Towards Quality Evaluation in Embedded Model Driven Incremental Product Lines: Analyzing Performance

Lorea Belategi[1], Joseba Andoni Agirre[1], Leire Etxeberria[1], Goiuria Sagardui[1], and Maider Azanza[1]

[1]Mondragon Unibertsitatea, Mondragon, Spain
{lbelategui, jaagirre, letxeberria, gsagardui, mazanza}@eps.mondragon.edu

Abstract. Although they often go unnoticed, embedded systems are becoming essential in our everyday lives. From a software development perspective, embedded systems present interesting challenges, some of which can be addressed using advanced development paradigms such as *Model Driven Engineering (MDE)* and *Software Product Line Engineering (SPLE)*. Nevertheless, these paradigms do not explicitly address quality attributes (e.g., performance), which are primary concerns in embedded systems. Within the general aim of ensuring that each member of the product line complies with the quality requirements, this paper presents a proposal for the performance analysis of products of embedded model driven incremental product lines.

Keywords: Embedded Systems, Model Driven Engineering, Software Product Line Engineering, Quality Attributes, Performance, MARTE.

1 Introduction and Motivation

In recent years, embedded systems have substantially increased their presence both in industry and in our everyday lives. Already, over 98% of computing chips are embedded in all sorts of devices (e.g., planes, cars, mobile phones, etc.). Hence, more and more effort is being dedicated to the development of such systems.
Embedded systems consist of hardware, software and an environment. It is important to note that there is an essential difference between embedded and other computing systems: since embedded systems involve computation that is subject to physical constraints, the separation of computation (software) from physicality (platform and environment) does not work for embedded systems. Instead, the design of embedded systems requires a holistic approach that integrates hardware design, software design, and control theory in a consistent manner [10].
If we focus on the software part, embedded system software characterizes itself, among others, by heterogeneity, distribution (on potential multiple and heterogeneous hardware resources), ability to react (supervision, user interfaces modes), criticality, real-time and consumption constraints [1]. As a consequence, quality attributes become a main concern when developing such software and *Validation and Verification (V&V)* from early development stages is crucial. In a nutshell, the need to

cater for all these requirements on top of the functional ones makes the development of software for embedded systems a complex endeavor.
Advanced software development paradigms such as *Model Driven Engineering (MDE)* and *Software Product Line Engineering (SPLE)* can assist when dealing with this complexity. The former abstracts from system complexity by the use of models, where information related to the critical quality attributes can be attached in order to support V&V through model analysis [1]. As a case in point, performance can be analyzed using stochastic techniques such as queuing theory or Petri nets to calculate response times, delays and resource requirements [7].
As for *SPLE*, it permits to build families of related systems by capitalizing on their common features in the form of core assets [6]. There are two fundamentally different ways of deriving each product from the common core assets. First, negative variability selectively deletes parts of an artifact, which comprises the realization of all the features of the product line. The second alternative uses positive variability. We start with a minimal core and incrementally add the features we want the product to contain [15].
The combination of both paradigms in *Model Driven Product Line Engineering (MDPLE)* combines the benefits of both paradigms. In this case models are the central artifact and become main core assets of the product line from which products will later be derived.
In software product lines, core assets are the means to promote reuse, as they will be used in different products of the family. As mentioned above, in embedded systems quality attributes play an essential role. We need to assure that each product meets its quality requirements, which can vary among the different products of the line [11]. This is where model based analysis becomes a valuable aid [1]. However, it needs to be tailored to a product line setting. To minimize development cost and effort while assuring desired quality attributes at the same time, model analysis should not be performed from scratch in a per product basis but, whenever possible, analysis information present in the core assets should be reused.
This work presents a proposal for increasing reuse in quality attribute product analysis for embedded model driven product lines implemented incrementally (i.e., using positive variability). Performance is used as the illustrative quality attribute, which is analyzed using MARTE [1] and *Feature Oriented Software Development (FOSD)* [4] is the SPL development paradigm of choice.

