NordDesign 2014
August 27 – 29, 2014
Espoo, Finland / Melbourne, Australia

Feature models supporting trade-off decisions in early mechatronic systems design

Arno Kühn[1], Christian Bremer[1], Roman Dumitrescu[1], Jürgen Gausemeier[2]

[1]Fraunhofer IPT, Project Group Mechatronic Systems Design, Paderborn, Germany
{Arno.Kuehn;Christian.Bremer; Roman.Dumitrescu}@ipt.fraunhofer.de
[2]Heinz Nixdorf Institute, University of Paderborn, Paderborn, Germany
Juergen.Gausemeier@hni.upb.de

Abstract
Many development projects cause serious challenges to their developers: a rich set of new features has to be implemented within a restricted schedule and tight budget constraints. Often, this necessitates trade-offs: Certain features are incorporated into later releases to reduce risk of schedule delay. To support release planning, transparency regarding possible feature-trade-offs is needed: What exactly is the set of all planned features, what alternatives exist and how do they depend on each other? Especially in mechatronic systems design defining possible feature-trade-offs results in a big challenge. Due to its multidisciplinary character, dependencies from mechanics, electronics and software have to be considered. Within this contribution, we introduce feature-models as an appropriate approach to bring transparency into trade-off decisions. The model is integrated into a modelling technique for early mechatronic systems specification that supports the emerging approach of Model-Based Systems Engineering. By this, we show how feature-models enrich the engineering of complex mechatronic systems.

Keywords: Trade-off decisions, Feature modelling, Model-Based Systems Engineering, mechatronic systems

1 Introduction

Todays' technical systems are based on the close interaction of mechanics, electronics, control engineering and software engineering. These so called mechatronic systems, offer fascinating possibilities for the integration of a continuously increasing number of new features. This results in an increasing complexity of the technical system and its development process. Additionally, shortened time-to-market and increasing cost pressure cause a serious challenge to many development projects: a rich set of new features has to be implemented within restricted schedule and tight budget constraints [1, 2, 3].
Product managers and developers conquer this challenge by prioritizing features and negotiating schedule objectives and additional resources. They intuitively search for flexibility and decide on trade-offs between competing project objectives, namely between product performance, time-to-market as well as development- and product unit costs [4]. Engineering in releases is a promising approach adding flexibility to development projects and opening up possibilities for trade-offs [5], [6]. Figure 1 illustrates this approach.

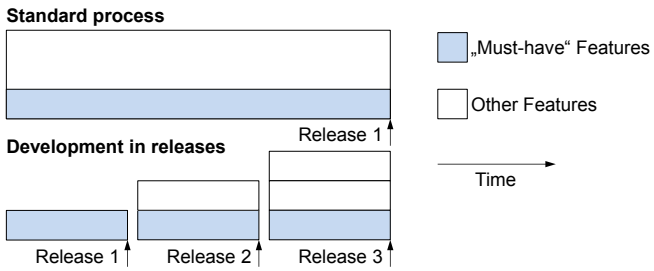

Figure 1 Development in releases [5]

Certain features are incorporated into later releases to reduce risk of schedule delay or to get along with scarce resources. This increases flexibility and allows for balancing project objectives. Which features are incorporated into which release, is decided during release planning. Here, features are prioritized by defining satisfactory trade-offs that accommodate competing stakeholder expectations [5].
To support release planning, early transparency regarding possible feature-trade-offs is needed: What exactly is the set of all planned features, what alternatives and options exist and how do they depend on each other? A basis for communication and cooperation is needed bringing together product- and project managers as well as developers. While typical requirement specifications do not provide the needed transparency, feature modelling offers a possible solution by capturing and visualizing optional and alternative features [7].
However, when deciding on a trade-off it is also necessary to consider dependencies and interactions between features. This is especially meaningful in mechatronic systems design. Due to its multidisciplinary character, dependencies and constraints from mechanics, hardware and software have to be analyzed. Therefore, it is necessary to analyze the system from a technical perspective. In this context, the emerging approaches in the field of Model-Based Systems Engineering (MBSE) seem to be promising. MBSE focuses on a model that encompasses different views of the system. The system model forms the basis for communication and cooperation in a multidisciplinary project environment and pursues the goal of controlling product complexity with the help of being transparent [8]. A key aspect of the system model is the system structure. It describes the system elements and their relationships. Mapping features to the system structure helps to reason on feature dependencies and thus supports possible trade-off decisions.
Within this contribution we will introduce an approach supporting trade-off decisions in the above mentioned context. In section 2, we will give a brief overview on the fundamentals of feature modelling. The concept of Model-Based Systems Engineering with focus on modelling the system structure is introduced in section 3. In section 4, we will show how feature modelling in combination with a system structure model supports feature-trade-off decisions. In section 5, we will give a conclusion and an outlook.

