Survey, Approach and Examples of Modeling Variants in Industrial Automation

C. R. Maga* and N. Jazdi**

*Institute of Industrial Automation and Software Engineering (IAS), University of Stuttgart, Pfaffenwaldring 47, 70569 Stuttgart e-mail:{*Camelia.Maga; **Nasser.Jazdi}@ias.uni-stuttgart.de*

Abstract: In order to meet requirements of different stakeholders, product lines for industrial automation systems have a high degree of variability. Despite the remarkable effort made by researchers and practitioners, modeling variants remains a challenge. It is often difficult to decide what does vary, how it varies and which interdependencies have to be considered between variants of an industrial automation system. This paper discusses the state of the art in modeling variants and proposes a new approach based on SysML for modeling variants of an industrial automation system within a product line.

Keywords: variability, product line engineering, SysML, modeling approaches

1. INTRODUCTION

An industrial automation system encompasses a technical system with the contained technical process, a computer and communication system, and the process operators involved. Thereby a technical system can be a technical product or a technical plant, in which a technical process takes place [Göhner 2009]. Examples of industrial automation systems are cars, elevators, or oil refineries.

For industrial automation systems, engineering is rarely performed from scratch. In most cases, there is a legacy principle within the domain, intended to systemize and increase the reusability of industrial automation systems. In consequence, one develops product lines instead of single products [Maga, Jazdi et al. 2009].

A product line is "a set of systems sharing a common, managed suite of features that satisfy a particular market segment or mission's needs and that are developed from a common set of core assets in a prescribed way" [Clemens, Northrop 2002]. A typical example of a product line for elevators includes freight elevators for aircraft-carriers or car elevators, as pictured in the Fig.1.

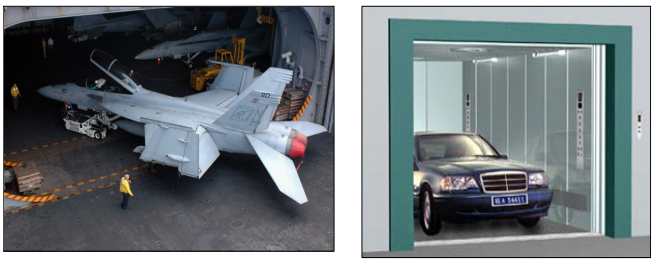

Fig. 1. Elevator variants [ThyssenKrupp 2010].

Industrial automation systems within a product line share commonalities, but possess at the same time some differences. In other words, variants occur within the product line. A variant is an alternative solution, which is created by varying qualitative or quantitative parameters and which serves solving the same or approximately the same problem as the initial solution [Franke, Firchau 1998]. Variability is the property of a system to have variants. For industrial automation systems, different types of variants can be identified. [Baumgart 2005] distinguishes between structural, functional, realization and service variants.

- **Structural variants** of industrial automation systems differ from each other regarding their structures. Examples are two elevators, one with hydraulic drive and the other with electric drive. Both elevators serve to transport passengers between different floors in a building. Despite this, they have different drives, different components and a different placement of the components within their structures.
- **Functional variants** of industrial automation systems provide different functionalities within the product line. For example, let us consider two elevators: one with implemented firefighter functionality, the other one without this functionality. Although the two elevators have the same structure, they exhibit different functions, in accordance with the country's legitimate regulations.
- **Realization variants** are industrial automation systems having different implementations. Examples are two elevators having DC door motors from Schindler and from Montgomery. Although both DC motors serve in the same function (doors movement), they may have different speeds or heat development profiles. Hence, the two elevators vary regarding their realizations.

- **Service variants** are industrial automation systems with different services provided. To illustrate, consider two elevators, one of them is delivered together with 24 h/day surveillance and service contract, the other has only a service contract. Although the two elevators are similar, their usage is different. Hence, two service variants are necessary.

The examples listed above emphasize two aspects. First, there are different types of variants to manage within a product line. In the praxis, an industrial automation system within a product line can belong simultaneously to two or more categories of variants, being for example structural and service variant at the same time. Second, the number of variants within a product line can grow tremendously, when all possible combinations are considered. Consequently, models of variants in a product line become increasingly difficult to understand and to use.

The current paper discusses the issue of modeling variants of industrial automation systems within a product line. The paper is organized as follows: the second section presents the state of the art in modeling variants of industrial automation systems. In the third section, we propose and illustrate a new approach for modeling variants with SysML, including required notations and stereotypes. Based on this, a configuration tool has been realized, which is presented in the fourth section. Section five concludes with a summary.

