*Transdisciplinary Engineering for Complex Socio-technical Systems – Real-life Applications*
*J. Pokojski et al. (Eds.)*

*doi:10.3233/ATDE200103*

# scone – A Requirements Management Tool for the Specification and Variability-Based Analysis of Product Lines

Sara ALLABAR[a], Christian BETTINGER[a], Michael MÜLLEN[a] and Georg ROCK[a,1]
[a] *Trier University of Applied Sciences, Germany*

**Abstract.** Nowadays, industrial products as well as software applications are expected to be tailored to the user's needs in an increasingly distinct manner. This often makes it necessary to design a vast number of customized variants, which leads to complex and error prone analysis and development processes. Generally, requirements engineering is considered to be one of the most significant activities in software and system development. Variant management has proven to play an important role in handling the complexity arising from mass-customization of products. However, there are only a few, often rather complex-to-use, applications which allow adding variance information directly to requirements. Especially in case of small and medium sized enterprises, approaches to meet this challenge often result in isolated solutions that are not driven by state-of-the-art analysis methods and do not cope with future requirements. This paper introduces a lightweight requirements management tool called scone, which will be embedded into an overall variability management methodology. scone enables the user to create and manage requirement specifications and augment them with variability information. Based on this specification, the requirements can be analyzed in a formal way with respect to their variability using the variability management tool Glencoe. scone was created as a single-page web application to eliminate the need for installation and allow it to run on many devices, while offering the experience of working with a native application, rather than a website. Both tools are designed to provide a proof of concept for the seamless integration of variability information within a system development process as well as to show how variability can be handled in an easy-to-use way from the very beginning within this process.

**Keywords.** Requirements engineering, product line engineering, variant management, collaborative teamwork, design of personalized products and services

## Introduction

The quality of a product significantly depends on the extent to which it meets its requirements. Therefore, it is commonly accepted that requirements management plays a key role in the development process. Defects and errors that are noticed and fixed at the beginning of a development cost very little, whereas a mistake that is discovered after delivery can cause great damage. This situation becomes even worse when variability is considered, which greatly increases the complexity of the products developed. Growing customer demands and expectations make it necessary to create multi-variant products that are able to meet a broader spectrum of needs. In industrial projects the difficulties

[1] Corresponding Author, Mail: g.rock@hochschule-trier.de.

associated with the described situation are apparently often avoided by initially ignoring variability. Only the requirements are described and variants are added at a later and therefore often inappropriate stage in development. The automotive industry has learned from this situation and has established a professional platform-based variant development process that allows to cope with the described difficulties. Although well-established companies have partly mastered the problem, it remains a challenge for small and medium-sized companies to integrate requirements management and variant management successfully and cost-effectively into their overall development processes.

At Trier University of Applied Sciences an application called Glencoe[2] is being developed since 2012 with which multi-variant product lines can be specified, visualized and formally analyzed. It is a web application which provides the user with the possibility to create and edit feature models in an accessible and user-friendly way. Since requirements management and variability are closely connected we developed an easy-to-use tool for requirements management and extended it with functionality for adding variance information to the requirements specification.

In this paper we present an easy to use platform independent requirements engineering tool that besides the usual functionality allows to specify variance information on requirements level. A loose, partly automated, integration with Glencoe gives the user the possibility to build a variability perspective on its requirements specification. This perspective can be used by several stakeholders involved in the project to clarify open questions concerning variability across disciplines.

## 1. Initial considerations

### *1.1 Obstacles for small teams*

As part of the planed overall development methodology [3] scone has to follow the development paradigm to keep things as simple as possible for the user. As part of the previous work [1], in which a first prototype of scone was designed and implemented, a survey on requirements management was conducted with the aim to find reasons why requirements engineering was not successfully applied during a product development. The participants' main reasons for not carrying out requirements management were:

- There was no time. Requirements management would have been too time consuming.
- The Participants stated that there was no need for requirements management because the project was so small.