2 Feature modelling fundamentals

2.1 Definition of a “feature”

The definitions and discussions of the term feature are manifold. Features are interpreted according to the context. There are very general definitions like *“anything about the thing being designed that’s from interest”* [9] to context-sensitive definitions e.g. in the field of Computer Aided Design (CAD), where a feature represents the engineering meaning of the geometry of a part or assembly [10]. Within this paper, definitions are relevant that are used in conjunction with feature-modelling as an emerging approach within the field of software

engineering. Riebisch defines a feature as a representation of an aspect that is valuable to the customer [11]. Czarnecki and Eisenecker broaden this perspective. They define a feature as a property that is relevant to some stakeholders and is used to discriminate between concept instances [12]. In the case of technical systems, these can be structural (e.g. shape, size), behavioural (e.g. an operation mode) as well as functional properties (e.g. cruise control of a car) [9]. Properties are usually described by requirements. Therefore, features are also defined as abstractions or groupings of requirements that both customers and developers understand [13]. To conclude, for this research, features are defined as a set of requirements describing structural, behavioural or functional properties of a system that are relevant and understandable for different stakeholders. They are described by a single word or a short line of text [7].

2.2 Feature modelling

The feature concept is a common approach for managing variability. It is applied e.g. in variant management for technical systems [14] and software product line engineering [7]. It is used to distinguish between different product line members, each representing a possible configuration, differing in certain features from other configurations. For that purpose, conceptual relationships among features are usually expressed in feature models. These differ in their notation and content according to the intended use. Figure 2 illustrates two alternate representations.

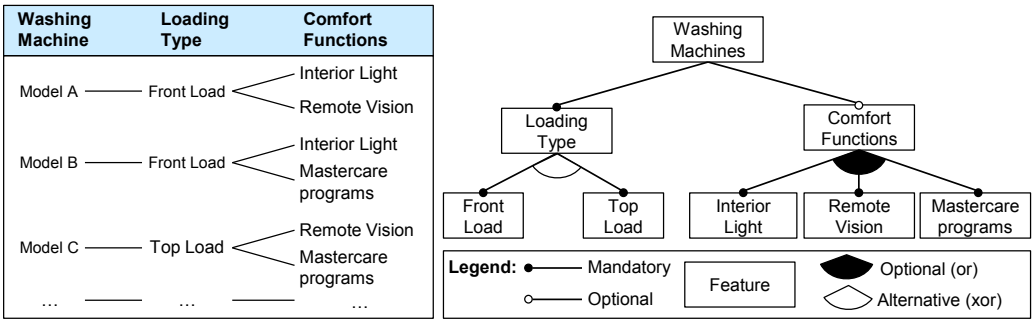

Figure 2 Different representations of feature configurations

The representation on the left emphasizes the entire number of variants that is planned according to possible combinations of the given features. These possible combinations can be derived from the representation on the right, which in a slightly different notation has first been proposed in the context of Feature-oriented Domain Analysis (FODA) [15]. Here, a tree-like notation shows the hierarchical organization and the relations of the features. A feature might be optional (depicted by a white dot) or mandatory (depicted by a black dot). Features can have a set of alternative sub-features, depicted by an arc. A white arc means that exactly one sub-feature from the set of alternatives can be chosen. A black arc is used to describe features that may be combined with the other sub-features of this junction. Due to its clarity, this representation is used in many different applications especially in software product development. These reach from strategic product planning over configuration management to release planning and project management [16].

