Product Certification of Component Performance Specifications

Henning Groenda
FZI Forschungszentrum Informatik
Haid-und-Neu-Str. 10-14
76131 Karsruhe
groenda@fzi.de

Abstract: In software engineering, performance analyses and predictions play an important role in the selection of components and the evolution of complex component-based systems. These analyses and predictions are based on parameterized performance specifications. However, the quality of the specifications and their trustworthiness usually remain unspecified. In existing approaches, it remains unclear if a specification can be reused in another context and which effect its use may have on the quality of the analysis or prediction. In this paper, we propose a test-based approach to validate parameterized performance specifications against deployed component implementations. The validation is used to certify the quality and valid parameter ranges of the specifications.

1 Introduction

Performance analyses and predictions play an important role in each life cycle phase of a component-based software system. Their application in the development phase enables software engineers to improve architecture and design by comparing design alternative based on objective measures. In the deployment phase, performance specifications guide the selection and sizing of an appropriate execution environment as well as the deployment of components within this environment. In the maintenance or evolution phase, analyses allow to examine performance bottlenecks or the effect of modifications on the performance and allow guiding maintenance effort.

All of these performance analyses are based on specifications of the component's performance-relevant behavior. The specifications are always abstractions of the performance-relevant behavior to allow faster analyses than by executing the software itself. The degree of parameterization of these specifications depends on the supported scenarios. Example scenarios are changes in the usage profile, assembly or deployment of components. Due to the complexity of these scenarios the specifications are usually only valid for a certain combination of parameter ranges. However, the quality of the specifications for such a combination is usually not stated. Having such statements enables identifying potential risks and reasoning about the overall quality of the analysis or prediction.

Existing performance prediction approaches rely on the capability of software engineers

to select suitable performance specifications of a component. Research focused on validating prediction approaches with the assumption that the specifications were suitable for the situation at hand. Validating performance specifications was a human activity in the selection process. Missing validity statements require substantial effort to reason about the suitability or quality of an performance specification. Especially if specifications should be reused in different context, for example because of late composition, effort has to be put in revalidation or maybe even recreation. This process step requires expert knowledge and is time-consuming which reduces the usefulness of specifications of unknown quality.

In this paper, we propose an approach to certify performance specifications and explicitly state the quality and limitation of these specifications. It is based on a test-based validation of the specifications against deployed component implementations. The validation is trustworthy as it is based on objective and verifiable information. This enables third party validation and eases testing the validity. Additionally, it aids in the protection of interests and trade secrets in marketplaces as it is sufficient to publish the certified specifications.

The paper is structured as follows. Section II shows what needs to be covered to reason about the quality of performance specifications. Section III sketches the certification process. Section IV point out and discusses related work. Section V concludes the paper.

2 The Quality of Performance Specifications

Validation of specifications is often made based on test suites. For example, the functional validation if an application server fulfills the Java platform enterprise edition requirements. Each test in such a test suite provides a simple pass or fail outcome by comparing expected and experienced results.

However, if performance is considered the comparison of expected and experienced results is much more difficult. Performance is for example influenced by input parameters, internal state, the performance of other required services, resource contention, and the deployment platform. Depending on the used measurement method and granularity, performance measurements often influence the performance itself. Measuring wall-clock time is error prone due to interruptions in multi-threading systems, whereas profiling has a heavy performance impact itself. See [KKR09] for details on measurement methods.

Performance specifications consists of connected resource demands which may depend on parameters. They can be as easy as a single value (e.g. mean) covering huge code areas but also as complex as an executable specification (e.g. byte code). The presented approach hence focuses on the following three aspects to ensure the overall specification quality:

Input Range The input range specifies the parameter ranges of parameterized specifications for which the specification is valid. For example, if a specification has the parameter *file size* and it is validated for file sizes between 3 and 50 MB this is stated as input range.

Resource Demand Resource demands are measured in the number of experienced byte code instructions. This metric is not influenced by the measurement itself and can, in combination with Kuperberg's approach [Kup09], be validated for any given hardware and

software environment. For each considered byte code instruction, a *Validation Estimator* expresses the maximal accepted deviance from the specified value.

3 Certification of Performance Specifications

Our certification approach implements a test-based validation of performance specifications against deployed component implementation. The certification process is explained in the following paragraphs.

