INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED13
19-22 AUGUST 2013, SUNGKYUNKWAN UNIVERSITY, SEOUL, KOREA

OBSTACLES AND DEVELOPMENT OF SUPPORT FOR TRANSLATION OF CONFIGURATION RULES

Anna TIDSTAM, Johan MALMQVIST
Chalmers University of Technology, Sweden

ABSTRACT

Collaborative product development is a method for reducing costs, but there is a risk of severe obstacles during the product documentation exchange. Product documentation for configurable products describes how parts can be combined with so-called ‘configuration rules’. This paper is based on a case study of the configuration rule exchange in a collaboration project between two automotive firms. The research approach was to create information and process models for the configuration rule exchange, and then to discuss obstacles and improvements. The results showed that the main obstacles were the difference in authoring methods, causing difficulties in detecting changes as well as a time-consuming reformulation of configuration rules. An implemented algorithm addressing the change detection serves its purpose to demonstrate the effort required to overcome this obstacle during a configuration rule exchange. The identified obstacles, and the effort to overcome them, are crucial to understand in order to motivate and direct further research of improved configuration rule exchange support.

Keywords: product lifecycle management, product modelling, collaborative design, exchange

Contact:
Anna Tidstam
Chalmers University of Technology
Product and production development
Gothenburg
41296
Sweden
tidstam@chalmers.se

1 INTRODUCTION

Collaborative product development is a means for reducing development time and lowering organizational risk (Bruce et al., 1995). In collaborative product development for configurable products, the need for exchanging product data goes beyond single parts and assemblies, and also has to include ‘configuration rules’. Configurable products, such as cars, are defined by a set of features, e.g. ‘engine size 1.8 liters’ and ‘exterior color red’. The configuration rules are logic expressions that control the selection of feature variants, for example that the ‘exterior color red’ should not be possible to choose by customers who also want ‘engine size 1.8 liters’.

Three approaches to product data exchange are known in the literature: (1) to use a neutral file, (2) to remodel the product data, or (3) to use a data instance mapping (Markson, 2007). The neutral file approach involves a translation from the native format to a neutral file format and then a translation again to the receiving system (Pratt, 2001). One well-known product information model standard, first published in 1994, using the neutral file approach is the STandard for Exchange of Product model data (STEP) (ISO, 1994). The use of a standardized information model can be efficient when there are available translators, as is commonly the case for part geometries. When there are no translators, as for configuration rules, they need to be developed for the specific exchange and may be difficult to motivate financially for a single case. Consequently, there is a lack of industrial validation for neutral file formats for configuration rules (Viel, 2003; Hirel and Hug, 2009). The second exchange approach is to remodel the data. This approach means that one of the companies needs to modify its configuration information model in order to be able to exchange the configuration rules. This second approach is too expensive for collaborative product development, as a remodeling affects downstream systems, e.g. manufacturing. The third approach, the data instance mapping then needs to be the approach applied in practice for exchange of configuration rules. The data instance mapping uses the data instances from Company A to map data instances at Company B, see Figure 1. A mapping is ‘one-to-one’ if every data instance from Company B is mapped by *at most one* data instance of Company A (Zeuthen, 1870). As shown in the figure, the mapping is instead of the type ‘onto’ if a data instance at Company B is mapped by *more than one* data instance from Company A (MacDuffee, 1940). The data instance mapping is an exchange approach where little research has been conducted.

Company A	Company B
‘red’ AND ‘sport’	‘red’ AND ‘sport’
‘red’ AND ‘classic’	‘red’ AND ‘classic’

Company A	Company B
‘red’ AND ‘sport’	‘red’ AND (‘sport’ OR ‘classic’)
‘red’ AND ‘classic’	

Figure 1. Data instance mappings illustrating left) ‘one-to-one’ and right) ‘onto’.

For configuration rules, one-to-one mappings have the benefit that a single configuration rule at company A is equivalent to a single configuration rule at company B. In onto mappings, a reformulation of the configuration rules needs to take place as more than one configuration rule has to be equivalent to a single configuration rule. ‘Onto’ mappings are probable during configuration rule exchanges, as companies have specific needs and practice for authoring configuration rules (Tidstam and Malmqvist, 2011).

During an inter-organizational exchange of product data, it is common with product data exchange issues (Domazet et al., 2000). There are exchange issues independent on information models, such as correctness, completeness, consistency, coverage and currentness (Killick, 1993). That type of exchange issues is however not studied in the present study, as it is assumed that the activities to assure high quality data have already taken place before the exchange. This is a recommended practice from (Chow and McElroy, 2002). Other exchange issues are described in (Fang et al., 1991; Naiman and Ouksel, 1995) as three aspects of information model comparability: (1) ‘naming’, refers to class, attribute or instance name issues, e.g. synonyms, (2) ‘abstraction’ refers to relationships between classes, e.g. ‘x’ is a generalization of ‘y’, and (3) ‘heterogeneity level’ is naming/abstraction conflicts on classes, attributes or instance levels. Another exchange issue that is mentioned in (Fang et al., 1991) is differences in IT systems, as for example how configuration rules are visualized in the two in-house developed IT systems in the present study. Some research on information exchange issues has thus been carried out, but, to the authors’ knowledge, no study has specifically focused on an exchange of configuration rules. In order to identify obstacles during a configuration rule exchange, this paper will describe the exchange process in detail. The following research questions have been addressed:

RQ1: What does an exchange process for product configuration rules between collaborating companies look like?

