Beagle

Software Requirements Specification

Annika Berger, Joshua Gleitze, Roman Langrehr,
Christoph Michelbach, Ansgar Spiegler, Michael Vogt

at the Department of Informatics
Institute for Program Structures and Data Organization (IPD)

Reviewer:
Second reviewer:
Advisor:

–

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Contents

Reference notation

This document uses a fixed notation for all of its contents, making them referenceable:

/C#/	mandatory criterion
/OC#/	optional criterion
/B#/	purpose boundary
/A#/	application attribute
/G#/	target group
/E#/	software environment attribute
/F#/	mandatory functional requirement
/OF#/	optional functional requirement
/Q#/	mandatory non functional requirement
/OQ#/	optional non functional requirement
/T#/	test case
/M#/	model

1 Purpose and Goals

When developing software, specifying its architecture in a sophisticated way is a crucial, yet challenging task. Decisions made at this point often highly influence software's properties, such as maintainability and performance. Palladio is a software enabling developers to analyse performance and other such metrics of component-based software at the definition phase, before actually writing any code. To achieve that, the software is meta-modelled by the Palladio Component Model (PCM).

If no code exists yet, models are created completely manually. But in many scenarios some to all source code has already been written. In such cases however, analysis with Palladio might still be wished. SoMoX is a tool for static code analysis of source code, to re-engineer the software's architecture into a PCM. Unfortunately, SoMoX' results are limited, as they do not contain resource demands which are essential for performance analysis.

The purpose of this project is to analyse software dynamically by conducting performance measurements on its source code, in order to determine the resource demands of its component's internal actions. Adding this information to the software's PCM will enable developers to use Palladio to analyse and optimise their existing software with minimal effort.

1.1 Must Have Criteria

/C10/ Beagle shall enable the user to analyse given source code for its internal actions' resource demands.

/C20/ Beagle shall automatically annotate its resource demand findings in a given instance of the software PCM.

1.2 Nice to Have Criteria

/OC10/ Beagle should determine the approximate probability in SEFF conditions, which are annotated in the PCM, for each case. If the probability depends on the input parameters, Beagle should determine this dependency.

/OC20/ The branch probability as a function of the input parameters should be annotated in the PCM.

/OC30/ Beagle should determine in SEFF loops, which are annotated in the PCM, too, how often the loop body is executed approximately.

/OC40/ The number of loop executions as a function of the input parameters should be annotated in the PCM.

1.3 Boundary

/B10/ Beagle does not perform actual measurements on source code, as this is done by other software and transferred through the Common Trace API (CTA).

/B20/ Beagle does not reconstruct a model of softwares' architecture from their source code, as this is done by other tools like SoMoX.

/B30/ Beagle does not reconstruct the internal structure of components like their service effect specification (SEFF), as this is done by other tools like SoMoX.

/B40/ Beagle only analyses Java programs.

/B50/ Beagle does no performance analysis or prediction, as this is done with Palladio.

2 Application

2.1 Application Field

/A10/ Re-engineering of existing software.

/A20/ Prototyping

/A30/ Software development. Early implementations of components can be analysed with Beagle in order to predict their performance in interaction with the software system.

/A40/ Verifying of design and implementation

2.2 Target Group

/G10/ Software architects will use Beagle predominantly for /A10/ and /A40/.

/G20/ System deployers will use Beagle predominantly for /A40/.

/G30/ Component developers will use Beagle predominantly for /A20/ and /A30/.

2.3 Operation Conditions

/A100/ Desktop System (with Eclipse) with JRE

/A110/ Server (optional), add possibility to run on Server as additional function

3 Environment

/E10/ Beagle requires Java and Eclipse to run.

/E20/ Beagle requires a PCM instance modelling the software to be analysed. The model must contain all components and their SEFFs.

/E30/ Beagle requires the CTA to communicate with performance measurement software.

4 Functional Requirements

Given a software's source code together with a valid PCM instance correctly describing the software's components, including their SEFF, Beagle must fulfil the following requirements:

4.1 Measurement

/F10/ Using the information provided in the PCM, Beagle determines the sections in the source code to be measured in order to find internal actions' resource demands.

/F20/ Beagle conducts one or multiple measurement softwares to measure the sections found in /F10/.

/F30/ Beagle uses existing measurement software to conduct performance tests on the source code.

/F40/ Beagle supports the CTA to communicate with measurement software

/F50/ Beagle does not modify the provided source code files.

/OF10/ Beagle approximately determines relations between components' interface parameters and their resource demands.

/OF20/ Beagle determines the probability for each case to be taken in encountered SEFF conditions.

/OF30/ Beagle determines the probability for each case to be taken in encountered SEFF conditions depending on the component's interface parameters.

/OF40/ Beagle determines the probable number of repeats in encountered SEFF loops.

/OF50/ Beagle determines the probable number of repeats in encountered SEFF loops depending on the component's interface parameters.

/OF60/ The user may choose whether Beagle will analyse the whole source code or only parts of it.

/OF70/ The user may choose to re-test the source code or parts of it, in order to either gain more precision or to reflect source code changes.

4.2 Result Annotation

/F100/ Beagles stores all its results in the software's PCM. (Hereafter to be called "Beagle's result PCM instance").

/F110/ Beagle's result PCM instance is a valid PCM.

/F120/ As far as technically possible, Beagle's results can be read from Beagle's result PCM instance by a Palladio installation without Beagle.

/F130/ Beagle's result PCM instance contains all contained components' internal actions' resource demands.

/F140/ Any information found in the PCM instance provided to Beagle will be found in Beagle's result PCM instance.