Depending on the application, feature models differ in the aspects they emphasize. For example, they can refer to different objects reaching from a product line to a single component. Furthermore, the included features vary according to the stakeholders involved. In some cases, the focus is on customer-relevant features, while in other cases also internal features e.g. from production are considered.

2.3 Feature models as a basis for trade-off decisions

The concept of features is a promising approach when it comes to trade-off decisions in development projects, because a trade-off will be made between the features to be implemented. Trade-offs require a decision process, in which priorities are determined according to the project objectives. According to [17], priority can be seen as the relative right of a feature to the utilization of limited resources. Dispensable features can be delayed to later releases to save time, costs or any other limited resource. Alternative features can be implemented that may have lower performance but are achievable from a resource perspective.

Determining feature priority and agreeing on a particular trade-off should follow a systematic decision-making process. Within this process, two main steps have to be distinguished before making the actual decision [17]: 1) specification of the decision problem by defining alternative and optional features 2) determining priority by evaluating these features.

For step 1, feature models are an appropriate approach. They enable a transparent overview of the set of all planned features and their conceptual relations. In contrast to other requirement specification techniques (e.g. requirements lists, use-cases), they explicitly emphasize variability and point out optional and alternative features. Feature models are thus a good input for determining priorities in step 2.

In step 2, priority for each feature is determined taking into account different aspects like importance to the customer, effort for implementation, time and risk. By taking into account more than one aspect, high priority features from a customer's perspective may turn out to be less important if they are very expensive to satisfy. Typical methods for determining priority are Analytic Hierarchy Process (AHP), rankings or numerical assignments [18]. These methods deliver a prioritization list indicating which features should be included in the next release and which features may be delayed.

However, following this prioritization list is not always possible as dependencies between features restrict certain decisions. Such dependencies might force a less important feature to be implemented because it is required by a highly prioritized one. Therefore, it is necessary to analyze for dependencies before going ahead with determination of feature priority in step 2. Tracing features to system components helps to bring transparency into those dependencies and thus supports trade-off decisions. Traceability is one key element of the emerging approach of Model-Based Systems Engineering.

3 Model-Based Systems Engineering

The concept of Systems Engineering encompasses a holistic consideration of a system in order to strengthen the understanding of the system and to solve a complex development task efficiently. Model-Based Systems Engineering (MBSE) contributes to this idea. It aims at a holistic description and analysis of a system based on models, beginning in the early phases of the product development throughout the whole product lifecycle. This approach thus constitutes the evolution of the mechatronic systems design process that is changing from a document-based process to a model-based one [3, 8].

MBSE encompasses the formalized application of modelling for supporting requirements engineering, design, analysis, verification and validation. The focus of MBSE is a system model which allows a holistic perspective on the system by describing its requirements, structure and behavior in a domain-spanning way. The system model constitutes the basis for communication and cooperation in a multidisciplinary project environment; it helps to reason about a problem and pursues the goal of controlling product complexity by being transparent [8, 19]. Figure 3 illustrates two key elements of MBSE facilitating transparency.

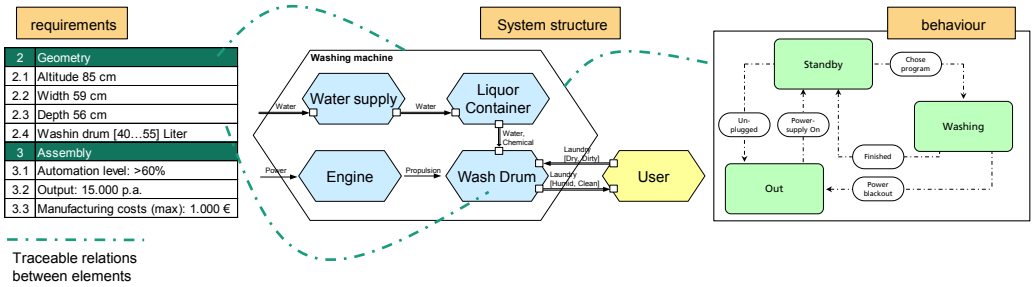

Figure 3 Diagrams and traceability links support transparency in system development

