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 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's PCM instance.

1.2 Nice to Have Criteria

/OC10/ Beagle should determine the approximate probability in branch constructs, 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 loop constructs, which are annotated in the PCM, too, how often the loop body is executed aproximalty.

/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.

/B20/ Beagle does not reconstruct a model of softwares' architecture from their source code.

/B30/ Beagle does not reconstruct the internal structure of components like their service effect specification (SEFF).

/B40/ Beagle only analyses Java programs.

/B50/ Beagle does no performance analysis or prediction.

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 Common Trace API (CTA) to communicate with performance measurement software.

4 Functionality

4.1 Functional Requirements

/F10/ Reasonable measurement practice (e.g. speed vs. accuracy)

/F20/ Software should only depend on Eclipse, the PCM and the source code of the analysed software.

/F30/ Parameterisation of results

4.2 Functional Requirements

/OF10/ Analyse source code for its architecture and performance in one turn (e.g. with SoMoX & Kieker).

5 Non Functional Requirements

5.1 Must Have

/Q10/ Beagle shall be an Eclipse plugin. As both Palladio and its plugins are Eclipse plugins, this ensures good usability for users.

/Q20/ Beagle shall be controlled by context sensitive menus in Eclipse.

5.2 Nice To Have

/OQ10/ Beagle should run on every system, Eclipse and Palladio run on.

/OQ20/ The code measurement software should be exchangeable with any software which supports the CTA.

6 Test Cases

7 Models

Terms and Definitions

branch construct a construct in the source code only executing none or one of one or multiple cases. In Java the `if`, `if-else` and `switch-case` statements form branch constructs. . 1

Common Trace API an API developed by NovaTec GmbH for measuring the time, specific code sections need to be executed. . 5

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

component developer builds composite components, specifies components, interfaces, and data types, and specifies data types. Creates Palladio Component Models, 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. 5, 9

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

Kieker . 7

loop construct a construct in the source code executing a specific code block none, one or multiple times. In Java the `for` loop, the `while` loop and the `do-while` statements form loops. . 2

Palladio an approach for the definition of component-based software architectures with a special focus on performance properties. . 1, 9

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

resource demand foobar . 1

SEFF service effect specification. 2, 5

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, 7

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