At the beginning, the component creator issues a validation request to the certification authority. He has to provide the component specification, the corresponding implementation, and the quality statements including the parameter range combinations which should be used for validation. Validity statements contain the information discussed in chapter 2.

An evaluator within the certification authority then assesses the validity of the specification. Execution of the following steps is necessary to decide upon issuing a certificate.

Generate Testcases In this step, test cases are generated automatically which are used to evaluate the specification. Automatic generation is chosen to ensure reproducibility of validation results and reduce the necessary human effort. Stratified random sampling is used to generate parameters. If the initial seed is stored, the results can be reproduced but unwanted implementation optimization on test values is still hindered. Overall, the samples cover the specified input range for each parameter.

Both directions, specification against executed implementation and vice versa, are checked by the test cases. The forward direction checks if the statements in the specification are correct, for example that a resource demand depends linearly on an input parameter. The backward direction checks that the implementation does not contain more dependencies than the ones specified, for example unspecified calls to external services, another sequence of calls, or unspecified dependencies to return values of calls.

Instrument Implementation The implementation is instrumented to log the CPU resource demands in form of issued byte code instructions. These demands are correlated to the specified demands.

Deploy Implementation The instrumented implementation and mock-ups for required services are deployed in the target environment.

Run Testcases The testcases derived in the first step are run on the deployed implementation and measurements of the issued CPU resource demands are gathered. A test case is run until the requested sample size is reached, which can be either specified directly or being calculated by a confidence level, e.g. if confidence interval of the sample mean is below the deviation threshold. The coverage of the test cases on the implementation allows to reason about the quality of the test.

The presented approach currently has the following limitations and assumptions:

- The approach currently considers only the validation of resource demands for processors. Hard disk or network requests are not validated yet.

- It focuses on the performance needs of business information systems. It does not consider guarantees, for example by a schedulability analysis, which are important in many embedded systems. The methodology is intended to reason about common and not best or worst case execution times.
- In a former prototype, resource demand specifications are only validated for fixed resource demands which are measured execution times and not byte code instructions. It is described with a small case study in [Gro09].
- Checks that the implementation does not contain more dependencies than specified are currently not implemented.
- Data dependencies of components are currently not validated yet. However, the support for parameters of the used performance specifications allows taking these into account later on.
- Plain java objects are considered. Code weaving which is for example used in Java Enterprise Edition application servers is not considered yet.

4 Related Work

The research on component certification started in the early 90s and is still ongoing as shown in Alvaro et al.'s survey [AAM05]. The survey shows the history of component certification and that certification approaches developed in the 90s focus on statements about the reliability of software components using test cases or mathematically analyzable models. Starting around 2000, the focus of the approaches shifted towards the certification of extra-functional aspects in general and the prediction of systems built out of certified components, e.g. [SW01, HMSW03].

Wallnau also did some basic work on classifying certification approaches, for example the 10 useful distinctions for certification approaches in [Wal04]. According to this classification the approach presented in this paper aims at reducing the gap between the knowledge about what a component does to what it actually does. It supports a *descriptive* certification of *objective* measures. The software *products* will be examined *empirically* with a given *context* in a *procedural* manner.

Hissam, Moreno, Wallnau et al. introduced the concept of predictable assemblies in [HMSW03] and later extended it to the *Predictable Assembly from Certifiable Components (PACC)* approach, described in [Wal03]. PACC allows the prediction of the runtime behavior of a software system from the properties of its components and their patterns of interactions. PACC's performance reasoning framework currently focuses on fixed-priority preemptive scheduling, making it suitable to analyze hard real-time systems [MM08]. In contrast to the approach proposed in this paper, the authors concentrate on the support for hard real-time systems instead of business information systems. Additionally, they focus on the small areas and conditions in which some correctness properties can be proven.

Alvaro et al. show in [ALC07] the need of component certification within component-based software development for business information systems and discuss similarities and interdependencies between component selection and certification. The authors also provide a framework for component selection [AAM07b] as well as an selection process [AAM07a]. The approach proposed in this paper focuses solely on the extra-functional aspect performance and could later on be integrated this framework.

The effort of testing and reusing components was addressed by Weyuker in [Wey98]. She states that high reliability and availability requirements lead to enormous costs. Additionally, components have to be tested in isolation and after integration so savings of components-of-the-shelf are not sure. Reusing components requires retesting for stability, reliability, stress, and performance testing.

The approach presented in this paper will allow testing components for specified range combinations. This allows reasoning if a specification is suitable for the planned analysis.