The first is, to gain transparency by describing the aspects of a system in suitable diagrams. A method (e.g. SysMod [20], CONSENS [19]) in combination with a modelling language (e.g. SysML [20]) define what aspects have to be considered and in what kind of diagrams they are described. Secondly, transparency is reached by associating related information objects via traceability links. For example, these links allow requirements to be connected to system elements satisfying those requirements. This enables the identification of change impacts, supports the understanding why certain system elements have been created and helps to ensure that all requirements are fulfilled. The system elements themselves may then be linked to the behavior they implement or the functions they realize.

A key aspect of nearly all MBSE-approaches is the system structure, which describes the elements and their relationships. Figure 4 illustrates an abstract system structure of a pump-system based on the method and modelling language CONSENS [19]. The pump-system serves as an example throughout this contribution.

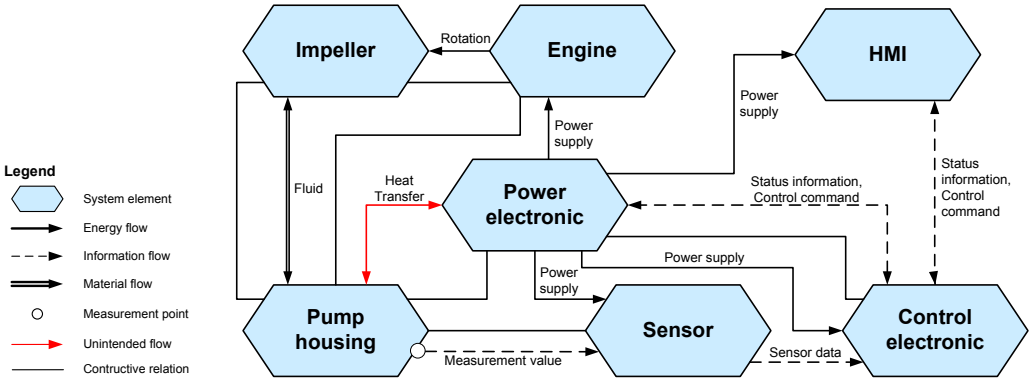

Figure 4 Elements of the system structure

The system structure contains all relevant system elements, no matter whether they are based on hardware, software or electronics. The dependencies and impacts between these system elements are described by flows derived from the known design methodology: energy, information and material flow (see [21]). In addition, a few more special relationships can be used, e.g. measurement point or a constructive relation. They have been implemented to meet the explicit needs of modelling mechatronic systems.

By means of the active structure the working principles and the technical interdependencies between elements of a system to be developed can be modeled and visualized. The diagram is the basis for defining interfaces between components or modules and serves as a common basis for reasoning and communication.

The system structure is helpful to overlook the system being developed holistically and to reason about technical issues. Thus, it is suitable to be considered in a trade-off decision process. In chapter 2, we indicated the link between a feature and its technical implementation to be of high importance. The MBSE approach implies the concept of mapping functions, requirements etc. to the system elements. Mapping features to systems elements may thus be a promising approach to support the understanding of the technical implications of a certain feature.

4 Feature-models supporting trade-off decisions

We pointed out, that a trade-off decision should stand on two pillars. First, one needs a basis, specifying the decisions that can be made. What are the possible trade-offs? What kind of alternatives exist and what are the consequences? To make a sound decision, the implementation of a certain feature into the system, must be considered. Doing so, the implications, challenges or influences induced by a certain feature can be overviewed, enabling a balanced decision in the second step. Here, priority for each feature is determined considering aspects like importance for the customer, effort for implementation, time and risk. We address these aspects by creating a feature model and linking it to the system's structure which is given by the MBSE approach. In the following, we present a method to create this entity. Figure 5 presents the process in four steps.

