Generating Multi-objective Programs from Variant-rich EAST-ADL Product Line Architectures

Tobias Wägemann[1] and Albert Werner[2]

Abstract: The design of robust architectures for software-intensive systems in the automotive domain is a complex task and a considerable engineering challenge. Even conventional systems are usually subject to a multitude of conflicting design goals, such as unit cost and weight minimization, dependability augmentation and timing control. One feasible approach to resolve these conflicting levers on a system's architecture is to perform an optimization analysis on a concretely defined design space, which in general is vast. For the purpose of the approach presented here, this design space is represented by an automated identification of variation points relevant for design space exploration. The identified variability information is then transformed into a convenient mathematical representation for product-line-aware architecture optimization.

Keywords: Model-based Analysis, Multi-objective Programming, Architecture Optimization, Pareto Optimality, Product Line Engineering

1 Introduction

The field of automotive engineering has seen a steady increase in relevance of software-intensive electronic systems over the last decade. In the early stages of this development, software-based components were mainly used for replacing or complementing electromechanical functionality. More modern approaches have instead striven to explore entirely new application areas within the scope of the automotive domain. As a consequence, the complexity of employed system architectures has grown rapidly, including those architectures of safety-critical applications and controls.

The current challenge in this field of research is therefore not merely the exploration of additional use cases for software applications, but also the optimization of existing and future system architectures taking a multitude of different quality attributes, i.e. design objectives, into consideration [Wa13]. This challenge is further complicated by the fact that many real-world objectives conflict with each other, which necessitates a mechanism for trade-off analysis and resolution.

The resulting trade-off analysis is a hard combinatorial problem, which can often not be satisfactorily resolved by hand for all but the most trivial cases. In absence of adequate automated solutions, industry standard often relies solely on manual decision making when

[1] Technische Hochschule Nürnberg Georg Simon Ohm, Fakultät Informatik, Hohfederstr. 40, 90489 Nuremberg, Tobias.Waegemann@th-nuernberg.de
[2] Technische Hochschule Nürnberg Georg Simon Ohm, Fakultät Informatik, Hohfederstr. 40, 90489 Nuremberg, Werneral43493@th-nuernberg.de

it comes to finding optimal or near-optimal configurations for variant-rich architectures. Due to the complex interdependencies and decision ramifications in sufficiently large design spaces, it is reasonable to assume that a large number of currently employed system designs in the automotive domain are based on potentially suboptimal configuration decisions.

In order to establish a robust foundation for automated optimization approaches, a system's information model has to facilitate a high degree of expressiveness in regard to its variability management capabilities and its support of relevant quality attribute data. The utilization of a domain-specific ADL like EAST-ADL[3] allows for model-based analyses on an information model of ample expressiveness comprised in a single monolithic model structure [Bl13]. A comprehensively engineered EAST-ADL model can effectively provide all relevant information necessary for an optimization analysis; including the system's variability information and a bulk of data eligible for quality attribute assessment.

To make use of such information repositories for the purpose of architecture optimization, our goal is to devise an approach for deriving multi-objective programs from variant-rich EAST-ADL models. This can be realized by representing quality attribute data as objective functions and variability information as program constraints. We are also taking care in creating a heuristic variability interpretation process, which allows for product-line-aware optimization analyses, i.e. the possibility of family-based optimization of entire product lines in one sweep, as opposed to the optimization of individual products [Th12].

In this paper, we provide a summary of our ongoing work on the subject of generating multi-objective programs from variability information and quality attribute data of given EAST-ADL models and detail our progress in regard to different aspects of the intended implementation.

2 Variant-rich EAST-ADL System Architectures

The EAST-ADL is an architecture description language with an explicit focus on the specific requirements of architecture design and management in the automotive domain. The language was developed and refined in a number of international research projects, notably ITEA EAST-EEA (www.itea3.org/project/east-eea.html), ATESST and ATESST2 (www.atesst.org), and MAENAD (www.maenad.eu), and is currently being overseen by the EAST-ADL Association (www.east-adl.info). The language has been tailored towards compatibility with the well-established AUTOSAR standard (www.autosar.org), which in turn serves as an integral part of the EAST-ADL language by realizing one of its abstraction layers outright. The purpose of the EAST-ADL is to provide a standardized solution for creating exhaustive representations of software-intensive electronic systems in the automotive domain, by introducing additional layers of abstraction above the software architecture centric perspective of AUTOSAR. EAST-ADL models can further be enriched by language-inherent constraints for requirements and other supplementary information.

