Embedded Software Product Lines: Domain and Application Engineering Model Based Analysis Processes

<table>
<thead>
<tr>
<th>Journal</th>
<th>Journal of Software: Evolution and Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manuscript ID</td>
<td>JSME-12-0034</td>
</tr>
<tr>
<td>Wiley - Manuscript type</td>
<td>Special Issue Paper</td>
</tr>
<tr>
<td>Date Submitted by the Author</td>
<td>15-Mar-2012</td>
</tr>
<tr>
<td>Complete List of Authors</td>
<td>Belategi, Lorea; Mondragon Unibertsitatea, Sagarduí, Goiuria; Mondragon Unibertsitatea, Etxeberria, Leire; Mondragon Unibertsitatea, Azanza, Maider; University of the Basque Country UPV/EHU,</td>
</tr>
<tr>
<td>Keywords</td>
<td>Software Product Lines, Model Based Analysis Process, Model Driven Development, Quality Attributes, Performance</td>
</tr>
</tbody>
</table>
Embedded Software Product Lines: Domain and Application Engineering model based analysis processes

Lorea Belategi∗, Goiuria Sagardui1, Leire Etxeberria1, Maider Azanza2

Mondragon Unibertsitatea, Goiru 2, 20500 Arrasate, Spain1
{lbelategui,gsagardui,letxeberria}@mondragon.edu
University of the Basque Country UPV/EHU, San Sebastián, Spain2
maider.azanza@ehu.es

SUMMARY

Nowadays, embedded systems are gaining importance, while the development of their software is increasing its complexity having to deal with quality, cost and time-to-market issues among others. With stringent quality requirements such as performance, early verification and validation becomes critical in these systems. Advanced development paradigms such as Model Driven Engineering and Software Product Line Engineering bring considerable benefits to the development and validation of embedded system software. However, these benefits come at the cost of increasing process complexity. This work presents, a process, based on UML and MARTE, for the analysis of embedded model driven product lines. It specifies the tasks, the involved roles and the workproducts that form the process and how it is integrated in the more general development process. Existing tools that support the tasks to be performed in the process are also described. A classification of such tools and a study of traceability among them, allowing engineering teams to choose the most adequate chain of tools to support the process is described. Copyright © 2010 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Software product line; model based analysis process; model driven development; quality attributes; performance

1. INTRODUCTION

Embedded systems are becoming ubiquitous both in industry and in our everyday lives and the software that runs on them is fundamental. As a result, embedded system engineering teams need to face the challenges these systems pose to the development of software.

While cost, quality and time-to-market have always been main concerns in software engineering, these concerns are even more pivotal in embedded system software. Not only development time must be met and changing requirements managed as in other domains, but embedded software architectures are usually complex and fragile, technological platforms evolve and change constantly and requirements such as reliability or safety add even more complexity to development. Specifically, embedded systems distinguish themselves by the following characteristics: heterogeneity (hardware/software), distribution (on potential multiple and heterogeneous hardware resources), ability to react (supervision, user interfaces modes), criticality, real-time and consumption constraints [1], and this impacts on how their software is developed. It is important to note that, considering the context these systems normally operate in where malfunction

∗Correspondence to: lbelategui@mondragon.edu

Copyright © 2010 John Wiley & Sons, Ltd.
Prepared using smrauth.cls [Version: 2010/05/10 v2.00]
can even result in casualties (e.g., healthcare systems), we need to be able to ensure the quality of the software.

Model Driven Engineering (MDE) is a paradigm that helps to address software development by reducing the gap between the problem domain and the software implementation domain using of abstractions (i.e., models), which are systematically transformed to concrete software implementations [2]. Models become the central artifacts in the software development process, hence increasing productivity and shortening development times. However, the advantages of MDE are exceed these time issues. In the particular case of embedded systems, software Verification and Validation (V&V) from early development stages is crucial to ensure software quality and one of the most spread means to ensure such quality is analysis [3]. Nevertheless, performing embedded software V&V is not trivial. Among other difficulties, in most cases embedded software is hardware-dependent (the hardware imposes requirements on the software, which the software has to cope with) and has to run under different configurations (communicating with different number and kind of devices). MDE supports this early V&V, as models can be annotated with information related to quality attributes (using UML and profiles or domain specific metamodels) [1, 4, 5].

