A Decision-making Process for Exploring Architectural Variants in Systems Engineering

Jérome Le Noir, Sébastien Madelénat, Christophe Labreuche, Olivier Constant, Grégory Gailliard, Mathieu Acher, Olivier Barais

A Decision-making Process for Exploring Architectural Variants in Systems Engineering

Jérome Le Noir,
Sébastien Madelénat,
Christophe Labreuche
Thales Research & Technology, France
firstname.name@thalesgroup.com

Olivier Constant
Thales Global Services, France
firstname.name@thalesgroup.com

Grégory Gailliard
Thales Communications & Security, France
firstname.name@thalesgroup.com

Mathieu Acher,
Olivier Barais
Université de Rennes 1, IRISA, INRIA, France
firstname.name@inria.fr

ABSTRACT

In systems engineering, practitioners shall explore numerous architectural alternatives until choosing the most adequate variant. The decision-making process is most of the time a manual, time-consuming, and error-prone activity. The exploration and justification of architectural solutions is ad-hoc and mainly consists in a series of tries and errors on the modeling assets. In this paper, we report on an industrial case study in which we apply variability modeling techniques to automate the assessment and comparison of several candidate architectures (variants). We first describe how we can use a model-based approach such as the Common Variability Language (CVL) to specify the architectural variability. We show that the selection of an architectural variant is a multi-criteria decision problem in which there are numerous interactions (veto, favor, complementary) between criteria.

We present a tooled process for exploring architectural variants integrating both CVL and the MYRIAD method for assessing and comparing variants based on an explicit preference model coming from the elicitation of stakeholders' concerns. This solution allows understanding differences among variants and their satisfactions with respect to criteria. Beyond variant selection automation improvement, this experiment results highlight that the approach improves rationality in the assessment and provides decision arguments when selecting the preferred variants.

Keywords: Systems engineering, Decision-making, Multi-criteria decision analysis, Design Exploration, Architecture, Model-driven engineering

1. INTRODUCTION AND MOTIVATION

In system and software engineering, the analysis of architectural variants is most of the times subjective and manual. The justification of a variant is seldom based on the pros and cons of the different options. Ideally, assessing or comparing several candidate architectures (variants) should be based on some decision criteria – corresponding to a Multi-Criteria Decision Aiding (MCDA) problem [31]. Among the classic hand-made "Decision Analysis Report" widely practiced in the industry the existing state-of-the-art industrial method and tools present several weaknesses.

In response, variability modeling allows system engineers to design and construct a set of candidate architectures. Then one aims at finding the design choices that best fit with the preference of the various stakeholders. This choice problem can be formulated as the maximization of a set of decision criteria. The difficulty is that the decision criteria are usually numerous and conflicting. One may indeed have performance criteria versus cost criteria which cannot be met both at the same time. Some tool-supported approaches exist to visualize and explore potential variants (e.g., through multi-objective optimization and Pareto front [17, 25]). Yet the practical difficulty remains manifold. In particular system engineers first need to identify and define a set of relevant metrics. The computation of metrics is complex and requires the derivation of architectural models so that specialized tools can be used to measure the criteria.

Second, the selection of one variant among several on the basis of a set of metrics is complex since there are commensurateness issues (combine "apples with oranges" as the metrics are given in different units), and one aims at making arbitrage between the metrics. Overall the selection of an architectural variant is a multi-criteria decision problem in which there are numerous interactions (veto, favor, complementary) between criteria.

The current practice in systems engineering is to perform a manual analysis. It is most of the time empiric and subjective ([8], [7], [1] and [2]). Moreover the justification of the choice of an alternative is very seldom based on the pros and cons of the different options. A central challenge in systems engineering is thus to provide decision support to explore variants and aid in the process of choosing the best architecture and create reports to justify some choices between several options.

The first contribution is a tool-based methodology for deriving, assessing and comparing several variants. We use the Common Variability Language (CVL) [30] and our early effort with reusable patterns [12] and custom derivation engine [15] to specify the variability in architectural models. We then integrate the CVL part to a MDCA off-the-shelf tools for evaluating and comparing variants. Our evaluation approach is based on the tooled method called Myriad [20]. This method is based on the design of a preference model [27] to make **explicit the evaluation strategies** (which are usually hidden), reach **objectivity in the analysis**, come up with a **recommendation** from a well-established methodology, and

the possibility to **justify the results**. This model is elicited from interviews with the stakeholders and expresses their preferences.

The second contribution of this paper, based on the preliminary work presented in [22], is to elaborate more on the practical difficulties faced by systems engineers when exploring architectural variants. We report on an industrial case study in which several variants have to be considered for a secure radio communication architecture.

The remainder of this paper is structured as follows: Section 2 presents an overview of our approach for exploring architectural variants. Section 3 reports an industrial practical experience based on advanced MCDA method and tool for comparing a set of a Software-Defined Radio (SDR) variants[1]. Related work is discussed in Section 4. Section 5 concludes the paper.

2. MODELING, DERIVING AND EXPLORING ARCHITECTURAL VARIANTS

In systems engineering, the difficulty to embrace the whole complexity of the concerns and the difficulty to manage their inter-relations have raised the interest for domain specific modeling techniques. System engineers usually employ domain-specific, model-based environment such as Capella[2]. Capella proposes a tooling approach structured on successive engineering phases which establishes clear separation between needs (operational need analysis and system need analysis) and solutions (logical and physical architectures), in accordance with **ARCADIA** method[3]. On top of such environment, one needs to provide domain-independent language for specifying and resolving variability, such as the *Common Variability Language (CVL)* [30].

Systems engineer have the basic tool-supported blocks for engineering model-based variants. Yet they still need an integrated solution for *deriving* and *assessing* such variants. Figure 1 gives an overview of our approach. It will be futher described, illustrated, and evaluate along this paper.

The approach involves the following:

1. At architecture principles level :

   architecture principles (key design choices) are captured in decision models (or feature models). Feature models can capture variability in terms of features or decisions. A realization model is used to map features/decisions to architectural elements (see left-hand top side of Figure 1).

   Prior to the assessment, system engineers are involved to elicit a preference model (see left-hand bottom side of Figure 1) in three stages according to the MYRIAD method to perform a multi-criteria decision analysis .

2. The principle is to derive an architectural variant based on a configuration conforms to the Architectural decicion model (see right-hand top side of Figure 1). A derivation engine automatically creates an architectural derived model (architecture alternative into a Capella descriptive model) based on a set of resolved choices (also called a configuration), a Capella base model and a catalog of reusable patterns.

3. At architecture variant level : Once architectural variants are derived, evaluation of architecture properties (with analysis or simulation) are computed. We connect them to the preference model to perform a multi-criteria Assessment of architecture alternatives (see right-hand bottom side of Figure 1).

We now describe each part.

