Automatic Derivation of AADL Product Architectures in Software Product Line Development

Javier González-Huerta, Silvia Abrahão, Emilio Insfran

ISSI Research Group, Universitat Politècnica de València
Camino de Vera, s/n, 46022, Valencia, Spain
{jagonzalez, sabrahao, einsfran}@dsic.upv.es

Abstract. Software Product Line development involves the explicit management of variability that has to be encompassed by the software artifacts, in particular by the software architecture. Architectural variability has to be not only supported by the architecture but also explicitly represented. The Common Variability Language (CVL) allows to represent such variability independently of the Architecture Description Language (ADL) and to support the resolution of this variability for the automatic derivation of AADL product architectures. This paper presents a multimodel approach to represent the relationships between the external variability, represented by a feature model, and the architectural variability, represented by the CVL model, for the automatic derivation of AADL product architectures through model transformations. These transformations take into account functional and non-functional requirements.

Keywords: AADL; Software Product Line Development; Architecture Derivation; Variability Representation.

1 Introduction

Software Product Line (SPL) development is aimed to support the construction of a set of software products sharing a common and managed set of features, which are developed from a common set of core assets in a prescribed way [1]. Thus, in SPL development variability must be defined, represented, exploited and implemented [2]. The external variability (relevant to customers), usually represented by a feature model [3], should be realized by the internal variability (relevant to developers) of the software assets used to build up each individual software product [4].

Software Architecture is a key asset in SPL development and plays a dual role: on the one hand the product line architecture (PLA) should provide variation mechanisms that help to achieve a set of explicitly allowed variations and, on the other hand, the product architecture (PA) is derived from the PLA by exercising its built-in architectural variation points [1].

In order to enable the automatic resolution of the PLA variation points is required not only that the architectural description languages provide variation mechanisms, but also to explicitly represent how the different variants realize the external variability usually represented in feature models.

Although AADL [5] incorporates different variation mechanisms that allow describing variability in system families [6], the explicit representation of the variation points and its variants is also required to cope with the problem of product configuration and architecture derivation.

To tackle this problem, in previous works, we presented the quality-driven product architecture derivation, evaluation and improvement (QuaDAI) method [7], which uses a multimodel to guide the software architect in the derivation, evaluation and improvement of product architectures in a model-driven software product line development process.

The Common Variability Language (CVL) [8] is a language that allows the specification of variability over any EMF-based, and supports the resolution of the variability and automatic derivation of resolved models. CVL incorporates its own variability mechanisms (e.g., fragment substitution) that can be used to extend those provided by the ADLs.

In this paper, we extend the multimodel approach to incorporate the explicit representation of the architectural variability, by using CVL, and to establish relationships among architectural variants and variation points with: i) the features that represent the SPL external variability; ii) the non-functional requirements and iii) the quality attributes. Once the application engineer has selected the features and non-functional requirements (NFRs) and the priorities of the quality attributes that together conform the product configuration, the relationships defined in the multimodel allow us to automatically derive the product AADL specification from the PLA.

The remainder of the paper is structured as follows. Section 2 discusses existing approaches that deal with the explicit representation of architectural variability and the derivation of product architectures in SPL development by using CVL. Section 3 presents our approach for the derivation of AADL product architectures by introducing the explicit representation of the architectural variability with CVL. Finally, Section 4 drafts our conclusions and final remarks.

2 Related work

AADL incorporates different architectural variation mechanisms that support the development of system families (e.g., multiples implementation of a system specification, component extension or the configuration support through alternative source code files) [6]. However, in a real SPL scenario is difficult to manage the derivation of the architectural specification of a product (PA), especially when the SPL allows a wide range of variability. To cope with the problem, in the last years, several approaches had been presented that support the representation of architectural variability and the derivation of product architectures in SPL development by using CVL (e.g., [9], [10], [11]).

Nascimento et al. [10] present an approach for defining product line architectures using CVL. They apply the Feature-Architecture Mapping Method (FArM) to filter the feature models in order to consider only the architectural-related features. These features will form the CVL specification that will allow obtaining the COSMOS* architectural models. They do not define relationships between the external variability model

(features model) and the architectural variability expressed in CVL and thus the derivation of the product architecture taking as input the configuration is not supported. They explicitly omit the non-functional requirements when applying the FArM method.

Svendsen et al. [11] present the applicability of CVL for obtaining the product models for a Train Control SPL that are defined using a DSL. They only consider the explicit definition of the internal variability and consequently, the configuration should be made directly over the CVL specification of the internal variability.