Furthermore, the participants argued that in case they did apply requirements management they decided against using a professional tool because:

- The effort to become familiar with the tool was too high.
- There had to be too much research to do.
- The tools are too complex.

[2] https://www.glencoe.hochschule-trier.de

We could not find any studies on the influence of the existing range of applications on the actual implementation of requirements management. Thus, the following considerations are only based on the survey conducted.

The reasons for the rejection of a professional tool may be due to the fact that standard industrial applications are suitable for a great variety of projects and therefor possess a vast number of – for small projects often – unneeded features that can initially be overwhelming to the user. If these tools are not documented well enough, or conversely have a relatively extensive documentation resulting from the number of features and use cases, users may find it difficult and too time consuming to learn how to use the tool and to familiarize themselves with it. According to [2] helpful resources like user guides and example solutions are often not available, making it harder for the user to learn how to use a tool. Novice users in particular may be put off and prefer using non-professional or not well-suited tools that are easy to use and well understood, or even stop performing requirements management altogether.

Another reason for small teams to reject professional tools may stem from the time needed to set up the tool or rather that there is no easy access. According to a systematic literature review [2] which compared 41 requirements management tools, the vast majority of the tools are not online, meaning the user needs to set up a tool on premise. More than half of the tools come as plugins, that require other already established and understood software and cannot be used alone, which means an additional obstacle for small teams and inexperienced users [2].

## *1.2 Mapping a requirements specification to a feature model*

Requirements management is often the first step in system development, which nowadays often involves several disciplines. Together, they try to work out a specification that reflects the requirements of different users in different variations. In order to overcome the before mentioned shortcomings we started to implement the tools to realize our before mentioned overall methodology. Since requirements management and variability are closely connected and Glencoe was already in place, we wanted to leverage the advantages of both tools to provide the user with the possibility to specify requirements for product lines and to visualize and analyze variability related properties in Glencoe. The variability information of the requirements specification has to be converted (automatically) into a corresponding feature model, that is delivered to Glencoe. This way the variability perspective on the specified product could be automatically generated and all the variant dependent information could be visualized to the user in a so-called Glencoe Feature Model (GFM) (see [3] for more on GFMs). Using this perspective gives the user also the possibility to analyze the specified variability using the built-in formal analysis operations, as there is a consistency checker, a dead feature checker or to count the number of different product variants to name just a few.

One challenge was to make sure that the considerations described in 1.1 are not neglected and that the features for specifying requirements and variability in scone do not collide with the ease-of-use paradigm. Emphasis was placed on enabling the user to specify requirements without having to know product line management and without the need to understand and learn the corresponding features of scone, so that the user will not be discouraged.

## 2. scone

Considering the results of the aforementioned survey and our own experiences in using tools for requirements management while simultaneously learning how to perform requirements management and variant management, we wanted to create a tool that allows even small teams to carry out requirements management without having to invest much time in mastering the tool. Since we wanted to develop an application that is intuitive to use and can be utilized without much extra effort, we reduced the functionality to the most basic functions needed to perform requirements management. This scope of the functionality is described in detail in [1].

The first prototype of scone was developed in 2017 with the intention of making requirements management accessible to students in university sized projects. In 2018 the application was redesigned and extended with a connection to Glencoe in order to deal with variability management. Nevertheless, its use should remain as straightforward as possible. scone currently enables the user to create requirement specifications and offers the possibility to augment requirements with additional variability information. The requirements can then be visualized and analyzed with the help of Glencoe and the results of the analysis can be leveraged by the user to adjust the requirements specification and the corresponding variability information in scone.