2.1 Modeling Variability

Numerous approaches, being annotative, compositional or transformational, have been proposed to develop *model-based product lines (MSPL)* [11, 32, 30, 10, 15]. We use CVL and a series of variability tools (e.g., pure::variants[4], FAMILIAR [3]).

The overall principle of CVL is close to many MSPL approaches. First a *variability abstraction model* formally represents features/decisions and their constraints in a tree-based structure, and provides a high-level description of the product line (domain space). We will use the terminology *feature model* in the rest of the paper. Configurations, corresponding to combination of features, can be specified in line with the constraints of the feature models. Second, a mapping with a set of models is established and describes how to change or combine the models to realize specific features (solution space). We will use the terminology **variability realization model (VRM)** in the rest of the paper. An engineer can define in the VRM what elements of the base models are removed, added, substituted, modified (or a combination of these operations, see below). Third, realizations of the chosen features are then applied to the models to **derive** the final product model.

Using CVL, the selection or deselection of features will specify whether a condition of a model element, or a set of model elements, will change after the derivation process or not. In this way, these choices must be linked to the model elements, and the links must explicitly express what changes are going to be performed. The aforementioned links compose the VRM, determining what will be executed by the **derivation engine**. Therefore, these links contain their own meaning. We consider that these links can express three different types of semantics:

- **Existence**. It is the kind of Variation point (VP) in charge of expressing whether an object (*ObjectExistence* VP) or a link (*LinkExistence* variation point) exists or not in the derived model.
- **Substitution**. This kind of VP expresses a substitution of a model object by another (*ObjectSubstitution* variation point) or of a fragment of the model by another (*FragmentSubstitution*)
- **Value Assignment**. This type of VP expresses that a given value is assigned to a given slot in a base model element (*SlotAssignment VP*) or a given link is assigned to an object (*LinkAssignment VP*).

We use a "custom" CVL derivation engine for adding and specializing the semantics of variation points in the VRM [15]. In [12], we have introduced the EMF Diff/Merge Patterns[5]

[1]The research activities were conducted in the context of ITEA2-MERgE (Multi-Concerns Interactions System Engineering, ITEA2 11011), a European collaborative project with a focus on safety and security and the Clarity French research project

[2]https://www.polarsys.org/capella/

[3]https://www.polarsys.org/capella/arcadia.html

[4]http://www.pure-systems.com/

[5]http://wiki.eclipse.org/EMF_DiffMerge/Patterns

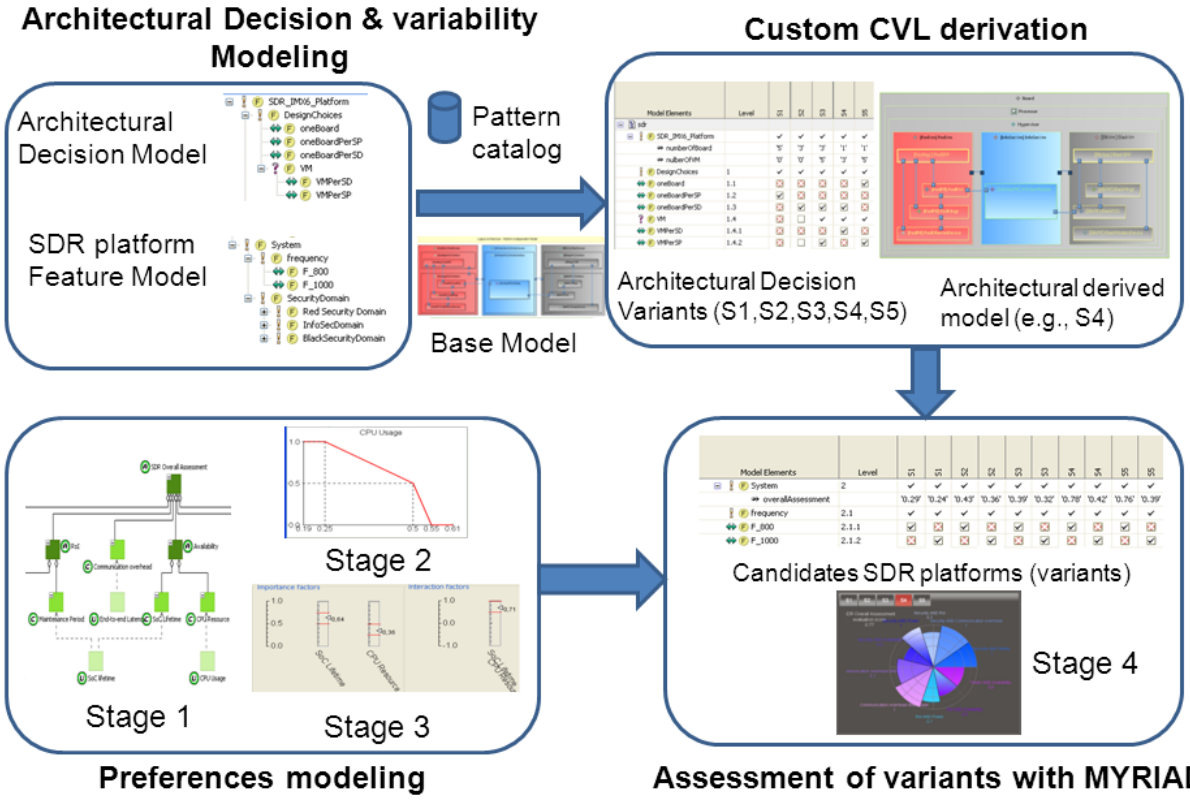

Figure 1: A tool-supported approach: modeling and derivation of architectural variants with CVL and assessment with MYRIAD

technology that provides conformant-by-construction deployment models thanks to the definition of a secure deployment pattern stored in our patterns catalog. We augmented CVL with new types of CVL variation points dedicated to the manipulation of patterns, and their instantiation on derived models. They can be seen as high-level constructs for weaving model components, with specific semantics based on engineers' expertise. Overall systems engineers can specify and derive model-based variants (see left-hand side of Figure 1).

2.2 Decision-making process for exploring architectural variants: MYRIAD overview

To explore architectural variants, we follow an approach based on a *preference model* to elicit the criteria and metrics. In this section we present the notion of preference model commonly used in *multi-criteria decision analysis (MCDA)* method. We then detail a tooled-process for exploring architectural variants named MYRIAD. The stages correspond to the bottom side of Figure 1 (see "Preferences modeling" and "Assessment of variants with MYRIAD"). Several concepts will be presented in this section; they will be illustrated in Section 3.4.