2. STATE OF THE ART IN MODELING VARIANTS

Motivated by the increasing diversity of industrial automation systems during the last years, researchers and practitioners have been dealing with modeling variants with high interest [Pohl, Böckle, van der Linden 2005], [Czarnecki, Eisenecker 2000], [Kaeding 2009]. Decision tables, decision trees, feature diagrams and orthogonal variability models are the most commonly used techniques for modeling variants. Therefore, they are discussed in this section.

2.1 Decision Tables

Explanations

Decision tables are a simple possibility to capture information about variability of a product line. According to [Kaeding 2009], each raw of the table represents a source of variability. In order to create a new industrial automation system, a decision for each raw is necessary. A sample decision table is depicted in the following table.

Table 1. Example of a decision table for elevators

ID	Description	Subject	Constraints	Resolution	Effect
material cabin	Which material is used for the elevator's cabin?	cabin	Panorama elevator recommends glass	a) metal b) glass	a) test cases 10 and 15 b) test cases 13, 24 and 14

The ID is a unique identifier of the variability point. The description is formulated as a question that has to be answered for the considered variability point. The column "Subject" represents the affiliation to a given item or topic. By doing this, more variation points can belong to the same item. Possible preconditions are listed in the column "Constraints". The column "Resolution" presents possible variants that are valid for the question in the second column. Finally, the last column contains consequences related to the chosen variants. [Kaeding 2009]. Regarding usage and functionality, decision tables are similar to morphological boxes presented in [Pahl, Beitz 2006]. These contain possible variants for developing new industrial automation systems.

Advantages and Disadvantages

A significant advantage of decision tables is their understandability. Since decision tables capture information about variability in textual form, it is easy for everyone to follow which variants are covered. There are no specialized notations or symbols required [Kaeding 2009], [Pahl, Beitz 2006]. In addition, decision tables are easy to use. There are no specialized software tools necessary to depict them.

Information in textual form is the main strength, but also the main weakness of decision tables. Because of the textual form, there are no formal proofs possible. In addition, industrial automation systems are nowadays complex and source of numerous variants. Many variants are interdependent. It is necessary to model different relations between variants, like mandatory, optional, recommended or alternative relations. These interdependencies are difficult to model in a decision table. Moreover, decision tables become confusing when they contain a large number of variants.

2.2 Decision Trees

Explanations

Decision trees represent an improvement of decision tables. They make it possible to capture information about variants graphically [Kaeding 2009]. Decisions yielding variants are represented as nodes in the tree. Tree's edges represent possible variants that can be chosen for engineering new industrial automation systems. In a decision tree, the number of end nodes (leaves, here represented as circles) corresponds to the total number of possible configurations within the product line. An example of a decision tree for elevators is shown in figure 2.

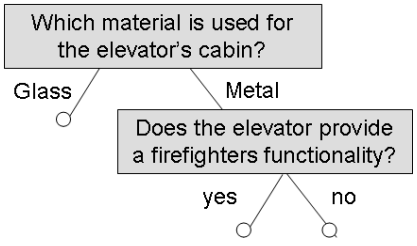

Fig. 2. Example of a decision tree for elevators.

Advantages and Disadvantages

Similar to decision tables, decision trees are easy to understand and to use. It is not necessary to get first familiarized with a modeling language or with a modeling software tool before using them. Furthermore, the graphical form enables a clear representation of possible variants.

Unfortunately, decision trees become large and unclear in case of numerous variants. Some decisive questions have to occur more than once, in order to cover all possible variants. This yields redundancies in decision trees. Because of the textual content, it is impossible to prove contained variants of a decision tree in a formalized way. Finally, it has been observed that some crucial interdependencies between variants cannot be captured in decision trees.

2.3 Feature Diagrams

Explanations

According to [Kang, Cohen et al. 1990], a feature is an end-user visible characteristic of a system. For instance, the firefighter functionality of an elevator or the existence of background music in the elevator's cabin are typical features for elevators. It is important to distinguish between features and variants. While features are characteristics of a system, variants occur only when two or more industrial automation systems have different features implemented. In our example, this would be the situation of two elevators: one with firefighter functionality, the other without it. The feature "firefighter functionality" has been selected and implemented only in the first situation. The two elevators are variants within the same product line.

Feature diagrams provide the possibility to model both features and relations between them within a product line. There are specialized notations for both direct relations between features like mandatory relations, options, alternatives, and cross relations between features like "recommends", "discourages", "conflicts" or "requires". A good introduction to feature diagrams is given in [Czarnecki, Eisenecker 2000]. Figure 3 shows a feature diagram for elevators. It can be observed that each elevator must have a drive, either an electric or a hydraulic one. Furthermore, the diagram considers features related to the usage form of the building, in which the elevator is located. The building can be either residential or non-residential. In case of non-residential buildings, the diagram considers the possibility to engineer a high-rise elevator, depicted as optional feature in the figure.