2 Performance Analysis in Feature Oriented Model Driven Product Lines

The *MARTE (UML Profile for Modeling and Analysis of Real-Time and Embedded systems)* profile [1] allows annotating temporal aspects (information related to schedulability and performance) in models in order to analyze or determine whether they will meet those temporal requirements. It defines quantitative performance annotations (such as resource demands made by different software execution steps, performance requirements, etc.). Thus they are aimed at determining the rate at which a system can perform a function [7]. Through these annotated models, analysis

models, which are guided by a specific formalism in order to perform analysis, can be obtained (e.g., LQN, petri nets, etc.).
Feature Oriented Software Development (FOSD) is a general paradigm for product synthesis in SPLE [4]. It advocates for the incremental development of the product line family members. Here, features are not only increments in program functionality that customers use to distinguish one application from another, but are the actual building blocks of the artifact at hand (i.e., features are the actual core assets). In MDPLE models are built incrementally by successively adding model deltas that realize features [3].
The process proposed in this paper follows the two phases of the traditional SPL development [6]:

2.1 Domain Engineering

Domain Engineering is the process of the SPL engineering in which the commonality and the variability of the product line are defined and realized. Hence, the feature model that defines the common and variable features of the product line and core assets that realize such features are created in this phase. In the same way, the information that will be used to perform product model analysis is also defined here.
When performing analysis using MARTE extra annotations for analysis are attached to an actual design model (application, platform and deployment models), rather than requiring a special version of the design model to be created only for the analysis. This is done using stereotypes and tagged values [1].
AnalysisContext is the main concern of MARTE to perform model analysis. It allows analyzing hypothetical real-time situation of the system by describing a specific behavior and the execution platform through design models with non-functional annotations. But variability is the key aspect of SPLs and it must be considered when analyzing models: not all products of the SPL have the same functionalities; often, some of the hardware devices and other performance-affecting factors can vary from one product to another [13]; software can be allocated in different ways in a specific platform to optimize system objectives [7]; and two products with the same functionality may require different quality attributes, as well as the degree or priority of them [9]. Therefore, analysis can vary from one product to another, depending on its functional, platform or allocation features. In the case of FOSD, each feature is modeled separately in the form of model deltas. By composing such deltas, products will be created [3]. In the same way, deltas for analysis can be composed to obtain product specific analysis models. Each delta must be annotated using MARTE analysis stereotypes to obtain the product specific analysis models by reusing such deltas. However, two exceptions exist: the *GaAnalysisContext* and *GaWorkloadEvent* analysis stereotypes. These stereotypes must be applied once the analysis model is composed.
GaWorkloadEvent specifies a flow of events that give way to system-level behavior initialization. Thus, the *GaWorkloadEvent* stereotype should go on the first action in the model representing the behavior of the analysis model giving way to the following operations.

On the other hand, *GaAnalysisContext* is the stereotype that identifies models that gather information about the system's behavior and workload, execution platform and allocation for the analysis and specifies global parameters (i.e., properties that describe different cases being considered for analysis) and is also annotated on the complete model.
We have identified another case that deserves special attention: the *PaStep* and *PaCommStep* stereotypes. *PaStep* is an execution step on a host processor, while a *PaCommStep* unit may be executed by a combination of host middleware and network services. Depending on the selected execution platform and allocation, a different stereotype should be annotated.
Analysis input/output variables must be defined in order to perform model analysis. As not all analyses have same variables, some standard variable values can be defined, which allow performing standard model based analysis for different product specifications. In case other analysis and variables are required, the user should provide the necessary information manually using the required annotations.
Finally, two approaches can be applied in order to evaluate products depending on the desired goal:

- When the goal of the evaluation is to ensure that the required quality levels for all the products are achieved in the software product line: In order to reduce the evaluation effort, from all the products of the line, a subset of representative products can be selected and evaluated, and in this way, data can be extracted to estimate the results of all the products [8].
- If the goal is to evaluate a specific product: That product is derived and then evaluated. This is the goal of this work and is performed in application engineering, which is described in the following section.

2.2 Application Engineering

Application Engineering is the process of the SPL engineering in which the applications of the product line are built by reusing domain artifacts and exploiting the product line variability.
Considering that we want to ensure that the performance is adequate, the steps that need to be followed to perform model based analysis would be the following:

1. Functional and platform features must be chosen first to decide the software allocation of the desired product.
2. Once the desired features have been selected, the model deltas that realize each feature can be composed to obtain the product specific analysis models.
3. The obtained models will contain MARTE analysis stereotypes, although, as it was aforementioned, some stereotypes such as *GaAnalysisContext* and *GaWorkloadEvent* must be annotated once the analysis model is derived. Considering that these annotations are general, i.e., do not change from product to product; this can be done automatically using a model transformation.

4. If product specific analysis variables are required, they can be defined at this point. Otherwise, the standard variable values defined in Domain Engineering can be used.
5. The now complete MARTE analysis model can be transformed using a bridge tool into the input model of the analysis tool, where the actual analysis will be performed.

