feature structure (● = *high*): The feature structure is created by using the configuration rules for calculating the tree structure for features.
Configuration rules (○ = *low*): Configuration rules exist and are the foundation for the feature structure.

Software Configuration Management [18]
Item structure (● = *high*): Items may include e.g. requirements, design documents, source code, test plans, user and maintenance manuals and interface control documents. During the identification of configuration items, also their (tree) structure is defined. Within the item structure, also dependencies as "requires" and "conflicts" between items may be formalized, as within Debian package management [18].

Discussion
The review of both theoretical and empirical product information models has shown that the framework presented in Table 1 is capable of describing all models. The empirical study and the literature review resulted in different ratings of the expressiveness of the product configuration models, which are going to be presented and discussed in the next section.

Gap analysis and discussion
The aim of the gap analysis, see Table 3, was to identify possible aspects in which the complexity is indeed special in the automotive industry, and to find suggestions for how to use the product information model for defining recommended practices for formulation of item usage rules and configuration rules. The feature structure, item structure, item usage rules and configuration rules were evaluated according to the criteria set out in Table 2, with *high* fulfilment (●), *low* fulfilment (○), and *non-existence* (□). For more details about the motivation for the evaluations, please refer to section 5. The table can be read as follows: The *Feature model* does not contain any items, which gives the evaluation *non-existence* for both the item structure and the item usage rules, but high support for the feature structure. Etc.

Table 3. Gap analysis

□ *non-existence*
○ *low*
● *high*

	ITEM STRUC-TURE	ITEM USAGE RULES	FEATURE STRUC-TURE	CONFIG-URATION RULES
Empirical study				
Automotive product configuration model	○	○	○	○
Literature review				
Generic BOM	●	●	(●)	○
Feature models	□	□	●	○
Probabilistic feature models	□	□	●	○
Product configuration view to software product families	□	□	(●)	□
Configurable components	○	○	○	●
System-based product modelling	●	○	○	□
Product family master plan	●	○	(●)	○
K- & V-matrix method	●	○	○	□
QFD method	●	□	□	□
Complexity Manager	●	□	●	○
Software Configuration Management	●	□	□	□

In Table 3, we notice that the automotive firm uses all four concepts but also that it does not utilize the full modelling expressiveness offered by some theoretical models. Mainly this has to do with hierarchical modelling (parent-child relationships): the firm's feature and item structures are "flat", as well as its set of configuration rules. Making use of the theoretical models capabilities for hierarchical modelling would enable the firm to modularize its feature structure and configuration rule set and thus parallelize development work and validation with benefits in terms of costs and improved quality. The integrated nature of the products is however a challenge: it is not known to what extent a configuration rule set can be modularized, even if the modelling mechanisms were available.

We further notice that there does not seem to be a theory that excels in all four categories. Selecting the "best" model would entail picking parts from several theories. It then seems that Generic BOM:s provide good support for item, feature and configuration rule modelling. It can be complemented with configurable components for formulation of configuration rules as they provide local configuration rules valid for limited scope of the vehicle, and by feature modelling for defining feature families and feature variants. A common weakness for all theories is the low ability to support users to formulate item usage rules.
Let us return to the issue of the uniqueness of configuration in the automotive industry. This uniqueness is not a myth; it is evidenced through the up-to-now failure of commercial software vendors to develop accepted solutions for the industry. However, the results from this study seem to suggest that this uniqueness has more to do with company-specific attributes, the integrated nature of the products, methods and processes than with the structural composition of the product configuration information model. Through adapting some concepts from the literature, the automotive firm we studied would obtain a more expressive information model, but other issues would require efforts with a method, process and product focus rather than information structure focus.

Admittedly, the analysis done based on Table 3 is coarse and can be refined, eg by examining in more detail which of the mechanical engineering models [12][13] that provides the best

support for the item domain. A more careful examination on the attribute level could also reveal the differences between the theoretical models as regards to attributes and the company-specific attributes. On this level, confidentiality will be an issue, however.

