Controlled Complexity for Future Mobility – Methodology, Guidelines and Tooling

Christian Reuter[1]

Abstract: The automotive industry is currently facing the most extensive changes since its invention. Connected, autonomous, shared, and electric crystallize as game changers whereupon new key player arise with expertise therein. Internet of things principles bring the vehicles online and so the product life cycle shortens. Although new functionalities should be rolled out quickly. This contradiction needs new methodologies, guidelines and tooling. In this paper, the current usage of combined textual and model-based requirement specification as well as variant management techniques at Daimler is presented in context to research outcomes. Furthermore, upcoming challenges in the field of handling complexity are described to give an example for ongoing investigations.

Keywords: Complexity, Variant Management, Systems Engineering.

1 Introduction

A bunch of innovations has led the automotive industry to a complex system architecture in modern vehicles. A Mercedes-Benz S-Class (W222) has about 150 systems with functions realized on 100 electronic control units with over 80 million lines of code [Sc15]. The degree of dependency is rising, since the number of sensors and actors nearly allows full automation up to autonomy. As known by modern smartphones, innovation drivers come by combining the persisting hardware with new software functionalities. These functionalities have to be implemented in development artifacts like requirements, models, test cases, and code. This principle is keeping the costs per piece down, but demands higher efforts in creating and integrating new concepts as well as validating the larger variability and mitigating malfunctions. Hereby, one of the biggest challenges is the short-term evaluation of changes by recognizing the development artifacts, which were affected.

In the next section, the current approaches for handling complexity are presented. In addition to the applied methodologies, guidelines, and tooling, also scientific analysis on these are provided.

Given that practical application falls short of academic research the pure form of approaches is not always one-to-one implementable into company processes. Therefore, dimensions of stakeholders of the development process and resulting realization tracks are described. Hence, appearing challenges, which need approaches to handle this

[1] Daimler AG, Research & Development, X426, Sindelfingen, 71059, christian.c.reuter@daimler.com

imperfection, are discussed in the third section. Finally, the presented approaches and challenges were summarized.

2 Foundations in Practical Handling of Complexity

Complexity pushes the need to establish a more high-level view on relations. Therefore, at Daimler Research & Development, for example, UML activity diagrams are used to structure functions and encapsulate them according to the IPO model [BVR17]. In addition, further diagrams are used to describe behavior (e.g. state machines, sequence diagrams) or structure.

One issue in splitting the development into different levels is keeping them consistent. A previous case study has shown that the combination of graphical models and a textual representation can match different needs of stakeholders. Although it should be considered, that the implementation follows an in advance with all participating partners agreed guideline [BRV18].

Regarding a software product line, changes can increase variability of development artifacts. Bauer et al. discussed "methods for the determination of change impacts" [Ba15]. The major problem in industrial application of these methods is a high effort to reach the initial conditions for using them. As well as the point of view, where based on a mainly product structure orientated approach the evaluation of efforts is getting more complex, since the derivation of development artifacts from the elements of a product structure needs further information.

Besides that, the concept of FODA (feature-oriented domain analysis) domain modelling [Ka90] is based on an easier implementable concept, of a tree of features. This furthermore allows extending details as of more feature levels over time. Aside from that, no differentiation of links is necessary. A link therein describes the relation of a development artifact to a feature, to such an extent as a development artifact relies on a feature.

At Daimler, this is used for a multi-level concept, which gives assistance to the developers in managing complexity. Here, all mentioned steps are based upon the same principle. The specification is called a 150%-specification, as it comprises different self-contained specifications with a common core and individual extents:

- On level one, complexity can be handled with explicit marks within a requirements specification or a graphical model, so that according to that mark, the document can be filtered and only relevant aspects appear. This concept is applicable for systems with a smaller complexity.
- On level two, a simplified form of feature-based variability management can be used, which is in daily use for systems with a medium complexity. Therefore, e.g. a requirements specification is linked to a feature tree within the same tool, so

every requirement is assigned to a feature. At the feature tree, there are features and their characteristics listed in a flat hierarchy and finally variants can be built up as a set of features.

- In an environment with high demands on managing complexity, on level three, the combination of a specialized variant management tool (e.g. pure::variants [PU17]) with interfaces to – for the specific company or department – most common development tools (e.g. DOORS [IB17], Enterprise Architect [SP17], Simulink [MA17]) can be used. In this case, the feature tree is independent of a single development tool. The mentioned specialized variant management tool supports a capable multi-level feature tree, which is furthermore extendable with in-depth logic-based constraints [Bo11].

