Towards Supporting Evolution of Service-Oriented Architectures through Quality Impact Prediction

Steffen Becker, Mircea Trifu
FZI Forschungszentrum Informatik Karlsruhe
Haid-und-Neu-Straße 10-14, 76131 Karlsruhe, Germany
sbecker|mtrifu@fzi.de

Ralf Reussner
Chair of Software Design and Quality (SDQ)
Am Fassanengarten 5, University of Karlsruhe (TH), 76131 Karlsruhe, Germany
reussner@ipd.uka.de

Abstract

The difficulty in evolving service-oriented architectures with extra-functional requirements seriously hinders the spread of this paradigm in critical application domains. This work tries to offset this disadvantage by introducing a design-time quality impact prediction and trade-off analysis method, which allows software engineers to predict the extra-functional consequences of alternative design decisions and select the optimal architecture without costly prototyping.

1 Introduction

The evolution of service-oriented architectures in critical application domains such as production control systems, enterprise software, or telecommunications is a very costly process. This is mainly due to the fact that software systems within these application domains have to achieve certain quality of service levels, which today can only be verified through costly trial and error testing, when the architectural changes are already implemented.

Although there are some methods which can predict performance or reliability of software architectures, they are not specifically designed to take into account software evolution scenarios like changing or adding components or services in the system. Existing methods completely ignore maintainability aspects and it is difficult to analyse trade-offs between multiple quality attributes. There are also some approaches for architecture trade-off analysis like SAAM, but they involve a lot of manual work and rely mainly on the personal experience of the software engineers involved.

The Q-ImPrESS approach presented in this paper supports design-time software evolution by enabling software engineers to analyse different architectural changes, predict the consequences of their design decisions on multiple quality attributes (performance, reliability, and maintainability) of the software system and perform a trade-off analysis between these quality attributes. The trade-off analysis is used to rank design alternatives based on multiple criteria and allow the software engineer to select the optimal design decisions before even writing a single line of code. The proposed approach works on a service architecture model and supports various change scenarios both at the level of service internals and the level of service composition.

The paper is structured as follows. Section 2 gives an overview of the competitive state of the art, while section 3 focuses on the related work this paper builds upon. Section 4 focuses on the scientific challenges targeted by Q-ImPrESS. The approach taken to solve the introduced issues is presented in section 5. Finally, the paper concludes in section 6.

2 Related Work

The Q-ImPrESS method aims at supporting trade-off analyses for extra-functional properties of evolving service-oriented architectures. Hence, related work stems from the areas of architecture trade-off analysis, evaluation of multiple quality attributes of software entities, and the prediction...
of extra-functional properties of software systems.

A large body of literature exists for early, architecture based trade-off analysis. Some of the most often applied methods are the Software Architecture Analysis Method (SAAM) [10] and the Architecture Trade-off Analysis Method (ATAM) [6]. Both rely on the guided definition of architecture change scenarios by the software architecture’s stake holders. The methods use the defined scenarios to identify potential design risks.

In the past, authors proposed several approaches to evaluate the utility of software entities (e.g., a component or an architecture). One of the first approaches in this area is the OTSO method [11] which aims at selecting software components which best fit the user’s needs. The method was refined later, for example, by the PORE method presented in [12].

In the area of software performance and reliability prediction, authors often used annotated UML models as input and derived quality metrics from generated prediction models (e.g., [14]). Most often, the UML profile for schedulability, performance and timing [13] has been applied in these approaches. However, performing trade-off analyses was left to the user of the method.

3 Foundations

The Q-ImPrESS method aims at using the prediction capabilities of several existing meta-models. In so doing, it uses mature prediction approaches and builds its own trade-off analysis on top of these. In Q-ImPrESS it is planned to create model instances for the following meta-models described in detail in this section: KLAPER, PalladioCM, ProgressCM, and SOFA. Additionally, it aims at integrating tools for architecture reverse engineering and an enhanced prediction model for concurrent CPU cache and file system access.

3.1 Component Models

KLAPER [8] is an intermediate model for performance and reliability predictions. Its aim is to decrease the effort necessary to write a transformation from an arbitrary design model like UML to different prediction models such as a queuing networks for performance or markov chains for reliability. Recent enhancements to the model also allow basic performability analyses [7]. KLAPER provides a small ECore-based meta-model with a minimal set of entities necessary for creating transformations into performance and reliability analysis models.

The Palladio Component Model (PCM) [1, 2] is a metamodel for the specification of component-based software architectures. Instances of the PCM contain annotations needed for the prediction of a given software architecture’s performance. An interesting feature of the PCM is that it allows to create parametrised component models with respect to component external service calls, hardware environment, and component input parameters. The PCM has an extensive ECore-based meta-model and includes mature tool support based on the Eclipse platform.

The Progress component model (ProCom) [4] is a component model supporting the engineering process of embedded systems mainly in the vehicular, automation and telecommunication domains. Its focus lies in the inclusion of quality aspects like the embedded system’s safety or reliability. An ECore-based meta-model is currently prepared.

The SOFA (Software Appliances) component model [15] is directed at checking the interaction protocols between software components. For this, each component specifies the allowed call sequences to its provided interfaces. Additionally, for each provided service it contains a specification of its interaction with its required components. SOFA has an ECore-based meta-model and its tool set contains a runtime environment capable of managing and executing components.

3.2 Reverse Engineering of Component Models

The SISSy tool chain uses static analysis techniques to reverse engineer an abstract programming language independent model of the code [16]. The model is basically an abstract semantic graph (ASG), which contains besides the usual OO structural constructs (classes and methods), abstract behaviour and resolved references.

The ArchiRec [5] tool uses a third party generated ASG, similar to the one produced by SISSy, to identify components and component architectures by means of an interactive hierarchical clustering approach.