2.2.1 Basic concepts

The three basic concepts in a preference model are [27, 20]:

Metric "U" (where "U" originally stands for "Universe") Usually, a metric is a numerical quantity that enables to assess the level of achievement of one objective. We consider the word "metric" as an instrument which synthesizes in qualitative or quantitative terms, certain information which should lay the foundation for a judgment of an alternative relative to certain of its characteristics, attributes or effects (consequence) [27]. In the SDR use-case, examples of the metrics are the number of software partitions or the recurring costs (see Section 3.4.1).

Criterion "C" A criterion is a specification of the preference that an individual has on the values of a metric relatively to a concern. This specification amounts to construct a function – called **utility function** – which returns for each value of the metric the relative performance level (goodness) which positions it on a preference scale. The underlying scale is often a numerical scale, such as the $[0, 1]$ interval in which the value 0 is judged unacceptable relatively to the concern of the criterion, and value 1 is judged perfectly satisfactory relatively to the concern of the criterion. One can give an absolute judgment on an alternative according to a criterion. Examples of criteria are given in Section 3.4.2.

Aggregation "A" Procedure producing an evaluation (e.g. in the $[0, 1]$ interval) of any subject by taking into account, in a comprehensive way, the performance levels of the subject according to the criteria corresponding to a set of concerns. There are often nested aggregations. The hierarchical organization of criteria in nested aggregations groups criteria according to similar concerns.

2.2.2 Preference model building

The construction of a preference model is composed of the following stages [20] (see top and right-hand side of Figure 1):

Stage 1 : Structuring phase. The goal is to construct a tree representing a hierarchy of concerns in which the root represents the overall evaluation, and the leaves are the elementary metrics. The nodes in the tree are the metrics,

criteria and aggregations. All nodes except the leaves return a numerical evaluation that is a satisfaction degree. The numerical evaluation is encoded in the $[0, 1]$ interval where the boundaries have a special meaning. Value 0 corresponds to the total absence of the property beneath a criterion, and value 1 corresponds to the complete satisfaction of the criterion. In the SDR use-case, the hierarchy of criteria and aggregations is given in Section 3.4.1.

A good practice is to limit the children of an aggregation to 6. According to Miller's law, humans can manipulate mentally only seven plus or minus two items at a time [24]. This is even less when we are talking about criteria as criteria are complex concepts, which yields to figure 6.

Stage 2 : Criteria construction. It consists in quantifying the evaluation tree on the criteria nodes. In other words, we need to construct a judgment for each metric separately This amounts to ask "Is this value for this metric is good or bad?". This is quantified by a utility function. The construction of this function results from an interview with the domain expert. It is characterized by some thresholds that need to be identified. For the construction of the curve on intermediate values of satisfaction, we use the MACBETH approach which comes from measurement theory [5].

Stage 3 : Aggregation construction. It consists in quantifying the evaluation tree on each aggregation node. One needs to aggregate the partial evaluations to obtain higher level evaluations. Considering for instance an aggregation of three criteria, this amounts to know whether the satisfaction attached to an subject that is for instance good over the first criterion, fair on the second criterion and bad on the last criterion, is considered as rather good, or rather bad. It is likely that the overall satisfaction will equal some value in between. A trade-off or compromise shall be made among all the criteria used to compute the aggregation node.

The most usual aggregation function that is used is the weighted sum. Its main drawback is that it fails to represent real-life decision strategies including interaction among criteria. We use a model (called 2-additive Choquet integral) able to represent veto, favor, complementarity among criteria and so on [16]. At the end of this stage, the preference model is thoroughly elicited.

Stage 4 : Decision support. The preference model is applied on derived models (variants) that the stakeholders wish to assess/compare (see bottom and right-hand side of Figure 1). The stakeholders are interested not only in the evaluations, but also in the explanations of these scores. The justification of the choice of the best variant is very important for the decision makers [19]. Graphical and textual explanations are proposed [19, 20]. The evaluations and explanations can be applied to a single architecture or to a pair of solutions in order to compare them. Finally, the pros and cons of each architecture are provided.

2.2.3 Stage 2: Construction of the utility functions

The aim of utility functions is to map the metric spaces onto a common scale $[0, 1]$ representing the degree of satisfaction. Then the utility functions for different criteria are thus **commensurate**[6]. **Reference levels** are classically used to ensure this condition. We identify on each metric a value that corresponds to each reference level. We will use three reference levels:

- **Completely Satisfactory** the criteria is completely met. It is a saturation level in the sense that one cannot do better than this level in terms of satisfaction;
- **Budget (target value)** This is the expected value in the requirement provided by the customer;
- **Not satisfied at all** The criterion is not met at all for this value. This is also a saturation level as one cannot be worse than this level.

[6]It means that a same score – e.g. evaluation 0.3 – shall have the same interpretation whatever the criterion.

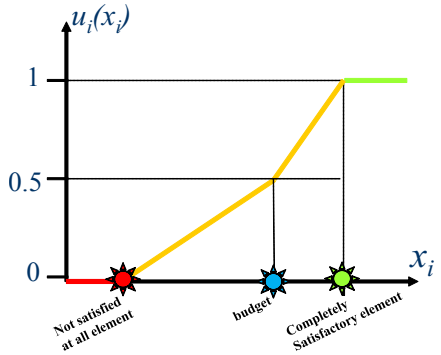

Figure 2: Shape of a basic utility function

The utilities of these three reference levels are 1, $\frac{1}{2}$ and 0 respectively (see Figure 2). We use a linear interpolation from these points to deduce the utility for other values of the metric. This is a standard practice in MCDA which allows to approximate any continuous function. Using a smoother function (like a spline) instead of linear interpolation would not improve the accuracy, as we just seek for an approximation of the decision maker preferences. In the SDR use-case, the hierarchy of criteria and aggregations is given in Section 3.4.2.

If the architect wishes to refine this base utility function, one can then use the MACBETH method, in which we can introduce other intermediate values on the metric and the MACBETH aims at identifying the utility of these new points that best fit with the architect preferences [5]. For space limitation, we do not describe this approach. However, its foundation will be used in the construction of aggregation.

2.2.4 Aggregation model

We consider an aggregation node. Its children are labeled $1, \ldots, n$. We denote by $u_1, \ldots, u_n$, and $H(u_1, \ldots, u_n)$ the evaluations on the n children, and the result of the aggregation respectively. The 2-additive Choquet integral is a good compromise between versatility of the model – and in particular the ability to represent *interaction* among criteria – and its simplicity and interpretability for the user [16]. It has the following expression:

$$H(u_1, \ldots, u_n) = \sum_{i=1}^{n} w_i \, u_i + \sum_{1 \le i < j \le n} \Big(w^{\wedge}_{i,j} \min(u_i, u_j) + w^{\vee}_{i,j} \max(u_i, u_j) \Big). \quad (1)$$

The first term with parameters w_i correspond to a weighted sum. The next terms model interaction among criteria: complementarity for parameter $w^{\wedge}_{i,j}$ [7], and redundancy for param-

[7]The complementary between two criteria is modeled by the minimum function. This basic strategy is fulfilled when both criteria are satisfied together.

eter $w^{\vee}_{i,j}$ [8]. A criterion is a veto if a bad evaluation on this criterion cannot be compensated by very good evaluations on the other criteria. Veto and favor can be represented in this model [16].