RQ2: Which configuration rule exchange obstacles between collaborating companies are identified?

RQ3: How can the exchange of configuration rules be improved?

The remaining parts of the paper are organized as follows: In Section 2 the research approach is described; Section 3 presents the results; Section 4 discusses the results; and in Section 5 the conclusions are stated. Section 6, finally, outlines the future work.

2 RESEARCH CONTEXT, QUESTION AND METHOD

The research process consisted of three phases: (1) analysis, (2) development and (3) testing, see Figure 2. The analysis phase included a set of parallel activities: creation of configuration information models, creation of exchange process model, as well as an analysis of automation potential. An algorithm development was iterated with a validation during the testing phase. Results addressing the three research questions are also listed in the figure.

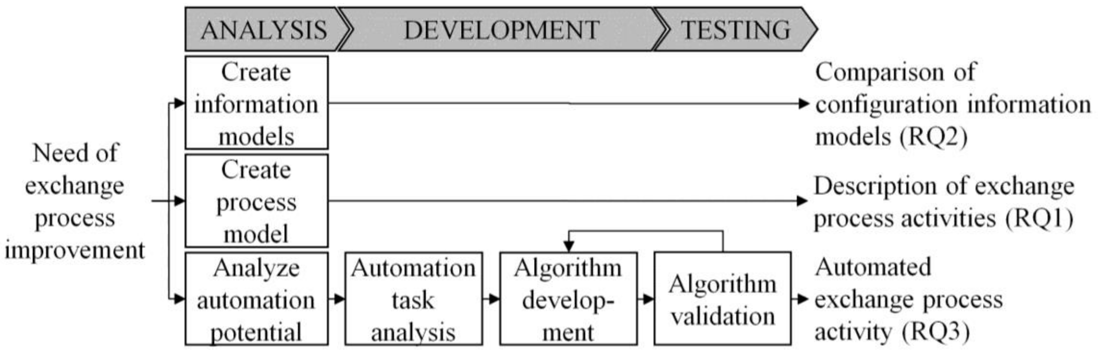

ANALYSIS
DEVELOPMENT
TESTING
Need of exchange process improvement
Create information models
Create process model
Analyze automation potential
Automation task analysis
Algorithm develop-ment
Algorithm validation
Comparison of configuration information models (RQ2)
Description of exchange process activities (RQ1)
Automated exchange process activity (RQ3)

Figure 2. Research activities and results with references to research questions.

3 RESULTS

The two collaborating automotive manufacturing firms that are involved in this study, henceforth called Alpha and Beta, are two major European manufacturing firms selling hundreds of thousands of cars per year. Consequently, these two companies have developed PDM systems that support their specific business needs for configuration rules. Alpha has the development responsibility during the studied collaboration project, and it is Alpha's responsibility to ensure that the exchanged configuration rule set is correct and complete. Beta receives the developed configuration rules, but has to translate them into its own language in order to be able to sell and manufacture the vehicle. The result section is structured in three subsections: 3.1 Analysis, 3.2 Development and 3.3 Testing.

3.1 Analysis

The analysis phase begins with describing Alpha's and Beta's information models, and then continues with the exchange process description, and finally the analysis of automation potential.

Configuration information models at Alpha and Beta

Alpha's configuration information model is shown in Figure 3, and Beta's configuration information model in Figure 4. Alpha's configuration information model contains a 'configuration rule matrix'. Such matrices show allowed and restricted combinations of feature variants with configuration rule values {S, D, O, -}, where 'S' = systematic, 'D' = default, 'O' = optional, and '–' = restricted. The systematic value indicates that it is the only allowed combination. The default value indicates a default combination among more several optional combinations. The restricted values indicate which combinations that are not allowed to be selected.

A configuration rule matrix is a user generated view based on a selection of 'minor' feature families, whose related major feature families are imported due to the minor to major relation. The user then authors configuration rules by assigning configuration rule values {S, D, O}. If a minor to major relation is modified, all configuration rule matrices with the minor feature family will be modified.