MDE offers multiple advantages both for the development and V&V of embedded system software. However, it does not deal with another of the challenges that embedded software poses: variability. Families of related embedded systems exist, which vary from each other in terms of their behavior, quality attributes, platform, network, physical configuration, middleware, scale factors and in a multitude of other ways*. In combination with MDE, Software Product Line Engineering (SPLE) can be an adequate alternative to traditional embedded software development in order to explicitly manage this variability.

Hence, MDE and SPLE are paradigms that bring benefits when dealing with the complexity of embedded software development. But these benefits come at the cost of increasing the complexity of the development process [6] and this complexity is even exacerbated when the software needs to comply with real-time constraints such as performance. As a result, the process to engineer these systems needs to cater for a diverse set of stakeholders (e.g., software, electronic or real-time engineers), an increased number of workproducts (e.g., feature models, system models, code, documentation, etc.) and new tasks (e.g., domain analysis or product derivation). If we perform V&V, this entails that an already complex process needs to deal with the analysis-related tasks. Moreover, the analysis process itself needs to consider the more diverse group workproducts required by MDE and SPLE (e.g., metamodels, transformations, feature models, etc.).

In order to tackle the aforementioned complexity this work focuses on the analysis process required for the V&V of an embedded model driven software product line, an analysis process, based on UML and MARTE (UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems). It specifies the tasks, the involved roles and the workproducts that form the process and how it is integrated in the more general development process. Existing tools that support the tasks to be performed in the process are also described. Specifically, based on [7], the contributions of this paper are the following:

- A more general process where the different roles, tasks and workproducts are identified.
- Existing tools that can support the process are also included in the process.
- A classification of existing tools and a study of traceability among them, allowing engineering teams to choose the most adequate chain of tools to support the analysis process.
- An enriched metamodel of feature models that includes the means to model software allocation and analysis variability.
- A metamodel for relationship models, the workproduct that relates the different pieces of the analysis process, managing variability and relationships, and suggesting relevant AnalysisContexts.

This paper is organized as follows: Section 2 presents how model analysis is performed in model driven embedded SPLs, a proposal process for such goal and tools to support the process are

*http://wwwsei.cmu.edu/productlines/frame_report/index.html
described in Section 3, Section 4 analyzes the related works and Section 5 presents the conclusions and future work.

2. MODEL ANALYSIS FOR MODEL DRIVEN EMBEDDED SOFTWARE PRODUCT LINES

Quality is an important aspect that needs to be taken into account from the beginning and during the whole software life cycle. Software quality is the degree to which software possesses a desired combination of attributes [8], and a quality attribute is a property of work products or goods by which its quality will be judged by some stakeholder or stakeholders\(^1\). However, performing software V&V is not trivial, and even less performing embedded software V&V due to critical quality attributes and limited resources involved in the system development.

There are many works that propose the use of models annotated with specific profiles that help performing model based analysis following MDE paradigm and in particular with MARTE profile [9, 10, 11, 12].

2.1. MARTE profile and its elements for model analysis: AnalysisContext

MARTE profile [1], standardized by OMG, facilitates validating and verifying temporal aspects through model analysis. MARTE analysis is intended to support accurate and trustworthy evaluations using formal quantitative analyses based on mathematical models, which may supplement designer intuition profile [1]. Quantitative analysis techniques determine the output values such as response times, deadline failures, resource utilizations, etc. based on data provided as input; e.g. execution demands or deadlines. System design models (application, platform and deployment models) must be modeled and extra annotations needed for analysis are to be attached to those design models (by the use of stereotypes, that map model elements into the semantics of an analysis domain, and tagged values) to be able to perform model based analysis using MARTE, rather than requiring a special version of design models to be created only for the analysis [1]. AnalysisContext is the main concern of MARTE to perform model analysis. It identifies models that gather information about system behavior and workload, execution platform and allocation (Y approach [13]) for the analysis and specifies global parameters (properties that describe different cases being considered for analysis). Therefore, stereotypes related to the AnalysisContext term are classified in two concepts:

- **WorkloadBehavior**: It is a container of a set of end-to-end system operations used for analysis and defined by a set of workload events triggered over time. These stereotypes are used in application models where constraints, scenarios and software design are specified, including functional and non-functional requirements.
- **ResourcesPlatform**: It is a container for the resources used by the system behavior represented by the design model. These stereotypes are used in platform and allocation models where resources and their properties are described and platform design is specified.

To conclude, AnalysisContext allows analyzing what it could be a real-time situation of the system by describing a specific scenario and the execution platform through analysis models with non-functional annotations.

2.2. Variability in AnalysisContext for model analysis in Embedded Software Product Lines

MARTE profile was defined for single systems modeling and analysis. So when model based analysis is performed for an embedded SPL, variability is a key aspect that must be considered: 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\(^1\) and so on. Therefore, analysis

\(^1\)Software Architecture Glossary, 2007: http://www.sei.cmu.edu/architecture/glossary.html

\(^2\)http://www.sei.cmu.edu/productlines/frame_report/index.html

Copyright © 2010 John Wiley & Sons, Ltd.

Prepared using smrauth.cls

DOI: 10.1002/smr

http://mc.manuscriptcentral.com/jsme
can vary from one product to another one. As a result, analysis process and AnalysisContext term must be extended to address SPL model analysis.

The wide variability issues involved in SPL context make V&V even more complicated and requires a proper management. SPL engineering paradigm separates development in two processes [14]: Domain Engineering, the process of SPL engineering in which the commonality and the variability of the product line are defined and realized; Application Engineering, the process of SPL engineering in which the applications of product line are built by reusing domain artifacts and exploiting the product line variability [15]. Thus, there are two architecture abstractions that must be considered also when carrying out model based analysis: 1) line architecture and 2) instance architecture; besides taking into account the development, realization and management of e.g., more workproducts, new tasks and multidisciplinary teams.

But two approaches can be applied when evaluating 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 [16].
- If the goal is to evaluate a specific product: That product is derived and then evaluated, which is performed in application engineering.

Any of the two evaluation approaches can be followed applying the process described on the next section. Specific product analysis models are obtained with the process, from a subset of the product line or specific product of the line.

Some fragments of an E-Commerce system SPL are shown in Figure 1, where new branches have been defined to support model analysis. This system has to run under different platforms and devices (Home Customers access to the system through PCs while Business Customers may also access by mobile) and functionalities may change from one to another (Home Customers have restricted information comparing with Business Customers). Even quality attributes or their degree are not the same for all products derived from the product line. On the first branch in (a), mandatory, optional and alternative functional features and quality attributes are specified, which describe the variability behavior of the AnalysisContext first concept (WorkloadBehavior). Each of the quality attributes has some critical scenarios to analyze, such as: the Performance quality attribute has critical scenarios like Browse Catalog Scenario or Status Follower Scenario. And features may require or exclude other features (by defined constraints). While Figure 1b facilitates information about platform and allocation variability that helps configuring a specific product model of the product line, describing diverse ways to allocate system threads and Figure 1c visualizes different analysis types for the critical scenarios and input/output analysis variables.

3. MODEL ANALYSIS PROCESS FOR EMBEDDED SPLS

MDE and SPLE are paradigms that have established their benefits for the development of software in general and embedded system software in particular [17, 14, 18]. However, these benefits come at the cost of increasing the complexity of the process followed to develop such software [6] and this complexity is even exacerbated when the software needs to comply with real-time constraints such as performance. As a result, the process to engineer these systems needs to cater for a diverse set of stakeholders (e.g., software, electronic or real-time engineers), an increased number of workproducts (e.g., feature models, system models, code, documentation, etc.) and new tasks (e.g., domain analysis or product derivation).