Combemale et al. [9] present an approach to specify and resolve variability on Reusable Aspect Models (RAM), a set of interrelated design models. They use CVL to resolve the variability on each model and then compose the corresponding reusable aspects by using the RAM weaver. They also consider just the internal variability, and the configuration should be made over the CVL specification.

In summary, none of the aforementioned approaches establish relationships among the SPL external variability and the architectural variability, even though some of them acknowledge that is a convenient practice in variability management [9]. Establishing connections between the SPL external variability, expressed by means of feature models, the non-functional requirements, represented in the quality model, and the architectural variability, represented by using CVL allows us:

i) To configure the product by using the feature model and the quality viewpoint;
ii) To check its consistency by using the constraints defined on the features model, on the quality viewpoint and on the multimodel;
iii) To solve the architectural variability automatically by using model transformations.

3 A Multimodel Approach for the Derivation of AADL Product Architectures

QuaDAI is an approach for the derivation, evaluation and improvement of product architecture that defines an artifact (the multimodel) and a process consisting of a set of activities conducted by model transformations. QuaDAI relies on a multimodel [12] that allows the explicit representation of different viewpoints of a software product line and the relationships among them. In this section, we focus on the representation of the architectural variability, its resolution and the derivation of the software architecture of the product under development.

3.1 Illustrating Example

The approach is illustrated through the use of a running example: a SPL from the automotive domain that comprises the safety critical embedded software systems responsible for controlling a car. This SPL is an extension of the example introduced in [13], and was modified in order to apply, among others the variation points described in [14].

This SPL comprises several features such as Antilock Braking System, Traction Control System, Stability Control System or Cruise Control System. The Cruise Control System feature incorporates variability. This variability is resolved depending on other selections made on the features model (i.e., the selection of the cruise control together with the park assistant implies the positive resolution of an extended version of the cruise control[1]). Fig. 1 shows an excerpt of the features model that represents the SPL external variability.

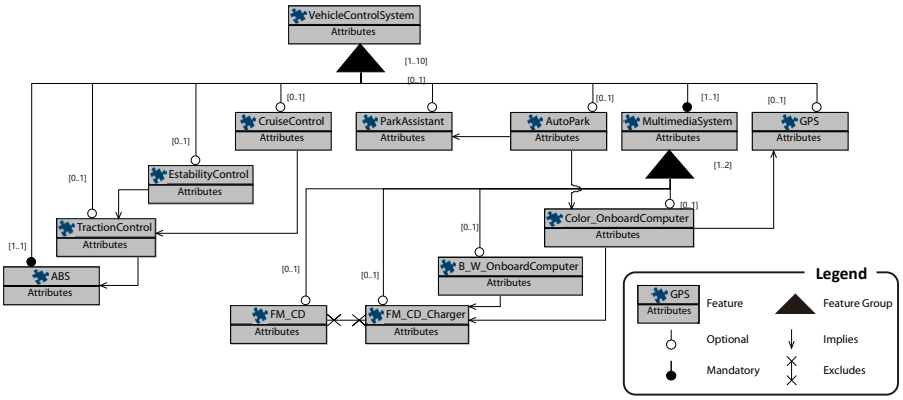

Fig. 1. SPL External Variability

3.2 A Multimodel for Representing Architectural Variability

In QuaDAI, a multimodel permits the explicit representation of relationships among entities in different viewpoints. A multimodel is a set of interrelated models that represent the different viewpoints of a particular system. A viewpoint is an abstraction that yields the specification of the whole system restricted to a particular set of concerns, and it is created with a specific purpose in mind. In any given viewpoint it is possible to produce a model of the system that contains only the objects that are visible from that viewpoint [16]. Such a model is known as a viewpoint model, or view of the system from that viewpoint. The multimodel permits the definition of relationships among model elements in those viewpoints, capturing the missing information that the separation of concerns could lead to [12].

The problem of representing and automatically resolving the architectural variability taking into account functional and non-functional requirements requires (at least) three viewpoints:

- **The variability viewpoint** represents the SPL external variability expressing the commonalities and variability within the product line. Its main element is the feature, which is a user-visible aspect or characteristic of a system [3] and it is expressed by means of a variant [15] of the cardinality

[1] The whole specification of the example is available at http://users.dsic.upv.es/~jagonzalez/CarCarSPL/links.html

based feature model (shown in Fig. 1).