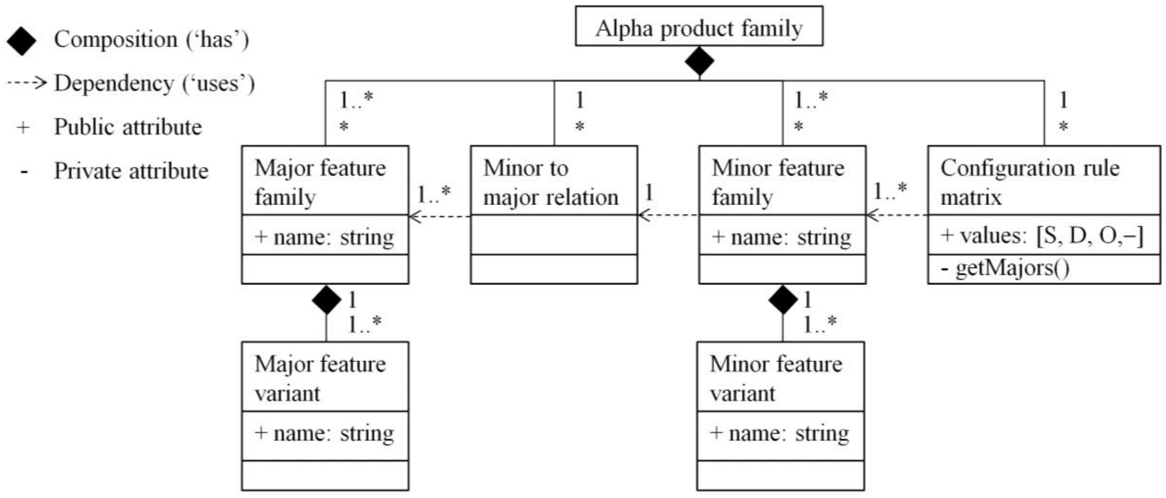


Figure 3. Configuration information model for Alpha in OMG UML notation (ISO/IEC, 2011).

Beta's configuration information model is shown in Figure 4. The configuration rules are grouped into 'positions' and 'position variants'. Position variants are used instead of OR operators, but also divide a configuration rule into two lines for readability reasons. The logic expression of a single configuration rule also uses operators such as IF-THEN, AND etc. From the company guidelines at Beta, it was found that the configuration rules are encouraged to be positive (not using the NOT operator) and short, but may admittedly become very long because of the OR operator and brackets.

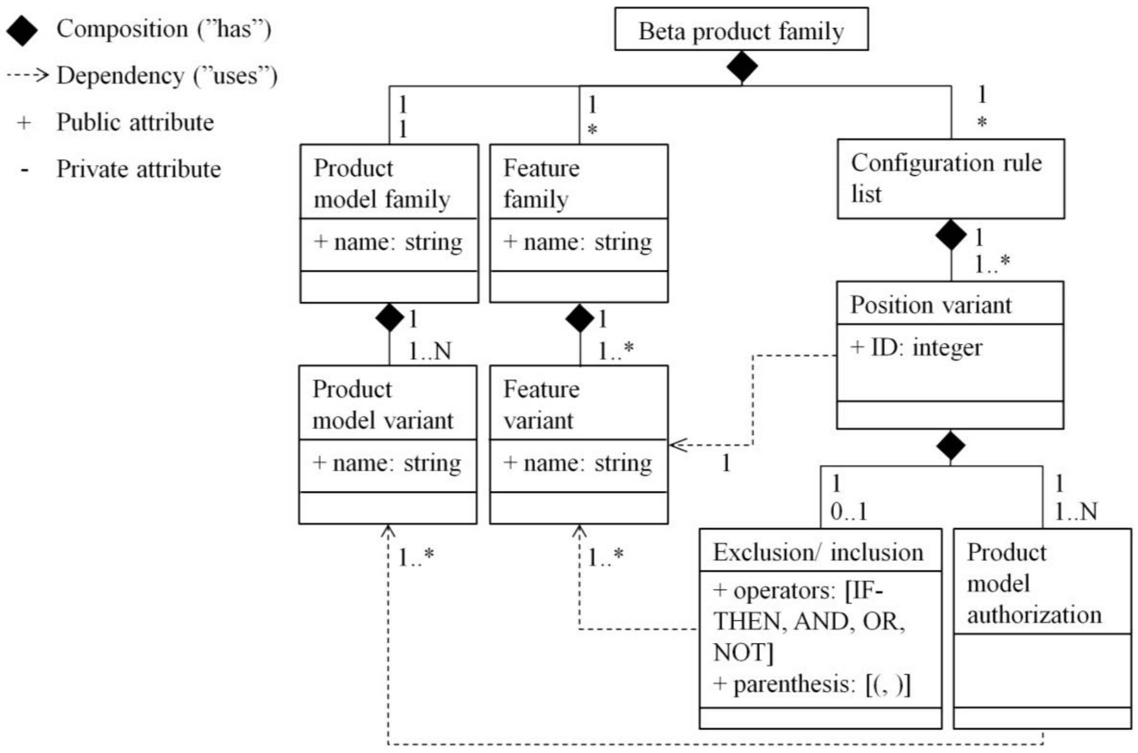


Figure 4. Beta's configuration information model in OMG UML notation (ISO/IEC, 2011).

Comparison of configuration information models

It is only possible to 'onto' map some of Alpha's and Beta's classes in the configuration information models: Alphas' 'major' and 'minor' feature family with Beta's feature family, as well as Alpha's 'major' and 'minor' feature variant with Beta's feature variant. The remaining classes at Alpha and Beta describe configuration rules. The mapping between the configuration rule classes is more difficult, as there are fundamental differences in Alpha's configuration rule matrix and Beta's list of configuration rules. The most obvious heterogeneity is, similar to programming, the explicit and relatively rich Boolean algebra at Beta, which are implicit in Alpha's configuration rule matrices. The consequences of this heterogeneity will be further discussed in the process analysis.

