TOWARD PRESERVATION OF EXTRA-FUNCTIONAL
PROPERTIES FOR MODEL-DRIVEN COMPONENT-BASED
SOFTWARE ENGINEERING OF EMBEDDED SYSTEMS

Federico Ciccozzi

2012
Abstract

Model-driven and component-based software engineering have been widely recognized as promising paradigms for the development of a wide range of systems. Moreover, in the embedded real-time domain, their combination is believed to be helpful in handling the ever-increasing complexity of such systems. However, in order for these paradigms and their combination to definitely break through at an industrial level for the development of embedded real-time systems, both functional and extra-functional properties need to be addressed.

This research focuses on the preservation of extra-functional properties; more specifically, the aim is to provide support for easing such preservation throughout the entire development process across different abstraction levels. The main outcome of the research work is a round-trip engineering approach aiding the preservation of extra-functional properties by providing code generators, supporting monitoring and analysis of code execution, and then enabling back-propagation of monitoring and analysis results to modelling level. In this way, properties that can only be roughly estimated statically are evaluated against runtime values and this consequently allows to optimize the design models for ensuring preservation of analysed extra-functional properties.

Moreover, a solution for managing evolution of computational context in which extra-functional properties are defined by means of validity analysis is provided. Such solution introduces a new language for the description of the computational context in which a given property is provided and/or computed, enables detection of changes performed to the context description, and analyses the possible impacts on the extra-functional property values when the context is changed.
Sammanfattning

Denna licentiatavhandling handlar om hantering av ickefunktionella systemegenskaper i modelldriven och komponentbaserad utveckling av mjukvara. Målet är att bevara systemets egenskaper genom de olika stadierna av utvecklingen; även när modeller översätts mellan olika språk och till kod. Vi är övertygade om att en sådan teknik är nyckeln till att bevara systemets ickefunktionella egenskaper och att den i förlängningen kommer att öka den industriella acceptansen för modelldriven och komponentbaserad utveckling av mjukvara.

I den här avhandlingen utforskar vi möjligheterna att studera ickefunktionella egenskaper genom att skapa en kedja i tre steg som: (i) genererar kod från systemmodellen, (ii) exekverar och analyserar den genererade koden och (iii) slutligen återkopplar analysvärden till systemmodellen. Introduktionen av steg (ii) och (iii) gör det möjligt att genomföra en detaljerad analys av egenskaper som är svåra eller till och med omöjliga att studera med hjälp av endast modeller. Fördelen med det här tillvägagångssättet är att det förenklar för mjukvaruutvecklaren att slipper att arbeta direkt med kod för att ändra systemegenskaper och optimera systemet. Istället kan utvecklaren fokusera helt och hållet på systemmodellerna och de ickefunktionella egenskaper som ska optimeras.

Vi föreslår dessutom en lösning på hur man kan hantera ickefunktionella egenskaper i ett system när det genomgår förändringar och uppraderingar. Vår lösning gör det möjligt att på automatisk väg upptäcka i vilka fall en komponents egenskaper kommer att påverkas av en förändring - och i vilka fall som egenskaperna kommer att bevaras. Detta förenklar, i sin tur, arbetet för mjukvaruutvecklaren som slipper att utvärdera samtliga komponenters alla egenskaper i samband med varje förändring av systemet.
Acknowledgements

There are many people to thank for making my Ph.D. studies and this licentiate thesis possible. I would like to start with my family that always believed in me and supported, even economically, my decision to move to Sweden for continuing my studies; without them I would not have been able to follow my dreams and achieve this result.

A special thanks goes to my main supervisor Mikael Sjödin and assistant supervisor Antonio Cicchetti that have continuously supported me, my curiosity and talkativeness, helping and driving me for achieving very satisfactory results. I would like to thank all my colleagues at IDT for all the moments, both fun and constructive, we spent together sure that many more are to come in the next years.

Being able to stand me and my vim, I would like to thank my office room mate Mehrdad. Many friends, both in Italy and in Sweden, have believed in me and made easier to move abroad for fulfilling my dreams; a special thanks goes to all of them. A final special thanks for standing by my side, believing in me and sharing her everyday life with me goes to my girlfriend Julia.