[3] *Electronics Architecture and Software Technology - Architecture Description Language*, an ADL specifically tailored to the requirements and characteristics of the automotive domain.

2.1 The EAST-ADL System Model

The EAST-ADL system model is segmented into four vertical abstraction layers, which can be augmented horizontally by an environment model and several language extensions (cf. Figure 1). Each layer has a specific purpose in the overall language concept and represents a different technical abstraction viewpoint, as well as a separation of concerns during the development process [EA13]. The EAST-ADL abstraction levels are as follows:

The *Vehicle Level* represents a top-level view of the system's characteristics and functions, as defined by stakeholder requirements, by means of cardinality-based feature modelling [CK05]. The focal point of this abstraction level is the technical feature model, which has a markedly technical perspective on the system's functional and non-functional composition. This technical feature model can be complemented by additional feature models, which realize different views on the composition of the system or any of its subsystems. The scope of these additional feature models can correspond to the requirements and practices of different departments or divisions, e.g. a specialized view for marketing purposes.

The *Analysis Level* implements an abstract functional description of the system as a functional analysis architecture (FAA), without exposing any implementation details. The FAA is comprised of analysis functions, which in conjunction represent the system's functional behavior in accordance with the specifications defined in the technical feature model. This kind of high-level functional decomposition allows for V&V tasks on a purely functional model, without interfering effects caused by implementation details.

The *Design Level* is divided into two parts: a functional design architecture (FDA) and a hardware design architecture (HDA). The FDA defines a concrete functional decomposition of the system's application software structure, with the intention of meeting certain non-functional constraints such as function-to-hardware allocation and reuse considerations. The FDA therefore comprises concrete design functions and and local device managers in a way that facilitates a derivation of AUTOSAR software components (SWCs), potentially using an n:m mapping. The HDA defines the system's hardware design by means of logical hardware functions, in order to establish a concrete representation of the physical device layout and circuitry, and to allow for allocation of design functions to modelled hardware entities.

The *Implementation Level* does not have an innate EAST-ADL implementation, instead the AUTOSAR standard is used for creating a suitable software architecture representation for the system. It should be noted that the EAST-ADL supports vertical traceability throughout all abstraction layers, including the Implementation Level realized by AUTOSAR.

The language extension components can be used to represent supplementary information, including requirements, variability information, timing and dependability constraints and generic system constraints, across all layers of an EAST-ADL architecture. An environment model, also called plant model, can be used to represent elements which influence and interact with the system architecture, but are out of scope of the system model itself, e.g. the near environment of the vehicle including other nearby vehicles.

