Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Selecting Test Cases for Mechatronic Products with a Variant and Version Management Approach based on a Consistent Toolchain

K. Land*, B. Vogel-Heuser*, A. Gallasch,**
M. Sagerer*, D. Förster****, G. Strobl***

** Institute of Automation and Information Systems, Garching near Munich, 85425 Germany (e-mail: {kathrin.land, vogel-heuser, georg.strobl}@tum.de)*
*** Software Factory GmbH, 85748 Garching near Munich, 85425 Germany (e-mail: gallasch@sf.com)*
**** Hirschmann Automation and Control GmbH, Neckartenzlingen, 72654 Germany, (e-mail: michael.sagerer@belden.com)*
*****SCHUNK GmbH & Co. KG, Lauffen am Neckar, 74348 Germany (e-mail: dorothea.foerster@de.schunk.com)*

Abstract: The number of variants and versions for mechatronic products increases. The high variability poses a challenge for test engineers in selecting suitable test cases upon a change. If a requirement or a feature of a mechatronic product changes, it is not necessary to retest the whole product but only the changed parts. To identify the product features that are directly or indirectly affected by the change, a connection of test, requirement, and variant management is necessary. Therefore, an approach to select test cases based on an occurred change using variant and version knowledge is needed. In this paper, such an approach and its possible application in a toolchain are introduced. The toolchain is built by combining established tools developed by the Parametric Technology Corporation (PTC) that are already used to manage parts of the product life cycle. The resulting PTC Integrity Toolchain and the applicability of the concept on it were evaluated together with industrial experts with positive results.

Keywords: Logical design, physical design, implementation of embedded computer systems, Product Variance, Computer-aided testing

1. INTRODUCTION

Manufacturers in the mechanical engineering industry must be able to adapt their products to individual customer requirements (Hinterreiter et al., 2018), also during operation. It is time-consuming and hardly economically feasible to thoroughly test the high number of resulting product variants and versions. Therefore, test engineers have to select a subset of suitable test cases when making a change. Due to the increasing complexity of mechatronic systems, adequate test case selection is very demanding, since dependencies between subcomponents or different disciplines and thus the propagation of changes are not clearly recognizable.

A modelling concept for the representation of dependencies and interfaces of interdisciplinary and cross-disciplinary modules or features makes changes comprehensible. Product line engineering is ideal for keeping development costs and time in check with the resulting variety of variants and versions. A product line consists of variants that have many common features, which theoretically enables the reuse of these features and thus the corresponding testing knowledge of existing variants. However, a common procedure for a suitable documentation or handling of the variability, which is needed for an efficient reuse, is not yet established for aPS (Vogel-Heuser et al., 2015). There are no standardised tools available, hence in industry, companies instead (mis)use a combination of different tools. Even most of the individual workflows currently used by companies for variant management lack the connections from variant or version knowledge to requirements and test management and thus they lack the systematic consideration of connections and restrictions between features in test selection. The knowledge about change propagation is usually only implicit available (Vogel-Heuser et al., 2015) which leads to the risk that important test cases are missed and not executed. Given a transparent representation of the cross-relations in-between features and the linkage of the features of a variant to requirements and test cases, changes could be tracked more easily and thus the test engineer could be supported in the selection of suitable test cases.

This paper presents an approach and its possible application to manage variants and versions of mechatronic products and link them to knowledge from requirement and test management. The realisation is with a combination of established PTC Integrity Management tools and the modelling language SysML. The effects of changes to requirements can thus be tracked transparently and the test engineer is supported in selecting the required test cases.

The remainder of this paper is structured as follows: Section two provides an overview on related work on variant and version management for mechatronic products and its use for test case selection. Section three introduces shortly the general concept for test case selection based on variant and version knowledge. Subsequently, the possible realisation is presented. The concept and its applicability with the toolchain are evaluated with industrial experts in Section five. Finally, the paper is concluded in Section six.

2. RELATED WORK ON VARIABILITY MANAGEMENT AND ITS USAGE FOR TEST CASE SELECTION

Model-based software engineering, for example with the Systems Modelling Language (SysML), is gaining interest for the development of complex and interdisciplinary systems (Barbieri et al., 2014). Product line engineering (PLE) is a model-based method in software engineering based on modular artefacts, which can be reused and adapted to individual customer needs (Vogel-Heuser et al., 2015). The information on the artefacts, interrelations in-between different artefacts and existing product variants that consist of those artefacts reduce the effort needed to maintain existing product variants and to develop new variants. For the graphical representation of the artefacts and their interdependencies, "feature models and tools based on feature modelling are clearly dominating" (Berger et al., 2013).