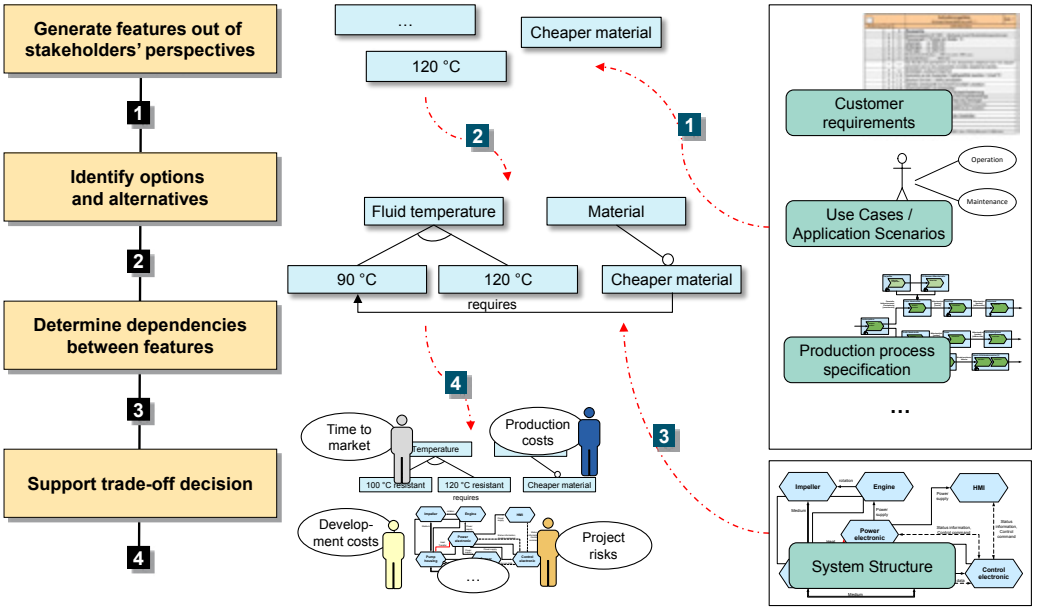

Figure 5 Process model for gaining a feature model to support trade-off decisions

Step 1: Generate features out of stakeholders' perspectives

First, features of the examined system must be captured. Usually, these features are already documented somewhere, e.g. in customer requirements lists or use-cases, and thus, just need to be extracted. In requirements lists, differentiations like "must-have", "should-have" and "nice-to-have" are already good indicators. "Should-haves" and "nice-to-haves" automatically induce a trade-off decision to be made. It is also helpful to take other stakeholders like the production-engineering into account and document their demands to the system. So, by scanning documents or carrying out interviews, features are gathered on which a decision (include/not include) can and must be made. To identify these features it is helpful to ask

questions like: “Is this requirement important for each application or does it aim at a special application or usage?”, “Does this required value actually need to be that high/low or can it be reduced for the typical application?” or “Are there alternatives to be considered?”. The engineer will then end up with a bunch of different features coming from several different stakeholders. In the example of the pump-system, a feature might be a new material for the impeller, which is promoted by the production engineering as it is cheaper but also uncertain in its technical behavior. Another feature might be a new requirement that demands to support applications with a fluid temperature up to 120°C.

Step 2: Identify options and alternatives

In the best case, each of the identified features can be implemented within time and budget. However, especially when it comes to risky and highly innovative projects, usually the contrary is the case. Time and budget are scarce and the technical solutions for new features are uncertain. Normally this is the point, when engineers start to negotiate on project objectives and search for optional features to be delayed and alternative features with lower risk to be implemented. Feature modelling systematically supports this process by providing systematic guidance and a transparent documentation. Optional and alternative features are searched by analyzing the features identified in step one. In this manner, a feature tree will be created that documents possible decisions to be made (see upper part of Figure 6).