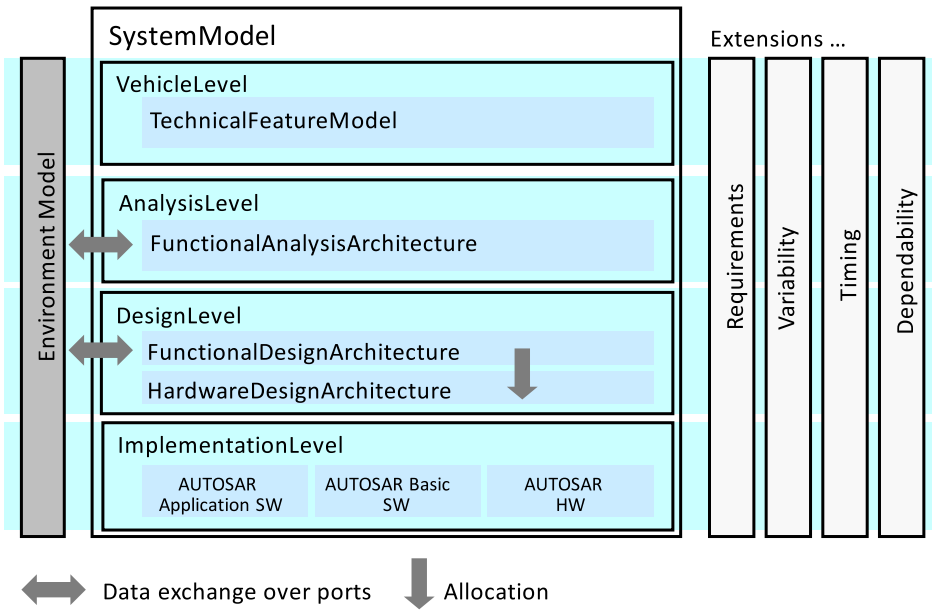

Fig. 1: The EAST-ADL System Model [Bl13, p.4]

2.2 Variability Language Concepts

Variability modelling in EAST-ADL can generally be divided into two different categories. The one category, variability on the Vehicle Level, is represented in the form of *cardinality-based feature models* (CBFM) [CK05]. Its purpose is to establish an abstract view of the system's overall variable content, including the interdependencies between variable elements. This view offers a distinction between a technical perspective and customer-specific perspectives, but does not disclose any particulars about actual implementation details. The other category, variability on the artifact levels, can be represented as part of the systems FAA, FDA, and HDA models by means of both CBFM and concrete variation points. This view is concerned with the actual technical implementation of variable content, by defining the relationships and interdependencies between variable architecture entities, i.e. actual functional components on the artifact levels of the system architecture.

In order to relate variability information of different feature models, as well as to allow for logical bindings from feature models to variation points on the artifact level, the language also supports configuration modelling by means of configuration links. These links contain atomized rules – so called configuration decisions – for the configuration of target model entities based on a given configuration of source model entities. Configuration links can be used to relay configuration information across EAST-ADL layers; but also in a more local context, for creating bindings of the internal variability of a container element with feature model representations of that internal variability – a so called *public feature*

model. The application of these variability concepts facilitates the use of *compositional variability management* (CVM) as a systematic language concept of the EAST-ADL language [RKW09, Re09].

2.3 Product Line Variability versus Architecture Design Space

In addition to the observance of different variability description methods described in section 2.2, a distinction between product-line-oriented variability [MP07] and the system's architecture design space, also called *architectural degrees of freedom* [Al13], is essential for realizing a product-line-aware architecture optimization approach. Table 1 gives a summary of the conceptual differences between these two kinds of variability.

Product Line Variability	Architecture Design Space
Purpose: satisfies specific needs of a particular market segment or mission (e.g different models of cars).	Purpose: represents different implementation variants (e.g. multiple suppliers for a subsystem).
Variations have functional influence on the target system.	Variations have influence on system characteristics like cost, performance or dependability.

Tab. 1: Characteristics of product line variability and architecture design space

The intended output of our proposed optimization approach is a product line with optimal architectural decisions, as opposed to an optimally configured product based on the product line. It is therefore necessary to establish a way to differentiate between the two types of variability, in order to be able to mark product-line-oriented variations as restricted for optimization. The approach presented in this paper achieves this distinction by tracing the propagation of individual variation points through the different layers of the given model.

In case of product line variability, a regarded variation point can always be traced all the way up to the features on the model's Vehicle Level (cf. section refsubsec:eadlSysModel), by means of configuration decision connections. The features represent product line variations and therefore allow for configuration of variation points located on underlying abstraction levels, i.e. all variability that ultimately is rooted on Vehicle Level is product line variability.

If no such traceable connectivity to the Vehicle Level exists, the variation point is instead considered to be part of the model's architecture design space. Such variation points are not configured as part of a product line configuration, but are instead decided at the system design time. Finding optimal allocations for these architecture variation point is the primary objective of the intended optimization approach.

3 Architecture Optimization Approach

In order to establish a robust foundation for optimization analyses of variant-rich EAST-ADL models, an expedient course of action is to first educe a rigorous mathematical model,

which reduces the complexity of the initial problem down to the fundamental drivers of the optimization process. This distillation procedure results in a lean representation of the optimization problem, which can be easily transformed into fitting input data for existing optimization applications.