Advantages and Disadvantages

Advantages of using feature diagrams are the dedicated notations and symbols. After familiarization, feature diagrams are easy to understand and to use. In addition, the existence of specialized tools such as pure::variants [Pure Systems 2010] enables formalized proofs of desired configurations. Another advantage is the possibility to capture cross relations between features.

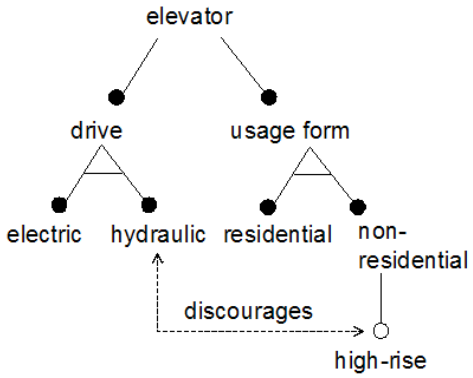

Fig. 3. Details of a feature diagram for elevators.

Unfortunately, feature diagrams may contain redundancies, in order to cover the entire variability of a product line. Some features occur more than once in a feature diagram. Hence, the clarity of the representation is affected. This disadvantage becomes even more intense when the modeled industrial automation system possesses many features. Another disadvantage has been observed during the usage of feature diagrams. There is a separation between variants models and realization of these variants. Systems are configured with help of feature diagrams, but designed and realized separately, using other tools, models and notations. Under these conditions, it is hard to determine, which impact is associated with the selection of a feature for decisions related to requirements, design, realization or tests. A similar observation is confirmed by [Pohl, Böckle, van der Linden 2005].

2.4 The Orthogonal Variability Model

Explanations

Variability information is spread across different models. One needs to model requirements, design decisions, structure, behavior and tests of industrial automation systems. The occurrence of a variant affects all these different models. According to [Pohl, Böckle, van der Linden 2005], it is almost impossible to keep the information consistent. In order to mitigate this problem, the so-called orthogonal variability model has been proposed. This is a "model that defines the variability of a software product line. It relates the variability defined to other software development models, such as design models, component models, and test models" [Pohl, Böckle, van der Linden 2005].

The orthogonal variability model considers variability of a product line from requirements specification, over design and realization, until test. It is called "orthogonal" because the variants model is placed orthogonal to the development models, as depicted in the following figure. In order to encourage the usage of this approach for modeling variants, the software tool VarMod has been developed at the Essen University [VarMod 2010].

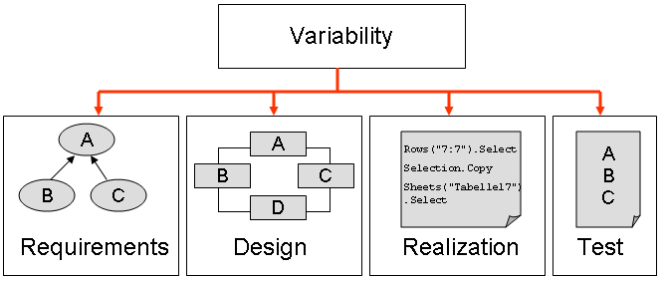

Fig. 4. Details of a feature diagram for elevators.

Advantages and Disadvantages

The orthogonal variability model is a helpful approach to model variants. It considers the effects of selecting a variant through all the development phases of a new product within a product line. Providing a specialized software tool increases the acceptance for using the orthogonal variability model. In addition, it offers an understandable way to depict and manage variation points of a product line.

Although the orthogonal variability model is a promising approach, it includes some disadvantages, as well. The approach proposes two separate models: one for capturing information about variants, along with other models for design and development of new industrial automation systems of a product line. This separation brings difficulties related to consistency and evolution of the two models. For instance, there is no complete tool chain for engineering industrial automation systems integrated in the approach. Usually, companies use many different tools from different suppliers during the engineering process. Because of this, there are some interruptions between the engineering phases depicted horizontally in the Figure 4. Despite the clear definition of the variation points in the variability model, there is no guarantee that the separate models created with different tools are consistent with it. In addition, each evolution of the product line necessitates the maintenance of at least two separate models. It has been observed that some helpful cross relations between variants (e. g. “recommends”, “discourages”, etc.) are not supported.