I dedicate this licentiate thesis to my uncle Alido who is always in my thoughts and would be surely proud of it.

Federico Ciccozzi
Västerås, 19 January, 2011
List of Publications

Papers Included in the Licentiate Thesis\(^1\)


**Paper B** *Toward a Round-Trip Support for Model-Driven Engineering of Embedded Systems*, Federico Ciccozzi, Antonio Cicchetti, Mikael Sjödin, EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Oulu, Finland, August, 2011.


\(^1\)The included articles are reformatted to comply with the licentiate thesis printing format
Related Publications not Included in the Licentiate Thesis


On the concurrent Versioning of Metamodels and Models: Challenges and possible Solutions, Antonio Cicchetti, Federico Cicozzi, Thomas Leveque, Alfonso Pierantonio, International Workshop on Model Comparison in Practice (IWCMP), Zurich, Switzerland, June, 2011.


Other Publications

Integrating Wireless Systems into Process Industry and Business Management, Federico Cicozzi, Antonio Cicchetti, Tiberiu Sceleanu, Johan Kerberg, Jerker Delsing, Lars Eric Carlsson, International Conference...
ence on Emerging Technology and Factory Automation (ETFA), Bilbao, Spain, September, 2010.

To my uncle Alido
# Contents

## I Thesis

1 Introduction
   1.1 Background .............................................. 4
   1.2 Contribution ............................................. 6
   1.3 Thesis Outline .......................................... 7

2 Research Summary
   2.1 Research Goal ............................................ 9
   2.2 Research Questions ....................................... 9
   2.3 Research Methodology ...................................... 11
   2.4 Research Contribution ..................................... 12
   2.5 Papers Contribution ....................................... 16

3 Conclusions
   3.1 Future Work .............................................. 22

Bibliography .................................................. 23

## II Included Papers

4 Paper A:
   Evolution Management of Extra-Functional Properties in Component-Based Embedded Systems 27
   4.1 Introduction .............................................. 29
   4.2 Motivation & Related Works ............................. 30
   4.3 A Case-Study: The ACC System ............................ 32
      4.3.1 The ProCom Component Model ....................... 33
xiv  Contents

4.3.2  The Attribute Framework ................................. 34
4.4  Proposed Approach ........................................ 36
4.5  Extending the Attribute Framework with Context Modeling 39
  4.5.1  Computation Context ................................. 39
  4.5.2  Computation Context Language: Expressing the Va-
        lidity Conditions .................................. 41
4.6  Modification Impact Analysis .............................. 43
  4.6.1  Difference Computation ................................ 43
  4.6.2  Validity Evaluation .................................. 50
4.7  Approach Validation ........................................ 51
4.8  Conclusions .................................................. 52
4.9  Acknowledgements .......................................... 52
Bibliography ...................................................... 53

5  Paper B:  Toward a Round-Trip Support for Model-Driven Engineering of
  Embedded Systems ............................................. 55
  5.1  Introduction ............................................... 57
  5.2  Background ................................................ 58
    5.2.1  Motivation ........................................... 58
    5.2.2  Related Works ....................................... 59
    5.2.3  Contribution ......................................... 60
  5.3  Proposed Approach ........................................ 61
    5.3.1  Code Generation and Traceability .................... 62
    5.3.2  Extra-Functional Properties Evaluation ............. 63
    5.3.3  Back-annotation to Source Models .................... 63
  5.4  Implementing the Round-Trip Support ..................... 65
    5.4.1  Back-annotation Meta-Model ........................ 65
    5.4.2  Code Generation and Traceability Management ....... 66
    5.4.3  Monitoring Results Management ....................... 67
    5.4.4  Back-annotating the Source Model .................... 68
  5.5  A Case-study: the ACC system ............................ 71
    5.5.1  From the ACC System Model to Generated Code and
           Back-annotation Model ........................... 72
    5.5.2  Code Execution Monitoring and Back-annotation to the
           ACC System Model .................................. 74
  5.6  Conclusion .................................................. 74
Bibliography ...................................................... 77
6 Paper C:  
Generation of Correct-by-Construction Code from Design Models for Embedded Systems  
6.1 Introduction ................................................. 83  
6.2 Related Work .............................................. 84  
6.3 The Proposed Solution ...................................... 85  
6.3.1 Validation .............................................. 89  
6.4 Summary .................................................. 89  
Bibliography .................................................. 91  