### *2.1 Extension of the requirements specification*

The first prototype of scone represented requirements as a simple list (see Figure 1, left side) with no possibility of defining a hierarchy. Sub-systems and different parts of products are often hierarchical in nature and feature models contain features in hierarchies. The first step in mapping a requirements specification to a feature model is to gather corresponding requirements for clearly distinguishable features and to provide a possibility to create hierarchies. For the sake of this we introduced Requirement Groups (RGs). Every project created in scone initially contains a root requirement group. Subsequently requirements and other RGs can be defined in it. Thus, the hierarchical nature of a product can be represented, as every feature is described by one RG containing all requirements for that feature and other RGs for its sub-features (see Figure 1, right side). The grouping of the RGs serves to improve exportability as a Glencoe feature model as well as to structure the requirements specification, since they also function as headings. Furthermore, they can be used to describe variants of product features, where one requirement group would gather all requirements and RGs for a variant.

To emulate the logical relationships between features, scone uses the same types as Glencoe. It is possible to define RGs as *Feature* (the content of the RG describes one distinct product feature), *Alternative* (the content of the RG consists of alternatives of which only one can be part of a valid product), or *Selection* (only a certain number of the direct children of the RG can be part of a valid product).

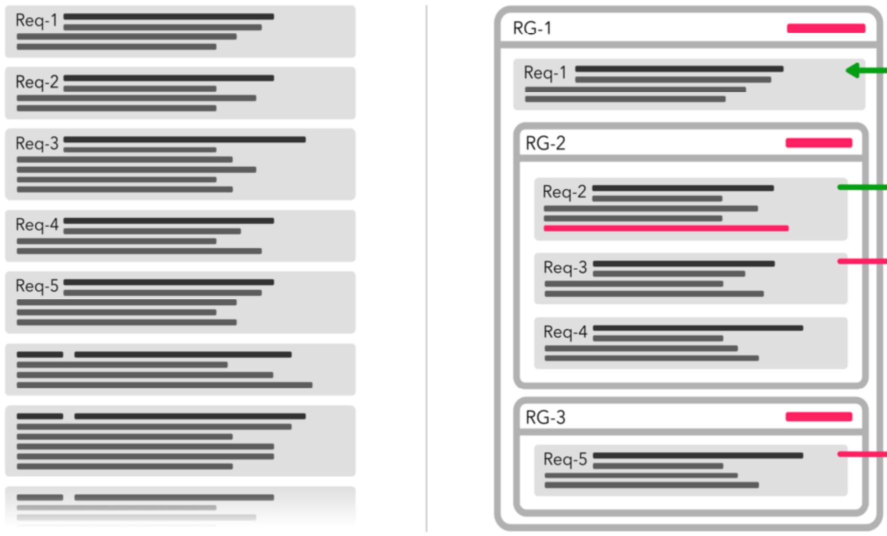


**Figure 1**. Schematic view of the representation of the requirements at the starting point and in the as-is situation in scone. The left side shows a list of requirements without hierarchical order and variance information. The right part of the figure shows the requirements aggregated in requirement groups with colored variance (excludes and implies) information.