This entails that an already complex process needs to deal with the analysis-related tasks. Moreover, the analysis process itself needs to deal with a more diverse group workproducts required by MDE and SPLE than when engineering is performed on a traditional development process.

As a result, and in order to manage and perform model analysis for ensuring the quality of the products of the SPL, it is necessary to establish a process with the aim of tackling the above
complexity. Following established product line development practices [14], two separate processes were defined for model based analysis.

- **Model based Analysis Process in Domain Engineering**: This process sets the infrastructure to perform the analysis of the products of the product line. That is, it defines the *analysis core assets*.

- **Model based Analysis Process in Application Engineering**: Based on the above mentioned analysis core assets, this process specifies how they are applied to perform model analysis.

### 3.1. Model based Analysis Process in Domain Engineering

If the goal is to develop and analyze an embedded SPL, the core assets developed in Domain Engineering (e.g., models, as MDE is applied), must be prepared to perform analysis later on. This is the aim of the Model based Analysis Process in Domain Engineering.

This process takes the feature model and the design models, as input and outputs core assets that will serve as the building blocks for model analysis in Application Engineering. When SPL development must begin from scratch, these models will be developed at the required stage. Figure 2 depicts the process using *Software & Systems Process Engineering Meta-Model Specification (SPEM)* [19]. The following subsections describe its tasks in detail, the workproducts that are developed in them and the main roles involved in the model based analysis process in Domain Engineering. The different participating roles are central, as embedded systems are developed by multidisciplinary groups of stakeholders who contribute to the team with specific knowledge.

#### 3.1.1. Feature Model Elaboration

In the first task, a FODA based feature model is modeled [20]. This model provides a global vision of the variability of the line. Traditionally this feature model contains information about features that different products derived from the SPL must give response to. In the case of embedded systems this model also needs to be extended with the quality feature tree, emphasizing the importance of the quality attributes in these systems [16]. This branch gathers information about quality attributes and the desired quality that needs to be assured, as not all products of the line may require same quality attribute or degree and thus, it should be considered from the beginning of the process. It also contains constraints among functional and platform features that facilitate managing impossible configurations. These latest features refer to the possible platform resources and design patterns for the embedded software derived from the SPL. An example of a feature model with these characteristics can be seen in Figure 1. An excerpt of the E-Commerce SPL is modeled where some functional features and a quality feature tree...
L.B. OTHER

are described in a). For example, as it is specified, Performance and Schedulability attributes are optional in the quality feature tree and the functional mandatory feature called Business Type can be the Home Customer or Business Customer feature alternative.

On the other hand, when performing model-based analysis other variability issues apart from functional and quality attributes must be considered since the beginning such as allocation and analysis variability. As this information is not of the same abstraction level, it is worth defining and developing different branches in the feature model gathering the different required information.

"The process of specifying a family member may also be performed in stages, where each stage eliminates some configuration choices" [21]. These branches will be explained more thoroughly in the following tasks.

There are many tools that support feature modeling for SPLs such as: FMP [22], RequisitePro [23], FAMA [24], QFMP [16], etc. but not all of them are suitable for time-related model analysis purposes. The tool chosen to aid in this task must be able to handle:

- Temporal quality attributes: Quality attributes such as performance or schedulability.
- Constrains among features: require and exclude type relationships among features.
- Allocation and analysis variability required in later steps: possible software allocations for each platform and different analysis types with diverse variables to perform model analysis to contemplate all the wide variability of products that can be derived from the SPL.

None of the tools above satisfy all requirements completely. Hence, based on FODA and [16] a metamodel for feature models has been defined, which is described in the following subsection.

During Feature Model definition, domain engineers and domain experts are the ones responsible for the definition of the Feature Model where functional and quality attributes are described. Domain engineers together with software architects identify real-time and embedded systems critical scenarios to be analyzed. As a result of this task a feature model that gathers quality

attributes, functional and platform features besides the constraints among them is obtained that will serve as input for the following tasks.