7 Paper D:  
Supporting Extra-Functional Properties Preservation in Model-Driven Engineering of Embedded Systems  
7.1 Introduction ................................................. 97  
7.2 Background .................................................. 98  
7.3 A Generic Round-trip Support for Preservation of Extra-Functional Properties .............................................. 101  
7.3.1 Code Generation and Traceability ......................... 103  
7.3.2 Extra-Functional Properties Evaluation ................... 106  
7.4 A Case-study: the AAL2 subsystem ............................ 107  
7.4.1 The CHESS Framework .................................... 108  
7.4.2 Modelling the AAL2 Subsystem ............................ 109  
7.5 Code Generation Process from Source Models .............. 113  
7.5.1 Executable Code Generation ................................ 113  
7.5.2 Traceability Links and Back-propagation Model ........ 115  
7.6 Code Execution and Back-propagation to Source Models .... 118  
7.6.1 Monitoring Results Management .......................... 118  
7.6.2 Performing the Back-propagation ......................... 121  
7.7 Discussion .................................................. 123  
7.8 Conclusion .................................................. 125  
7.9 Future Work ................................................ 126  
Bibliography .................................................. 131
I

Thesis
Chapter 1

Introduction

The thesis presents the research work carried out to help management of extra-
functional properties (EFPs) during the development of embedded real-time
systems. The desire would be to exactly preserve EFPs of a model when it
is transformed between different abstraction levels (e.g., model-to-model and
model-to-code). However, for some EFPs\(^1\) it is impossible, or impractical, to
validate such preservation before an actual runnable system has been gener-
ated.

Therefore we propose a new step in the development process which serves
to back-propagate EFPs that are determined from the generated code to the
model and then their preservation is evaluated. In addition, we propose a new
technique to determine when changes to a system can impact an EFP and to
what extent the EFP is impacted. The purpose of this new technique is to, e.g.,
allow impact analysis before modifying a system and to identify which EFPs
need to be re-evaluated after a system modification.

As the base for our work we use the increasingly popular development tech-
niques Model-Driven Engineering (MDE) and Component-Based Software En-
gineering (CBSE). MDE and CBSE aim at mitigating software-development
complexity by tackling different issues with dedicated solutions. Moreover,
their combination has been recognized as very promising and identified as an
enabler for them to definitely break through for industrial development of em-

\(^1\)This would be the case, e.g., for the EFP "worst-case execution time" for a code-block to be
executed on a modern hardware platform with accelerator features like caches, branch-predictors,
and translation lookaside buffers (TLBs). This time cannot be tightly bounded until machine in-
structions and memory allocation have been determined.
bedded systems; that is the reason for which we embrace such combination for achieving our goals. While, functionally speaking, both paradigms are already able to offer diverse and effective capabilities for modelling and evaluating the system under development, when coming to the extra-functional perspective there is still a gap to be filled for them to be widely adopted.

To a certain extent, one could argue that MDE is highly suitable for management of EFPs, since it promotes specification, modelling, and analysis of them. However, in practice very little work has been done to ensure preservation of EFPs during model transformations in MDE, especially when dealing with code generation; to the best of our knowledge, no work has previously used the concept of back-propagation to evaluate EFP-preservation from generated code.

CBSE is still struggling with EFPs management [1] too. In this case such management entails a two-fold issue: (i) how to express the properties (in which language and where), (ii) how to express the rules to compute them. Furthermore, a well-established pattern of concepts and mechanisms for automating the discovering of EFPs validity when their computational context\(^2\) evolves is missing.

The outcome of this thesis is composed by two approaches:

- Support for easing preservation of EFPs throughout the entire development process by adding the further step, after code generation, of back-propagation of monitored results from code to model level;

- Support for managing evolution of EFPs' computational context through proper context description, detection of context evolution, and analysis of possible impacts on the involved EFPs.