In EAST-ADL models, however, the information relating to variability and design objectives is interwoven with other aspects of the given architecture description. There exists no separate depository for this kind of supplementary data; the identification and interpretation of the relevant information together with the transformation into a proper mathematical formulation is the purpose of the research project presented in this paper.

In order to accomplish this goal, we propose an approach for generating multi-objective programs from given EAST-ADL models, by representing the design objectives as a set of objective functions, and the system design space as a corresponding set of program constraints (cf. Figure 2). In order to allow for product-line-aware optimization, a number of specific considerations have to be made in regard to the examined variability information (cf. section 2.3).

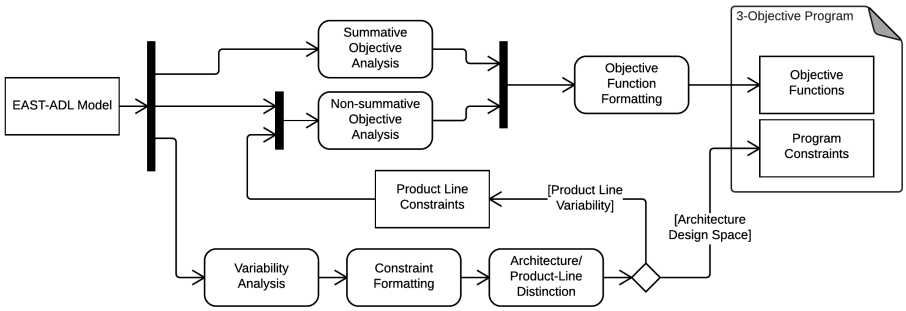

Fig. 2: Deriving a multi-objective program with both summative and non-summative design objectives

Our efforts to devise a comprehensive approach are part of an ongoing work in progress; our current, preliminary method still has certain restrictions in terms of the kind of variability and the kind of design objectives that can be handled and successfully converted. These restrictions are (a) the limitation to strictly boolean variability spaces, as opposed to non-boolean aspects introduced in particular by parametrization and cloned features, and (b) the limitation to strictly summative design objectives, like unit cost and weight minimization. The formal effect of these limitations on the generated output is that the resulting program is both linear and exclusively disjunctive. Future iterations, assuming that the current limitations can be successfully lifted, may instead produce multi-objective non-linear mixed-integer programs (cf. section 5).

3.1 Multi-objective Optimization

When considering multiple objectives as part of an optimization analysis, it is usually impossible to find a single perfect solution which is truly optimal with respect to all

objectives. This complication is caused by the often conflicting nature of typical design objectives, e.g. improving safety will decrease performance, and vice versa. One possible solution for this dilemma is to convert the multi-objective analysis into a single-objective analysis, by formulating an entirely new objective function from a weighted aggregation of the original objectives – a process known as scalarization or weighted normalization [GR06]. This approach, while widely used, has significant drawbacks, in particular the fact that the weighting factors for the objectives have to be defined beforehand, i.e. before there is tangible knowledge about the properties of possible optimal solutions.

An alternative and more exhaustive approach involves the simultaneous evaluation of all distinct objective functions in a single optimization sweep. This approach is usually regarded as genuine multi-objective optimization, also called *Pareto* optimization due to the particular composition of the analysis result. The typical outcome of a Pareto optimization is a Pareto set, also called Pareto frontier, which constitutes the set of all Pareto-optimal solutions of the optimization analysis [MA12b, p.54f].