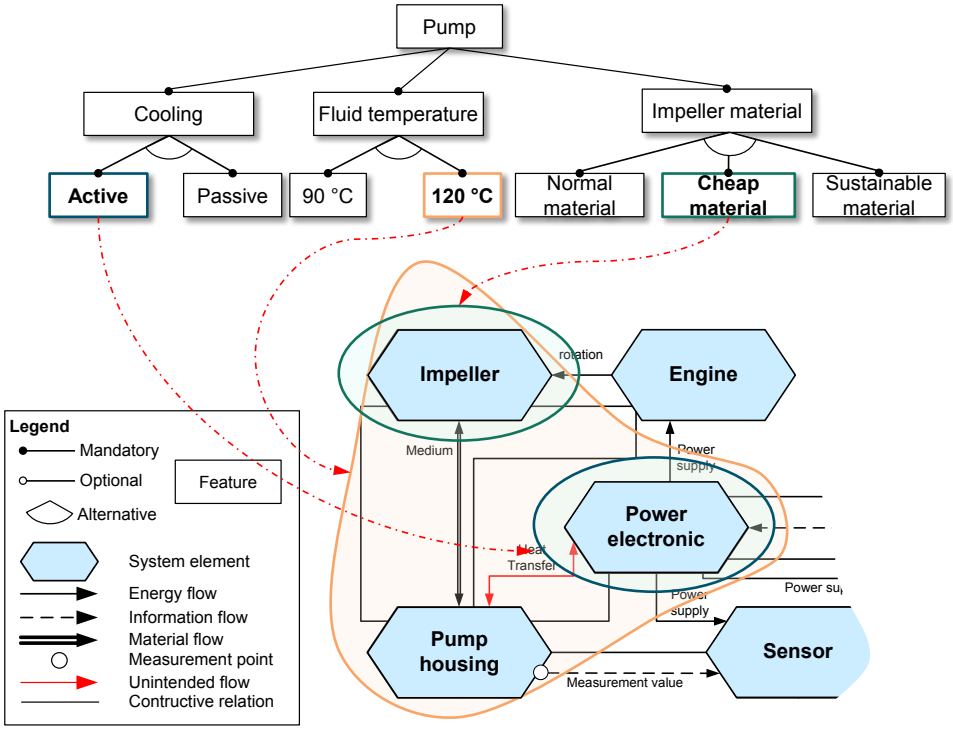

Figure 6 Mapping features to system structure

The graphical representation and modelling of a feature tree have already been presented in chapter 2. Good sources for alternatives are e.g. preceding products or competitor products. In the example of the pump-system, an alternative to the 120°C fluid temperature would be to implement 90°C as it has been the case in the preceding product. If there are features to which no alternatives or options can be found, the feature does not have to be investigated any further. Apparently, no decision can be made on this feature if there is no alternative.

Step 3: Determine dependencies between features
Until now, the features have been modeled in a way making them seem to be separated from the other features. However, the technical implementation of these features may cause significant interdependencies. Therefore, it is important to also take the technical integration of a feature in the system structure into account. This link is enabled by mapping the features to the system elements realizing them as shown in Figure 6. Hereto, the features are first linked to the system elements implementing them. Then, the interdependencies of these system elements to other system elements have to be analyzed.
To analyze the technical realization of a feature, the engineer should ask: "What system elements are involved in realizing a certain feature?", "To what other elements are they linked and what features are mapped to those elements?" and "Does this technical interconnection gain a (maybe indirect) relation between associated features?". Typically, "requires"- and "exclude"-connections are then used to model the identified interdependencies in the feature model. This is shown in Figure 7.