In the remainder of this chapter background notions, motivations that led us to the research problem, goals formulation, as well as contribution and outline of the thesis are described.

1.1 Background

Design of embedded software systems requires support capable of properly dealing with their ever-increasing complexity. MDE and CBSE can be considered as two orthogonal ways of reducing development complexity: the former

\(^2\)For computational context is meant a set of validity expressions that must hold in order for the EFP's value to be valid.
shifts the focus of application development from source code to models in order to bring system reasoning closer to domain-specific concepts; the latter breaks down the set of desired features and their intricacy into smaller sub-modules, called components, from which the application can be built-up and incrementally enhanced [2].

Embracing this premise an integrated Model-Driven and Component-Based (MDCB) process could help in handling embedded systems intricacy and thus reducing development effort and risks by: (i) enabling efficient modelling of EFPs such as, for instance, safety, reliability, availability and dependability, and (ii) providing automation for the management of modelled EFPs throughout the development process [3]. In the last six years, such integration has been covered by a large number of works [4] and recognized as extremely promising; tools and frameworks have been developed for supporting such kind of integrated development process [5].

According to the MDE vision, a system is developed by refining models starting from higher and moving to lower levels of abstraction until code is generated; refinements are achieved by means of transformations between models. A model transformation converts a source model to a target model while preserving their conformance to the respective meta-models [6]. Moreover, as a model is an abstraction of the real system, rules and constraints for building the model have to be properly stated through a corresponding language definition. In this respect, a meta-model describes the set of available concepts and well-formedness rules a correct model must conform to [7].

One of the major challenges of MDE is to provide automated code generation to be executed on specific target platforms ensuring that EFPs of the system modelled at abstract levels are preserved at code level for achieving a correct-by-construction product. Preservation of EFPs throughout the development process entails the ability to generate code by ensuring that values\(^3\) expressed at system specification level and modelled at modelling level are actually matching the ones observed at runtime executing the generated code. In the remainder of the thesis as well as in the included papers, generated code is defined as \textit{correct-by-construction} if it behaves as expected according to the functional and extra-functional definitions given in the source models.

Conflicting to MDE’s goal is the fact that the code generation is too often considered only as the conclusive step of an MDE approach [8]. Particularly in the embedded domain, resources limitation stresses the criticality of having EFPs evaluation at code level: for instance, when dealing with resources

\(^3\)While EFP means the extra-functional property, EFP’s value represents the actual value, expected and/or observed, associated to the EFP.
consumption or execution time, it is generally only possible to provide rough estimates of values boundaries and derive statistical behaviours of their variations at model level. Hence, many of these properties are best analysed at modelling level by using results gathered at code level. Moreover, in order to preserve EFPs and achieve correctness-by-construction, the generated code should not be manually edited after its generation. Instead, the prescribed manner to perform modifications is to edit the source models until generated code conforms with both desired behaviour and EFPs.

Proper description, verification and preservation of EFPs throughout the development process can help in reducing final product verification and validation effort and risks by providing generated code which is correct-by-construction [9], [10]. Notably, many EFPs are cross-cutting the functional definition of the application, making compositionality of analysis results for each component to hold only under particular conditions [1]. Moreover, each type of EFP demands for corresponding theories precisely defining how such a property can be specified, the kind of values it can assume, the manipulations it can undergo, and the impact a particular class of changes can have on it. Such information are critically relevant when considering evolution and maintenance activities, since changes performed on the current version of the system should allow preservation of its EFPs.

Assuming that the whole modelled software system in terms of components is considered to be at system level, while a subsystem represents a non-empty subset of the components and each of the operations defined for them are considered as function level, in this research work we take into consideration two sets of EFPs specified at these different levels of granularity:

- System level: throughput, mean time between failures (MTBF) and overall CPU usage;
- Subsystem and Function level: execution time and memory allocation.