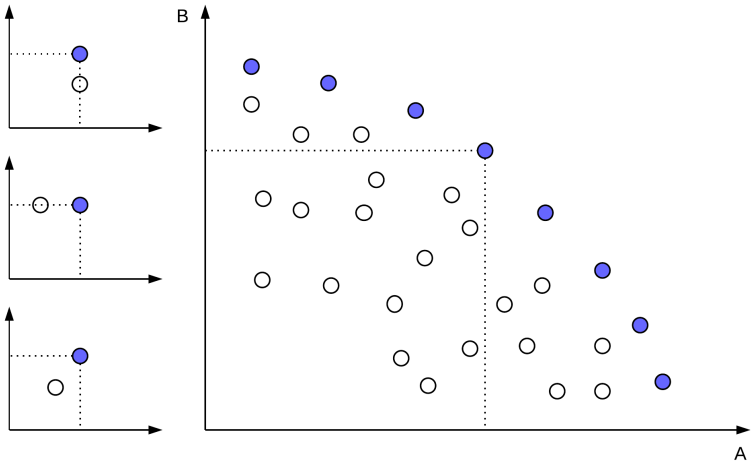

Fig. 3: Depiction of Pareto dominance for an exemplary solution space

Pareto optimality is based on the concept of dominance. A solution is called dominant if and only if no other solutions exist, which outperform it with respect to at least one objective without degradation in any other objective [BK05, p.414ff]. Figure 3 depicts a solution space for an optimization analysis with two conflicting design objectives A and B plotted against the axes. In this example, both objective functions were to be maximized – think safety and performance. The Pareto set is represented by the filled bullets in the main graph. The smaller graphs depict different scenarios of solution dominance.

3.2 Architecture Representation as Multi-objective Program

In order to attain a suitable mathematical representation of a given EAST-ADL optimization task, we propose an approach which makes use the formulations of multi-objective programming [Be71][MA04]. Our current exclusion of non-boolean variabilities and non-

summative design objectives, as noted in section 2, results in a program in which the constraints and objective functions are (a) linear, and (b) strictly binary. The application of such programs is called 0-1 integer linear programming, or *disjunctive programming* [Ba79]. Such a program can be formulated as follows:

$$\begin{aligned} \text{Minimize} \quad & Cx \\ \text{subject to} \quad & Ax \geq a_0 \\ & x \in \{0,1\} \end{aligned} \tag{1}$$

where C is a (m,n)-Matrix representing a mapping of objective coefficients; A and $a0$ are a (p,n)-Matrix and a p-vector, respectively, representing the mapping of the variability space to constraint coefficients; and x is an n-vector of strictly 0-1 binary architecture decisions.

The Matrix Cx is in principle just an alternative representation of a set of linear functions $F(x) = (f_1(x), f_2(x), ..., f_m(x))^T$, where each function represents one distinct design objective, and thus one of the target parameters in the Pareto trade-off analysis. The coefficients can be parsed directly from extension constraints affiliated to the respective variable architecture elements in the given EAST-ADL model (cf. section 2.2). The methodology of ascertaining the program constraints $Ax \geq a0$ from the model's architecture design space is highlighted in section 3.3.

In order to illustrate the composition of such a program, suppose an exemplary optimization task on a system with two variable elements X_1 and X_2, which are mandatory alternatives, i.e. exactly one of the elements can and must exist in each valid system configuration. Further suppose that the relevant design objectives for this optimization task be unit cost and weight, and that the elements have the properties $X_1(weight : 150, cost : 20)$ and $X_2(weight : 100, cost : 30)$. This supposition results in the following 2-objective program:

$$\begin{aligned} \text{Minimize} \quad & 150x_1 + 100x_2 \\ \text{Minimize} \quad & 20x_1 + 30x_2 \\ \text{s.t.} \quad & x_1 + x_2 = 1 \\ & x_1, x_2 \in \{0,1\} \end{aligned} \tag{2}$$

where each of the objective functions aims to minimize the regarded design objective unit cost and weight, respectively, and the program constraints ensure that one and only one of the elements exists once and only once in every solution.

Each of the Pareto-optimal solutions to this mathematical program represents a corresponding Pareto-optimal configuration of the initial system. It should be noted that, while the scale of constraints is rather handy in this small example, the size and complexity of program constraints grows rapidly with increasing complexity of a regarded system's variability space and might be problematic when considering very large systems. This is one of the so far unresolved challenges for future work on this project (cf. section 5).

3.3 Generation of Program Constraints from System Design Space

The basis for our optimization space definition is a variant-rich EAST-ADL model in the standardized exchange format EAXML, which is specified by the EAST-ADL Association by means of an XML schema[4]. The hierarchical XML structure of the input model can be transformed into a DOM tree in order to apply recursive search and pattern recognition operations.

The format of the generated optimization constraints is conform to a set of program constraints for a disjunctive program. Taken together, these constraints are a representation of all valid configurations of the optimization space.