/OF100/ Beagle annotates resource demands in its result PCM instance dependent on the component's interface's parameters using the PCM's expression language for resource demands.

/OF110/ Beagle's result PCM instance contains all contained SEFF conditions' estimated branch probability.

/OF120/ Beagle's result PCM instance contains all contained SEFF conditions' estimated branch probability dependent on the containing component's interface's parameters.

/OF130/ Beagle's result PCM instance contains all contained SEFF loops' estimated branch probability.

/OF140/ Beagle's result PCM instance contains all contained SEFF loops' estimated branch probability dependent on the containing component's interface's parameters.

todo Parameterisation of results

todo Analyse source code for its architecture and performance in one turn (e.g. with SoMoX & Kieker).

5 Non Functional Requirements

5.1 Dependencies

/Q10/ In order to use Beagle, the user is not required to have any software installed but Java, Eclipse, Palladio and a measurement software supported by Beagle.

/Q20/ Beagle does not depend on any specific measurement software.

/Q30/ Beagle does not require its input artifacts to be generated by a specific software.

/Q40/ Beagle definite a maximum time to run in dependencey of the measurement software.

/Q50/ In every component to be measured Beagle set a few measurement points.

/OQ10/ Beagle can be used on every combination of operating system and hardware platform Eclipse and Palladio runs on.

/OQ20/ The measurements runs without user interaction.

/OQ30/ The measurements shall also possible over the network e.g. on a Server.

/OQ40/ During the measurements Beagle saves the results of measurement to a hard disk.

/OQ50/ The user has the possibility to pause the measurements. As well the user can continue a measurement from save.

/OQ60/ If it is requested Beagle shut down the PC after it finished working.

/OQ70/ Beagle reports progress of measurements in kind of a progress bar.

/OQ80/ Beagle is robust against software errors e.g. exceptions.

5.2 User Interface and Experience

/Q100/ Beagle is implemented as an Eclipse plugin. As both Palladio and its extensions are Eclipse plugins, this ensures good usability for users.

/Q110/ Beagle can be controlled by context sensitive menus in Eclipse.

/OQ100/ Beagle is integrated in SoMoX, such that it is automatically executed after SoMoX has finished.

/OQ110/ Beagle can obtain its input artifacts from SoMoX, such that the user does not need to provide further information after SoMoX was started. If Beagle requires more information than SoMoX provides, the user can already submit it while configuring SoMoX.

6 Test Cases

7 Models

Terms and Definitions

Common Trace API an API developed by NovaTec GmbH for measuring the time, specific code sections need to be executed. . 2

component an artifact of a software development process with a description of its application.. 1, 2, 5, 7, 8

component developer builds composite components, specifies components, interfaces, and data types, and specifies data types. Creates service effect specifications, stores modelling and implementation artecfacts in repositories, and implements, tests, and maintains components. . 3

component-based software a software constituted of components. . 1

CTA Common Trace API. 2, 5, 7

internal action sequence of commands a component executes without leaving its scope (e.g. without calling other components). 1, 7, 8

Kieker a Java-based application performance monitoring and dynamic software analysis framework.

[3] A measurement software Beagle aims to support. . 9

measurement software software capable of measuring the time, given source code needs to execute some task. The software's results are usually returned in a time unit like nanoseconds. Beagles interactes with such software through the CTA and uses it to find resource demands. . 7, 11

Palladio an approach for the definition of component-based software architectures with a special focus on performance properties. . 1, 2, 11, 12

Palladio Component Model a domain-specific modeling language (DSL) used by Palladio.

It is designed to enable early performance predictions for software architectures and is aligned with a component-based software development process.

[2] . 1

PCM Palladio Component Model. 1, 2, 5, 7, 8

resource demand how much of a certain resource—like CPU, Network oder hard disk drive—a component needs to offer a certain functionality. A resource demand is ideally specified platform independently, e.g. by specifiying required CPU cycles, megabytes to be read, etc. If such information is not available, resource demands can be expressed platform dependend, e.g. in nanoseconds. In this case, a certain degree of portability can still be achieved if information about the used platforms' speed relative to each other is available. . 1, 7, 8

SEFF service effect specification. 2, 5, 7

SEFF condition conditions (like Java's `if`, `if-else` and `switch-case` statements) which affect the calls a component makes to other components. Such conditions are—contrary to conditions that stay within an internal action—modeled in the component's SEFF.. 1, 7, 8

SEFF loop loops (like Java's `for`, `while` and `do-while` statement) which affect the calls a component makes to other components. Such loops are—contrary to loops that stay within an internal action—modeled in the component's SEFF.. 2, 7, 8

service effect specification describes the inner behaviour of a component on a highly abstract level by specifying the relationship between provided and required services of a component. . 2

software architect a person who planes a software architecture from existing components and interfaces. Uses architectural styles and patterns, analyses architectural specifications, and makes design decisions. . 3

software architecture todo. 1, 2

SoMoX a Palladio plugin for static code analysis to reengineer a software's architecture from its source code. Constructs a PCM instance including the reconstructed components and their SEFF.. 1, 2, 9, 12

system deployer models the resource environments and allocations of components to resources. Also sets up the resource environments, deploys components onto resources, and maintains the running system. . 3

Bibliography

[1]

[2] Palladio component model, 09 2015.

[3] André van Hoorn, Jan Waller, and Wilhelm Hasselbring. Kieker: A framework for application performance monitoring and dynamic software analysis. In *Proceedings of the 3rd joint ACM/SPEC International Conference on Performance Engineering (ICPE 2012)*, pages 247–248. ACM, April 2012.