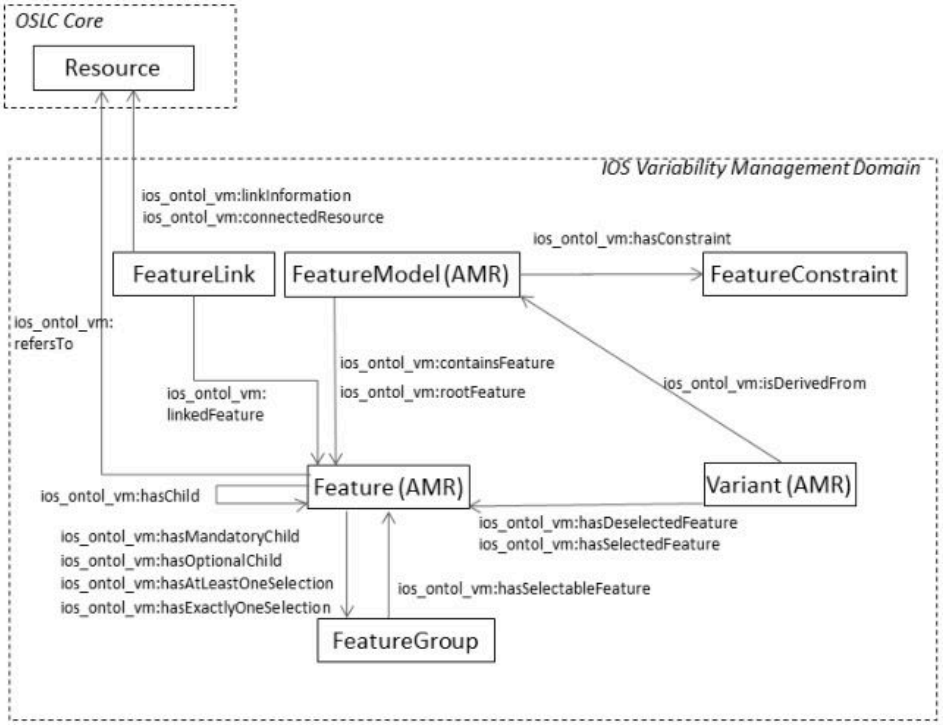

Fig. 1: Draft version of Variability Management Domain for OSLC [Re15]

Besides these in-tool, one-to-one tool, or one-to-many tools solutions, previous work has shown that a continuous systems engineering environment would bring the highest benefit to the users and IT operations. To reach this aim, an extension of the OSLC (Open Services for Lifecycle Collaboration) standard was proposed as part of the ARTEMIS EU project – CRYSTAL (CRitical sYSTem engineering AcceLeration). Part of this was to strengthen the concept of an Interoperability Specification and an

interrelated Reference Technology Platform [CR17]. As a contribution to the Interoperability Specification, the diagram in Fig. 1 pictures the objects “FeatureLink” and “Feature”, which can relate to other OSLC resources in order to attach variant information to a variety of development artifacts [OS17]. Several tool implementations (e.g. PTC Integrity [PI17], PTC Modeler [PM17], Enterprise Architect) were realized to demonstrate the capabilities of this concept [Re15].

In the area of alternative drive systems, an interview-based study of Beckmann et al. [BRV18] depicts that overlooking todays challenges the trade-off between the separation of concerns and the interlink of different levels and views like function orientation, component mapping, signal flow, coding of electronic control units and also safety and security topics is difficult to handle.

3 Upcoming Challenges for a Variety of Development Artifacts

The described methodology gives advice to the developer how variant handling can be pursued. However, in daily use, the comprehension for variant management is essential for developing a variant friendly architecture and therein comprised functions. This means that widely generic interfaces between the components as well as a clearly defined hierarchical decomposed structure inside the components is a prerequisite for the flexibility the industry is dependent on nowadays [CN02].

To illustrate the stakeholders of the development process resulting in different dimensions, which have to be handled, the following list gives assistance:

- Development Process (from system design till system validation with stakeholders like strategy, sales, user interaction)
- Additional Processes (change requests after contracting or issues in the supply chain)
- External regulations (legislation, state of the art e.g. documented by ISO or GB/T standards)
- Manufacturer/OEM-specific formalities (e.g. hardware/software architecture, communication matrix, security requirements)

These influences lead to a functional structure with a variety of requirements, models, software with related calibration, and more. For handling their complexity, two major tracks should be considered:

- The first track includes all information, which is well known and can be considered from the beginning of the development process. That applies also to manufacturer/OEM-specific formalities and mostly to external regulations.