Constraints regarding features connected by configuration decisions follow the VSL notation [Re09]. Unlike artifacts, feature names are not identified by their fully qualified names but with a combination of the feature model's name and the feature's name separated by the hash symbol '#'. If the feature has a parent feature, the resulting path is arranged as follows: Featuremodel#Parentfeature.Childfeature (cf. Figure 4). Additionally, the VSL language allows for several alternative notations[5]. To get a consistent format, all VSL-based feature names are replaced by their fully qualified names and notations are replaced by a uniform version.

Next, the fully qualified names of variation points are substituted with simplified versions. These substitutions have the advantage of containing no special characters (e.g. '/'), which might otherwise cause problems during execution of the optimization procedure. In order to be able to re-obtain already placed substitutions, an assignment table is generated as part of this procedure. This step is necessary to avoid creating duplicates with differing name substitutions.

After these preparations have been finished, the variability information can be parsed into program constraints. Table 2 gives a summary of transformation rules for different kinds of variability modelling concepts. Applying these rules on the example in Figure 4, the variability information is first transformed into propositional logic.

In the subsequent example (cf. Figure 4), the features are depicted with their fully qualified names, which are later replaced by abbreviations.

Propositional logic for single features:

BS			Braking System is always present
b	$\rightarrow$	BS	when basic, then also Braking System
a	$\rightarrow$	BS	when advanced then also Braking System

Propositional logic for the feature group:

BS	$\rightarrow$	$(a\|b)$	when Braking System, then advanced or basic or both
BS	$\rightarrow$	$(!b\|!a)$	when Braking System, then not advanced and basic together

Propositional logic for the feature link:

a	$\rightarrow$	ABS	when advanced then ABS

[4] http://www.east-adl.info/Specification.html

[5] Different variations of operators (e.g. and, &, $\wedge$)

Next, the propositional logic is transformed into program constraints according to the second set of rules located in the third column of Table 2:

$$\begin{array}{lcl} BS & = & 1 \\ BS-b & \geq & 0 \\ BS-a & \geq & 0 \\ a+b-BS & \geq & 0 \\ -a-b-BS & \geq & -2 \\ ABS-a & \geq & 0 \end{array}$$

The set of these constraints, together with the elements' co-domain $BS,a,b,ABS \in \{0,1\}$, constitutes the set of all valid system configurations for this exemplary system.

Variability Method		Propositional Logic	Disjunctive Integer Program
Feature Tree	Feature has parent	$f \rightarrow f_{parent}$	$f_{parent} - f \geq 0$
	Feature is mandatory	f	$f = 1$
	Feature is excluded	$!f$	$(1-f) = 1$
	Feature Group	$\bigwedge_m \left(f_{parent} \rightarrow M_m(f_1,..,f_n)\right)$,	$\bigwedge_m \left(M_m(f_1,..,f_n) - f_{parent} \geq 0\right)$[6]
Feature Link	needs	$f_{start} \rightarrow f_{end}$	$f_{end} - f_{start} \geq 0$
	optional alternative	$f_{start} \rightarrow !f_{end}$	$f_{end} + f_{start} \leq 1$
	mandatory alternative	$f_{start} \oplus f_{end}$	$f_{start} + f_{end} = 1$
Variation Group	needs	$f_1 \rightarrow (f_2 \wedge f_3 \wedge \ldots \wedge f_n)$	$\bigwedge_{k=2}^{n} (f_k - f_1 \geq 0)$
	optional alternative	$\bigwedge_m (M_m(f_1,..,f_n))$	$f_1 + f_2 + \ldots + f_n \leq 1$
	mandatory alternative	$f_1 \oplus f_2 \oplus \ldots \oplus f_n$	$f_1 + f_2 + \ldots + f_n = 1$
Configuration Decision		$criterion \rightarrow effect$	$effect - criterion \geq 0$

f	: feature or artifact	!	: boolean operand 'not'
f_{parent}	: parent feature	$\oplus$	: boolean operand 'xor'
$\rightarrow$	: boolean operand 'implication'	$\bigwedge_m$	: for all m
f_{start}	: start feature regarding a dependency	f_{end}	: end feature regarding a dependency
$\wedge$	: boolean operand 'and'	$\bigwedge_{k=2}^{n}$	: for all $k \in \{2,\ldots,n\}$
M_m	: Maxterms[7] of all illegal combinations of features f_n in a feature group		

Tab. 2: Transformation rules

[6] Negated variables are replaced with the transformation rule for excluded features.
[7] Maxterms are true only for one single combination of all variables.