Java2PCM [9] and JAbstractor are two similar tools for extracting abstractions of the behavior inside components from existing source code. The extracted abstractions, although semantically close, are specifically tailored for different types of analyses such as performance prediction and protocol checking.

3.3 Resource Contention Models

Research results from experiments with component performance [3] should be used for Q-ImPrESS to increase the accuracy of performance and reliability predictions. The adapted prediction methods aim at incorporating concurrent accesses to the CPU’s second level cache and the operating system’s implementation of the file system layer into the predicted quality metrics.
4 Scientific Challenges

Q-ImPrESS addresses several scientific challenges. The following list gives an overview on the major ones.

- **Software Evolution**: Redesigning existing systems into service-oriented architectures is often a need to keep systems flexible, adaptable, and maintainable. However, this software evolution during design time still faces unsolved issues. First, existing systems exhibit known extra-functional properties, e.g. for reliability or performance. When changing these systems, it has to be ensured by the system architect that the systems maintain these properties or even improve some of them. For this, a systematic evaluation of a number of potential evolution scenarios, i.e., different design alternatives, with focus on all important quality attributes is needed. To the best of our knowledge, a unified and automated approach for this is not available today. Second, the prediction of quality attributes involving existing systems requires models of these existing systems. However, these systems are huge with respect to their complexity and code size. Semi-automated abstraction of models of the code of these systems is needed to come up with solvable models. Existing methods and tools often need further development to deal with this type of complexity.

- **Trade-off Analysis for Multiple Quality Attributes**: Q-ImPrESS aims at evaluating the impact of system design changes on different quality attributes. This allows the software architect to validate that different extra-functional requirements of the evolving system hold after the change. Additionally, it allows the selection of the best alternative with respect to different quality attributes. An approach which combines multiple prediction methods for different quality attributes in a single comprehensive method is still missing. In particular this involves the definition of a meta-model able to store sufficient information to generate instances of the different prediction models. Such a unified meta-model would give insights in commonalities and differences of the various approaches. After the project, this meta-model can serve as basis for standardization activities to overcome the drawback of current UML and profile based approaches.

- **Usage Dependent Predictions**: The usage of a system can often not be foreseen by the software architect. This is especially true for service-oriented architectures where the customers requesting services are often unknown to and independent from the service provider. Because of this, prediction methods are needed which allow a parameterization with workload and input parameter information. Q-ImPrESS aims at including such usage dependencies in all its predictions.

- **Large Scale Model-Driven Quality Predictions**: While model-driven prediction methods have been presented for the last few years, model transformation languages, meta-modelling environments, or editor generators are still in their maturation phase. Q-ImPrESS tries to apply these techniques on a broad scale. The gained experiences will also help to further identify needed research in the area of model-driven methods and tools.

5 Approach

To realise the envisioned trade-off analysis between quality attributes for evolving service-oriented architectures in Q-ImPrESS the method needs to include several steps (see Figure 1 for an overview). First, an already existing software architecture which is subject to evolutionary changes has to be reverse engineered into appropriate models. For this, static analysis is used to extract the static structure from the source code (components, interfaces, connectors). The gathered information is then enhanced with measurements taken on existing and running systems using dynamic analysis. These measurements are needed in order to capture an abstraction of the system’s behavior with respect to the considered quality attributes. Because this abstraction is in essence an approximation of the behavior, the resulting models may also need some manual corrections or additions to compensate the loss of precision. This activity is supported by model editors, generated using MDD techniques.

The reverse engineered information is stored in a model called the Service Architecture Model, which is an instance of a newly created meta-model. This meta-model attempts to unify common aspects in the previous described metamodels used later for quality predictions. The meta-model consists of several packages each of them dealing with a certain aspect of the system’s design. There are packages for the static structure, behaviour, system resources (i.e., hard- and software layer), and usage of the system. While a common understanding of the static structure of a service-oriented architecture and the underlying hardware and software resources is likely to be defined, a unifying behaviour model is still missing. This is partly due to the fact that different prediction techniques use different behavior models. Since one of the key requirements for the meta-model defined in Q-ImPrESS is to support several different prediction techniques, each with its strength and weaknesses, this meta-model contains an extensible behaviour package, covering at least process algebra expressions and UML-like activity specifications.
Newly developed transformations then transform instances of the Q-ImPrESS meta-model into the specific meta-models described in Section 3 (Klaper, PalladioCM, ProgressCM, and Sofa). Each of these meta-models uses in turn its own tool-support to make predictions on their supported quality attributes (e.g., mean response time, response time distribution, mean time to failure, ...) and the results are collected and fed back into the Service Architecture Model. They are then used in the trade-off analysis to support the user in selecting the optimal design decisions for the given evolution scenario.

6 Conclusions

This paper presents an initial overview of the envisioned Q-ImPrESS method for making trade-off analyses for evolving service-oriented software architectures. The method relies on a common meta-model which serves as a reference meta-model for transformations. These transformations reverse engineer on the one hand existing source code and measurements taken on existing systems into instances of the Q-ImPrESS meta-model while on the other hand they generate the instances of the various prediction models to estimate the quality attributes used in the trade-off analysis.

Software architects will benefit from the presented method such that they can design the evolving system in a way which complies with the system’s extra-functional requirements like performance or reliability but also maintainability. The envisioned trade-off analysis enables a rational decision for a specific target architecture based on cost/utility ratios.

However, some limitations can be identified already. On the one hand, there are some limitations in the underlying prediction methods like limited support for stateful components or the abstraction from memory accesses. On the other hand, not all metrics are supported. For example, currently there is no worst-case execution time prediction method available in the target meta-models.

Future work is directed at an initial release of the Q-
ImPrESS meta-model, the corresponding transformations, and the realisation of the reverse engineering tool suite.

References