- **The architectural viewpoint** represents the architectural variability of the Product Line architecture that realizes the external variability of the SPL expressed in the variability viewpoint. It is expressed by means of the Common Variability Language (CVL) and its main element is the Variability Specification (VSpec). We only represent in the multimodel the PLA architectural variability, the PLA is represented in an AADL base model, which is referenced by the CVL specification. A base model, under the CVL terminology, is a model on which variability is defined using CVL [8]. The base model is not part of CVL and can be an instance of any metamodel defined via MOF [8]. Fig. 2 shows an example of the CVL variability specification on an AADL base model, allowing to express whether some AADL elements should exist or not in the resolved model (e.g., the ABS) and to configure some internal values (e.g., the range value assignment).
- **The quality viewpoint** represents the hierarchical decomposition of quality into sub-characteristics, quality attributes, metrics and the impacts and constraints among quality attributes. Is expressed by means of a quality model for software product lines [17], which extends the ISO/IEC 25010 (SQuaRE) [18] and allows the representation of NFRs as constraints affecting characteristics, sub-characteristics and quality attributes.

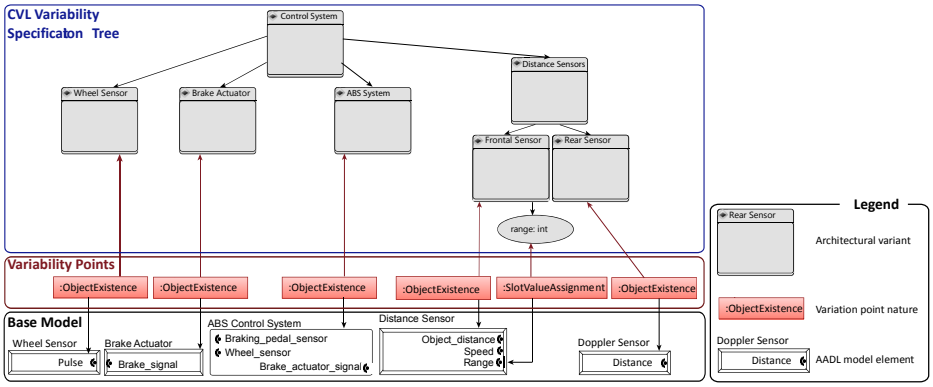

Fig. 2. CVL Variability Specification on an AADL Base Model

The multimodel also allows the definition of relationships among elements on each viewpoints with different semantics as *is_realized_by* [19] or *impact* relationships [12]. An excerpt of these relationships is shown in Fig. 3.

We can describe in the multimodel: i) how the ABS feature *is_realized_by* a set of VSpecs (e.g., the WheelRotationSensor); ii) how the user_safety NFR *is_realized_by* a set of features (e.g., the ABS or the Stability Control); iii) how the selection of a given feature VSpec impacts positive or negatively on a quality attribute; or iv) how the positive resolution of a given VSpec impacts positive or negatively on a quality attribute. These relationships are used to check the consistency of the configuration and to decide

which variation points should be resolved positively in the CVL resolution model driving the derivation of the AADL product architecture.

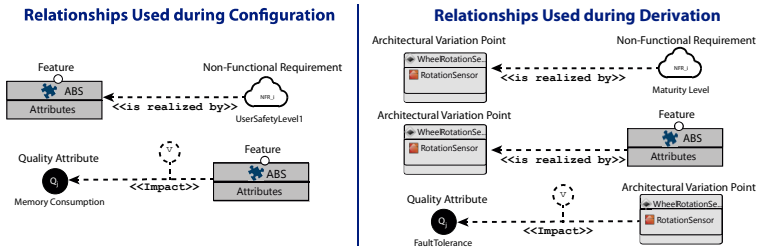

Fig. 3. Multimodel Relationships

3.3 Automating the Derivation of Product Architectures

The QuaDAI derivation process for obtaining AADL product architectures based on the functional and non-functional requirements comprises two main activates: the *product configuration* and the *architecture instantiation*. Fig. 4 shows an excerpt of the specification of the activities with the main input and output artifacts using the Software & Systems Process Engineering Meta-Model (SPEM) [20].