All comparability issues between information models categorized by Fang et al. (1991) could be exemplified based on the present study's findings:

- *Abstraction, heterogeneity on class level:* There is a distinction of two types of feature families/variants ('major' and 'minor') at Alpha which does not exist at Beta.
- *Naming, heterogeneity on instance level:* There is a product model family/variant in Beta's configuration information model, which corresponds to one of the major feature families/variants at Alpha. For the configuration rules, Beta has numbered 'positions' for grouping the configuration rules, while Alpha creates groups with the 'configuration rule matrix'.
- *Naming, heterogeneity on attribute level:* The configuration rule class 'Exclusion/Inclusion' at Beta could be mapped to attribute values {S, D, O, -} for the configuration rule matrix class at Alpha.

The configuration information models also have heterogeneous scopes of the information models because of the distinction between 'D' and 'O' at Alpha which lack correspondence at Beta.

The process for the configuration rule exchange is described in the next section.

Process description of configuration rule exchange

The exchange process model is shown in Figure 5. When there is a new version of configuration rule matrices at Alpha, a file with those is created and exported to Beta. For the collaboration project, Beta created a new 'integration system' which is used to visualize configuration rule matrices as well as to map feature families and variants between Alpha and Beta. A configuration rule specialist manually inspects the configuration rule matrices and detects changes. Possibly new minor/major feature families/variants at Alpha have to be matched with feature families/variants at Beta. Then configuration rules are authored, sometimes requiring a reformulation in order to follow Beta's company guidelines. The process activities (A3-A5) will now be discussed in more detail:

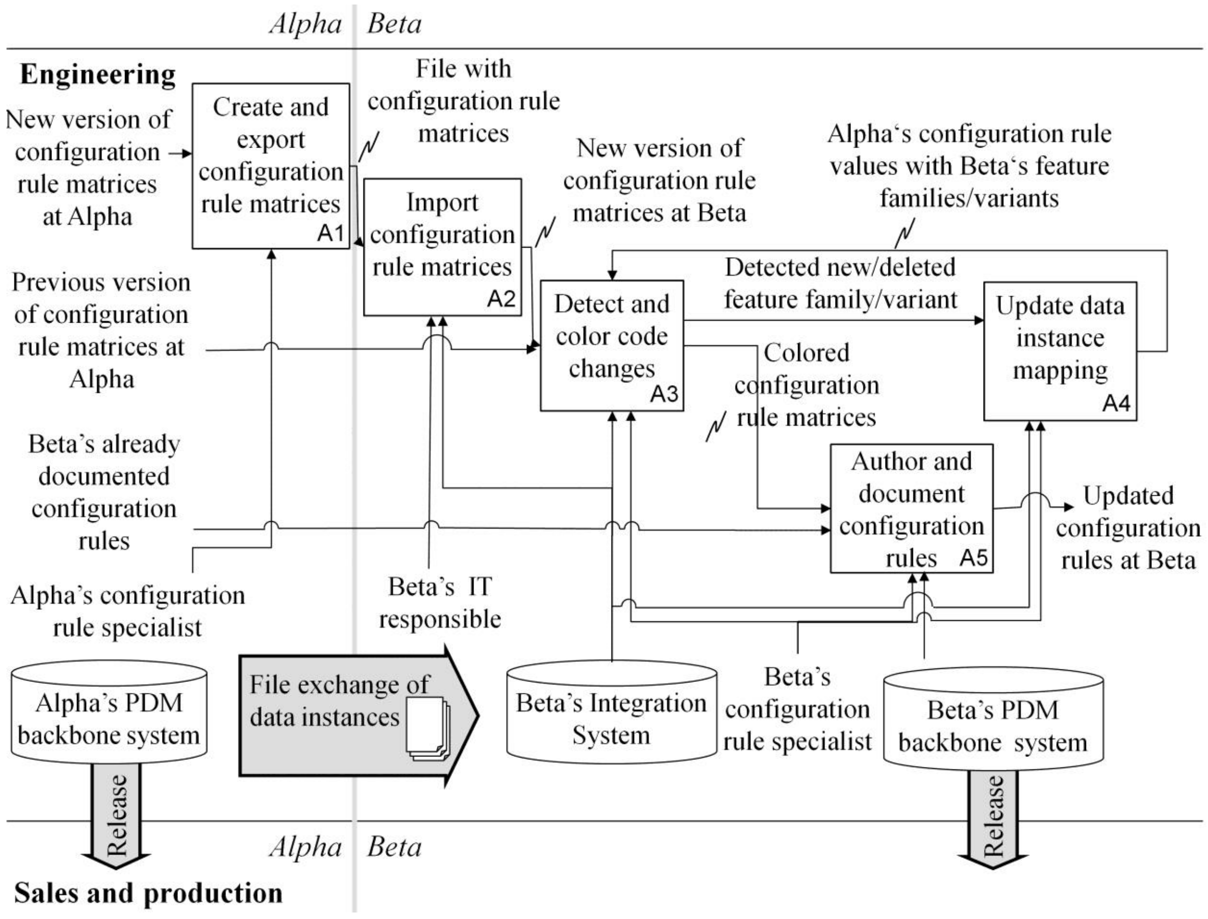

Alpha
Beta
Engineering
New version of configuration rule matrices at Alpha
Create and export configuration rule matrices A1
File with configuration rule matrices
Import configuration rule matrices A2
New version of configuration rule matrices at Beta
Alpha's configuration rule values with Beta's feature families/variants
Previous version of configuration rule matrices at Alpha
Detect and color code changes A3
Detected new/deleted feature family/variant
Update data instance mapping A4
Colored configuration rule matrices
Beta's already documented configuration rules
Author and document configuration rules A5
Updated configuration rules at Beta
Alpha's configuration rule specialist
Beta's IT responsible
Beta's configuration rule specialist
Alpha's PDM backbone system
File exchange of data instances
Beta's Integration System
Beta's PDM backbone system
Release
Release
Alpha
Beta
Sales and production

Figure 5. Exchange process for configuration rules from Alpha to Beta.

A3: Detect and color code change. In the present study, Beta's configuration rule specialist uses three different colors: blue cells indicate new, red indicate deleted and purple indicate modified data instances.
A4: Update data instance mapping. New feature families/variants from Alpha are interpreted and matched with feature families/variants at Beta.
A5: Author and document configuration rules. The authoring of configuration rules is done by analyzing configuration rule matrices from Alpha when the feature families/variants are expressed with feature families/variants from Beta. The authored configuration rules have to be documented into Beta's PDM backbone system, which is done manually today. Unnecessary work-load for this activity is when Alpha changes the list of minor feature families for creating a configuration rule matrix without doing any changes in the configuration rule values. This causes a major reformulation of configuration rules at Alpha. This issue is related to the visualization methods used for documenting the configuration rules.
The next section presents an evaluation of the automation potential for these three process activities.

Evaluation of automation potential

Looking at the exchange process in Figure 5, there are three process activities that are interesting to study from a configuration rule exchange perspective: A3) the detection of changes in the configuration rule matrices, A4) the update of data instance mapping tables, and A5) the authoring of configuration rules at Beta. The first step, the detection of changes, was found to be the best candidate for automation:
A3: Detect and color code changes. Alpha's configuration rule matrix does not include any information about which configuration rule values that have changed. The detection of changes is a manual process where the user compares two computer screens with different versions of configuration rule matrices. By comparing the screens cell for cell, it is possible to detect the differences in the configuration rule matrices. This step seems to be a suitable candidate for automation, since it is a repetitive and time-consuming task. Evaluation: suitable for automation.
A4: Update data instance mapping tables. The update of instance mapping tables has already been studied and found difficult to automate (Kementsietsidis et al., 2003). Consequently, there are according to Kementsietsidis et al. no tools facilitating the update of mapping tables. The difficulty to automate the update of data instance mapping tables was emphasized in the present study as several inter-organisational workshops had to discuss the naming conventions for feature families and variants. One real case example: is Alpha's feature family 'secondary color' equal to Beta's 'upper color' or 'lower color'? Evaluation: difficult to automate.
A5: Author and document configuration rules. The last process activity to be discussed is the automated authoring of configuration rules. The easiest automation approach would be to create a one-to-one mapping between configuration rules at Alpha and Beta. This is however creating far more (~100 000) string-based configuration rules at Beta than what is normally authored in a car project. The possibility to manually check these configuration rules becomes limited for the configuration rule specialist who has work practices adapted to much fewer configuration rules. The manual inspection is an important task, as when mistakes are made, the mistake has to be possible to be tracked. To reduce the number of list-based configuration rules, Beta has chosen to apply their authoring guidelines. Consequently, a re-formulation is required from the configuration rule matrices at Alpha to the much longer string-based configuration rules used at Beta. This is a re-formulation that has to mimic the configuration rule specialist's preferences of how to author configuration rules. In order to do that an in-depth study has to be conducted. Evaluation: difficult to automate but worthy of future examination.

Summary of analysis phase

The main obstacles identified during the analysis phase are mapping between data instances as well as heterogeneity in visualization methods:
- *Mapping of data instances:* Different naming of data instances from feature families and feature variants causes a time-consuming update of one-to-one mapping tables. Update of these mapping tables is known to be difficult to automate, and an improved exchange process should therefore address other exchange issues. Another type of mapping, which is rather unknown in the research literature, is the onto mapping of configuration rules. This onto mapping causes a re-formulation of configuration rules, which is difficult but possible to be automated.

- *Visualization methods:* Process activities where heterogeneity in visualization methods are causing issues are the detection of changes and the authoring of configuration rules. Both of these process activities are negatively influenced by the fact that the configuration rule matrices are only user generated 'views'. Any view can be generated, each requiring a unique set of list-based configuration rules to be authored. This complicates change detection, as it becomes difficult to know if it is the view or the configuration rule values that have changed. When the major feature families in the view changes, the complete list of configuration rules from that configuration rule matrix has to be re-formulated (updated). It is therefore in Beta's interest to automate both the change detection and the authoring of configuration rules.