1.2 Contribution

In this thesis we aim at providing possible solutions for the challenges described in the previous section by means of two approaches:

1. An automated support for the preservation of EFPs throughout an entire MDCB development of embedded systems. Specific contributions are code generation as well as back-propagation of monitoring results
to modelling level for evaluation of EFPs. Moreover, the code generation is meant to facilitate monitoring activities by performing controlled code injections and to enable back-propagation capabilities by maintaining explicit traceability and consistency between source models and generated code.

2. A support for managing evolution of EFPs' computational context by means of validity analysis. That is achieved by: (i) introducing a new language for the description of the computational context in which a given property is provided and/or computed, (ii) detecting model changes whenever the context evolves, and (iii) analysing the possible impacts on the EFPs' values based on differences representation between previous and current versions of the modelled context.

The research work has been carried out to target a MDCB development process. More specifically, while CBSE contributes with design patterns and mechanisms for modelling the system, MDE provides, through model transformations, the mechanisms for enabling generation of code from models and back-propagation of monitoring values.

1.3 Thesis Outline

This thesis consists of two parts. The rest of the first part is organised as follows: Chapter 2 describes a summary of the conducted research in terms of a description of focus, goal, questions, methodology and contribution. Thesis conclusion and identified possible future work are outlined in Chapter 3. The second part of the thesis consists of Chapters 4 through 7 which describes the research contribution in details by means of research publications.
Chapter 2

Research Summary

In this chapter a research summary is given in terms of the identified research goals, research-driving questions, methodology followed for pursuing the goals and outcome of the research work in terms of actual contribution.

2.1 Research Goal

The goal of this research work is two-fold:

1. Aiding MDCB development of embedded systems in achieving the generation of code that fulfils both functional and extra-functional specifications given at modelling level by providing mechanisms for enabling preservation of EFPs from modelling to code level.

2. Providing detection of computational context evolution by detecting modifications at modelling level and providing impact analysis on the involved EFPs.

2.2 Research Questions

The following research questions have been formulated and considered as main drivers for the research work:
Question 1. How can specific facilities in an MDCB approach facilitate discovery of computational context evolutions and thereby evaluation of their impact on related EFPs at modelling level?

Firstly, a language for the description of computational context at modelling level needs to be defined. Then, whenever such context evolves, the model changes are detected and, based on a precise representation of differences between old and new context model versions, the impact caused on the involved EFPs' values is analysed. In this way, it is possible to provide a development environment with a validation feature, able to detect computational context evolution and deal with its impact on EFPs values validity as early as possible in the system lifecycle.

Question 2. How can we provide specific support for an MDCB approach to enable preservation of EFPs throughout the entire development process?

The focus of our approach is to provide a round-trip support for achieving a required level of quality preservation in terms of EFPs from the modelling artefacts to the generated code. In the embedded domain, resources limitation sharpens the criticality of having EFPs evaluation at code level. As a consequence, some of the information and values needed for the developers to perform a thorough extra-functional evaluation of the modelled system cannot be gathered at modelling level before code is generated. Hence, a round-trip support for: (i) generating instrumented code from source models, (ii) performing monitoring of its execution and (iii) finally providing back-propagation of monitoring results to the source models, aids the MDBC process in achieving preservation of EFPs.

From question 2 two sub-questions are formulated:

Question 2a. How is the traceability information created and manipulated for back-propagation activities within the round-trip approach?

Giving monitoring results as source for the back-propagation transformations is not enough. In fact, the traceability chain defined and maintained along the path from design models to monitoring results is also part of the source artefacts to be fed to the transformations in order to correctly back-propagate results to the corresponding model elements. Depending on the decisions taken when defining traceability, actions to manipulate it and feed it to the transfor-
mations will be needed. As well as for the monitoring results, tracing information is maintained using the structures provided as part of a back-propagation meta-model.

**Question 2b.** *Which solutions can be adopted in the generation of target code and in the monitoring of its execution in order to enable back-propagation of monitoring results to modelling level?*

According to the MDE vision, a system is developed by refining models starting from higher and moving to lower levels of abstraction until code is generated; refinement is implemented by transformations over models. In this case, the code is meant to be automatically generated and executed in a way that enables back-propagation capabilities by maintaining explicit traceability and consistency between source models, generated code and monitoring results. The crucial step is the enrichment of source models with information concerning computation of EFPs from code execution monitoring activities. Such enrichment should be performed by propagating the property values computed at code level back to the related model elements for completing the extra-functional specification of the design models. The effort to be put in such injection depends on the modelling language's capabilities in modelling EFPs.