2.1 Variability modelling with Feature Models

A feature model (also: 150% model) represents all features that are equal or different among similar products of a product line in a tree structure. It visualises the different types of interdependencies between features (mandatory, optional, alternative). A feature model that additionally represents versions is called 175% model (Lity et al., 2018). Feature models have a simple notation, which makes them easy to understand and it is possible to apply formalized analysis (Schröck et al., 2015). In industry, they are also used as communication base for product manager, software architects and developers (Hinterreiter et al., 2018) due to its useful abstraction. Hence, feature models are well suited to link different management views – requirements, test, variants and versions – during the product development and life cycle.

As this paper focuses on mechatronic products, not only software but also the mechanical and the electrical domain and their interdependencies have to be considered (Bąk et al., 2016). Feature modelling in the mechatronic domain is hardly researched (Vogel-Heuser et al. 2015). Feldmann et al. (2016) propose an approach to model variants and versions for interdisciplinary product lines (IPL). They split the feature model view into domain-specific subviews and model interdependencies separately by "feature interactions". Due to the separation, they obtain a clearer view of the variability. However, the feature interactions become complex and difficult to track. Kowal (2018) translates the relations to explicit propositional formulas to make the relations between the different feature models visible.

Boutkova (2011) presents an alternate approach to model all features in one model. She proposes a hierarchical view with different levels of abstraction. As the levels are interlinked, the abstract view is getting more detailed in lower levels. Furthermore, Boutkova (2011) links requirements to features in order to track changes. Therefore, she uses the management tool IBM Rational Doors. As the views are not separated as in Feldmann et al (2016), consistency checks for the whole model can be conducted. However, test cases are not linked nor selected in these approaches.

Papakonstantinou and Sierla (2013) propose a method for modelling interdisciplinary systems and their variability with feature models using the Systems Modelling Language (SysML). Their approach is based on a package structure, which resembles a classic digital folder structure. Packages in SysML can have specific connections to each other as the ones known from feature models. The packages can contain requirements, structures, functions or test cases (Haber et al., 2011). The advantage of this is a hierarchical structure which minimises redundancies and improves the clarity of the product line, its features and their interdependencies. A further advantage is that SysML models are feasible for model driven development due to their standardisation. In comparison to other modelling languages, SysML is especially promising for mechatronic products and the implementation of programmable logic controllers as parts of the domain-specific programming language IEC 61131-3 can be generated from it (Legat, 2018). However, SysML is only limited suitable for the modelling of product variability. For this reason, an own modelling language with a suitable diagram, the Orthogonal Variability Model (OVM), was explicitly developed that can be connected to SysML model elements. OVMs represent only the variation points of a product line and the dependencies between these variation points. "Exclude"-relationships between features and variation points visualise which features anticipate which decisions. Thus, they provide a clearer view than feature models (Metzger and Pohl, 2014) for the user when creating a new variant from a product line. In contrast to the classic feature model, the possible variants are directly visible in the OVM, but the structure of the variants from features and the link to requirements and test cases are missing.

2.2 Test Case Selection based on Variants and Versions

In most companies the knowledge needed for efficient testing is only implicit available and based on the experience of the test engineers. In order to automatically select test cases based on changes and variant and version knowledge, variant and version management must be linked to requirements and test management. Feature models are particularly suitable for managing variants and versions with the goal of precise test case selection due to the formal configuration check methods and the representation of cross-relationships between the features. If the interdependencies between the features are known and visible, changes and their effects are easy to track. Hence, the test engineer can be supported in selecting the relevant test cases. Thüm et al. (2014) developed an algorithm to determine changes between feature models mathematically, but the interdisciplinary character of mechatronic systems is not considered and the results are not used in order to reduce the necessary test suite for new variants or versions. Lochau et al. (2014) proposed

"Variability-Aware Product Line Testing", in which the tested software must be modelled as software product lines. The IMoTEP tool they developed uses a feature model, a feature-tagged state diagram test model, and a test coverage criterion to generate the required test suite. However, this approach is hardly applicable to IPL and their mechatronic character. There are already model-based approaches to linking requirements and test cases, e.g. by modelling the requirements in OWL (Web Ontology Language). The approach provides a suitable method to specify mechatronic systems in a structured way, but is not applicable for requirements and test case management due to the lack of support for time dependency and mathematical relationships. All in all there are hardly approaches available that focus on test selection for SPL (Engström, 2010), especially for mechatronic products and with a consistent tool support. Wang et al. (2013) model a test feature model in parallel to the original feature model and mapped the elements to select test cases automatically based on manually selected features.