3. APPROACH FOR MODELING VARIANTS IN SysML

As mentioned in the previous section, existing approaches for modeling variants have both advantages and disadvantages. In our opinion, a new approach to model variants requires a paradigm shift. We need models of variants within a product line to manage existing or future commonalities and differences. Existing approaches separate more or less variants models from models of the product line. Our idea is to integrate them in the product line instead of separating them. Hence, we propose to model variants of an industrial automation system together with its requirements, its structure, its functionality, its realization or its service packages. Shortcomings like inconsistency, overloaded models, or difficulties in modeling variants using the same concepts as for development [Pohl, Böckle, van der Linden 2005] have to be avoided. For this, we use a modeling language to model industrial automation systems, namely SysML [OMG 2008], and extend it with necessary stereotypes for modeling variants.

A promising concept for modeling only software variants exists already in form of a UML extension [Riebisch, Böllert et al. 2000]. This concept is based on feature diagrams. Hence, it suffers from similar disadvantages. It is confusing for numerous variants and necessitates a separate configuration map. This contains static design references, which are difficult to maintain consistently. Moreover, cross relations between variants are missing. In UML, requirement diagrams and parameter diagrams are not supported. To conclude, modeling variants of industrial automation systems requires a new approach, able to cover the entire engineering process. SysML has been chosen because it provides a comprehensive support for modeling, covering all the required phases.

3.1 Syntax Elements

The basic concepts for the new approach are inheritance and package modeling. We distinguish between variability relations between packages and within packages.

Elements of industrial automation systems that are mandatory for each member of the product line are modeled in a core package. This package is included through a package import relationship provided with the stereotype “mandatory” to each new industrial automation system. Since packages contain different SysML diagram types, we can include in such a core package mandatory requirements, mandatory structures, mandatory functionalities, and mandatory test cases. Optional elements are modeled in a separate package, under usage of the package import relationship and the stereotype “optional”. In this case, either the entire package is selected or no element is included in a new industrial automation system.

Alternatives (“xor”) and selections (“or”) are modeled with an element import relationship provided with the “requires” stereotype. These situations are depicted in the following figure.

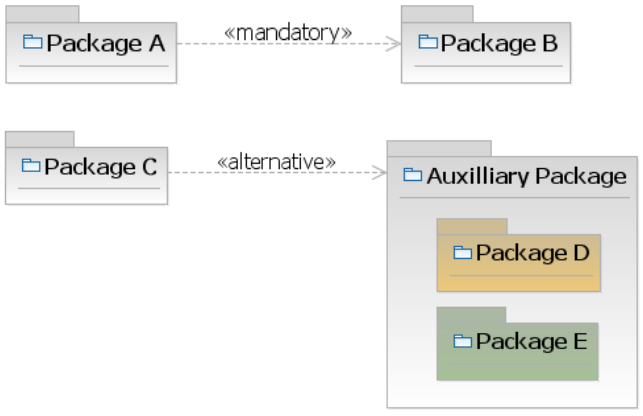

Fig. 5. Examples of possible relationships between packages.

The usage of packages and relationships between packages ("package import" and "element import") brings two advantages. First, the variants are structured hierarchically, which increases the understandability of the model. Second, the subordinate packages or elements are imported only if required. This minimizes the model's redundancy.

Variants within packages are modeled using compositions, aggregations, and inheritance relations. If required, cardinalities are used, as well. As depicted in figure 6, mandatory relations are represented by shared associations with a black diamond end. Optional elements of an industrial automation system are modeled as shared associations with a white diamond end. In case that a selection between two or more elements is necessary, we introduce a superior element encompassing commonalities of the proposed alternatives. The differences are modeled with the help of inheritance.

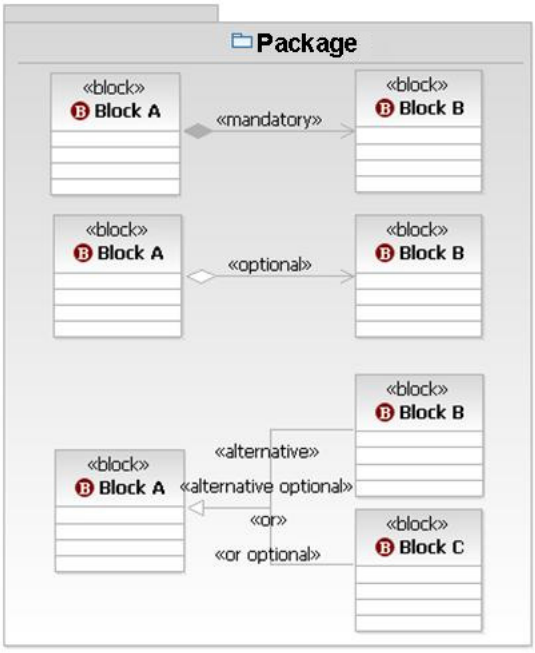

Fig. 6. Possible relationships within packages.