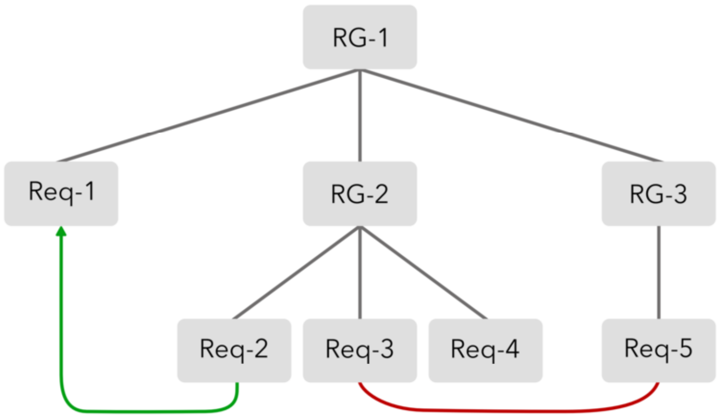


**Figure 2**. The example requirement specification from figure 1 displayed as a feature model.

### *2.2 Handling of constraints*

To express inter-dependencies between features and variants, we added the possibility to define constraints for requirements and RGs which are called *links* in scone. There are mainly three types of links: *excludes* (if A is part of the product, B cannot be and vice versa), *requires* (if A is part of the product, B must be, too), and *refines* (A further defines B), with refines being a constraint that can only be used for requirements. The first two translate directly to their Glencoe equivalents *Requirement* and *Mutual Exclusion*. The refines-constraint translates to two linked features, A and B, with B being the child of A. A schematic view of requirements with constraints can be seen in figure 1 on the right side.

### *2.3 Working with scone and Glencoe*

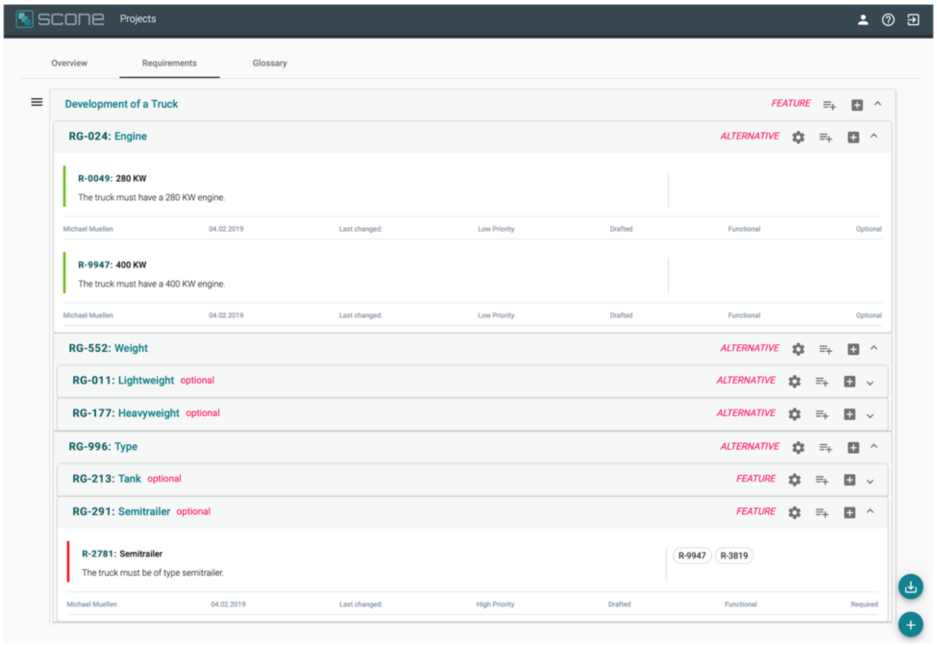
The transmission of variant information from scone to Glencoe can be performed by the user at the desired stage of development. Thus, a variability view (see figure 2 for an example) is generated which can be visualized and analyzed in Glencoe. If errors or defects are detected, the corresponding experts have to be contacted in order to fix these problems at a very early stage during the development. Besides the possibility to formally analyze the variability of the product line the requirements engineer and the variability engineer can work together on the basis of the visualized feature model to define the appropriate variability.

As both tools put their emphasis on being easy to use and the visualization provided by Glencoe serves a better understanding of the product, also non-technical users can be more closely involved in the process. Furthermore, the visualization of the requirements specification can be used to create a shared understanding of the product and to facilitate cooperation in a project as mentioned before.

Figure 3 shows a screenshot of scone illustrating requirement groups, subgroups and requirements. An export of this requirement specification results in a Glencoe feature model that can be directly imported and visualized in Glencoe, for example using the Tree view – a representation based on the FODA notation, which was introduced in 1990 by Kang et al. [4] –, as shown in Figure 4.