3. GENERAL TEST CASE SELECTION CONCEPT

Feature models visualise possible different variants of a product. Each variant consists of several features that can be described by requirements. The requirements are verified by test cases (cf. Figure 1). Ideally, the test cases are created or generated automatically (Sinha et al., 2016) based on the individual requirements so that a direct connection is already known. In order to select test cases based on a change in requirement or in feature combination, all these elements have to be connected to each other. Upon a change in requirements, the affected features and corresponding test cases can be chosen directly due to this linkage.

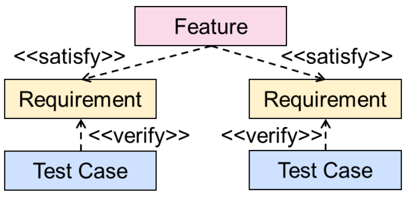

Figure 2: Connection of model elements

Figure 3 presents an exemplary feature model of a crane (in SysML). The crane consists of a turning table and can have two different kinds of grippers – mechanic or pneumatic. The pneumatic gripper variant requires an air compressor so there is a dependency between those two feature modules (block). If one of the blocks changes, the blocks affected by this change can be determined based on this interrelations. Also, blocks that are not affected can be determined and so the amount of test cases to be executed to verify the changed parts is reduced.

To explain the method, it is assumed that there already exists a variant with the pneumatic gripper which is already tested. If a change in requirement occurred for the air compressor, its test cases, the test cases of the dependent pneumatic gripper and those of the general gripper are chosen. As the turning table is independent of the air compressor, it is assumed to be not affected by the change and thus does not need to be retested. All necessary test cases are selected based on the (dependency) links within the feature model. The same applies if a new variant with the mechanic gripper is created. As the gripper is independent of the turning table, the change in gripper type is assumed to not affect it and hence it does not need to be retested within this new variant.

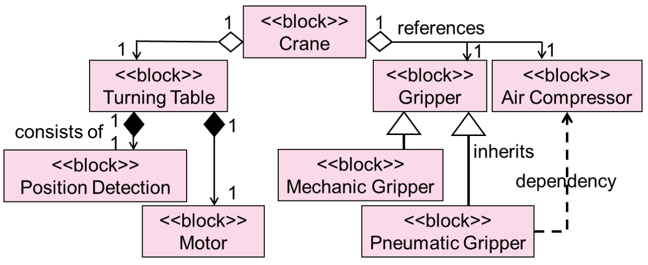

Figure 3: Excerpt of Feature Model in SysML

4. REALISATION APPROACH WITH PTC INTEGRITY TOOLS COMBINATION

In order to elaborate a possible application of the approach proposed, a toolchain was created from existing commercial PTC Integrity tools. The PTC Integrity tools are used in industry separately for different management tasks – e.g. for requirement management, for variant management or as module library. Figure 4 illustrates how the individual tools were used in order to realise the concept proposed. The use of established tools promises higher acceptance in industry. The triangular symbolizes the rising complexity of the different management views over time. In the following, the three tools Integrity Lifecycle Manager, Integrity Modeler and Integrity Asset Library are presented. Further, Figure 5 shows as result the combination of the several tools so that a toolchain that assists the tester or engineer in consistent management is established. The tools themselves and the synchronizer were already available. However, several adaptions for their combination as well as the creation of a variant library were necessary.