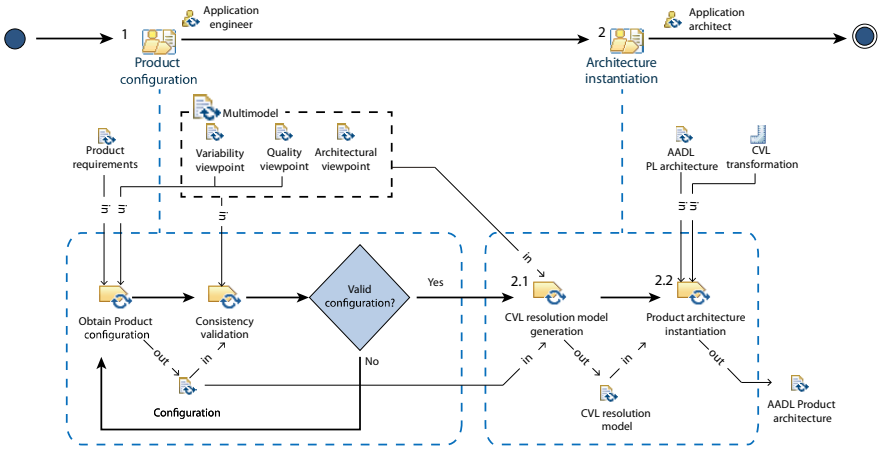

Fig. 4. SPEM specification of the QuaDAI Derivation Process

In the *product configuration* activity, the application engineer selects the features and NFRs that the product must fulfill and establishes the quality attributes priorities in the *obtain product configuration* task. Those priorities will be used during the derivation to choose from a set of architectural variants that having the same functionality differ in their quality attribute levels. In the *consistency validation* activity, first we check the variability viewpoint consistency (i.e., whether the selection of features fulfills the constraints defined in the feature model) and the quality viewpoint consistency (i.e., whether the priorities of the quality attributes defined in the configuration satisfy

the impact and constraint relationships among them defined in the quality viewpoint). Once the intra-viewpoint consistency is satisfied we check the inter-viewpoint consistency (i.e., whether the configuration satisfy the *is_realized_by* and *impact* relationships defined among elements on different viewpoints). The multimodel relationships had been formalized and operationalized in OCL and are checked at runtime by using the OCLTools validator [21]. This consistency checking mechanism allows us to assure that the product configuration meets the SPL constraints facilitating the architecture instantiation activity that focus on the resolution of the PLA architectural variability.

In the *architecture instantiation* activity, the application architect generates the AADL product architecture by means of two model transformation activities. The first transformation, *CVL resolution model generation*, takes as input a valid product configuration and the multimodel (i.e., the relationships between the architectural viewpoint with variability and the quality viewpoint) and, through a Query View Transformation (QVT) [22] model transformation, generates a CVL resolution model. With the multimodel relationships, the QVT transformation decides which architectural variants have to be positively resolved in each variation point. Finally, the *product architecture instantiation* activity, through a CVL transformation, takes as input the CVL resolution model and generates the AADL product architecture. This AADL product architecture represents the resolution of the PLA architectural variability taking into account not only the functional requirements and non-functional requirements selected in the configuration. The use of CVL alleviates part of the computational complexity since it allows us, at design time, to describe the PLA architectural variants and, in derivation time, to only focus on the resolution of the PLA architectural variability. Fig. 5 shows an outline of the AADL architecture derivation with the CVL resolution model generation and the AADL Product architecture instantiation.

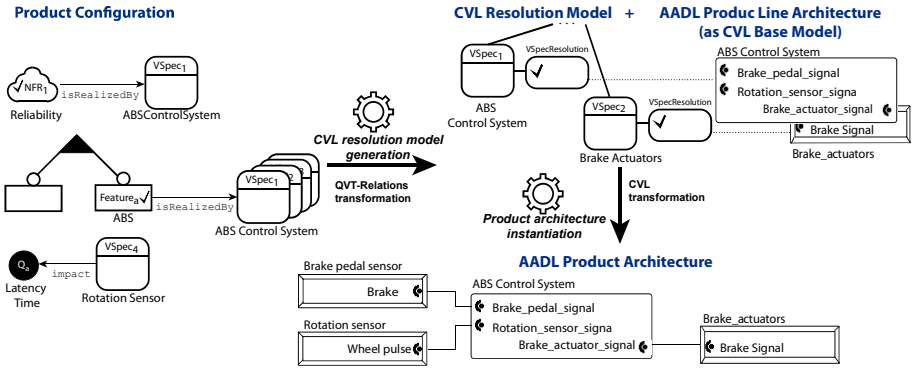

Fig. 5. AADL Product Architecture Instantiation

3.4 Tool Support

The approach is supported by a prototype[2] that gives support to the configuration, consistency checking and generation of the CVL resolution model. The prototype allows importing feature models and CVL specifications defined using third party tools and to establish the relationships among them described in the paper so as to automate the AADL product architecture derivation.