3.1.2. Allocation Variability Elaboration. Allocation variability specification (Figure 1b) starts once the functional features and quality attributes have been defined. Therefore, allocation variability branch is described on the Feature Model by domain engineers with the help of domain experts (the role with the domain knowledge but often no software development expertise) and hardware experts, identifying and describing constraints among features. To be able to do that, aforementioned metamodel has been described. A metamodel is an abstraction of the model that defines the language for expressing a model*. The metamodel described in Figure 3 gathers information related to all variability information contained in the feature model required for model analysis (i.e., functional, platform, allocation and analysis variability). It considers functional, platform and implementation features besides constraints among features as defined by Kang et al. [20]; critical quality attributes and impacts among features previously extended by [16]; and allocation and analysis related features defined in this paper. Based on [16] this metamodel introduces some concepts to facilitate the use of feature models with the aim of model analysis. Two new concepts, i.e. AllocationNode and AnalysisNode, have been introduced. The AllocationNode concept is related to allocation variability (Figure 1b). It allows to identify a feature as allocation type making it possible to define a feature model branch with allocation variability features, where AllocationNode inherits FeatureNode and PlatformNode’s properties. Thus, threads identified as AllocationNode can be bound to platform resources due to PlatformNode. On the other hand, AnalysisNode helps to specify diverse analysis types for critical scenarios by the relationship with the Scenario metamodel and also input/output variables. These concepts are used when developing a feature model where analysis related variability is described (Figure 1c).

To summarize, the input feature model is modified adding an allocation variability branch (based on the metamodel specified above) which is the input model for the next task.

3.1.3. Analysis Variability Elaboration. In the same way allocation variability branch has been described, the analysis variability branch must be developed. It gathers information related to critical

*Defined by: http://www.omg.org/mof/
scenarios to be analyzed and analysis types for each quality attribute defined before. Later in this
model only those possible scenarios related to the selection made in the quality feature tree are
shown somehow suggesting the analyses that can be performed for the configured product model.
Taking as basis the metamodel (see Figure 3), different types of analysis and input/output variables
are specified and constraints are also defined in the Feature Model. Type node of the metamodel
gathers information about the type of analysis to be carried out on the models (e.g., sensitivity
analysis) while input/output variables (Variable, Input and Output nodes of the metamodel) make
possible to perform diverse analysis cases of a specific analysis type and combining both concepts,
a wide variety of analysis can be performed.
Some of these features have constraints that select or deselect other features of the feature model
(e.g., when Browse Catalog Scenario is selected in the quality feature tree for a configuration in
Figure 1a, Browse Catalog Scenario will be selected in the analysis variability branch (Figure 1c),
as a critical scenario to perform model analysis). This feature model is the output workproduct of
this task that will be take into account when annotating SPL design models.

3.1.4. Real-Time Specification. SPL design models (the set of models that represent the SPL:
 system’s structure, behavior, platform and deployment) must be modeled considering variability
 issues and then annotated with extra information that will later be used for analysis. Models must be
developed in a generic way, thus making them capable of representing common and variant aspects
of the products that make up the product line. For this purpose Gomaa [25] and MARTE profiles
[1], for variability and temporal information respectively, have been used; where these annotations
add temporal aspects information to the models using stereotypes and tagged values, for example:
hostDemand or speedFactor. When modeling and annotating SPL design models software engineers
and domain experts take into account the Feature Model defined before.

![Figure 4. Relationship Model Metamodel](image)

3.1.5. Transformation Definition. To be able to generate the analysis models automatically, the
mechanisms developed above must be connected between them assuring a traceability of the models.
Therefore, an analysis environment that gives response to this need has been developed. Having as
inputs the feature model and the annotated SPL design models, transformation specialists must
define the transformation rules for the Relationship Model. Variability in functionality, quality
attributes, platform devices and allocation, and relationships among all those variability issues make
analysis variable from one product of the product line to another one [26]. Considering the existence

Copyright © 2010 John Wiley & Sons, Ltd.
DOI: 10.1002/smr

http://mc.manuscriptcentral.com/jsme
of multiple concepts that are related to each other, a model to manage them all systematically is required. Moreover, and in an SPL setting, where these artifacts contain commonalities and variabilities that must be managed among all products of the product line. The Relationship Model specifies these relationships and helps managing them making possible to perform different AnalysisContexts.