- The second track of late changes is more difficult to handle. Depending on the scale of the change, this track starts earliest after contracting with Tier-1 suppliers and latest when the milestone of 100% functionality is reached. This track also applies to a software product line related development, when the newer software version relies on the previous one.

For the first track, the strategies mentioned in Section 2 are applicable, although the efficiency of exchanged variant data between the different development tools could be increased. Also, a comprehensive controlling of variability over the single levels from product over system to components and across the horizontal departments (like development, validation, after sales) would bring a better integration to fasten realization phases.

Quite more challenging are late changes. Here, usually time, cost, quality, and package are the most significant measurements. If market influences or misleading design decisions result in a late change, the appreciation of values needs more information. At first sight, it needs a clearly defined aim. This is the basis for alternating concepts. In turn, this leads to a scope, what has to be adapted (e.g. components, communication, documentation, testing). For getting a fast decision the more helpful it is for the involved departments the closer the scope is that they have to investigate for implementing the concept. Finally, the evaluated concepts need to be weighed, which concept fits better or is even one at least suitable.

4 Conclusion

The handling of complexity questions will be a key skill for developers as well as for companies. The foundations must be well known and day-to-day routine. Therefore, the knowledge base has to be broadened and common guidelines have to be agreed. Areas of high variability can be a starting point for that, but cross-sectional support is required. Because managing complexity cannot be reached by finding a local optimum, also the interaction of tools has to be pushed to prevent isolated applications. Furthermore, some challenges still persists, which need new methodologies. A short-term evaluation of a rating of change impacts and their extent with preferably low entry requirements rises the attractiveness of such an approach.

Bibliography

[Ba15] Bauer, W.; Bosch, P.; Chucholowski, N.; Elezi, F.; Maisenbacher, S.; Lindemann, U.; Maurer, M.: Complexity Costs Evaluation in Product Families by Incorporating Change Propagation. In: 9th Annual IEEE International Systems Conference, 2015.

[BVR17] Beckmann, M.; Vogelsang, A.; Reuter, C.: A Case Study on a Specification Approach using Activity Diagrams in Requirements Documents. In: International Requirements Engineering Conference (RE), 2017.

[BRV18] Beckmann, M.; Reuter, C.; Vogelsang, A.: Coexisting Graphical and Textual Representations of Requirements: Insights and a Guideline. In: International Working Conference on Requirements Engineering — Foundation for Software Quality (REFSQ), 2018.

[Bo11] Boutkova, E.: Experience with Variability Management in Requirement Specifications. In: 15th International Conference on Software Product Lines (SPLC), 2011.

[CN02] Clemens, P.; Northrop, L.: Software Product Lines – Practices and Patterns, Addison-Wesley, Boston, 2002.

[CR17] Crystal, http://www.crystal-artemis.eu/, accessed 21/12/2017.

[IB17] IBM Rational DOORS Family, https://www.ibm.com/us-en/marketplace/rational-doors, accessed: 21/12/2017.

[Ka90] Kang, K.; Cohen, S.; Hess, J.; Novak, W; Peterson, A.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. In: Technical Report CMU/SEI-90-TR-21 ESD-90-TR-222, 1990.

[MA17] MathWorks, Simulink – Simulation and Model-Based Design, https://www.mathworks.com/products/simulink.html, accessed: 21/12/2017.

[OS17] Specifications – Open Services for Lifecycle Collaboration, http://open-services.net/specifications/, accessed: 21/12/2017.

[PI17] Integrity | PTC, https://www.ptc.com/en/products/plm/plm-products/integrity, accessed: 21/12/2017.

[PM17] Integrity Modeler | PLM | PTC, https://www.ptc.com/en/products/plm/plm-products/integrity-modeler, accessed: 21/12/2017.

[PU17] pure-systems – The leading provider of software for product line and variant management tools | pure::variants, https://www.pure-systems.com/products/pure-variants-9.html, accessed: 21/12/2017.

[Re15] Reuter, C.: Variant Management as a Cross-Sectional Approach for a Continuous Systems Engineering Environment. In: Graz Symposium Virtual Vehicle, 2015.

[Sc15] Schneider, J.: Software-innovations as key driver for a Green, Connected and Autonomous mobility. In: ARTEMIS-IA/ITEA-Co-Summit, Berlin, 2015.

[SP17] Sparx Systems, Enterprise Architect – UML Design Tools and UML CASE Tools for software development, http://www.sparxsystems.com/products/ea/, accessed: 21/12/2017.