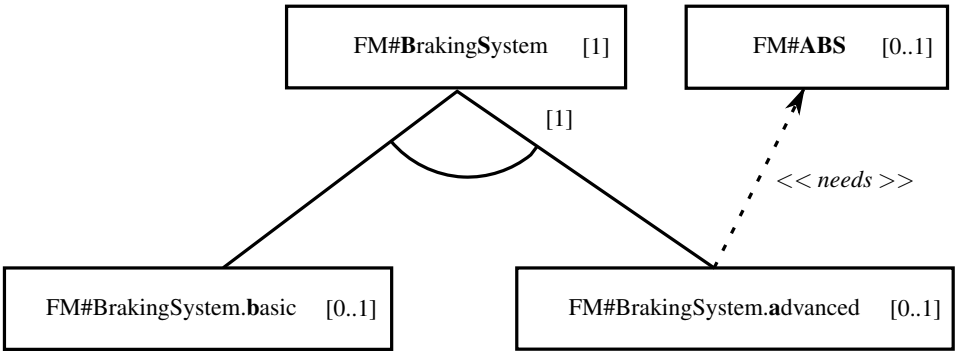

Fig. 4: Example feature model

4 Results and Discussion

The presented approach for deriving multi-objective programs from variant-rich EAST-ADL models has been shown to work for a particular subset of models with boolean variability spaces and summative design objectives. The developed mechanism is able to achieve this result by (a) formulating a set of *N* linear objective functions for *N* considered quality attributes, and by (b) formulating a set of linear program constraints, which together constitute a comprehensive representation of a given model's architecture design space.

By means of additional analyses on the model's variability space, the mechanism can further distinguish between product-line-oriented and architecture-oriented variation points, thereby allowing for product-line-aware optimization. In order to achieve the same capability for non-summative design objectives, we expect that the separated product line variability has to also be considered as part of the formulation of the – then potentially non-linear – objective functions (cf. section 5).

In its current iteration, the approach has proven effective when applied to the particular subset of EAST-ADL models as curtailed in this paper. Due to the flexibility of the utilized mathematical representation, the approach also shows promise in regard to future expansions for considering the full range of variability options in the EAST-ADL language, as well as a coverage of more complex design objectives – in particular non-summative objectives.

5 Outlook

At the time of release of this paper, the presented approach has only been realized for a particular subset of all feasible variant-rich EAST-ADL models, namely those with strictly boolean variability spaces and strictly summative design objectives. In order to expand the approach to cover a broader range of use cases, a number of changes have to be made to the existing mechanism.

One intended goal for future iterations is the support of non-boolean EAST-ADL-based variability information, in particular parametrization and cloned features. Since these

extensions can give rise to non-binary decision variables, it essentially entails the subsequent implementation of an additional change, namely the explicit evaluation of the respective co-domain of each decision variable.

Other projected extensions are concerned with realizing the interpretation and transformation of non-summative quality attribute data into corresponding objective functions, e.g. safety characteristics or performance constraints. We expect that most non-summative design objectives will have to be addressed by individually tailored approaches. Therefore, a reasonable first step for such an effort could be a real-world evaluation of the relative relevance of different quality attributes in the domain of automotive software-intensive systems.

Once a complete or near-complete coverage of variability characteristics and relevant design objectives has been established, a reasonable next step would be a conversion of the analysis output to a standardized optimization format like MPS. Existing tools for multi-objective programming could then be used to conduct case studies on the feasibility of our approach as a basis for optimization analyses on large test models. A sound implementation of this functional coverage would also be a useful contribution to the official EAST-ADL tool platform EATOP [MA12a].

Acknowledgments

We want to express our sincere gratitude and appreciation to Prof. Dr. Ramin Tavakoli Kolagari of the Technische Hochschule Nürnberg Georg Simon Ohm, and Prof. Dr. Klaus Schmid of the University of Hildesheim, for their continuous support in our research efforts and for sharing with us their knowledge and enthusiasm for variation management and automotive software engineering in general.