The metamodel in Figure 4 was defined for this aim. It relates all the variability described in the feature models to themselves, through constraints among features from three branches. A set of desired features define a specific configuration and an instance of AnalysisContext is the desired result of this analysis environment (an analysis configuration or set of desired features for model analysis). Specifically, it is an analysis type of a critical scenario representing a specific behavior, a concrete platform model and how software is allocated on it (deployment) and the specific value instances for that analysis (analysis case).

The transformation specialists are the ones that take care about the analysis environment which makes possible the derivation to a specific product model analysis.

3.2. Model based Analysis Process in Application Engineering

Previously developed core assets in Domain Engineering are used in Application Engineering process to perform model analyses, prioritizing the critical scenarios of each specific product model of the SPL.

Desired features must be selected from the different branches of the feature model. In this way, the specific product model configuration is defined regarding the specified constraints. Meanwhile, the Relationship Model will be modified with each configuration decision taken in the branches deselecting all those possible choices that were not selected in the configuration feature set (i.e., in the features selected by the user for a specific product). This is possible through the analysis environment developed in the Domain Engineering process above. Thus, this model will give specific instances of AnalysisContexts using the transformation tool to get the specific product model derived from the SPL design models.

![Diagram of Model Based Analysis Process in Application Engineering](Figure 5. Model Based Analysis Process in Application Engineering)

To achieve the aim of model based analysis the following tasks must be realized where analysis core assets developed in Domain Engineering are used. Through the analysis environment developed in Domain Engineering, all mechanisms (Feature Model, annotated SPL design models and the Relationship Model) are connected to each other making possible to configure the specific analysis product model. Thus, once the configuration is performed, transformations would be applied automatically and the specific AnalysisContexts would be obtained (see Figure 5).

3.2.1. Feature Model Configuration. The main objective of this task is to select desired features related to functionality, platform and quality attributes. In this way, the configuration of a specific
product and the derivation of a specific product model analysis starts by application engineers using the reusable assets built in Domain Engineering that are managed by the model analysis environment. The set of selected features restricts possible choices from the allocation and analysis variability branches and limits the possible analyses to perform gather together in the Relationship Model. Those restrictions and limits are defined and handled by the constraints and transformations rules specified before, hence capitalizing on the previous work and increasing reuse.

3.2.2. Allocation Variability Configuration. The allocation variability input branch is the result of the selection made in the Feature Model functional and quality branches. It allows application engineers to select just the possible features for that particular Feature Model Instance (i.e., only the allocation variability available once the functional and quality variability have been bound are presented to the engineer). At this step, features related to software deployment into the platform are selected and the Relationship Model deselects those AnalysisContexts that do not satisfy the selected set of features (configuration) minimizing the wide range of AnalysisContexts to carry out.

3.2.3. Analysis Variability Configuration. And finally, analysis variability branch configuration must be done. Features related to the variability that can be found in analysis are selected in this task, resulting in an analysis variability instance. The Relationship Model considers this instance to choose the specific AnalysisContexts to be performed. Therefore, the transformations are applied to annotated design models to get the specific product analysis models where specific values are bound to input analysis variables that are suggested by the Relationship Model. Later, depending on the analysis tool used, MARTE analysis models must be transformed to its input model using an existing bridge tool or if not developed before, defining one (see Table I). Analysis experts can study the results obtained after performing model analysis and use them as feedback for the software product line model, considering if the desired quality attributes are assured.

3.3. Summary

A model based analysis process for embedded SPLs has been described in this paper where MDE and SPL mechanisms have been applied to assure quality attributes like performance. There are a wide variety of tools that facilitate software modeling but not all of them support MARTE profile annotations. Therefore, the available tools, how they can give support and where (on what tasks) has been studied. The main issue for this study (Table I) has been to analyze how these tools can be combined to assure the traceability of MARTE analysis models, as different tools for different purposes are required during the process.