The variability viewpoint consistency checking is carried out by transforming the feature model and the features selection to the FAMA [23] metamodel and by calling the FAMA validator. The quality viewpoint and the inter-viewpoint consistency checking are carried out through OCL constraints checked at runtime by the OCLTools validator. The tool is also capable of calling the QVT transformation that generates the CVL resolution model.

Fig. 6(a) shows the call to the CVL resolution creation functionality. Fig. 6(b) shows the resulting CVL resolution model when for a configuration formed by the feature configuration *features*={`Vehicle Control System, ABS, TractionControl, StabilityControl and FM_CD`} (see Fig. 1) and the NFRs configuration *nfrs*={`EuroNCAP`[3]}.

Finally, with the integration of the CVL supporting tool [24] the CVL transformation can be called so as to generate the resulting AADL product architecture.

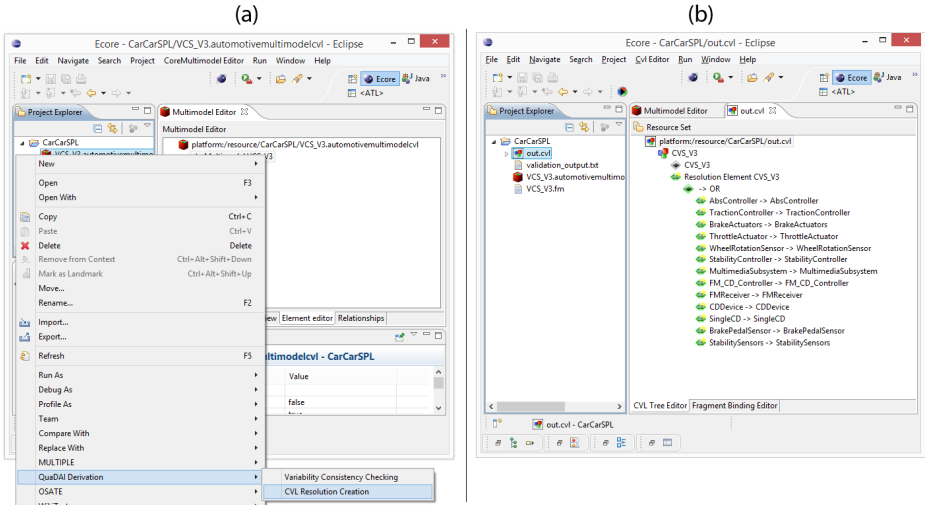

Fig. 6. Generation of the CVL Resolution Model with the Prototype

---

[2] The prototype is available for download at: http://users.dsic.upv.es/~jagonzalez/CarCarSPL/index.html

[3] EuroNCAP is a voluntary EU vehicle safety rating system. In our example, the EuroNCAP NFR is realized by the ABS, the Traction Control, and the Stability Control features.

4 Conclusions and Further Works

In this paper, we have presented an approach to explicitly represent architectural variability on AADL architectural models by using CVL. We include the architectural variability in a multimodel in which we also represent the SPL external variability in a feature model, and the non-functional requirements in a quality model. In this multimodel, we can establish relationships among elements on the CVL model, the feature model and the quality model. This information is used to drive the model transformation that resolves the architectural variability and obtains the AADL product architecture. The approach is supported by a tool with which the user can edit a product configuration, check its consistency and automatically derive the CVL resolution model. The CVL resolution models allow us to obtain the AADL product architecture by using the CVL supporting tool.

In this work, we apply model-driven engineering principles to provide a feasible solution to an open, complex, error-prone and time-consuming problem in the software product line development community, which is the derivation of product architectures takin into account functional and non-functional requirements.

As further work, we plan to empirically validate the approach through controlled experiments and case studies. We plan also to analyze how to incorporate more powerful CVL variation mechanisms (i.e., the use of VInterfaces that can be used to configure CVL configuration units) and its possible use in combination with the AADL syntax. Finally, although the approach has been initially defined for working together with AADL, the use of CVL isolates the approach from the architectural description language and we want to analyze its applicability to other MOF-based ADLs.