2.2.5 *Stage 3: Construction of the aggregation*

Following Stage 2, the "Completely Satisfactory" and "Not satisfied at all" elements have been identified for the metrics associated to the n children of the aggregation node under consideration. They are denoted by $\top_k$ and $\bot_k$ respectively, for node $k \in \{1, \ldots, n\}$. In order to learn the preferences of the expert regarding the aggregation node, we consider partial alternatives taking values only on the n children (the values on the other nodes are left unspecified). We define special partial alternatives (called prototypical alternatives) that can take only values in $\{\top_k, \bot_k\}$ for all children. These alternatives are [23]:

- $\top_\emptyset$ which takes the unsatisfactory value $\bot_k$ on the n children;
- $\top_i$ which takes the the satisfactory value $\top_i$ on child i and the unsatisfactory value $\bot_k$ on the other children;
- $\top_{i,j}$ which takes the the satisfactory value $\top_k$ on children i and j, and the unsatisfactory value $\bot_k$ on the other children.

If we consider a simple weighted sum model, the weight assigned to criterion k corresponds to the difference of satisfaction between the two alternatives $\top_k$ and $\top_\emptyset$. The presence of $\top_{i,j}$ involving two children allows us to express interaction between criteria.

The approach presented here is the extension of the MACBETH approach to the 2-additive Choquet integral [23]. We first ask the architect to rank order the previous alternatives (i.e. the elements of $E = \{\top_\emptyset, \top_1, \ldots, \top_n, \top_{1,2}, \ldots \top_{n-1,n}\}$). The architect is then asked to specify intensity of preferences among the pair of elements in E. This intensity is expressed in an ordinal scale in 0 (indifference), 1 (very weak preference), 2 (weak preference), 3 (moderate preference), 4 (strong preference), 5 (very strong preference) and 6 (extreme preference). For instance an intensity of 5 between $\top_{1,2}$ and $\top_3$ means that being completely satisfactory on children 1 and 2 (and unsatisfactory on the other children) is *strongly* preferred to being completely satisfactory on child 3 (and unsatisfactory on the other children). This approach is illustrated in Section 3.4.3.

3. INDUSTRIAL EXPERIENCE REPORT

The design of complex systems such as radio communication products requires taking into account various and sometimes contradictory concerns such as security and performance. Indeed, radio communication equipment exhibits strong requirements in terms of size, weight, power consumption, security and real-time performance. One of the most challenging aspects in systems engineering is to analyze the combination of numerous concerns; our case is an instance of this problem. We now describe how we instantiate the process of Figure 1 on the radio communication system.

3.1 Secure Radio Architecture

[8]The redundancy between two criteria is modeled by the maximum function. This elementary strategy is fulfilled when only one of the two criteria is satisfied.

A secure radio platform is basically divided into three parts: The **Red security domain** receives sensitive information from the user point of view (data plan) such as plain text data that need to be ciphered; The **Black security domain** deals with nonsensitive information that are ciphered for data information and may be ciphered or not for control information; An **Information security domain** (InfoSec) handles communications between Red and Black domains. It ciphers data information from Red to Black domain and deciphers them from Black to Red domain using cryptographic channels. Control information may go between Red and Black domains without ciphering using bypass channels. For strong security and safety needs, a physical separation is enforced for the Red, Black and InfoSec domains. Each domain is implemented by a dedicated board in the radio equipment and has its own independent processor. The introduction of multi-core processors, hypervisor and separation kernel technologies in embedded systems allows a new security/safety architecture with a logical separation between the Red, Black and InfoSec domains. Basically, each domain may be implemented on a single multi-core processor. Multiple processors may be replaced by a single multi-core processor at lower frequency. This reduces power consumption as it roughly grows linearly with the processor frequency and the number of processors.

A radio platform is the set of software and hardware layers that provide the services required by the Software Radio Protocol (SRP) application layer through Application Programming Interfaces (APIs). A radio platform includes system components: Radio Devices (RD) (e.g. Ethernet Device, Audio device) and others Services (e.g. management service, IP and routing service). The SRP application and Software-Defined Radio (SDR) platform components may be designed for different security/safety levels (e.g. Common Criteria (CC) for security and/or DO178 for safety).

Figure 3 presents the SRP application high level architecture.

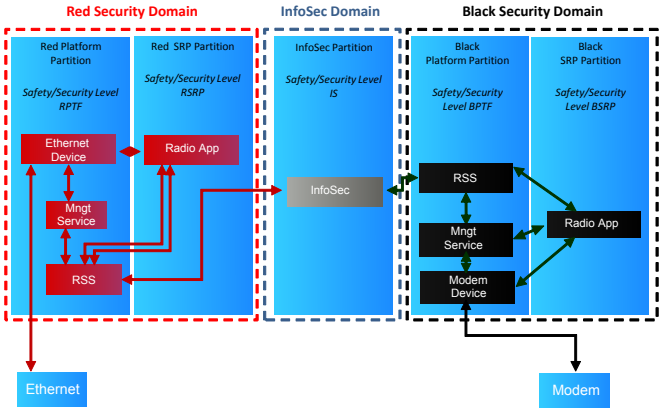

Figure 3: SRP application high-level architecture

In addition to the SRP application components (Red and Black Radio App), the use case architecture consists of the following SDR platform components:

- **the Ethernet Device** abstracts an Ethernet network interface, of the target SDR platform,
- **the Management Service** checks and dispatches control and management requests to SRP and platform components, For instance, it allows the configuration of

component properties such as the MAC address or the transmission power of the radio equipment,

- **the Radio Security Service** (RSS) provides security channels to cipher/decipher user information, and forward control information without encryption (bypass),
- **the Modem Device** abstracts the Physical layer implemented on DSP, FPGA and the Radio Frequency (RF) front-end(s).

This experiment focuses on modeling, deriving and exploring architectural variants on the SRP sub-system of the SDR. The exploration shall conclude to the selection the best design for a hand-held SDR.

3.2 Systems engineering environment

For designing the SDR System, system engineer used the following tools and methods:

Pure::variants is a tool for variant management that supports developers of SPLs throughout the entire product lifecycle. Its user interface is an extension to the Eclipse Integrated Development Environment (IDE). Pure::variants supports the definition of the problem and solution space with different types of models.

FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning) is a environment for composing, decomposing, configuring, and reasoning about (multiple) feature models [3].