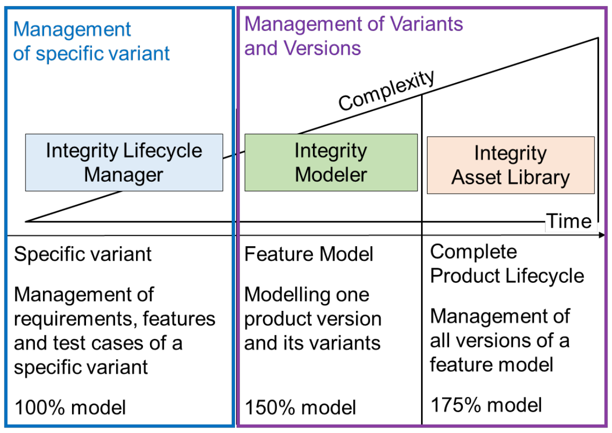
Management of specific variant
Management of Variants and Versions
Complexity
Integrity Lifecycle Manager
Integrity Modeler
Integrity Asset Library
Time
Specific variant
Management of requirements, features and test cases of a specific variant
100% model
Feature Model
Modelling one product version and its variants
150% model
Complete Product Lifecycle
Management of all versions of a feature model
175% model

Figure 4: PTC Integrity Tools and Usage for Approach

The Integrity Lifecycle Manager manages the requirements and test cases of specific product variants. The tool is used to manage the lifecycle of a variant of a specific customer and thus provides an overview of which variant and which version of the product is present at the customer. This information also facilitates subsequent maintenance and further development at the customer. The Lifecycle Manager had to be connected to the Modeler via a synchronizer, which

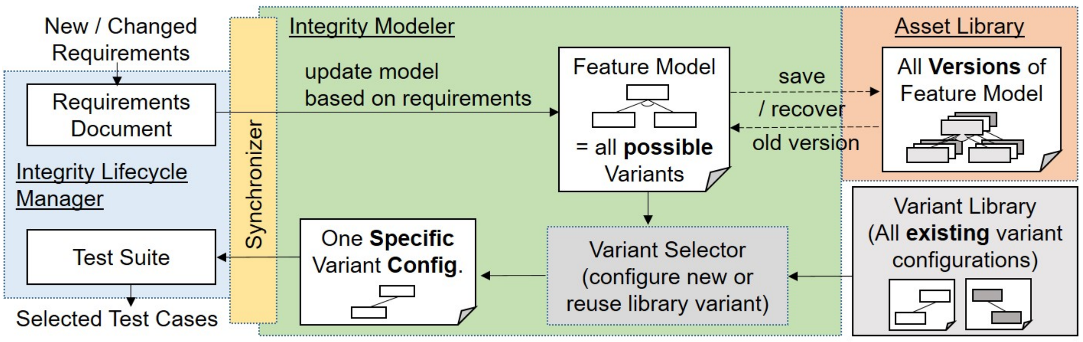
New / Changed Requirements
Requirements Document
Integrity Lifecycle Manager
Test Suite
Selected Test Cases
Synchronizer
Integrity Modeler
update model based on requirements
Feature Model
= all possible Variants
save
/ recover
old version
Asset Library
All Versions of Feature Model
Variant Library (All existing variant configurations)
Variant Selector (configure new or reuse library variant)
One Specific Variant Config.

Figure 5: Requirement Change Workflow using PTC Integrity Tools Combination

maps the Modeler feature packages (from feature model) to the data and documents of the Lifecycle Manager (Figure 5). If there are new requirements or an existing requirement is modified, the corresponding (requirement) documents are created or adapted within the Lifecycle Manager.

The synchronizer between the Modeler and the Lifecycle Manager uses a mapping function to connect requirements and test cases of the Modeler (cf. Figure 2) to requirements and test cases in the Lifecycle Manager. The corresponding mapping function had to be adapted and configurated manually so that the desired data is synchronized correctly. The synchronizer was mainly intended for transferring requirements into the Modeler. Thus, if there is a change that was not triggered due to a change in requirements, the requirements cannot be automatically updated. Upon a changed requirement and the thus new documents, a new feature version is created with the Integrity Modeler.

The Integrity Modeler can be used to graphically model software in SysML and manages both, the product line as 150% model and the specific variants as 100% models. The SysML requirement diagram is used to represent the relationships between the model elements "blocks", "test cases" and "requirements". The requirement diagram (cf. Figure 2) is therefore well suited for modelling features, their requirements and test cases. The combination of features modelled in SysML as well as their associated requirements and test cases are summarized as a feature module based on the package representation proposed, in order to be particularly clear for industrial applications. A block diagram (cf. Figure 3) is used for the feature model in which the feature modules are integrated.