### 2.3 Research Methodology

This research work has been carried out by following the deductive method depicted in Figure 2.1 and composed by the following steps:

1. Identification of software engineering hot topics in the fields of both MDE and CBSE.

2. Research definition, consisting in the following iterative process: (i) definition of the research problem, (ii) systematic literature review for rounding off the problem, and (iii) definition of research-driving questions.

3. Achieving and presenting the research results. This task is performed by iteratively run the following steps: observation, analysis, evaluation and refinement of the research results.

4. Validation of the research results by implementing a prototype and testing it against a set of case-studies; multiple iterations among this and the previous phases were needed too.
2.4 Research Contribution

The contribution of this research is two-fold:

1. Provide an automated round-trip support for MDCB development that is able to generate **full**\(^1\) code, which fulfills the functional specification

\(^1\)For **full** is meant code that can be executed directly after its generation without any manual manipulation.
given at modelling level, and to provide apposite back-propagation facilities from code to source models for enabling evaluation of EFPs at modelling level and thereby validate EFPs preservation.

2. Propose a solution for managing computational context evolutions and impacts on involved EFPs at modelling level.

The goal of the round-trip support is to provide code generation, traceability and back-propagation features for an MDCB process to achieve preservation of those EFPs which cannot be accurately predicted at modelling level, but rather measured at code level. By adhering to the MDE vision, code for the platform taken into account is generated from the source models; then, by monitoring the execution, registered values are automatically propagated back to modelling level. The aims are multiple: at an initial development stage, estimated values can be iteratively refined through executions of the generated system and corresponding monitoring and back-propagation activities; in turn, design models can be evaluated and possibly refined to achieve EFPs preservation and hence correctness-by-construction [11] at code level.

In Figure 2.2 the round-trip support is depicted. Once design modelling tasks have been successfully completed, the objective is to enable automatic generation of code from source models. Taking design models as source artefacts\(^2\), we generate target code through an appropriate set of model-to-model and model-to-text transformations (Figure 2.2a).

Information regarding tracing of source (e.g. model elements) and target (e.g. code segments) artefacts has to be defined and maintained for further back-propagation activities. Therefore, code generating transformations have to be properly defined by encoding apposite rules for the generation of traceability links (i.e. explicit traceability [12]) between models and code (Figure 2.2b); such rules populate the back-propagation model with traceability information.

Once the code has been generated as well as the traceability links, quality attributes of the system can be evaluated by selected code execution monitoring and analysis tools (Figure 2.2c). Depending on the capabilities of such tools and their output format, different actions varying from text-to-model to model-to-model transformations (Figure 2.2d) are required to extract and formalize execution results that would complete the information needed for back-propagation activities.

\(^2\)In the remainder of the thesis as well as in the included papers source and design models refer to the artefacts at the highest level of abstraction as shown in Figure 2.2
Figure 2.2: Round-trip support for EFPs preservation.

The last step of the round-trip approach aims at finally enriching the source models with the monitoring and analysis results (Figure 2.2e) through dedicated in-place [6] model-to-model transformations. The details regarding the round-trip support are given in:

- **Paper B**: the approach is described considering code generation and traceability links creation as black-box; a preliminary validation is performed against a small-sized case-study on a specific framework. Focus is given to feasibility and usefulness of the round-trip support;

- **Paper C**: the code generation process is sketched and presented as a set of transformations (e.g. model-to-model/model-to-text) acting on diverse intermediate artefacts for generating code from source models;
• **Paper D:** the entire approach is given in terms of: (i) generators able to produce 100% of target code from source models, (ii) traceability information generation and maintenance, (iii) monitoring of code execution and (iv) back-propagation of monitoring results to the source models. A further validation is performed on a more complete and extensive industrial case-study on top of a different modelling framework in order to assess scalability and reusability of the approach.