Conclusions

Theoretical product configuration information models typically include items, features, configuration rules and item usage rules. There is no commonly accepted model, but rather different model have different strengths: generic BOM:s for item structures and item usage rules, configurable components for configuration rule modelling and feature models for features. The real automotive product configuration information model that we studied is based on the same fundamental components, and differences as compared to the theoretical models were found on the attribute rather than structural level. The theoretical models provide concepts that could increase the expressiveness of the industrial one for items, features and configuration rules, but item usage rules is a less developed area. Practical challenges further emerge from method and process aspects rather than the information content – how should an information model be used in practice? Theoretical product configuration models need to go beyond pure information modelling to include methods for writing, eg, "good" configuration rules and item usage rules in order to increase their utility for industry.

Future work

This paper has reported on the results from the first phase of a PhD project on "Configuration rule management", and has served to establish an information model that will be further explored in the remainder of the project. Future work will be more method and process-oriented and address the design engineer's ability to formulate configuration rules by support of the product information model: *Is it possible to develop a product configuration information model with associated methods that enables design engineers rather than to state item usage and configuration rules?* Future work also includes the evaluation if it is possible to generalize the results for automotive industries by conducting more empirical studies. *Is the automotive product configuration information model found during the empirical study possible to generalize across automotive companies?* In parallel, we are collaborating with a team of computer scientists to develop computationally efficient configuration algorithms.

Acknowledgements

This work was carried out within the Wingquist Laboratory VINN Excellence Centre at Chalmers University of Technology. It was supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA). This support is gratefully acknowledged.

References

[1] Suh, N. P., "Axiomatic Design Theory for Systems", *Research in Engineering Design*, Vol. 10, 1998, pp 189-209.

[2] A.T. Kearney, "Study on: Complexity Management – Chances Amid the Crisis", 2009.

[3] Novak, S. and Eppinger, S. D., "Sourcing By Design: Product Complexity and the Supply Chain", *Management Science*, 2001, pp. 189 – 204.

[4] Mesihovic, S., "On the Development and Sales-Delivery Process of Configurable Products", LicEng thesis, Department of Mechanical Engineering, Chalmers University of Technology, Göteborg, Sweden, 2004.

[5] Nyström, B., and Smidek, F., ”Hit men inte längre!”, *Verkstadsforum*, Vol. 2, 2008, pp 13-14. In Swedish.

[6] McKay, A., Erens, F. J., and Bloor, M. S., “Relating Product Definition and Product Variety”, *Research in Engineering Design*, Vol. 2, 1996, pp 63-80.

[7] Miles, R., and Hamilton, K., “Learning UML2.0”, O'Reilly Media, 2006.

[8] Euwe, M. J., and Shuwer, R. V., “Configuration of complex products”, *Computers in Industry*, Vol. 21, 1993, pp 1-10.

[9] Bühne, S., Lauenroth, K. and Pohl, K., “Why is it not Sufficient to Model Requirements Variability with Feature Models?”, *Automotive Requirements Engineering*, AURE04, 2004, pp 5-12.

[10] Czarneck, K., She, S. and Wasowski, A., “Sample Spaces and Feature Models: There and Back Again”, *Software Product Line Conference*, 2008.

[11] Männistö, T., Soininen, T. and Sulonen, R., “Product Configuration View to Software Product Families”, *10th International workshop on Software Configuration Management*, SCM-10, 2001.

[12] Claesson, A., “A Configurable Component Framework Supporting Platform-based Product Development”, PhD thesis, Product and Production Development, Chalmers, 2006, pp 98-100.

[13] Collier, W., “A Common Specification for Systems-Based Product Modelling”, DH Brown Associates, 1999.

[14] Harlou, U., “Developing Product Families based on Architectures”, PhD Thesis, Department of Mechanical Engineering, Technical University of Denmark, 2006.

[15] Bongulielmi, L., Henseler, P., Puls, C., Meier, M., “The K- & V-Matrix Method – An Approach in Analysis and Description of Variant Products”, *Proceedings of the 13th International Conference on Engineering Design*, ICED 01, 2001, pp 571-578.

[16] Akao, Y., “Development History of Quality Function Deployment”, Asian Productivity Organization, 1994, pp 339.

[17] Schuh & Co., ”Complexity Manager – Software for visualizing and optimizing variant variety”, www.schuh-group.com

[18] Di Cosmo, R., “Report on Formal Management of Software Dependencies”, Environment for the Development and Distribution of Open Source Software, FP6-IST-004312, 2006.