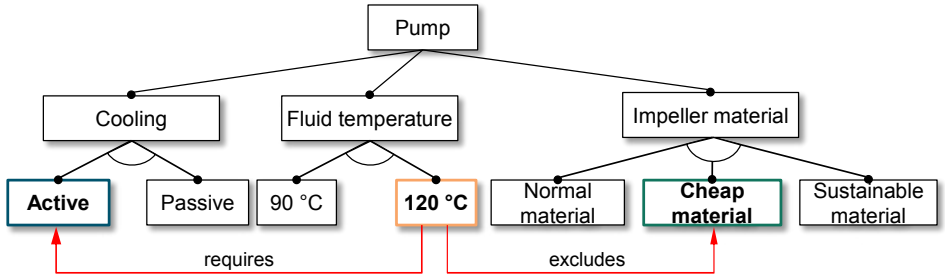

Figure 7 Feature-dependencies visualized within a feature-diagram

Coming to the example we applied, we first need to link the features to the system elements. This means, that e.g. the feature "120°C" is mapped to the pump housing. It will then have also effects to the power electronics because of the "Heat transfer" energy flow, modelled in the system structure. As the regarded temperature is also linked to the medium flowing through the pump, the influence on the impeller arises by itself. Secondly, other features, associated to the three just identified system elements have to be checked. Looking at the impeller, we notice the feature "cheap material" allocated to this system element. The engineer now has to determine whether the two features influence each other. It may be, that one feature is needed for the other. It may also be, that one feature excludes another – this is the case in our example. The cheaper material does not fit to the higher temperature. In addition, there is another feature linked to the power electronic. The higher temperature makes an active cooling necessary. Of course, it is also possible that some features have not been identified before and are figured out at this stage of modelling. The process of identifying features, modelling them and mapping them to the system structure is an iterative one. The interdependencies, that just have been identified give valuable input to the feature model. The gained information help reasoning about the trade-off decisions to be done.

Step 4: Support trade-off decision
For a sound trade-off decision, both feature model and system structure must be taken into account. While the first primary defines, on what a decision can be taken, the second allows for analyzing the technical implications and reasoning about the restrictions of feature-combinations. Together, both models support the trade-off decision. However, to make a sound decision on a certain trade-off, further aspects have to be taken into account. This

might be costs, customer importance or technical risks. These aspects can be specified by attributes within the feature model. However, the actual decision and the weight, each aspect brings into consideration, depend on the characteristic and the objectives of the project. In this paper we aimed at providing a transparent basis for this decision process.

5 Conclusion and outlook

The fascinating possibilities of mechatronic systems come along with an increasing complexity of their development projects. Project managers and engineers are forced to decide on trade-offs concerning systems capability, time-to-market as well as development- and product unit costs. Nowadays, these decisions are often made either intuitively, causing unsubstantial decisions, or too late, inducing significant change effort. Therefore, early transparency regarding possible feature-trade-offs is needed. Originating from software engineering, feature models are a feasible approach. By capturing and visualizing optional and alternative features and interdependencies between them, they offer an important contribution to make sound trade-off decisions. They help to specify the decision problem before deciding on a particular alternative. Within this contribution we demonstrated the integration of feature models within a state-of-the-art design process for technical systems, namely the MBSE-approach. We showed how feature models make a basis of decision-making. Mapping features to the system elements providing the implementation helps to reason about the technical dependencies, to understand possible impacts and to communicate which other decision makers involved. Based on these two pillars – feature model and linked system model – a sound trade-off can be made.

The method has been successfully applied within an industrial development project. The considered system was much more complex and in accordance with this, the created models were bigger. Still, the method was applicable. In this paper, we used a much simpler example for illustration purposes.

However, potential applications of feature models within the context of MBSE reach far beyond the field of trade-off decisions. Linking feature models to the system model describing the technical aspects of a system is especially promising when it comes to systems with high variability. Feature modelling allows describing commonalities and dependencies between different product concepts with only one model. By linking this information to the technical aspects of a system, for example, the variability of the system structure could be configured and visualized. This would allow for analysing and managing compatibility and inconsistencies of different product variants in an early stage of development. Furthermore, the technical impacts and challenges induced by a certain feature could be estimated. That way, the integration of feature modelling and MBSE would support product line planning of complex technical systems. However, further research still needs to be done especially when it comes to the integration of modelling-tools and the configuration of system structures that include hardware- and software elements in just one model.

Acknowledgement

This research and development project is funded by the German Federal Ministry of Education and Research (BMBF) within the Leading-Edge Cluster “Intelligent Technical Systems OstWestfalenLippe” (it's OWL) and managed by the Project Management Agency Karlsruhe (PTKA). The authors are responsible for the contents of this publication.