![Diagram](image)

Figure 2.3: Management of Context Evolution for EFPs.

Concerning the impact analysis of computational context evolutions on related EFPs, the solution aims at synthesizing evolution side effects as early as possible by providing the developer with a validation tool showing the impact of the ongoing adaptation to the current model, hence before the changes are stored. The approach is based on the specification of component attributes as context
sensitive, that is, each property value is specified together with a set of pre-
conditions that must hold in order to keep it valid. In this way, whenever a
component is put in a different computational context it is possible to check
if the values associated to its EFPs can be considered still reliable and hence
usable to update the overall system properties.

Taking the hypothesis that multiple values are available for EFPs, the goal
is to provide checking of attribute values validity when the system evolves ac-
cording to selected scenarios. It provides the possibility to model and store the
context of each EFP, while the evolution of the system is tracked by means of
effects to the context (Figure 2.3b). In particular, two or more context mod-
els representing the embedded system in different points in time (Figure 2.3a
and Figure 2.3c), are given as input to the differences calculation engine (Fig-
ure 2.3d) which outputs the detected manipulations through an appropriate dif-
ferences representation structure (Figure 2.3e). Starting from this result, the
technique is able to check the validity of existing property values and in case of
invalid values (Figure 2.3f), to produce estimations of possible impacts (Fig-
ure 2.3g). The approach described in this section is completely unwound in its
details in Paper A.

2.5 Papers Contribution

In Table 2.1 the contribution of the papers in relation to the research questions
is depicted.

<table>
<thead>
<tr>
<th>Paper</th>
<th>Q1</th>
<th>Q2</th>
<th>Q2a</th>
<th>Q2b</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>D</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Table 2.1: Contribution of the papers in relation to the research questions.

Included Papers

**Paper A.** Evolution Management of Extra-Functional Properties in Compon-
ent-Based Embedded Systems, Antonio Cicchetti, Federico Ciccozzi, Thomas
Leveque, Sverine Sentilles, Proceedings of the 14th International ACM SIGSOFT Symposium on Component Based Software Engineering (CBSE), Boulder, Colorado (USA), June, 2011.

Summary: This paper presents concepts and mechanisms that allow to automatically discover whether an EFP's value is still valid when related computational context evolves. In particular, we first introduce a new language for the description of the computational context and then, whenever such context is modified we detect the model changes and, based on a precise representation of differences between old and new context model versions we analyse the impact caused on the related EFPs. In this way, it is possible to provide a development environment with a validation feature, able to detect evolution issues as early as possible in the system lifecycle. The impact analysis is based on EFPs given values and definition of their computational context, hence assuming that such information is correct and thereby leaving aside any reasoning about how such values are computed. The approach has been plugged in in an existing framework, namely ProCom, that enables the definition of quality attributes for components.

My Contribution: This paper was written with equal contribution from all the authors. My main responsibility entailed differencing calculation, representation and management; that was achieved by means of ad-hoc meta-models, models and model-to-model transformations.


Summary: In this paper the round-trip support for back-propagation of monitoring/analysis results from code to model level is introduced. While code generation facilities and traceability links generation were considered as black-box, the back-propagation activities as well as the description of designed structures for maintaining trace links and values to be back-propagated are described in detail. The proposed approach is decomposed in three fundamental issues, namely how to store trace links between source models and generated code, how to retrieve useful information from monitored code execution results, and how to propagate the registered values back to the source models; for each of them, challenges and possible solutions, together with their implemen-
tation, are discussed. A preliminary validation of the approach is performed on top of the ProCom platform.

**My Contribution:** I was the main driver and principal author of this paper as well as the sole developer of the given implementation for the proposed solution.


**Abstract:** In this paper we provide a solution for the problem of automatically generating target platform code from source models focusing on producing code artefacts that facilitate monitoring activities and enable back-propagation of values from code to model level. Challenges and solutions are described together with a proposal of the intended implementation to be developed. This work lays the foundations for the development of the code generators and transformations for creation of explicit traceability that were left aside in Paper B.