KCVL is an implementation CVL, bundled as a set of Eclipse plugins and consists of a set of integrated components that support the different aspects of our approach. A textual editor, implemented with Xtext[9], that allows to express in a concise syntax the different parts of a CVL model: the feature model, the variability realization model, the modeling artifacts, and the configuration model; A derivation engine that accepts as input a valid CVL model and derive appropriate variants of the base models depending on the configuration choices expressed in the configuration model.

The **EMF Diff/Merge Patterns**[10] technology provides tool support for the creation, application, evolution and management of modeling patterns.

ARCADIA [11] is a model-based engineering method for systems, hardware and software architectural design. The **Capella**[12] modeling workbench is an Eclipse application implementing the ARCADIA method providing both a domain-specific modeling language and a tool-set which is dedicated to guidance.

In the SDR case study, pure::variants is used for feature/decision modeling. FAMILIAR is used for variability model reasoning. The pattern technology is used to define a secure deployment pattern. KCVL is used for variability realization model for specifying and resolving variability over Capella base model. Our evaluation approach is based on the tooled method called **Myriad** presented in Section 2.2.

3.3 Feature model and variants

3.3.1 *Feature model*

Numerous variability modeling approaches exist today to support domain (problem space) and application engineering (solution space) activities. Most are based on feature modeling (FM) or decision modeling (DM). In our case study we have used feature modeling for capturing the design choices and the relation between them (e.g., requires, exclude, etc.) as illustrated in Figure 4. Features refer here to high-level architectural decisions[13]. Each security/safety partition may be physically isolated in boards or logically isolated in virtual machines (VMs). As described above there are three security domains: Red, InfoSec and Black domains, and five security/safety partitions for trusted/safe Red platform components, untrusted/unsafe Red application components, trusted/safe InfoSec components, trusted/safe Black platform components, untrusted/unsafe Black application components.

For the case study, we designed two feature models for the SDR system: i) A first one is in charge of showing the variation point from the architectural point of view (Figure 4.a) We call it the *architectural decision model*. The second one is linked to the variation realization model. It defines all the variation points that exist on the system (Figure 4.b). We call it *system feature model*. The first feature model constrains the second one to keep only relevant configurations. We use FAMILIAR composition operators [3] to check the consistency between these two feature models.

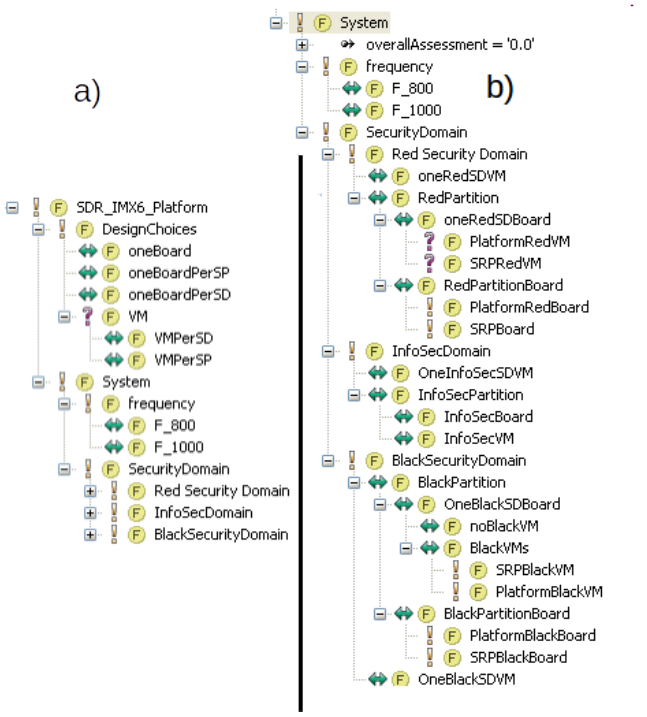

Figure 4: Feature models of the SDR platform

3.3.2 *Variant description model*

For deriving variants, we have to set the design choices (see Figure 5 and pure::variants matrix editor).

To this end, we define the different Variant Description Models (VDMs). The use of CVL as a pivot language for modeling the problem spaces enable connector with FAMIL-

[9]https://eclipse.org/Xtext/
[10]http://wiki.eclipse.org/EMF_DiffMerge/Patterns
[11]https://www.polarsys.org/capella/arcadia.html
[12]https://www.polarsys.org/capella/

[13]It should be noted that other low-level features, more related to architecture sizing and processor frequencies, are documented in other feature models. They are typically used to fine-tune the parameters once a given architecture has been chosen

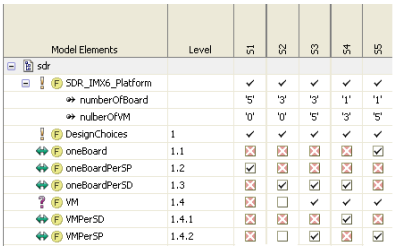

Figure 5: Architectural Decision Variants matrix of SDR platforms.

Id	Criteria	Not satisfying at all value	Budget value	Completely satisfactory value
SSe	SW part. for Security	0	3	3
HSe	HW part. for Security	1	3	3
SSa	SW part. for Safety	0	3	3
HSa	HW part. for Safety	1	3	3
CPU	Used CPU Res.	0.55	0.50	0.25
SocLT	SoC Lifetime (h)	50000	60000	90000
CO	Comm. overhead (us)	1500	1000	200
MP	Maintenance period (h)	90000	70000	60000
CpE	Cost/equipment (Euro)	1000	600	200
PC	Pow. cons. (mWh)	2000	1700	1000

Table 1: Evaluation criteria.

IAR that can be used to check the feature model consistency. In the SDR use case, the architect specifies constraints and limits the number of architectural variants to only five variants – called S1-S5 in Figure 5. Without these constraints the original feature model presented in Figure 4 allows one to derive 144 products. The feature model for our case is available online[14].

3.3.3 Base model and variability realization model

Features are mapped to architectural elements using a pattern technology on top of CVL. Specifically, a pattern consists of a set of elements and their interconnections, and a set of roles that associate some of the elements with OCL (Object Constraint Language[15]) constraints. Applying the pattern consists in binding all roles to existing model elements: the model elements and the pattern elements associated to the roles are merged while the remaining pattern elements are added in the model with their interconnections. We have equipped the EMF Diff/Merge patterns technology with variability support so that numerous architectural variants can be automatically derived out of a base model. In [12], we showed a concrete example showing how in the variability realization model, the different roles of the secure deployment pattern are bound to concrete elements of the base model.

3.4 Preference model

This section describes the preference model designed for the secure radio, using the MYRIAD method and tool.

3.4.1 Stage 1 : Structuring phase

This stage structures the problem and identifies a hierarchy of criteria. There are 10 criteria listed in Table 1.