5 Conclusion

In this paper, we showed how the combination of Input Range, Resource Demand, and Validation Estimator is used to reason about the quality of performance specifications. We presented our approach and explained the process of assessment and certification as well as listed current limitations and assumptions.

The presented approach aids companies in offering components in marketplaces. The publication of certified performance specifications in a marketplace is sufficient for potential customers to evaluate and select components using prediction approache like the PCM [1]. However, the interests and intellectual properties of the offering companies are still protected as the specifications only contain a highly abstract view on the component's behavior and keep the disclosure of details on the used algorithms and techniques to a minimum. Having certified performance specifications additionally supports software engineers in late composition of components. The software engineers gain the knowledge if the performance specifications fulfill their requirements for the intended composition which can in turn ease the evaluation of components and reduce the necessary effort. The certification of specifications also supports a predictable assembly of components. The information contained in these specifications allows increasing confidence in the results produced by validated performance prediction approaches or identifying potential risks. Last but not least, certification by independent authorities provides a mean for quality assurance. If the development of components is given to a contractor the stipulation of certification enables the contracting body to trust the performance of a developed component beyond a few test cases while keeping its own quality assurance effort low. The contracting body can focus on its own expertise and does not need to employ performance engineers just for quality assurance.

As a next step the measurement of byte code instructions instead of execution times is

[1] http://www.palladio-approach.net

planned. This allows the certification independent of the target environment. The execution times can be calculated based on benchmarking the instructions in the target environment. It is also planned to advance considered parameters to data dependencies like randomness which influence resource demand during compression. It is further planned to examine the necessary effort in terms of test cases and runs for the requested validation.

References

[AAM05] Alexandre Alvaro, Eduardo Almeida, and Silvio Meira. Software component certification: a survey. *SEAA*, pages 106–113, September 2005.

[AAM07a] Alexandre Alvaro, Eduardo Almeida, and Silvio Meira. Component Quality Assurance: Towards a Software Component Certification Process. *IRI 2007*, pages 134–139, August 2007.

[AAM07b] Alexandre Alvaro, Eduardo Almeida, and Silvio Meira. Towards a Software Component Certification Framework. *QSIC 2007*, pages 298–303, October 2007.

[ALC07] Alexandre Alvaro, Rikard Land, and Ivica Crnkovic. Software Component Evaluation: A Theoretical Study on Component Selection and Certification. Technical Report MDH-MRTC-217/2007-1-SE, Mälardalen University, November 2007.

[Gro09] Henning Groenda. Certification of Software Component Performance Specifications. In *WCOP 2009*, pages 13–21, 2009.

[HMSW03] Scott Hissam, Gabriel Moreno, Judith Stafford, and Kurt Wallnau. Enabling predictable assembly. *J. Syst. Softw.*, 65(3):185–198, 2003.

[KKR09] Michael Kuperberg, Martin Krogmann, and Ralf Reussner. TimerMeter: Quantifying Properties of Software Timers for System Analysis. In *QEST'09*, pages 85–94, September 2009.

[Kup09] Michael Kuperberg. FOBIC: A Platform-Independent Performance Metric based on Dynamic Java Bytecode Counts. In Felix C. Freiling, Irene Eusgeld, and Ralf Reussner, editors, *Dependability Metrics Research Workshop*, pages 7–11. Department of Computer Science, University of Mannheim, May 2009.

[MM08] Gabriel A. Moreno and Paulo Merson. Model-Driven Performance Analysis. In Steffen Becker, Frantisek Plasil, and Ralf Reussner, editors, *QoSA 2008*, volume 5281 of *LNCS*, pages 135–152. Springer, September 2008.

[SW01] J. Stafford and K. Wallnau. Is Third Party Certification Necessary? In *4th ICSE Workshop on Component-Based Software Engineering*, 2001.

[Wal03] Kurt C. Wallnau. Volume III: A Technology for Predictable Assembly from Certifiable Components (PACC). Technical Report CMU/SEI-2003-TR-009, SEI, CMU, 2003.

[Wal04] Kurt C. Wallnau. Software Component Certification: 10 Useful Distinctions. Technical Report CMU/SEI-2004-TN-031, SEI, CMU, September 2004.

[Wey98] Elaine J. Weyuker. Testing Component-Based Software: A Cautionary Tale. *Software, IEEE*, 15(5):54–59, Sep/Oct 1998.