Acknowledgements. This research is supported by the Value@Cloud project (MICINN TIN2013-46300-R) and the ValI+D fellowship program (ACIF/2011/235).

References

1. Clements, P., Northrop, L.: Software Product Lines: Practices and Patterns. Addison-Wesley Professional (2001).
2. Van der Linden, F., Schmid, K., Rommes, E.: Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer Berlin Heidelberg (2007).
3. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. , CMU/SEI-90-TR-21 ESD-90-TR-222, Software Engineering Institute, Carnegie Melon University (1990).
4. Pohl, K., Böckle, G., van der Linden, F.: Software product line engineering. Springer, Berlin (2005).
5. Feiler, P.H., Gluch, D.P.: Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language. Addison Wesley (2013).
6. Feiler, P.H.: Modeling of System Families. , CMU/SEI-2007-TN-047, Software Engineering Institute, Carnegie Mellon University (2007).
7. González-Huerta, J., Insfrán, E., Abrahão, S.: Defining and Validating a Multimodel Approach for Product Architecture Derivation and Improvement. 16th International

Conference on Model-Driven Engineering Languajes and Systems. pp. 388–404. , Miami, USA (2013).
8. Object Management Group: Common Variability Language (CVL) OMG Revised Submission. (2012).
9. Combemale, B., Barais, O., Alam, O., Kienzle, J.: Using CVL to operationalize product line development with reusable aspect models. Workshop on Variability Modeling Made Useful for Everyone. pp. 9–14. , Innsbruck, Austria (2012).
10. Nascimento, A.S., Rubira, C.M.F., Burrows, R., Castor, F.: A Model-Driven Infrastructure for Developing Product Line Architectures Using CVL. 7th Brazilian Symposium on Software Components, Architectures and Reuse. pp. 119–128. , Brasilia, Brazil (2013).
11. Svendsen, A., Zhang, X.: Developing a software product line for train control: a case study of CVL. 14th Software Product Line Conference. pp. 106–120. , Jeju Island, South Korea (2010).
12. González-Huerta, J., Insfran, E., Abrahão, S.: A Multimodel for Integrating Quality Assessment in Model-Driven Engineering. 8th International Conference on the Quality of Information and Communications Technology. pp. 251–254. , Lisbon, Portugal (2012).
13. Hudak, J., Feiler, P.H.: Developing AADL Models for Control Systems : A Practitioner ' s Guide. , CMU/SEI-2007-TR-014, Software Engineering Institute, Carnegie Mellon University (2007).
14. Shiraishi, S.: An AADL-based approach to variability modeling of automotive control systems. 13h Model Driven Engineering Languages and Systems. pp. 346–360. , Oslo, Norway (2010).
15. Gómez, A., Ramos, I.: Cardinality-Based Feature Modeling and Model-Driven Engineering : Fitting them Together. International Workshop on Variability Modelling of Software-Intensive Systems. pp. 61–68. , Linz, Austria (2010).
16. Barkmeyer, E.J., Feeney, A.B., Denno, P., Flater, D.W., Libes, D.E., Steves, M.P., Wallace, E.K.: Concepts for Automating Systems Integration. , NISTIR 6928, U.S. Department of Commerce (2003).
17. González-Huerta, J., Insfran, E., Abrahão, S., McGregor, J.D.: Non-functional requirements in model-driven software product line engineering. Proceedings of the Fourth International Workshop on Nonfunctional System Properties in Domain Specific Modeling Languages. pp. 1–6. , Innsbruck, Austria (2012).
18. ISO/IEC: ISO/IEC 25000:2005 Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE, (2005).
19. Janota, M., Botterweck, G.: Formal approach to integrating feature and architecture models. 11th Conference on Fundamenteal Approaches to Software Engineering. pp. 31–45. , Budapest, Hungary (2008).
20. Object Management Group: Software & Systems Process Engineering Meta-Model Specification (SPEM) v2.0. (2008).
21. Eclipse: Eclipse OCL, http://projects.eclipse.org/projects/modeling.mdt.ocl.
22. Object Management Group: Meta Object Facility (MOF) 2.0 Query / View / Transformation Specification. (2008).
23. ISA Research Group: Fama Tool Suite, http://www.isa.us.es/fama/.
24. Haugen, Ø., Moller-Pedersen, B., Olsen, G.K., Svendsen, A., Fleurey, F., Zhang, X.: Consolidated CVL language and tool. , MoSiS Project, D.2.1.4., SINTEF, Univeristy of Oslo (2010).