We have built a tree representing a hierarchy of the concerns in which the root represents the overall evaluation, and the leaves are the identified metrics. The Criteria are structured into five Aggregations mapping the principal concerns of the architecture description. The resulting Aggregations are the following:

- **RoI** aggregated Criteria CO and MP.
- **Availability** aggregates Criteria CPU and SocLT.
- **Safety** aggregates Criteria SSa and HSa.
- **Security** aggregates Criteria SSe and HSe.
- **SDR Overall Assessment** (top node) aggregates the previous Aggregations together with Criteria CpE and PC.

[14]https://github.com/barais/splc16.git
[15]http://www.omg.org/spec/OCL/

3.4.2 Stage 2 : Criteria construction

Following Section 2.2.3, the utility function are thoroughly constructed once we have identified the reference levels "Not satisfying at all", "Budget" and "Completely satisfactory" of each criterion as summarized in table 1. The construction of this function results from an interview of different experts.

3.4.3 Stage 3 : Aggregation construction

We apply the approach described in Section 2.2.5 on Aggregation "Availability" which aggregates "CPU Resource" (child 1) and "SoC Lifetime" (child 2). The intensities of preferences over the elements of $E = \{\top_\emptyset, \top_1, \top_2, \top_{1,2}\}$ were provided by the SDR architect: $\top_1$ is similar to $\top_\emptyset$ (not fulfilling *SoC Lifetime* is unacceptable), $\top_2$ is *weakly* preferred to $\top_1$, and $\top_{1,2}$ is *strongly* preferred to $\top_2$.

MYRIAD computes the model from the intensity of preferences. In our specific case, the aggregation function can handle the following expression (in line with Equation 1, page): $0.29\, u_2 + 0.71 \min(u_1, u_2)$, where u_1 and u_2 are the utility of "CPU Resource" (child 1) and "SoC Lifetime" (child 2) respectively.

In the aggregation function, "SoC Lifetime" is very important – its mean importance is 64% (64% is the sum of coefficient 29% of u_2, and half of coefficient 0.71 shared between the two criteria). Moreover, "SoC Lifetime" is a *veto* (thanks to the indifference between $\top_1$ and $\top_\emptyset$): the aggregation value is zero when "SoC Lifetime" has a zero score, whatever the value on "CPU Resource" , which is clear from the expression of the aggregation. This arises from the fact that for the SDR architect, "SoC Lifetime" is a requirement whereas "CPU Resource" is more a soft constraint.

At the end of the process, a thoroughly specified preference model is obtained.

3.4.4 Binding architecture and evaluation models

A technical challenge in automated evaluation is collecting metrics among architecture models. The evaluation model is bound to the derived model elements involved in the metrics computation. In this specific case we collect metrics from architecture models defined in Capella[16] augmented with a subset of the MARTE[17] OMG Standard called non functional properties viewpoint implemented thanks to the Capella Studio environment.

3.5 Decision support

Assessment. Figure 6 shows the assessment of variants according to the preference model. The comparison criteria is the overall score interpreted as a utility function, the higher

[16]http://www.polarsys.org/capella
[17]http://www.omg.org/spec/MARTE/

is the better: a 0 score is interpreted as "Not satisfactory at all", a 0.5 score is interpreted as "Budget" and a 1 score is interpreted a "Completely satisfactory". The objective is then selecting solutions evaluated above 0.5 and, in case of multiple selection, choose the one with the highest mark.

Model Elements	Level	S1	S1	S2	S2	S3	S3	S4	S4	S5	S5
System	2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
overallAssessment		'0.29'	'0.24'	'0.43'	'0.36'	'0.39'	'0.32'	'0.78'	'0.42'	'0.76'	'0.39'
frequency	2.1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
F_800	2.1.1	☑	☒	☑	☒	☑	☒	☑	☒	☑	☒
F_1000	2.1.2	☒	☑	☒	☑	☒	☑	☒	☑	☒	☑

Figure 6: overall assessment.

At 1000 Mhz, The Myriad-based evaluation make us conclude there is **no satisfying solution**. Looking into details, Power consumption is "not satisfying at all" in all cases. Power is outside acceptable range while CPU Resource exceeds the expectations, lowering SoC frequency may improve Power evaluation while keeping CPU Resource satisfying. At 800 Mhz, the Myriad-based evaluation exclude all solutions other than S4 and S5. S4 and S5 variants are operationally acceptable.

Figure 7 shows the evaluation of the S4 variant. On Security and Safety the variant is very good on SW partitioning but very bad on HW partitioning. There is a complete substitutability between these criteria (indeed, SW Partitioning is completely sufficient for the SDR architect). The score on Availability is equal to the score on "SoC Lifetime". This follows from the strong complementarity between "SoC Lifetime" and "CPU Resource" – see Section 3.4.3. Figure 8 shows the details of the aggregation at the highest level. Each piece in the pie chart corresponds to a decision strategy (here the complementarity between two criteria), where the aperture of the piece corresponds to the weight of this strategy and the radius of the filled part corresponds to the evaluation of this strategy (i.e. the minimum between the two scores in a complementarity strategy). At the end, the ratio between the surface that is filled and the surface of the disk is equal to the overall evaluation of the variant. This provides a graphical explanation of the multi-criteria assessment.

Argumentation report. When deciding, systems engineers have to justify their choices. MYRIAD generates an argumentation report justifying evaluation and comparison towards the evaluation model. This argumentation helps decision maker in producing its justification report and producing a complete argumentation of the evaluation for each Aggregation.

3.6 Lessons learnt

Through this experiment, we demonstrate that we can combine MCDA techniques with model-based system engineering in order to automate the variants generation and their comparison.

Better confidence in the decision. The aim of decision support is to bring objectivity in the decision process and increase the confidence of the decision maker. One often argues that subjectivity is impossible to eliminate and it is not possible to model all subtleties of the experts in a utility model, so that the proposed decision may be subject to biaises. The main advantage of decision support is to engage an incremental discussion between the tool (which improves constantly the relevance of its model) and the architects (who enriches his value system). The architects enter some initial

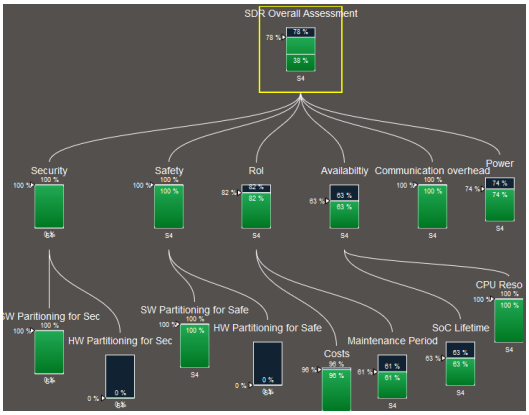

Figure 7: S4 variant assessment

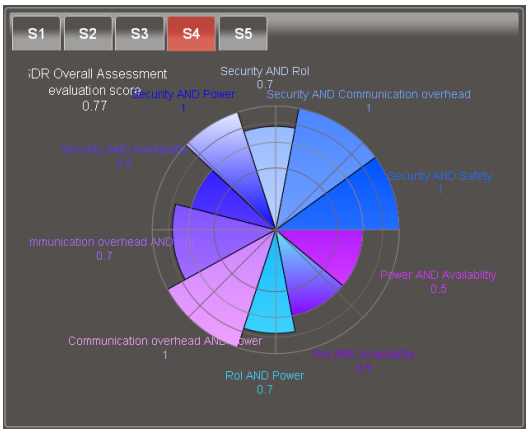

Figure 8: S4 variant assessment (800 Mhz)

preferences. If the outcomes are unexpected to him, then the decision support tool can pin-point to him some changes that should be done in the preference information he entered. This is very helpful as this increases the confidence the experts has in the preference model.