3.2 Development

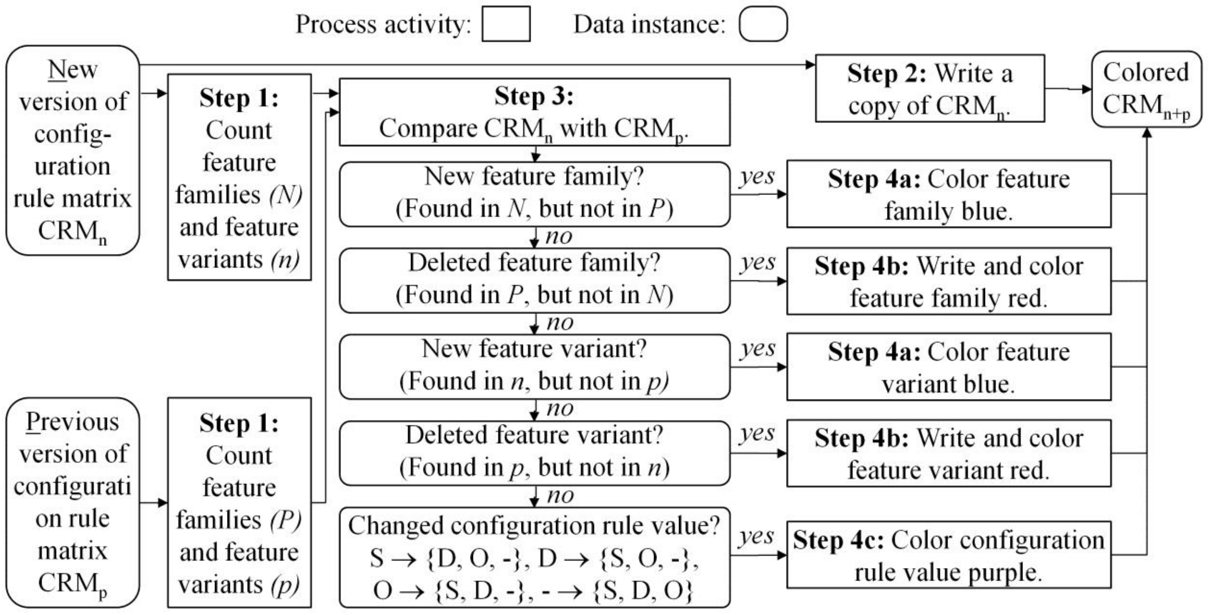
The development phase consisted of two sections: task analysis as well as the algorithm development.

Automation tasks

There are two main tasks for how to support the configuration rule specialist during the exchange of configuration rules: *Task A – Automated detection and coloring of configuration rule matrix changes*, and *Task B – Automated authoring of configuration rules.*
The number of errors is the most important criteria, and the selection of which task to develop a support for, should be based on this. Task A was selected due to its convincing effect on reducing the number of errors. Task B with the automatic authoring of configuration rules would give better time-savings, but is more complicated. Task B is therefore left to future work.

Algorithm development

Task A (automated coloring of configuration rule matrix changes) is here described in a step-by-step algorithm, which is also shown in Figure 6.
Step 1: Count feature families (N) and feature variants (n) in a new version of configuration rule matrix CRM_n, Count feature families (P) and feature variants (p) in a previous version of configuration rule matrix CRM_p.
Step 2: Write a copy of CRM_n into a new configuration rule matrix CRM_{n+p}.
Step 3: Compare data instances in CRM_n with CRM_p: feature families, feature variants and configuration rule values {S, O, D, -}.



Process activity:
Data instance:
New version of configuration rule matrix CRM_n
Previous version of configuration rule matrix CRM_p
Step 1: Count feature families (N) and feature variants (n)
Step 1: Count feature families (P) and feature variants (p)
Step 3: Compare CRM_n with CRM_p.
New feature family? (Found in N, but not in P)
Deleted feature family? (Found in P, but not in N)
New feature variant? (Found in n, but not in p)
Deleted feature variant? (Found in p, but not in n)
Changed configuration rule value? S → {D, O, -}, D → {S, O, -}, O → {S, D, -}, - → {S, D, O}
yes
no
Step 2: Write a copy of CRM_n.
Colored CRM_{n+p}
Step 4a: Color feature family blue.
Step 4b: Write and color feature family red.
Step 4a: Color feature variant blue.
Step 4b: Write and color feature variant red.
Step 4c: Color configuration rule value purple.

Figure 6. Change detection algorithm described with a step-by-step flowchart.

Step 4: Color new/deleted/modified values.
  Step 4a: If new feature family or feature variant, color it blue in CRM_{n+p}.
  Step 4b: If deleted feature family or feature variant, write and color it red in CRM_{n+p}.
  Step 4c: If modified configuration rule value, color it purple in CRM_{n+p}.

The algorithm was implemented as a Microsoft Excel macro in C#. The spreadsheets were imported from Beta's integration system. The next section will compare the results of this implemented algorithm with the manual change detection process performed by the product structure specialist.