If errors, like inconsistencies or dead features, are found, the requirements specification can be corrected in scone as needed. The cooperation of both tools makes it possible to detect errors or flaws in a very early stage of development, which can help to drastically shorten the development time and effectively reduce costs. Furthermore, it improves the understanding of the product and its variability as the analysis procedures of Glencoe help give some more insight into the specified variability as there are metrics to find common features, unique features and compute all possible configurations.

Figure 4 shows the analyzed sample project where Glencoe has found two dead features (crossed out red), meaning these features are not part of any configuration. Those could either result from errors in the specification that can subsequently be corrected, or the corresponding requirements are simply not needed and may be omitted. At this stage a decision has to be made how to handle such an error. Glencoe supports the user by providing an interactive proof representation that allows the user to identify possible causes for this defect. See [3] for more information on the formal analysis procedures used in Glencoe.


scone Projects

Overview Requirements Glossary

Development of a Truck FEATURE

RG-024: Engine ALTERNATIVE

R-0049: 280 KW
The truck must have a 280 KW engine.
Michael Muellen 04.02.2019 Last changed: Low Priority Drafted Functional Optional

R-9947: 400 KW
The truck must have a 400 KW engine.
Michael Muellen 04.02.2019 Last changed: Low Priority Drafted Functional Optional

RG-552: Weight ALTERNATIVE

RG-011: Lightweight optional ALTERNATIVE

RG-177: Heavyweight optional ALTERNATIVE

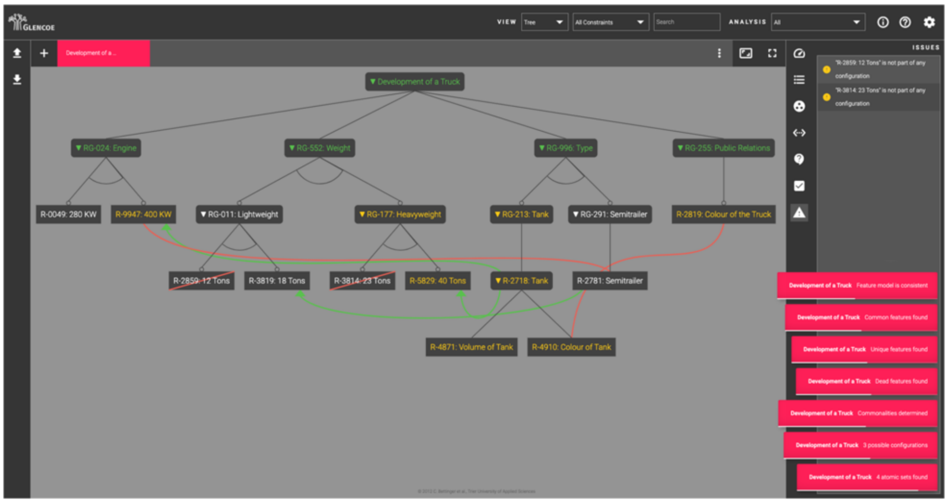
RG-996: Type ALTERNATIVE

RG-213: Tank optional FEATURE

RG-291: Semitrailer optional FEATURE

R-2781: Semitrailer
The truck must be of type semitrailer.
R-9947 R-3819
Michael Muellen 04.02.2019 Last changed: High Priority Drafted Functional Required

**Figure 3**. View of the requirements specification with variance information within scone.



"R-2859: 12 Tons" is not part of any configuration
"R-3814: 23 Tons" is not part of any configuration

Development of a Truck Feature model is consistent
Development of a Truck Common features found
Development of a Truck Unique features found
Development of a Truck Dead features found
Development of a Truck Commonalities determined
Development of a Truck 3 possible configurations
Development of a Truck 4 atomic sets found

**Figure 4**. The requirements from figure 3, exported from scone and visualized with Glencoe. The feature model has been analyzed.

## 3. Technical realization

In order to ensure that collaboration on the requirements specification is as simple as possible, scone was implemented as a single-page web application. This section gives a brief overview of the architecture of the application.