3 Related Work

Being able to perform analysis with the aim of V&V of quality attributes at early phases facilitates obtaining a product with the same functionality but different quality levels through model based analysis. Over the last years, some works have been proposed, related to quality attributes, analysis based on models following SPLE and MDE, but most of them focus on negative variability.
MeMVaTEx methodology [2] proposes the decomposition of the design process in different abstract levels of the EAST-ADL2 framework. System requirements are analyzed in the design phase and taken into account in verification. It concerns the *verify* links that show how requirements can be verified by test cases. Tawhid and Petriu propose a SPL modeling approach [14] with functional variability and annotated in a general way (using variables) with MARTE in order to validate performance aspects of products. And Belategi et al. [5] describe a process for model based analysis of product lines where analysis variability aspects (functional, platform, allocation, quality attributes and analysis) are taken into account from a negative variability point of view. Moreover, the defined process is oriented to the evaluation of the complete product line instead of evaluating specific products as is the case of this proposal.
On the other hand, Espinoza [7] proposes a methodology that describes a set of steps to perform complex model analysis. In this methodology different computation blocks must be defined, adequate non-functional properties specified, etc. before reusing model elements and it has been defined for a single product model analysis. Variability is not taken into account. Thus some modifications are needed before applying this proposal in embedded SPLs.

4 Conclusions and Future Work

In this paper a model based quality attribute analysis for specific products in an embedded product line is proposed, where functional, quality attribute, platform and allocation variability issues need to be taken into account. MARTE and FOSD have been used for the analysis with the aim of reusing the performance information when analyzing each particular product. Besides, based on the analysis models composed from deltas, analyses that are not included in the previously defined standard set can be specified by hand by the user, thus taking advantage existing analysis models and minimizing effort and time.

The future work to be carried out includes a real case study to analyze the results obtained following this proposal. Moreover, the selected architectural pattern has an impact in the resulting system quality (e.g., communication patterns impact performance) [12] and should be taken into account in the proposal (e.g., in the corresponding feature model). Considering our general aim, we also intend to complement our proposal by studying other quality attributes, distinguishing between operational or execution quality attributes and development or non-execution ones, and to compare the results attained with both types of quality attributes.

Acknowledgments. This work was supported by COMODE Project (Basque Government under grants UE2011-4). The project has been developed by the embedded system group supported by the Department of Education, Universities and Research of the Basque Government.

References

1. UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems. formal/2009-11-02 (2009)
2. Albinet, A., Begoc, S., Boulanger, J. -. et al.: The MeMVaTEx Methodology: From Requirements to Models in Automotive Application Design. (2008)
3. Azanza, M., Batory, D., Díaz, O. et al.: Domain-Specific Composition of Model Deltas. pp. 16-30, Proc. of the Third int. conf. on Theory and Practice of Model Transformations (2010)
4. Batory, D. S., Sarvela, J. N., Rauschmayer, A.: Scaling Step-Wise Refinement. IEEE Trans. Software Eng., 30 (2004) 355-371
5. Belategi, L., Sagardui, G., Etxeberria, L.: Model Based Analysis Process for Embedded Software Product Lines. ICSSP'11 (2011)
6. Clements, P., & Northrop, L.: Software product lines: Practices and patterns. Addison-Wesley Professional (2001)
7. Espinoza, H.: An Integrated Model-Driven Framework for Specifying and Analyzing Non-Functional Properties of Real-Time Systems, Thesis (2007)
8. Etxeberria, L., & Sagardui, G.: Variability Driven Quality Evaluation in Software Product Lines, pp. 243-52, 12th Int. Software Product Line Conference (SPLC), (2008)
9. Etxeberria, L., Sagardui, G., Belategi, L.: Quality Aware Software Product Line Engineering. Journal of the Brazilian Computer Society (JBCS), 14 (2008)
10. Henzinger, T. A., & Sifakis, J.: The Embedded Systems Design Challenge, pp. 1-15, 14th Int. Symposium on Formal Methods (FM), Hamilton, Canada (2006)
11. Montagud, S., & Abrahao, S.: Gathering Current Knowledge about Quality Evaluation in Software Product Lines, pp. 91-100, 13th Int. Software Product Lines Conference (2009)
12. Ovaska, E., Evesti, A., Henttonen, K. et al.: Knowledge Based Quality-Driven Architecture Design and Evaluation. Information & Software Technology, 52 (2010) 577-601
13. SEI A Framework for Software Product Line Practice, Version 5.0. http://www.sei.cmu.edu/productlines/frame_report/index.html (2008)
14. Tawhid, R., & Petriu, D. C.: Towards Automatic Derivation of a Product Performance Model from a UML Software Product Line Model. pp. 91-102, In: WOSP '08 (2008)
15. Voelter, M., & Groher, I.: Product Line Implementation using Aspect-Oriented and Model-Driven Software Development. pp. 233-242, Int. Software Product Line Conference (SPLC), (2007)