3.3 Testing

The validation of the software with the implemented algorithm for change detection was based on tests, inspection and reviews. The tests were based on two different inputs to the algorithm, and the outcome was compared to the manual process of change detection. The manual process was conducted by a configuration rule specialist. The inspection of the implemented algorithm was conducted with a programmer who was responsible of creating the production version of the software. The reviews were conducted together with the IT project manager who had been writing the requirement specification for the integration system, and also the person who had assumed that it was too complicated to create a change detection algorithm. Both the programmer and the IT project manager accepted the prototype software and work started to develop the production code that would be a functionality of the integration system. The remaining technique to be discussed is the validation tests. There were two evaluations of the algorithm: Test 1 and Test 2, see Table 1. These evaluations were performed on complete version collections of configuration rule matrices, which were 16 configuration rule matrices for all tested version collections. The characterization of Test 1 was configuration rule matrices from early development of vehicles, and the most frequent changes were new/deleted feature families/variants. The characterization of Test 2 was configuration rule matrices from a more mature development of the vehicle, and the detected changes were dominated by modified configuration rule values.

In Test 1, both the manual and automate process found 100% of the changes. In Test 2, the automated algorithm again found 100% of the changes, but the manual process only found 95%. The access to manual process data for Test 1 was limited to only after the documentation update to Beta's PDM system, and the coloring had therefore been verified several times. The manual process data was for Test 2 possible to be accessed before the documentation update to Beta's PDM system, and it therefore remained some verification of the coloring. All discrepancies were modified configuration rule values that had not been detected in the manual process. It was thereby discovered that this type of change is the most difficult to detect in the manual process.

Table 1. Comparison between automated and manual process in two evaluation tests. ('CRM' = configuration rule matrix, 'del.' = deleted, 'mod.' = modified)

Test ID	Test results (16 CRM_n and 16 CRM_p)							Colored data instances		Time [min]	
	Major & minor feature families		Major & minor feature variants		Configuration rule values (D, S, O, -)			Automated process	Manual process	Automated process	Manual process
	New	Del.	New	Del.	New	Del.	Mod.				
Test 1	4	1	77	11	2955	1424	627	5099	100%	5	900
Test 2	11	2	33	8	0	1490	1847	3391	~ 95%	5	900

The above tests show what the automation is essential to secure the quality of the configuration rules. In addition, the algorithm saves time. The process from detection of changes until and including documentation in Beta's PDM backbone system takes about 100 working hours. The time spent for the manual detection and color coding of configuration rule matrix changes is 15 hours of the 100 working hours, which would with the automation suggested in this paper be reduced to few minutes. Also, from a quality point of view, the change detection algorithm is fundamental for getting it right the first time without iterating verification of the configuration rules.

4 DISCUSSION

This section will go through the research questions and discuss their answers, and discuss the results' generalization.

4.1 Answering research questions

RQ1: What does an exchange process for product configuration rules between collaborating companies look like?

The exchange process for configuration rules in this paper was taking place between Alpha and Beta. Alpha has the responsibility to export correct and complete configuration rules, but in order to update Beta's PDM system there is a need to detect changes, update a one-to-one data instance mapping table, as well as author configuration rules. Beta has developed a separate integration system to support the configuration rule exchange process. This integration system contains the one-to-one data instance matching table and a visualization of Alpha's configuration rule matrices.

RQ2: Which configuration rule exchange obstacles between collaborating companies are identified?

During the exchange process between Alpha and Beta there are several kinds of data heterogeneity, e.g. different naming of data instances and different scopes of information models. The most challenging heterogeneity was however the heterogeneity in visualization methods. Alpha uses configuration rule matrices, which after the exchange has to be translated to configuration rule lists at Beta. This heterogeneity affects how configuration rules are authored, and is what causes the reformulation of configuration rules. Generally speaking, visualization method heterogeneity is the most important criteria when evaluating the effort required for an exchange of configuration rules. It is not the information modes which is the main obstacle, which could have been assumed based on previous comprehensive research work on information models. The visualization method heterogeneity causes major re-formulations of configuration rules at Beta from minor changes at Alpha. Another implication is the change management. The change management has been identified as an issue during product data exchange in (Jokinen et al., 2008).

RQ3: How can the exchange of configuration rules be improved?

Two main possibilities to automate process activities during a configuration rule exchange is an authoring of configuration rules, as well as change detection. More concretely, an algorithm should either address the 1) automated authoring of list-based configuration rules from a configuration rule matrix 2) the automated detection and color coding of configuration rule matrix changes. The algorithm that was developed in this paper concerns the later. The algorithm works for changes in configuration rule matrices in general, not only at Alpha. Future work could develop the algorithm further and also include the automated authoring of configuration rules.

4.2 Generalization

The algorithm developed in this paper works for configuration rule matrices. The applicability of the algorithm is therefore to companies that have a matrix-based visualization of configuration rules. Another application area is during collaboration projects in general. Beta's product structure specialists consider their list-based visualization unreadable for product structure specialists from other companies. They therefore would like to use a matrix-based visualization during coming future collaboration projects, with the in this paper developed algorithm as a PDM independent method.
The exchange process found during this study is based on collaboration between two companies. The use of neutral file formats requires a translation from the sender to the neutral file format, and then a translation again between the neutral file format to the receiver. If more companies would have been involved in the exchange process, it is possible that a neutral file format for the exchange had been used as well as that automated translators would have been developed.