The relations depicted in the Figure 6 are hierarchical relations. They express relations between a system component and its variant parts. An overview on hierarchical relations is shown in the following table.

Table 2. Overview on hierarchical relations

Variability Relation	Logical Term
mandatory	a AND b
optional	a OR b
alternative	a XOR b
alternative optional	(NOT a) OR (a XOR b)
or	a OR b OR ... OR x
or optional	(NOT a OR a) OR (NOT b OR b)...

- A "**mandatory"** relation means that the variant must be included in the configuration of a new product line member. For example, each elevator must have a cabin.
- An "**optional"** relation refers to the selection of a variant part, which can be done independently from its other peers (same level system parts). The variant part can be selected or not. An example therefore is the situation of an elevator cabin that can be made of laminated glass.
- **"Alternative"** relations mean that one and only one of the variants marked as alternative must be chosen. For instance, each elevator must have a drive, either an electric or a hydraulic one.
- **"Alternative optional"** relations mean that either no variant or exactly one and only one variant can be selected. An example is an elevator that can have a self-adapting dispatching software, either agent based or object oriented.
- **"Or"** relations mean that minimum one of the corresponding variants in the respective product line can be selected for the configuration. For instance, each elevator must have an overspeed governor. In order to execute the speed surveillance, the overspeed governor must evaluate the signals coming from velocity sensors, from acceleration sensors or it uses both.
- **"Or optional"** relations express the possibility to combine diverse variant components to a new industrial automation system within the product line. It is not mandatory to choose a variant. For example, an elevator can cover floors with restricted access. In order to authenticate passengers to access such floors, an identification detection mechanism is required a device to check the passenger's identity is required. For this purpose, the cabin can have a keypad for entering the password, a fingerprint scanner or both.

Cross relations are modeled with SysML usage relationships that are extended accordingly with the stereotypes "requires", "recommends", "forbids", "discourages", or "influences". Cross relations express relations between peer variant system parts (same level system parts). An overview on cross relations is shown in the table 3.

Table 3. Overview on cross relations

Variability Relation	Logical Term
requires	a AND b
recommends	a OR (NOT a)
forbids	a XOR b
discourages	a AND (NOT b)
influences	a ° b

- Variants connected though "**requires**" imply that at least one of the specified target variants has to

be selected once the source variant is selected. For instance, the selection of the fingerprint authentication functionality requires the inclusion of a fingerprint sensor in the cabin console.

- The "**recommends**" relation is weaker then "requires". One is advised to include a variant, but neglecting this advice does not lead to a failure in system. For example, to make the cabin look larger, it is recommended to include a mirror in small passenger elevators. Neglecting this advice will still yield to a valid elevator.
- The "**forbids**" relation is the opposite of "recommends." If all specified target variants are selected, the source variant cannot be a member of the selection. For example, it is forbidden for elevators with a glass cabin to be used as firefighter's elevators.
- The "**discourages**" relation is weaker then "forbids". One is advised not to include a variant if the source variant is selected, but neglecting this advice is not result to invalidity.
- The "**influences**" relation means that the target variant is influenced by the source variant according to certain parameters. The interpretation is up to the user.

The described syntax elements allow to model variants of an entire product line. For visualizing one variant of the product line, SysML offers the possibility to define views. By selecting a view, only elements related to the considered variant will be displayed.

3.2 Example of Modeling Variants with the new Approach

The approach mentioned above has been concretely deployed in a case study for the domain of passenger elevators. The goal of the case study was twofold: first, to evaluate a more extensive approach for product line engineering, which has been developed at the Institute of Industrial Automation and Software Engineering (IAS) of the University of Stuttgart in cooperation with the Siemens Company. We focused on identifying necessary refinements, changes or completions. Detailed results have been presented in [Maga, Jazdi et al. 2009], [Maga, Jazdi 2009 a], [Maga, Jazdi 2009 b]. Second, we aim at analyzing the deployment of the new approach for modeling variants of industrial automation systems for a concrete example.

In order to accomplish the case study, we proceeded as follows. First, we identified elevator variants by analyzing a large number of different elevators manufactured by the companies Kone, ThyssenKrupp Elevators, Otis and Schindler. These companies span a very broad field of products, from freight elevators over passenger elevators to escalators. For the sake of simplicity, the case study focused on passenger elevators spectrum.