In broad terms, the application can be divided into frontend and backend. Both of these parts are located on the server, but the code for the frontend is used for visualization and logic of the application in the user's browser, while the code for the backend handles the processing of data.

As shown in Figure 5, the backend uses a database, in which all persistent data required by the application is stored. The frontend is formed by a single-page application created with Angular, extended with an NgRx store for state management. The latter helps to improve the performance of the application while ensuring data consistency. The Angular application communicates with Django via the REST interface, which processes HTTP requests, or via Django Channels which, in concert with Redis, handles web socket connections used for real-time functionalities in scone.

The implemented real-time functionalities serve the purpose of keeping the requirements specification up to date for all members of a team, as changes are automatically propagated to them with an accompanying notification. This is especially helpful for remotely working teams.

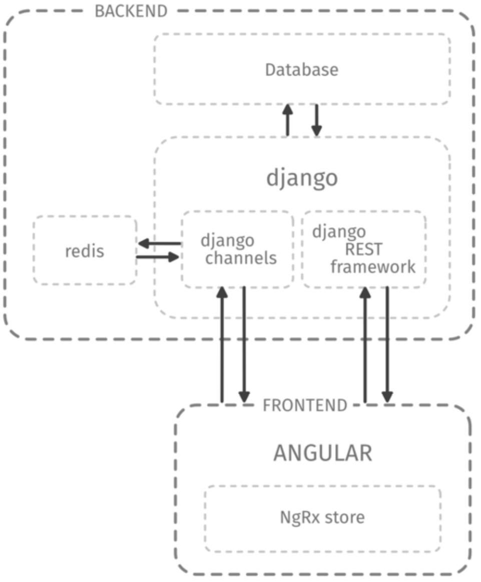


**Figure 5.** Server architecture of scone.

## 4. Validation of the implemented tool

At present, the tool is used in the context of lectures by students. This is the first step to evaluate the functionality according to the following questions:

- Is the functional scope of the application sufficient?
- How do users rate the usability?
- How long does it take users to become familiar with the application?
- Are the benefits of the application so great that users would prefer it to a text document or spreadsheet for small projects?
- Could the benefits of the application convince users who do not want to do requirements management because their project is so small?

As described above, scone provides only the most basic features needed to perform requirements management. It is now to be determined if the features at hand are sufficient to efficiently carry out requirements management and how they can be improved in any way.

Since the main goal was to develop an application that is easy to use, usability has to be determined in an acceptance test. As part of a university seminar, small groups of users who mostly have never applied a requirement engineering tool will use scone during a several weeks lasting project. After this period the users will evaluate the user experience and point out potential problems.

## 5. Related work

There are several product line applications available, including pure::variants [7], SPLOT [8] [9], and FaMa[10]. However, the study of Pereira et al. [2] shows that 51% of the reviewed applications are plugins for other applications and only 17% are online. Of those 17% only two, the DOORS extension [11] and Metadoc FM [12] provide support for the inclusion of requirements. Both tools are plugins for IBM Rational DOORS, with the DOORS extension being just a prototype. Neither of them is free to use [2].

## 6. Conclusion and future work

During the development of scone, the emphasis was placed on user-friendliness and simplicity of use. A basis was created to supplement the requirements specifications with variance information which can be analyzed and visualized with the help of Glencoe.

The result is an easy-to-use web application that allows both novice and professional users, regardless of the platform used, to create requirement specifications even in distributed teams. Due to the possibility to enrich the requirements with variance information and the integration with Glencoe for analysis, requirements management and product line management can be applied in conjunction. This can lead to an early error detection and a better understanding of the product to be created resulting in a more efficient and effective development process.

We argue in the paper that different disciplines must work together in the context of requirements and variant management. To our experience and to our best knowledge it is often the case, that variant management is (if at all) considered at the very end of a project. The reasons are in most cases obvious, but result often in a pure variability concept. The integration of the tools described in this paper gives the possibility to overcome this defect at a very early stage of development and with very little effort on the tool side.