Within the Integrity Modeler, there is a tool called "Variant Selector". It is used to generate a 100% model out of the 150% model. Based on the OVM model, the Variant Selector knows the structure of the model with its including and excluding dependencies and structures the variant creation process. If two features are mutually exclusive, the selection box for the other feature is automatically deactivated when one feature is selected. In addition, the SysML profile in the Integrity Modeler must be slightly adapted in order to successfully implement the developed concept and its implementation. For example, if one feature inherits from another, dependencies to other features must be inherited as well. The Variant Modeler is used to configure the new variant or version. To avoid redundant variants and versions in the database, the variant selector checks whether the variant configuration selected already exists in the variant library before creating the new variant (100% model). If an existing variant can be reused, no additional testing is required. The variant selector is only dependent on the Modeler and it is proven so that there were no adaptions necessary to make it work.

The Asset Library is a web-based asset management application. It contains all historical data and was used as version management and as a template for further variants. Since the model shown here contains all versions of each of the structural elements of the 150% model and can thus track the product lifecycle, it is also called the 175% model. By the modular structure of a system from features, the features can be separated well and can be reused and embedded into the structures of another model. Thus, the Asset Library saves the redundant development of existing features and supports central maintenance. The location of the feature is saved in the package structure of the SysML model and can be called up when it is inserted into another model again. An Asset Library bridge makes it possible for the Modeler to access the Asset Library databases (Figure 5). If the newly created variant or version is not yet in the Asset Library, the Library is updated by the Integrity Modeler. Finally, the data of the new variant or version is updated within the Lifecycle Manager. Hereby, all elements that were affected by the change are marked with a specific state for test case selection.

4.3 Test Case Selection

The method for test case selection is twofold if using this toolchain, depending on whether it is a change in variant configuration or requirements. Based on an existing variant, a new version of that variant or a new variant can be created using the "Variant Selector" tool integrated in the Modeler. In this case the user selects an existing variant and the Modeler displays its configuration. The user then applies his changes e.g. choosing another feature based on the underlying OVM or changing the requirement parameter of one of the features.

Alternatively, the requirements of an existing variant could also be changed or adapted through the Lifecycle Manager. The lifecycle manager reflects all requirements and test cases of one of the existing 100% models within the Integrity Modeler. If a requirement or test case is changed within the Lifecycle Manager, the corresponding 100% model in the Integrity Modeler is updated and the changed features and all affected features are marked for testing.

In both cases, the changes made by the user are marked for traceability. The marking is done by a status that was added to each changed element and all elements that are affected by this change according to the 150% model. The status changes from "checked" or "released" to "unchecked". This identifies all feature modules that need to be re-checked because of the change. Affected feature modules are those that either have a direct dependency relationship like one feature requiring another, or that have an indirect dependency relationship through inheritance, for example. The SysML profile was further adapted so that feature modules inherit their dependencies to their underlying instances. By creating a new variant or version, the 175% model of the asset library is updated. To test the new variant or version, all test cases that belong to a feature module that has the status "untested" can be selected. This is done with specific filter functions that are provided by the Lifecycle Manager.

5. APPLICABILITY EVALUATION OF TOOLCHAIN APPROACH

The workflow was evaluated using the model of a demonstrator production plant of the institute. The plant is made of various interdisciplinary components that include mechanical, electrical and software components. The crane which was partly shown in Figure 3 is part of that plant. The concept and the toolchain were rated by several industrial partners iteratively and partly separately as well as in a final joint meeting in use cases.

As evaluation use case, a new variant based on an existing variant was designed with the Variant Selector. First, a requirement parameter of the air compressor (cf. Figure 3) was changed. After making sure that this variant does not yet exist by checking the variant library, the resulting variant was created. As expected, the air compressor feature module and all dependant feature modules were thereafter marked as “untested”. After synchronising this information with the Lifecycle Manager, all test cases that also obtained the “untested” tag could be selected. However, the user can change only one requirement parameter in the Variant Selector. For a comprehensive parameterization of requirements and test cases this is not sufficient and has to be added in the future for a better usability. Also the new variant is saved to the same data storage as the 150% model, which causes the 150% to be locked for further use until the variant is moved to another data storage field. This results in a less efficient procedure. This issue could be solved by adapting the tools so that the new variant is directly saved elsewhere.