The ability of the preference model to capture real-life decision strategies is important here. We have initially tried to model the SDR use-case using a simple weighted sum for the aggregation function. At frequency $1000MHz$, $S3$ were recommended as the best variant and variants $S2, S3, S4, S5$ were assessed all satisfactory (with score over 0.5). The SDR architect were not satisfied with these results as he felt that $S4$ is the best variant and it is the only satisfactory variant. This is actually exactly what the model using a 2-additive Choquet integral returns, thanks to its ability to model complementarity between criteria (variants $S2, S3, S5$ are thus penalized, which was not the case with a weighted sum). This is the main asset of more elaborate preference model: we have seen that in most of use-cases, they return the correct ranking at the very first iteration.

The visual comparison provided by Myriad helps the system architect to justify its design choices. Overall the decision-making process brings practical benefits and improvements (e.g., evaluation strategies made explicit, objectivity in the analysis, and the possibility to justify the results). We now discuss the effort needed to realize the process as well as some limitations.

Effort. The design of the variability model takes one week: a variability realization model (two days) and patterns spec-

ification/reuse (three days). It requires a model-driven engineering expert to master pure::Variants, CVL and the pattern technologies. An architect was also involved to master software defined radio domain. When the variability model is designed, the criteria have been identified for structuring the preference model during two working sessions (two days) involving the architect of the SDR, a decision-maker and a decision-support facilitator. The architect has to bring the metric of the preference model to the derived architecture model in order to compare automatically the different architectural variants.

Current limitations. During this experimentation, we mainly see three main limitations for such kind of approaches.

i) The preference model is an experts' subjectivity synthesis built by a method attempting to keep maximum objectivity: utility and weights are computed by automated learning on experts' decisions. The Criteria has been structured into five Aggregations mapping the principal concerns of the architecture description. Such mapping is often questionable and requires experts' consensus. The presented tooled method requires the capability of sorting criteria, not possible when the criteria number is high and/or preference sorting is not possible.

ii) The concrete binding between the preferences model and the architectural variants is currently weaved manually at the derived model level. This is a clear limit of the described approach in terms of automation. We thought this binding has to be taken into account in the variability model.

iii) The presented tooled method still requires some works for improving the tooling integration. Indeed, in this methodology we combine six different tools (pure::variants, KCVL, FAMILIAR, EMF Diff/Merge Patterns, Capella and Myriad). Even if most of them are built on top of the Eclipse Modeling Framework and if we create some connectors for going from one to another, the system architect effort for mastering the resulting tooling environment is still too high.

4. RELATED WORK

The issues raised by safety assessment are apparent in many software-intensive systems [18, 26, 6]. In our context, the process of assessing and checking safety properties has to be applied to each (potential) solution (or variant). The variability substantially increases the complexity. In practice, the amount of possible variants is exponential to the number of alternatives. From an industrial perspective, some papers report on their experience in managing safety in a product line and systems engineering context [28, 9]. For example, Schulze et al. [28] demonstrated how a comprehensive model-based tool can manage safety-related artifacts with variability. Our industrial case study focuses on the problem of exploring architectural alternatives through an automated assessment and comparison.

In system engineering, stakeholders extensively rely on models as core part of their development process (see, e.g., [11, 32, 10, 15]). We rely on similar model-based approaches for specifying architectural alternatives and deriving variants. One challenging problem is to integrate systems engineering tools as part of the process [13]. We show how, in our context, model-based systems engineering tools can be epxloited to assess variants.

Another important problem is to assist systems engineers in exploring and comparing alternatives. Many tools and approaches have been developed [25, 14, 4, 21, 29], including in the context of multi-objective optimization (see, e.g., [17]). For example, ClaferMoo [25] is a tool to visualize potential variants. The principle is to depict a Pareto front based on quality attributes associated to features and set of objectives. Specifically, a Bubble graph is proposed to visualize 4 dimensions in terms of an horizontal axis, a vertical axis, a color and a bubble size.

These tools have many applications but are not directly applicable in our case. First, all quality attributes cannot be computed at feature model level. We rather need to use specialized tools to compute different metrics over a given variant (i.e., over a derived architectural model). Second, the composition of criteria is not straightforward (e.g., it is not a weighted sum of criteria). We have to define how criteria can be aggregated and what are their underlying interactions (e.g., veto, favor, complementary). Third, the number of criteria can be superior to 100 and thus hard to visualize/understand. For all these reasons, we first engineered a variability solution for integrating model-based tools as part of the derivation and assessment of variants. Second, we allow system engineers to define their preferences so that criteria can be composed and visualized.

5. CONCLUSION

In this paper we reported on the use of a tooled method for comparing evaluations of different architectural variants. We described the engineering of variability solution for deriving model-based variants with an extension of the Common Variability Language (CVL). We integrated systems engineering tools as part of the process to compute different metrics and eventually compare variants. We then described how experts can define their preferences and aggregate numerous criteria so that tools can eventually assist them in a multi-criteria decision analysis.

Key benefits are that systems engineers can now make explicit the evaluation strategies (which are usually hidden), reach objectivity in the analysis, come up with a recommendation from a well-established methodology, and the possibility to justify the results when choosing a variant. For achieving such benefits, the engineering of a variability-based solution is fundamental. It allows engineers to automate the derivation, assessment, and comparison of variants as well as the instrumentation of the decision-making process.

We believe the approach is generic enough, though some parts (e.g., the use of CVL and a derivation engine) can require some specializations. Further effort is thus needed to understand the applicability in other application domains.