Although some tools that appear above do not support MARTE, they are worth mentioning for their close relation with the profile, such as ArgoUML or Rational Rose that support SPT (MARTE predecessor) [27] or MAST [28] profiles respectively. Some tools like Papyrus come from academy while others like Rational Software Architect or MagicDraw are commercial tools from industry.

Other research that must be mentioned is the one done by Anssi et al. in [29] where they mention that the tool support for AUTOSAR* and MARTE still need to be improved further for an efficient use for industrial development. This work intends to be a step on such direction.

The concrete points that have been considered for the study are:

- Requirements: If there is a tool for feature modeling or requirements specification.
- Modeling: A modeling tool and the annotation profiles that are supported (including MARTE).
- Transformation: The model to model transformation mechanisms that are supported for product derivation and if there is a transformation bridge available for MARTE analysis model to analysis tool input model.
- Analysis/Simulation Tool: Performance and schedulability model analysis tools.
- References: References for tools, research works and case studies.

*Another environment used for developing software: http://www.autosar.org/
To sum up what is described on the Table I not all modeling tools that support MARTE profile guarantee a traceability of the design models with tools that exist nowadays. Some improvements are required to facilitate model based analysis with existing analysis tools. Although transformations can be applied from MARTE analysis models into analysis tools (e.g., Cheddar or MAST) input model, both metamodels need to be understand to specify the transformation rules. And thus, develop bridge tools that make possible the automatic transformation. An open source tool chain solution to build an analysis environment could be composed by FMP and Papyrus for modeling the SPL, MARTE profile to add temporal information, ATL transformations to derive to a specific product model and the analysis bridge for MAST scheduling analysis tool developed by the University of Cantabria.

4. RELATED WORK

The capability to be able to perform model analysis to validate and verify quality attributes variability at early phases facilitates obtaining a product with the same functionality but different quality levels. Quality variability must be another issue taken into account during embedded software development and will affect the decision-making. And in the same manner, variability modeling is important for managing variability in software product families [39]. Different approaches have been proposed related to variability modeling and management, but those techniques rely on different technical background and most variability modeling techniques lack a description of a process*.

*http://dissertations.ub.rug.nl/faculties/science/2008/m.sinnema/
Over the last years, several variability modeling techniques have been developed that are aimed to support variability management. Despite the fact that these mechanisms are suitable for managing variability and feature models are widely used in embedded systems domains [40] [41] [42], no standard way is defined yet (CVL is a request proposal [43]) and few works cover other phases like software V&V.

Research works like the one done in MeMVaTEx or [44] do not considered variability, although temporal aspects are annotated with MARTE. MeMVaTEx methodology presented in [42] proposes the decomposition of design process in different abstract levels of EAST-ADL2 framework. For each level, requirements and solution models are created in a separate way and the interrelations among the elements of these models are specified through traceability mechanism of the SysML profile while real-time issues are specified by MARTE profile. The methodology proposed focuses on requirements traceability from analysis to implementation phase, taking into account temporal issues and regardless the variability and V&V of them. Espinoza [44] 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. It has been defined for a single product model analysis. Thus some modifications are needed before applying it in embedded SPL. In [45] a model-based methodology oriented to distributed embedded and real-time applications development is proposed, that focuses on the requirements traceability management while variability and verification phase is left for future work. And Anssi et al. make a deep study about MARTE and AUTOSAR for enabling timing analysis for automotive domain in [29].

Works presented by Tawhid and Petriu [9] and [12] are the closest ones to our work as SPL and MARTE are considered. In [9] they propose an SPL modeling approach, with functional variability and annotated with MARTE profile for performance in a general way (using variables). In order to validate quality aspects, concrete values are assigned to general annotations through ATL transformations that also are used to obtain a specific product model. But variability management is slightly defined. While the research has been improved in [12] specifying each task, not all variability issues (e.g., allocation variability) have been taken into account when carrying out model analysis.