Despite those unsolved issues within the toolchain, the overall concept was shown to be applicable in an industrial toolchain. The industrial partners emphasized positively the workflow using the toolchain to add or change requirements and to detect and select all corresponding test cases. The introduction of a status such as "tested" and "untested" for requirements and blocks improves the clarity and traceability within the current testing progress in the Modeler and Lifecycle Manager. Further, relevant test cases are intuitive to select so that the manual effort (e.g. test case selection time) of the test engineer and the risk to miss test cases that cover the change are considered reduced. Also, the clear differentiation and usage of the three model types (100%, 150% and 175%) and their management within the tool was considered promising.

The toolchain links data of test, requirement, variant and version management. This way, it assists in synchronising the data, which reduces the risk of inconsistencies due to manual synchronisation. However, the industrial partners noted that a high level of initial effort is required to adapt their individual solutions and established tools and structures to the toolchain. The initial effort can be partially reduced as the Integrity Modeler offers standardised interfaces to import data or to connect other tools. Common tools as for example tool IBM Rational Doors for requirement management or pure::variants to model variability are supported. However, the actual effort to import and map the data provided by these tools was not evaluated in this scope. Also, the synchroniser between Modeler and Lifecycle Manager (cf. Figure 5) was only beta version and its main purpose was to enable the transfer of requirements from the Lifecycle Manager to the Modeler. Due to that, changes that occur on a test case (e.g. on-site adaption or aging of test cases) are not considered or traced back to the influenced features and requirements. This issue will be subject to future research. Further, the mapping to transfer the information from the Integrity Modeler back to the Lifecycle Manager with the synchronizer had to be added manually, which is time-consuming.

According to the experts, the feature-model-based approach was convenient due to the easy to understand notation and the overview of dependencies. The combination of SysML and OVM enables the modelling of interdisciplinary products and a separate, constraint view on the variation points. With the SysML feature model, the variant solution space can be modelled clearly. The dependencies between features are further detailed with OVMs. As the features are structured in packages, the view on the model is clear despite its complexity. However, a strict hierarchical modelling is needed to achieve this clearness and thus a good scalability. This hierarchical modelling is not always obvious for mechatronic products due to the interrelations and different modelling approaches (customer view, developer view, tester view) in industry. As the optimal suitable modelling approach differs depending on the use-case, a product-specific choice has to be made within the respective companies.

6. CONCLUSION AND OUTLOOK

In this paper, test cases were linked to variants and versions in order to select them based on occurred changes within the product line. Therefore, the feature modelling approach, which is widely used for product line engineering in the software domain was used and enhanced with test information. For the approach a toolchain made of established PTC Integrity Tools was build and adapted. Through the interaction of the modelling languages SysML and OVM and the Variant Selector of the Modeler, it is possible to display the entire variant space clearly. With SysML, the architecture of the model and the variant space can be represented in a structured way. However, dependencies between the feature modules cannot be defined in more detail. Using the OVM model, dependencies can be represented in more detail in the PTC Toolchain by including and excluding conditions and the assignment of feature modules to variants can be determined. The Variant Selector provides a structured selection process for feasible variants.

A structure could be created that allows the feature modules to be combined into product variants through a modular system. It was then adapted to the modelling language SysML. This approach serves the hierarchical and modular implementation in the toolchain. For evaluation, a practical adaptation of the concept into the toolchain could be presented. This showed that the developed concept and the Integrity Toolchain can be used to track the effects of requirement changes. This possibility represents a basis for the automation of the selection of affected test cases and the test coverage estimation of a new variant or version. The positive feedback from industrial experts confirms the concept and the approach with the toolchain. The concept has been successfully applied to the industrial toolchain, but there are still a number of technical challenges that need to be resolved before it can be fully implemented for practical use.

The variant selector within the Modeler allows requirement parameterisation. This is a promising approach to easily adapt generic test frameworks for automatic test execution. However, currently only one parameter per requirement is adaptable, which restricts the flexibility of the requirements. In future work, this restriction should be lifted. Furthermore, an advancement of the synchronizer is expected that solves its issue with the one-sided communication. Also, the approach would further benefit of an automatic mapping across the tools instead of the manual mapping. Currently, the changes that are made to an existing variant to create a new variant are tracked and marked. Hence, the new variant is compared to the variant the test engineer selected as basis for the creation of the new variant. This variant is not necessarily the one with the least difference to the new variant. Thus for future improvement of this approach, a newly designed variant should be compared systematically to all existing variants to determine the least difference and thus the least number of test cases required to cover the change. Furthermore, the applicability to different types of industrial examples or plants shall be considered in future work.