In a second step, we modelled a product line for passenger elevators in SysML. For this purpose, we used the EmbeddedPlus SysML Toolkit v2.0 in combination with the IBM Rational Software Modeler v7.0. The SysML model contains a large set of reusable artifacts: from reusable requirements, over reusable state machines, reusable activity diagrams, and reusable test cases to reusable reference architectures. For all these reusable items, we identified different types of variants, as presented in the first section of this paper. Structural variants were defined in the reference architecture of the elevator. Functional variants were captured in reusable requirements, reusable state machines and reusable activity diagrams. Moreover, realization variants were considered in the reference architecture and in reusable test cases. Finally, service variants were modelled with the help of reusable requirements and reusable test cases.

The different types of variants were grouped to packages. An overview of the modeled packages with variants is shown in the following figure. This uses the notations introduced in the previous section.

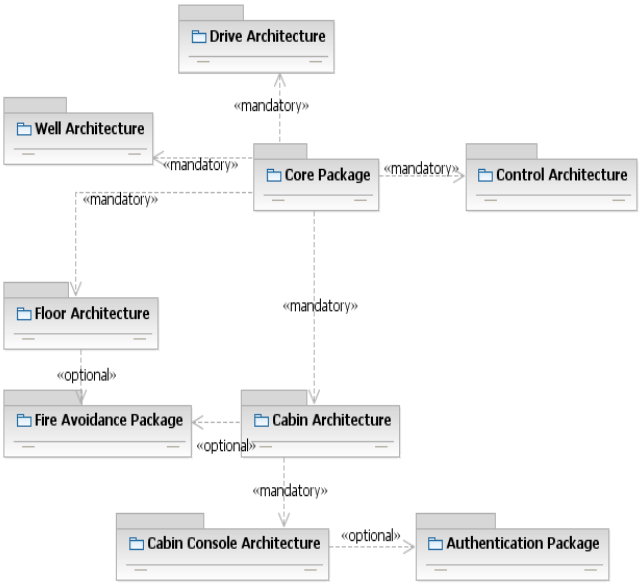

Fig. 7. Overview of packages with variants for an elevator product line.

Mandatory elements are imported to the core package from the different packages containing variants. If no optional elements have been selected, a closer look to the core package for elevators reveals the elements depicted in figure 8.

Selection and visualization of one variant is possible with the help of SysML-views. The following figure shows the situation of selecting the fingerprint authentication method

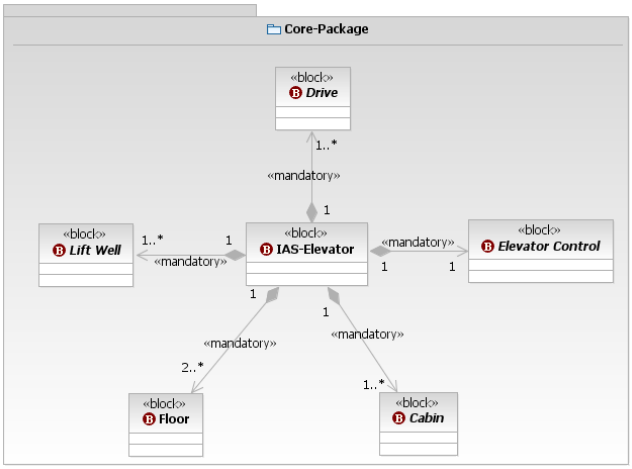

Fig. 8. Mandatory elements within the core package for elevators.

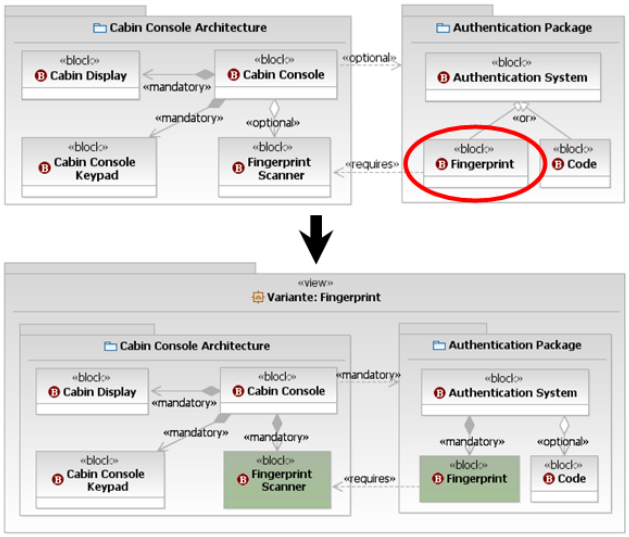

Fig. 9. Selection and visualization of one variant.