Citations and References

[1] Gausemeier, J., Frank, U., Donoth, J., Kahl, S., *"Specification Technique for the Description of Self-Optimizing Mechatronic Systems"*, Research in Engineering Design, Volume 20, No. 4, Springer, London, 2009.

[2] Fricke E., Schulz A.P., *"Design for Changeability (DfC): Principles To Enable Changes in Systems Throughout Their Entire Lifecycle"*, Systems Engineering, Vol. 8, No.4, John Wiley & Sons, Inc, New York, 2005.
[3] Gausemeier, J., Dumitrescu, R., Steffen, D., Czaja, A., Tschirner, C., Wiederkehr, O., *"Systems Engineering in der industriellen Praxis"*, Paderborn, 2013.
[4] Smith, P.G., Reinertsen, D., *"Developing Products in half the time"*, John Wiley & Sons, Inc., 2nd ed., New York, 1998.
[5] Smith P.G., *"Flexible Product Development- Building Agility for Changing Markets"*, Wiley & Sons, New York, 2007.
[6] Schuh, G., Eversheim, W., *"Release Engineering – An Approach to Control Rising System-Complexity"*, CIRP-Annals, Vol. 53/1, 2004.
[7] Thörn, C., Sandkuhl, K., "Feature Modelling: Managing Variability in Complex Systems", In: Tolk, A., Jain, L.C. (ed.) *Complex Systems in Knowledge-based Environments*, Springer-Verlag, Berlin, 129-161, 2009.
[8] International Council On Systems Engineering (INCOSE), *"Systems Engineering Vision 2020"*, INCOSE-TP-2004-004-02,Version/Revision: 2.03, September 2007.
[9] Brown, D.C., *"Functional, behavioral and structural Feature"*, ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago, Illinois USA, September 2-6, 2003.
[10] Shah, J.J., Mantyla, M., *"Parametric and Feature-Based CAD/CAM: Concepts, Techniques, and Applications",* Jon Wiley & Sons, Inc., 1995.
[11] Riebisch, M.: "Towards a More Precise Definition of Feature Models". In: Riebisch, M., Coplien, J.O., Streitferdt, D. (ed.) *Modelling Variability for Object-Oriented Product Lines*, Norderstedt, 2003.
[12] Czarnecki, K., Eisenecker, U., *"Generative Programming"*, Addison-Wesley, Reading, 2000.
[13] Ruhe, G., "Software Release Planning", in Chang, S.K. (Ed.): *Handbook of Software Engineering and Knowledge Engineering,* vol. 3, World Scientific Publishing, 2005.
[14] Schuh, G., *"Produktkomplexität managen"*, 2nd ed., Carl Hanser Verlag, München, 2005.
[15] Kang, K., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, S.A., *"Feature-Oriented Domain Analysis (FODA) - Feasibility Study"*, Technical Report CMU/SEI-90-TR-21, Carnegie-Mellon University, 1990.
[16] von der Maßen, T., Bacher, H.V., Dörr, J., Geisberger, E., Houdek, F., MacGregor, J., Müller, K., Paech, B., Singh, H., Wußmann, H., *"Einsatz von Featuren im Software-Entwicklungsprozess – Abschlussbericht des GI-Arbeitskreise "Features""*, RWTH Aachen, Aachener Informatik Berichte, 2005.
[17] Gilb, T., Maier, M.W., *"Managing Priorities: A key to systematic decision making"*, INCOSE, 2005.
[18] Berander, P., Andrews, A., "Requirements Prioritization", in Aurum, A., Wohlin, C. (Ed.) *Engineering and managing Software Requirements*, Springer-Verlag, Berlin, 2005.
[19] Kaiser, L., Dumitrescu, R., Holtmann, J., Meyer, M., *"Automatic verification of modelling rules in systems engineering for mechatronic systems"*, Proceedings of the ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 2013.
[20] Weilkiens, T., *"Systems Engineering with SysML/UML: Modelling, Analysis, Design"*, The MK/OMG Press, 2014.
[21] Pahl, G., Beitz, W., Feldhusen, J.,Grote, K.-H., *"Engineering design. A systematic approach. Third edition"*, Springer-Verlag, London, UK, 2007.