We expect greater acceptance and a lower hurdle in the introduction of an integrated variant management methodology, especially among small and medium-sized companies.

However, the integration of scone and Glencoe has to be improved, since it is currently still necessary to first export the requirements specification from scone and then import it into Glencoe. We are currently working on a closer integration of both tools allowing for a seamless roundtrip from scone to Glencoe and back again without the need for manual intervention. Thus, it would be possible to fix inconsistencies directly in Glencoe with the changes automatically being reflected to the requirements specification in scone. This would realize a completely seamless workflow during the development process, leading to an easy to use analysis and error correction mode of operation.

Furthermore, it is currently not possible to import requirement specifications to scone. It is planned to realize the import of ReqIF [4] conform specifications in order to support interoperability with well-established requirements engineering applications. However, the possibility to import other common formats like CSV should be added too.

In order to further consolidate the user-friendly approach, more research is needed in this area, particularly with regard to requirements management, as most studies focus on methodology and techniques rather than on usability and user-interaction.

## References

[1] S. Allabar, Design and Implementation of a Single-Page-Web-Application for Requirements Management, Thesis, Hochschule Trier, 2017.

[2] J. A. Pereira, K. Costantino and E. Figueiredo, A Systematic Literature Review of Software Product Line Management Tools, In: I. Schaefer, I. Stamelos (eds.) *Software Reuse for Dynamic Systems in the Cloud and Beyond*, Springer International Publishing, Miami, FL, USA, 2014, pp. 73-89.

[3] G. Rock, C. Bettinger and A. Schmitt, Glencoe - A Tool and a Methodology to Manage Variability within the Product Development Process, *International Journal of Agile Systems and Management*, Vol. 12, No. 4, 2019, pp. 332-353.

[4] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak and A. S. Peterson, *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, Carnegie-Mellon University, 1990.

[5] A. Schmitt, C. Bettinger and G. Rock, Glencoe – A Tool for the Specification, Visualization and Formal Analysis of Product Lines. 25th international conference on Transdisciplinary Engineering, Modena, Italy, 2018. In: M. Perruzzini, M. Pellicciari, C. Bil, J. Stjepandic, N. Wognum (eds.): *Transdisciplinary Engineering Methods for Social Innovation of Industry 4.0: Advances in Transdisciplinary Engineering*, Volume 7, 2018, pp. 665-673.

[6] Object Management Group (OMG), *Requirements Interchange Format (ReqIF)*, 2016. Accessed: 09.03.2020. [Online]. Available: https://www.omg.org/spec/ReqIF.

[7] Pure-systems GmbH, *Variant Management with pure::variants*, White Paper, Magdeburg, 2006.

[8] M. Mendonca, M. Branco and D. Cowan, S.P.L.O.T. - Software Product Lines Online Tools, In: *OOPSLA*, Orlando, Florida, USA, 2009, pp. 761–762.

[9] M. Mendonca, M. Branco, D. Cowan, 2009, *S.P.L.O.T.*, Accessed: 21.01.2020. [Online]. Available: http://www.splot-research.org/

[10] D. Benavides, S. Segura, P. Trinidad and A. Ruiz-Cortés, Tooling a Framework for the Automated Analysis of Feature Models. In: *1st International Workshop on Variability Modelling of Software Intensive Systems (VaMoS)*, 2007, pp. 129-134.

[11] S. Buhne, K. Lauenroth and K. Pohl, Modelling Requirements Variability across Product Lines, *13th International Conference on Requirements Engineering (RE)*, 2005, pp. 41-50.

[12] A. K. Thurimella and D. Janzen, Metadoc Feature Modeler: A Plug-In for IBM Rational Doors. In: *International Software Product Line Coference (SPLC)*, 2011, pp. 313-322