Please note that the presence of a fingerprint scanner is optional in the elevator's console architecture. When the variant "Fingerprint" is selected from the authentication package, then the fingerprint scanner becomes mandatory. This variant is displayed in the view with the same name.

4. ELEVATOR CONFIGURATION TOOLS

There are many tools available on the market for modeling variants. Widely-used in the industry and in the academia are CaptainFeature [BigLever 2010], Pure::Variants [Pure Systems 2010], Feature Plugin [Antkiewicz, Czarnecki 2004], XFeature [XFeature 2010] and VarMod [VarMod 2010]. A good overview on their features is given in [Djebbi et al, 2007]. Although technically very competitive, these tools are based on the separation of variability models and models of elevator's structure and behavior. In contrast, the approach presented in the previous section captures information about structure, behavior and variability of industrial automation systems in only one SysML model. This enables holistic modeling of the product line members.

Beyond modeling variants, the selection of one variant that matches the given project requirements deserves special attention. Particularly, in case of product lines with multiple variation possibilities it is difficult to decide which variants are appropriate. In order to mitigate this problem, the approach presented in the previous sections has been used at IAS for developing a configuration tool.

Elevator configuration tools are currently offered by many elevator providers. Free available tools are Kone MonoSpace® Toolbox [Kone 2010 a], Kone PlanulatorTM [Kone 2010 b], Otis Gen2 [Otis 2010], and Elevator Designer [Synthesis 2010]. Although these tools are useful and trivial to use, the configuration possibilities are limited. For instance, it is impossible to change the position of the machine room of a proposed elevator. Furthermore, the opening way of the elevator doors cannot be freely configured. Moreover, since the configuration tools are commercially motivated, only the variants available at the concerned providers are proposed. Hence, the decision to implement an own elevator configuration tool has been taken.

The configuration tool enables qualified recommendations of elevator variants to meet concrete project requirements. These requirements are regarded as input parameters. Examples of input parameters in the freight and passenger elevator domain are building type, number of passengers in the building, number of floors to be covered by the elevator and floor height. Depending on the input parameters and the valid norms and directives in the domain, the best matching variant is proposed. Moreover, the structure related to the variant is displayed, so that CAD-drawings can be generated. Same applies to behavioral models of the selected variant, which are used for generating simulations. A screenshot of the IAS elevator configuration tool is presented in figure 10.

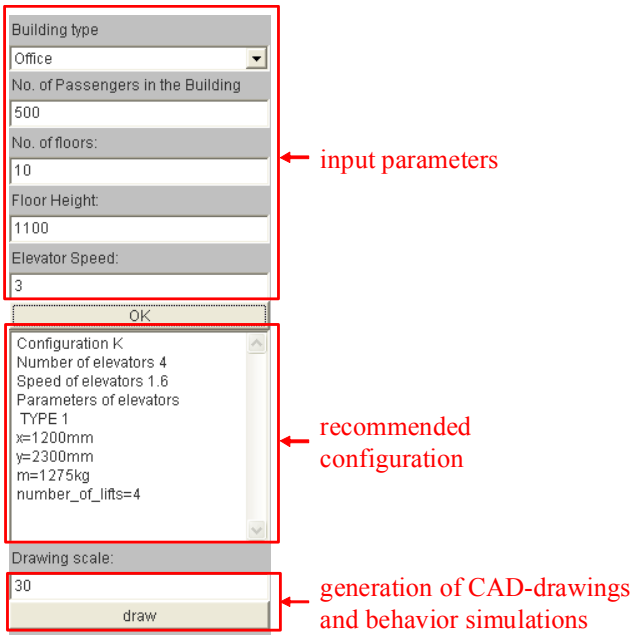

Fig. 10. Functionality of the IAS elevator configuration tool.

The decision for a member of the product line is taken according to the recommendations and best practices

described in [Schindler 2010]. By using the IAS elevator configuration tool it is possible to select the appropriate variant for different project specific requirements.

5. CONCLUSION

Product lines for industrial automation systems have a high degree of variability, in order to meet the requirements of the different stakeholders. This contribution addressed the issue of modeling variants of industrial automation systems. After an introduction in the used terminology, a state of the art survey in modeling variants was presented. Furthermore, a new approach for modeling variants in SysML was proposed and exemplarily deployed.