ACKNOWLEDGMENT

The research results presented in this paper were developed in the IGF-project 19090 N/1 of the research association Forschungskuratorium Maschinenbau eV. Thanks to all participating industrial partners. The project was funded by the AiF as part of the program for the promotion of industrial joint research (IGF) by the Federal Ministry for Economic Affairs and Energy.

REFERENCES

Bąk, K., Diskin, Z., Antkiewicz, M., Czarnecki, K., Wąsowski, A. (2016): Clafer: unifying class and feature modeling. In: Softw Syst Model 15 (3), S. 811–845.

Barbieri, G., Fantuzzi, C. and Borsari, R. (2014). A model-based design methodology for the development of mechatronic systems. Mechatronics 24(7), p. 833–843.

Berger, T., Rublack, R., Nair, D., Atlee, J., et al. (2013). A survey of variability modeling in industrial practice. *VAMOS*, pp- 7-14.

Boutkova, E. (2011). Experience with variability management in requirement specifications. *SPLC*, pp. 303–312.

Engström, E. (2010). Regression Test Selection and Product Line System Testing. ICST 2010, pp. 512 - 515.

Feldmann, S. and Vogel-Heuser, B. (2016). Interdisciplinary product lines to support the engineering in the machine manufacturing domain. *International Journal of Production Research,* 55(13), 3701–3714.

Haber, A., Rendel, H., Rumpe, B., Schaefer, I. and van der Linden, F. (2011). Hierarchical Variability Modeling for Software Architectures. *SPLC,* pp. 150-159.

Hinterreiter, D., Prähofer, H., Linsbauer, L., Grünbacher, P., Reisinger, F. and Egyed, A. (2018). Feature-Oriented Evolution of Automation Software Systems in Industrial Software Ecosystems. *IEEE ETFA*, pp. 107-114.

Kowal, M. (2018). Interdisciplinary Variability Modeling and Performance Analysis for Long-Living Software Systems. Dissertation. DOI: 10.24355/dbbs.084-201802071247

Legat, C. (2018). Automated orchestration planning of field level software functions in the domain of machine and plant automation. Dissertation. Göttingen: *sierke.*

Lity, S., Nahrendorf, S., Thüm, T., Seidl, C. and Schaefer, I. (2018). 175% Modeling for Product-Line Evolution of Domain Artifacts. *VAMOS*, pp. 27–34.

Lochau, M., Bürdek, J., Lity, S., Hagner, M., Legat, C., Goltz, U. and Schürr, A. (2014). Applying Model-based Software Product Line Testing Approaches to the Automation Engineering Domain. *at-Automatisierungstechnik,* 62(11), pp. 771-780.

Metzger, A. and Pohl, K. (2014). Software product line engineering and variability management: achievements and challenges. *ACM FOSE*, pp. 70-84.

Papakonstantinou, N. and Sierla, S. (2013) Generating an Object Oriented IEC 61131-3 software product line architecture from SysML. *IEEE ETFA*, pp. 1–8.

Schröck, S., Fay, A. and Jäger, T. (2015). Systematic interdisciplinary reuse within the engineering of automated plants. *IEEE SysCon*, pp. 508-515.

Sinha, R., Pang, C., Martínez, G., et al. (2016). Automatic test case generation from requirements for industrial cyber-physical systems. *at - Automatisierungstechnik*, 64(3), pp. 216-230.

Thüm, T., Kästner, C., Benduhn, F., Meinicke, J., Saake, G. and Leich, T. (2014). FeatureIDE: An Extensible Framework for Feature-Oriented Software Development. *Science of Computer Programming*, 79(0), pp. 70–85.

Vogel-Heuser, B.; Fay, A.; Schaefer, I.; Tichy, M. (2015) Evolution of software in automated production systems - Challenges and Research Directions. *JSS*, 110, pp. 54-84.

Wang, S., Gotlieb, A., Ali, S., Liaaen, M. (2013). Automated Test Case Selection Using Feature Model: An Industrial Case Study. *MODELS*, vol. 8107 pp. 237-253.