6. REFERENCES

[1] Defense acquisition guidebook. https://dag.dau.mil/

[2] Manual for the operation of the joint capabilities integration and development system

[3] Acher, M., Collet, P., Lahire, P., France, R.B.: Familiar: A domain-specific language for large scale management of feature models. Science of Computer Programming (SCP) 78(6), 657–681 (2013)

[4] Al-Naeem, T., Gorton, I., Babar, M.A., Rabhi, F., Benatallah, B.: A quality-driven systematic approach for architecting distributed software applications. In: 27th International Conference on Software Engineering, 2005. ICSE 2005. pp. 244–253 (May 2005)

[5] Bana e Costa, C.A., De Corte, J., Vansnick, J.C.: MACBETH. International Journal of Information Technology and Decision Making 11, 359–387 (2012)

[6] Belategi, L., Sagardui, G., Etxeberria, L.: Variability management in embedded product line analysis. In:

Advances in System Testing and Validation Lifecycle (VALID), 2010 Second International Conference on. pp. 69–74. IEEE (2010)

[7] Biltgen, P., Ender, T., Cole, B., Mavris, D., Biltgen, P., Mavris, D., Molter, G.: Development of a collaborative capability-based tradeoff environment for complex system architectures. In: 44th AIAA Aerospace Sciences Meeting and Exhibit (2006)

[8] Biltgen, P., Mavris, D., Molter, G.: A methodology for technology evaluation and capability tradeoff for complex system architectures. In: International Council of the Aeronautical Sciences (ICAS) (2006)

[9] Braga, R.T.V., Trindade, Jr., O., Branco, K.R.L.J.C., Lee, J.: Incorporating certification in feature modelling of an unmanned aerial vehicle product line. In: Proceedings of the 16th International Software Product Line Conference - Volume 1. pp. 249–258. SPLC '12, ACM, New York, NY, USA (2012)

[10] Clarke, D., Helvensteijn, M., Schaefer, I.: Abstract delta modeling. In: Proceedings of the 9th GPCE'10 conference. pp. 13–22. GPCE '10, ACM, New York, NY, USA (2010)

[11] Czarnecki, K., Pietroszek, K.: Verifying feature-based model templates against well-formedness ocl constraints. In: GPCE'06. pp. 211–220. ACM (2006)

[12] Degueule, T., Filho, J.F., Barais, O., Acher, M., Noir, J.L., Madelénat, S., Gailliard, G., Burlot, G., Constant, O.: Tooling support for variability and architectural patterns in systems engineering (2015)

[13] Domis, D., Adler, R., Becker, M.: Integrating variability and safety analysis models using commercial uml-based tools. In: Proceedings of the 19th International Conference on Software Product Line, SPLC 2015, Nashville, TN, USA, July 20-24, 2015. pp. 225–234 (2015)

[14] Esfahani, N., Malek, S., Razavi, K.: Guidearch: Guiding the exploration of architectural solution space under uncertainty. In: Proceedings of the 2013 International Conference on Software Engineering. pp. 43–52. ICSE '13, IEEE Press, Piscataway, NJ, USA (2013)

[15] Filho, J.B.F., Barais, O., Acher, M., Le Noir, J., Legay, A., Baudry, B.: Generating counterexamples of model-based software product lines. International Journal on Software Tools for Technology Transfer pp. 1–16 (2014)

[16] Grabisch, M., Labreuche, C.: A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid. Annals of Operation Research 175, 247–286 (2010)

[17] Guo, J., Zulkoski, E., Olaechea, R., Rayside, D., Czarnecki, K., Apel, S., Atlee, J.M.: Scaling exact multi-objective combinatorial optimization by parallelization. In: 29th IEEE/ACM International Conference on Automated Software Engineering (ASE). ACM, ACM, Västerås, Sweden (2014)

[18] Kolb, R., John, I., Knodel, J., Muthig, D., Haury, U., Meier, G.: Experiences with product line development of embedded systems at testo ag. In: Software Product Line Conference, 2006 10th International. pp. 10–pp. IEEE (2006)

[19] Labreuche, C.: A general framework for explaining the results of a multi-attribute preference model. Artificial Intelligence 175, 1410–1448 (2011)

[20] Labreuche, C., Le Huédé, F.: Myriad: a tool suite for MCDA pp. 204–209 (September 7-9 2005)

[21] Loesch, F., Ploedereder, E.: Optimization of variability in software product lines. In: Software Product Line Conference, 2007. SPLC 2007. 11th International. pp. 151–162 (Sept 2007)

[22] Madélenat, S., Gailliard, G., Labreuche, C., LeNoir, J.: Comparing several candidate architectures (variants) : An Industrial Case Study. In: Embedded Real Time Software and System (ERTS'16) (2016)

[23] Mayag, B., Grabisch, M., Labreuche, C.: A representation of preferences by the Choquet integral with respect to a 2-additive capacity. Theory and Decision 71, 297–324 (2011)

[24] Miller, G.: The magical number seven, plus or minus two: Some limits on our capacity for processing information. The Psychological Review 63, 81–97 (1956)

[25] Murashkin, A., Antkiewicz, M., Rayside, D., Czarnecki, K.: Visualization and exploration of optimal variants in product line engineering. In: Software Product Line Conference. Tokyo, Japan (2013)

[26] Polzer, A., Kowalewski, S., Botterweck, G.: Applying software product line techniques in model-based embedded systems engineering. In: Model-Based Methodologies for Pervasive and Embedded Software, 2009. MOMPES'09. ICSE Workshop on. pp. 2–10. IEEE (2009)

[27] Roy, B.: Decision aiding today: what should we expect? (1999)

[28] Schulze, M., Mauersberger, J., Beuche, D.: Functional safety and variability: Can it be brought together? In: Proceedings of the 17th International Software Product Line Conference. pp. 236–243. SPLC '13, ACM, New York, NY, USA (2013)

[29] Siegmund, N., Rosenmüller, M., Kästner, C., Giarrusso, P.G., Apel, S., Kolesnikov, S.S.: Scalable prediction of non-functional properties in software product lines. In: Proceedings of the 15th International Software Product Line Conference (SPLC). pp. 160–169. IEEE Computer Society, Los Alamitos, CA (8 2011)

[30] Svendsen, A., Zhang, X., Lind-Tviberg, R., Fleurey, F., Haugen, Ø., Møller-Pedersen, B., Olsen, G.K.: Developing a software product line for train control: A case study of cvl. In: Bosch, J., Lee, J. (eds.) SPLC. Lecture Notes in Computer Science, vol. 6287, pp. 106–120. Springer (2010)

[31] Thurimella, A.K., Ramaswamy, S.: On adopting multi-criteria decision-making approaches for variability management in software product lines. In: Proceedings of the 16th International Software Product Line Conference - Volume 2. pp. 32–35. SPLC '12, ACM, New York, NY, USA (2012)

[32] Ziadi, T., Jézéquel, J.M.: Software product line engineering with the uml: Deriving products. In: Käkölä, T., Dueñas, J.C. (eds.) Software Product Lines, pp. 557–588. Springer (2006)