References

[Al13] Aleti, Aldeida; Buhnova, Barbora; Grunske, Lars; Koziolek, Anne; Meedeniya, Indika: Software architecture optimization methods: A systematic literature review. In: Software Engineering, IEEE Transactions on. volume 39. IEEE, pp. 658–683, 2013.

[Ba79] Balas, Egon: Disjunctive programming. In: Annals of Discrete Mathematics. volume 5. North-Holland Publishing Company, pp. 3–51, 1979.

[Be71] Benayoun, R; De Montgolfier, J; Tergny, Jo; Laritchev, O: Linear programming with multiple objective functions: Step method (STEM). In: Mathematical programming. volume 1. Springer, pp. 366–375, 1971.

[BK05] Burke, Edmund K; Kendall, Graham: Search methodologies: Introductory Tutorials in Optimization and Decision Support Techniques. Springer, 2005.

[Bl13] Blom, Hans; Lonn, Henrik; Hagl, Frank; Papadopoulos, Yiannis; Reiser, Mark-Oliver; Sjostedt, Carl-Johan; Chen, De-Jiu; Kolagari, Ramin Tavakoli: , White Paper Version 2.1.12: EAST-ADL - An Architecture Description Language for Automotive Software-Intensive Systems, 2013.

[CK05] Czarnecki, Krzysztof; Kim, Chang Hwan Peter: Cardinality-based feature modeling and constraints: A progress report. In: International Workshop on Software Factories. pp. 16–20, 2005.

[EA13] EAST-ADL Domain Model Specification, 2013. Version V2.1.12. http://east-adl.info/Specification/V2.1.12/EAST-ADL-Specification_V2.1.12.pdf.

[GR06] Grodzevich, Oleg; Romanko, Oleksandr: Normalization and other topics in multi-objective optimization. In: Proceedings of the Fields–MITACS Industrial Problems Workshop. The Fields Institute, pp. 89–102, 2006.

[MA04] Marler, R Timothy; Arora, Jasbir S: Survey of multi-objective optimization methods for engineering. In: Structural and multidisciplinary optimization. volume 26. Springer, pp. 369–395, 2004.

[MA12a] MAENAD consortium: , EATOP: An EAST-ADL Tool Platform for Eclipse, 2012. Project Deliverable D5.3.1 Version 3.0. http://www.maenad.eu/public/Deliverables/MAENAD_Deliverable_D5.3.1_EATOP_V3.0.pdf.

[MA12b] MAENAD consortium: , Language Concepts Supporting Engineering Scenarios, 2012. Project Deliverable D3.1.1 Version 3.0. http://www.maenad.eu/public/Deliverables/MAENAD_Deliverable_D3.1.1_V3.0.pdf.

[MP07] Metzger, Andreas; Pohl, Klaus: Variability management in software product line engineering. In: Companion to the proceedings of the 29th International Conference on Software Engineering. IEEE Computer Society, pp. 186–187, 2007.

[Re09] Reiser, Mark-Oliver: Core Concepts of the Compositional Variability Management Framework (CVM): A Practitioner's Guide. Technical report, Technische Universität Berlin – Fakultät IV – Softwaretechnik, 2009.

[RKW09] Reiser, Mark-Oliver; Kolagari, Ramin Tavakoli; Weber, Matthias: Compositional variability-concepts and patterns. In: System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on. IEEE, pp. 1–10, 2009.

[Th12] Thüm, Thomas; Apel, Sven; Kästner, Christian; Kuhlemann, Martin; Schaefer, Ina; Saake, Gunter: Analysis strategies for software product lines. Technical report, Otto-von-Guericke-Universität Magdeburg Fakultät Informatik, 2012.

[Wa13] Walker, Martin; Reiser, Mark-Oliver; Tucci-Piergiovanni, Sara; Papadopoulos, Yiannis; Lönn, Henrik; Mraidha, Chokri; Parker, David; Chen, DeJiu; Servat, David: Automatic optimisation of system architectures using EAST-ADL. Journal of Systems and Software, 86(10):2467–2487, 2013.