5 CONCLUSIONS AND FUTURE WORK

The configuration rule exchange is a process that has been shown to need to detect changes, map data instance as well as re-formulate configuration rules. Several types of information model heterogeneity could be identified, but this heterogeneity was however not identified as a major issue. It was instead the visualization method heterogeneity that caused the most challenging issues during the configuration rule exchange. For example, Alpha's PDM system cannot guarantee that a collection of configuration rule matrices contains all feature families as those are user-generated views and not verified to be a complete data set. The conclusion is that tools that are working well for documenting configuration rules within a company do not necessarily suffice between collaborating companies. For example, the visualization method heterogeneity caused difficulties for the change management. The algorithm implemented for this study serves as an example of what is required to overcome the issue of visualization method heterogeneity. Future work should address issues identified in the present study in order to validate the generalizations and extend the knowledge of configuration rule exchange issues.

ACKNOWLEDGEMENTS

This work was carried out at the Wingquist Laboratory VINN Excellence Centre for Efficient Product Realization at Chalmers University of Technology, supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA). Further financial support was received from ProViking. The authors would like to thank Hr. Eiffinger.

REFERENCES

Bruce, M., Leverick, F. and Littler, D. (1995) Complexities of Collaborative Product Development. *Technovation*, Vol. 15, No. 9, pp. 535-552.

Chow, K. and McElroy, J. (2002) Searching for the Perfect BoM: a NEMI Team Comprised of OEMs and EMS Providers Offers Recommendations for Improving Bills of Materials. *Circuits Assembly*, Vol. 13, No. 10, p. 26.

Domazet, D.S., Yan, M.C., Calvin, C.F.Y., Kong, H.P.H. and Goh, A. (2000) An Infrastructure for Inter-organizational Collaborative Product Development. *International Conference on System Sciences - HICSS*, Hawaii, University of Hawaii/Shidler College of Business.

Fang, D., Hammer, J. and McLedo, D. (1991) The Identification and Resolution of Semantic Heterogeneity in Multidatabase Systems. In Kambayashi, Y., Rusinkiewicz, M. and Sheth, A. (eds.) *International workshop on Interoperability in Multidatabase Systems – IMS'91*, Kyoto, IEEE Computer Society/Interoperability Technology Association for Information Processing/Information Processing Society of Japan/Telecommunication Advancement Foundation.

Hirel, G. and Hug, H. (2009) Harmonized Exchange of Bill of Material Data for the Next Truck Generation. *Integrated Virtual Product Creation Symposium - iViP*, Berlin, ProSTEP.

ISO 10303-1:1994 (1994) Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 1: Overview and Fundamental Principles.

ISO/IEC 19505-1:2012 (2012) Information Technology – Object Management Group Unified Modeling Language (OMG UML).

Jokinen, K., Hajda, M. and Borgman, J. (2008) Challenges with Product Data Exchange in Product Development Networks. In Marjanovic, D., Storga, M., Parkovic, N. and Bojcetic, N. (eds.) *International Design Conference – DESIGN 2008*, Dubrovnik, University of Zagreb/The Design Society.

Kementsietsidis, A., Arenas, M. and Miller, R.J. (2003) Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues. In Halevy, A.Y., Ives, Z.G. and Doan, A. (eds) *International Conference on Management Of Data - SIGMOD'03*, San Diego, ACM Special Interest Group on Management Of Data.

Killick, J.H. (1993) Data Representation Using a Standard Data Transfer Specification. *Journal of Navigation*, Vol. 46, No. 2, pp. 223-237.

MacDuffee, C.C. (1940) *Introduction to Abstract Algebra*. New York: Wiley.

Markson, H. (2007) Achieving CAD Interoperability in Global Product Design Environments. *White Paper,* SpaceClaim Corporation.

Naiman, C.E. and Ouksel, A.M. (1995) A Classification of Semantic Conflicts in Heterogeneous Database Systems. *Journal of Organizational Computing,* Vol. 5, No. 2, pp. 167-193.

Pratt, M.J. (2001) Introduction to ISO 10303 - the STEP Standard for Product Data Exchange. *Journal of Computing and Information Science in Engineering,* Vol. 1, No. 1, pp. 100-101.

Tidstam, A. and Malmqvist, J. (2011) Authoring and Verifying Vehicle Configuration Rules. In Pels, H.J. (eds.) *International Conference on Product Lifecycle Management – PLM11*, Eindhoven, Technische Universiteit Eindhoven.

Viel, C. (2003) Management of Product's Diversity: Industrial Validation of AP214 Reference Model. *International Journal of Computer Applications in Technology,* Vol. 18, No. 1-4, pp. 62-77.

Zeuthen, H.G. (1870) Sur les Points Fondamentaux de Deux Surfaces Dont les Points se Correspondent Un à Un. *Comptes Rendus des Séances de l'Académie des Sciences,* Vol. 7, p. 742.