**My Contribution:** I was the main driver and principal author of this paper.


**Summary:** After the preliminary validation against a system designed on top of the ProCom platform given in Paper B, the round-trip support has been enriched with its own C++ code and trace links generators (partially following the anticipation given in Paper C) and validated at a larger scale against a more complete and extensive case-study on top of a different framework, CHESS [13]. From the design models, created using the CHESS modelling language and following the CHESS methodology, C++ code and trace links in terms of a back-propagation model are automatically generated. The execution of the code is then monitored and selected EFPS (e.g. execution time and allocated memory) measured at operation level; apposite transformations are defined to perform injection of such values into the back-propagation model.
Such model is then taken as input by an apposite model-to-model transformation for propagating the computed property values back into the related model elements' placeholders at modelling level. Once an iteration of the process is completed, the developers have at their proposal a set of enriched source models from which it is possible to extract precious information about the modelled system in terms of monitored quality attributes. Finally the models can be optimized and the process re-iterated until generated code achieves desired preservation of selected EFPs.

This paper extends Paper B and implements the ideas given in Paper C in order to provide the complete round-trip support.

**My Contribution:** I was the main driver and principal author of this paper as well as main developer of the whole round-trip support.
Chapter 3

Conclusions

In this research work we provide a support for easing evaluation of preservation of EFPs throughout the entire development process, that is to say among modelling artefacts at same levels of abstraction as well as between source models and generated code. Concretely we provided two approaches that can either cooperate in a single development process or be singularly adopted by different processes.

The first approach is meant to operate across different abstraction levels for achieving extra-functional preservation. In fact, for MDE-CBSE combination to be trusted, EFPs should be preserved not only at modelling level but even among different levels of abstraction, namely from source models to generated code. For this purpose we propose a full round-trip support able, from source models, to generate full code, to monitor selected EFPs at code level, and finally propagate measured values back to modelling level.

In order to evaluate preservation of EFPs at modelling level, we provided a support for managing evolution of computational context in which EFPs’ values are computed by means of validity analysis. A new language for the description of the computational context is introduced as well as a mechanism for detecting model changes whenever the context evolves; finally analysis of the possible impacts on the EFPs’ values based on a precise representation of differences between previous and current model versions is performed for easing early evaluation of extra-functional validity.

Precision and expressive power of the adopted modelling patterns, especially regarding the way to express EFPs, have shown to be very important for achieving and providing automation of full code generation as well as back-
Chapter 3. Conclusions

propagation features. Using the modelling approach proposed in the CHESS methodology, we have been able to start from very complete modelling artefacts from which full code could be generated. Nevertheless, the code generation process has resulted more complex than expected. Due to diverse issues coming from both the modelling language's expressiveness and the aim to provide a generation approach easily adaptable to different target languages, the process has been broken down in several sub-steps involving a number of intermediate artefacts appositely defined for it.

Moreover, being able to back-propagate monitoring and analysis results from code to modelling level implies the ability to correctly trace such information among different levels of abstraction. Therefore, apposite structures and mechanisms have been defined for maintaining traceability among related artefacts at different levels of abstraction and thereby allowing correct back-propagation. Also for back-propagation features, the aim of providing an approach which could be easily adaptable to different execution platforms, monitoring and analysis tools as well as modelling languages, led us to the definition of a multi-step approach involving intermediate artefacts appositely defined for this purpose.

3.1 Future Work

Future work should entail a complete integration of the two approaches under the same development process for evaluating if their single positive impact on the process would persist or even increase by their combination. Another future research direction is to encompass the extension of the round-trip support for enabling management and evaluation of properties like safety and security, which usually drastically differ from properties measurable by means of computed values. Moreover, new generation EFPs, such as energy-related properties, should be taken into consideration for future evaluations and possible extensions of the provided round-trip support. A very interesting direction would certainly be to focus on the adaptation of the round-trip support to be applied to the numerous embedded systems entailing heterogeneous platforms. Concerning the code generation process, interesting paths could be engaged concerning the provision of features for end-user customization of the code generation process by means of input parameters.
Bibliography