5. CONCLUSION AND FUTURE TRENDS

In this paper a model based analysis process for embedded SPLs have been proposed focusing on quality attributes. Throughout the process variability issues like e.g., allocation or analysis have been taken into account. This process is defined to supply the lack that exists in embedded SPLs model based analysis. In order to ensure all the requirements even quality attributes required for each specific product of the product line are met, model based analysis techniques can be applied.

An analysis environment has been developed which includes the process and binds the required elements to carry out the analyses. Profiles like Gomaa and MARTE for variability modeling and quality attributes annotation respectively have been used in models. And as analysis configuration might be performed in different stages, with different knowledge, etc. a staged configuration mechanism composed by allocation and analysis variability branches at the feature model besides the traditional ones, have been included in the process (enriching the feature model metamodel). As a consequence, features related to diverse abstraction levels can be managed facilitating the analysis configuration specifying more details in each stage with each decision-making. A Relationship Model has been also used (and its metamodel defined). It is a mechanism that gathers and manages relationships among all variability issues (functional, quality attributes, allocation, platform devices and analysis) that take part in embedded SPL development and model analysis. This model helps obtaining the specific AnalysisContexts for each specific product configuration of the embedded SPL. It facilitates prioritizing critical scenarios for specific configurations, helps binding concrete values to the input/output variables and gives feature traceability.

An study of the existing tools and how can help in different stages of the product configuration and derivation and how they can be combined and/or extended to assure models traceability has
been made. With that aim, tools that come from diverse domains but support MARTE profile have been taken into account.

The future work to be carried out includes the realization of another real case study where aforementioned variabilities are considered and properly managed and the developed analysis environment is applied in order to perform model analysis. In the same way, it could help checking the scalability of the proposal and identifying possible conflicts.

We also complement our proposal studying whether existing proposals or mechanisms could help in allocation variability and can be introduced in the proposed process. And we are working on developing an analysis environment tool that facilitates automatic model based analysis integrating a development and analysis tool chain.

ACKNOWLEDGEMENTS

This work is co-supported by the Spanish Ministry of Education, and the European Social Fund under contract TIN2011-23839 (Scriptongue) and Basque Government under grants UE2011-4 (COMODE Project). The project has been developed by the embedded system group supported by the Department of Education, Universities and Research of the Basque Government.

REFERENCES


32. Tribastone M, Gilmore S. Automatic Extraction of PEPA Performance Models from UML Activity Diagrams Annotated with the MARTE Profile. WOSP ’08: 7th International Workshop on Software and Performance, ACM.


AUTHORS’ BIOGRAPHIES

Leire Belatge was born in Mondragon, Spain in 1981. She received the M.Sc. degree in Computer Science Engineering from the University of Mondragon, in 2006. She is currently with the department of Electronics and Computer Sciences at the University of Mondragon, working towards the Ph.D. degree in the area of software engineering in topics like MARTE model based analysis and embedded SPLs.

Goizua Sagardui obtained the PhD degree in Computer Sciences from the University of the Basque Country (Spain) in 1999. From 2000 she is working as lecturer/researcher in the department of Computing and Electronics in Mondragon University and she is the coordinator of the software engineering group of the University of Mondragon. Her main research area is software engineering, including topics such as Software Product Lines and Model Driven Engineering.

Leire Etxeberria obtained the PhD degree in Computer Science from Mondragon University (Spain) in 2008. Previously she finished the BSc in Computer Science Engineering at Mondragon University in 2004. She is nowadays working as senior researcher in the department of Computing and Electronics in Mondragon University.

MaidorAzanza obtained the Ph.D degree in Computer Sciences from the University of the Basque Country in 2011. She is currently a lecturer at the Computer Languages and Systems Department of the same university, where she teaches Software Engineering courses. She is part of the Onexin Research Group and her research interest include Model Driven Engineering, Software Product Line Engineering, and the challenges their combination poses.

Copyright © 2010 John Wiley & Sons, Ltd.

Prepared using smrauth.cls

DOI: 10.1002/smr

http://mc.manuscriptcentral.com/jsme