The new approach uses predefined extension mechanisms of SysML. Hence, it is automatically supported by software tools conform to the SysML specification [OMG 2008]. Furthermore, variants, structure and behavior of industrial automation systems are integrated in one model. This has positive effects on consistency, since only one model has to be maintained. Different diagram types are modeled within packages, so that the model remains understandable even by numerous variants. SysML modeling skills are sufficient for modeling both industrial automation systems and their variants. Finally, a configuration tool for selecting appropriate variants was proposed. Currently, a patent for the proposed approach of modeling variants with SysML is pending.

REFERENCES

BigLever Software Inc., http://www.biglever.com, July 2010.

C. Maga, N. Jazdi.: *Concept of a Domain Repository for Industrial Automation*. Domain Engineering Workshop at the 21st International Conference on Advanced Information Systems (DE@CAiSE'09), Amsterdam, the Netherlands, 2009.

C. Maga; N. Jazdi: *A Survey on Determining Factors for Modeling Reference Architectures*. Proceedings of OOPSLA, Orlando, Florida, USA, 2009.

C. Maga; N. Jazdi; T. Ehben; T. Tetzner: *Domain Engineering – Improved Systematisation in Industrial Solutions Business*. Proceedings of the Automation Congress, Baden-Baden, Germany, 2009.

G. Kaeding: *Product Lines in the Automotive Domain* (Produktlinien im Automobilbereich). Seminary Automotive Concepts and Techniques, Sommer Term 2009, University Koblenz-Landau, 2009.

G. Pahl, W. Beitz: *Design Basics* (Konstruktionslehre Grundlagen). 7.Aufl., Springer-Verlag, 2006.

H.-J. Franke, N. Firchau: *Zusammenfassender Zwischenbericht des Kalenderjahres 1998 für das BMBF-Verbundprojekt „Methoden und Werkzeuge zur Kostenreduktion variantenreicher Produktspektren in der Einzeln- und Kleinserienfertigung - EVAPRO*". TU Braunschweig, 1998.

I. Baumgart: *Modularisation of Products in Plant Engineering* (Modularisierung von Produkten im Anlagenbau). 1. Auflage, Mainz GmbH Verlag, 2005.

K. Czarnecki, U. Eisenecker: *Generative Programming*, Boston, San Francisco, New York: Addison-Wesley Verlag, 2000.

K. Kang, S. Cohen, J.A. Hess, W.E. Novak, S.A. Peterson: *Feature Oriented Domain Analysis (FODA) Feasability Study*. Technical Report, Software Engineering Institute (SEI), Carnegie-Mellon University, 1990.

K. Pohl; G. Böckle; F. Van der Linden: *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer Verlag, 2005.

Kone: http://www.kone.com/countries/de_DE/tools/Pages/default.aspx, accessed in July 2010.

Kone: https://www.kone.com/toolbox/start.html?locale=de_DE#cad, accessed in July 2010.

M. Antkiewicz, K.Czarnecki: *FeaturePlugin: Feature Modeling Plug-In for Eclipse*, OOPSLA, Canada, 2004.

M. Riebisch, K. Böllert, D. Streitferdt, B. Franczyk: *Extending the UML to model System Families*. Integrated Design and Process Technology (IDFT), USA, 2000.

O. Djebbi, C. Salinesi, G. Fanmuy: *Industry Survey of Product Lines Management Tools: Requirements, Qualities and Open Issues*. 15th IEEE International Requirements Engineering Conference, 2007.

OMG SysML Specification. www.omg.org, 2008.

Otis, http://www.otisgen2.com/index.shtml, July 2010.

P. Clemens, L. Northrop: *Software Product Lines: Practices and Patterns*. Addison-Wesley, Reading, Mass, 2002.

P. Göhner: *Industrial Automation*. Lecture Notes, Summer Term 2009, Institute of Industrial Automation and Software Engineering, Universität Stuttgart, 2009

pure-systems GmbH, www.pure-systems.com, accessed in January 2010.

Schindler, *Schindler Planungsnavigator e3b*, http://www.schindler.de/deu_index/deu_ser/deu_ser_wissen/deu_ser_wissen_nv4.htm, accessed in July 2010.

SYNTHESIS Company: Instant custom AutoCAD drawings. http://www.synthesiscompany.com/asp/default.asp?page=elevator, accessed in July 2010.

ThyssenKrupp Elevators: http://www.thyssenkrupp-aufzuege.de, accessed in January 2010.

VarMod tool, University Essen, http://www.sse.uni-essen.de/wms/en/index.php?go=256, accessed in January 2010.

XFeature Modeling Tool, http://www.pnp-software.com/XFeature/, accessed in July 2010.