Beagle

Design and Architecture

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

10th of January 2016

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

Reviewer:	Jun.-Prof. Dr.-Ing. Anne Koziolek
Advisor:	M.Sc. Axel Busch
Second advisor:	M.Sc. Michael Langhammer

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Contents

List of Figures

1 Architectural Overview

1.1 Overview of the entire system

1.2 Components' interaction

1.3 Communication between Beagle and external tools

2 Component: Beagle Core

2.1 Overview

Controller classes

The classes `Beagle Controller` and `Measurement Controller` manage the invocation of `Measurement Tool` or `Result Analyser` components. `Beagle Controller#main` is the main control loop, managing the control flow throughout Beagle's measuring and analysis activity. There is always exactly one `Measurement Tool`, `Result Analyser` or `Final Judge` running at any given moment during the execution of `Beagle Controller #main` ("the main loop").

An iteration of the main loop starts by asking the `Measurement Controller` whether it wants to conduct measurements for the current blackboard state—which will usually be the case if there is something not yet measured—, and if so, calling its `#measure` method. The `Measurement Controller` will then decide which `Measurement Tools` to run. Usually it will tell every tool to measure as long as there is something left to be measured.

After that, the main loop invokes one arbitrary chosen `Result Analyser` reporting to be able to contribute. This analyser may then propose results for items that have measurement results. If there is no such analyser, the `Final Judge` will be called. It decides whether enough information has been collected and Beagle can terminate. If this is the case, it also creates or selects the final result for each item that has proposed results.

The main loop will then be repeated until the `Final Judge` was called and its `#judge` method returned `true`.

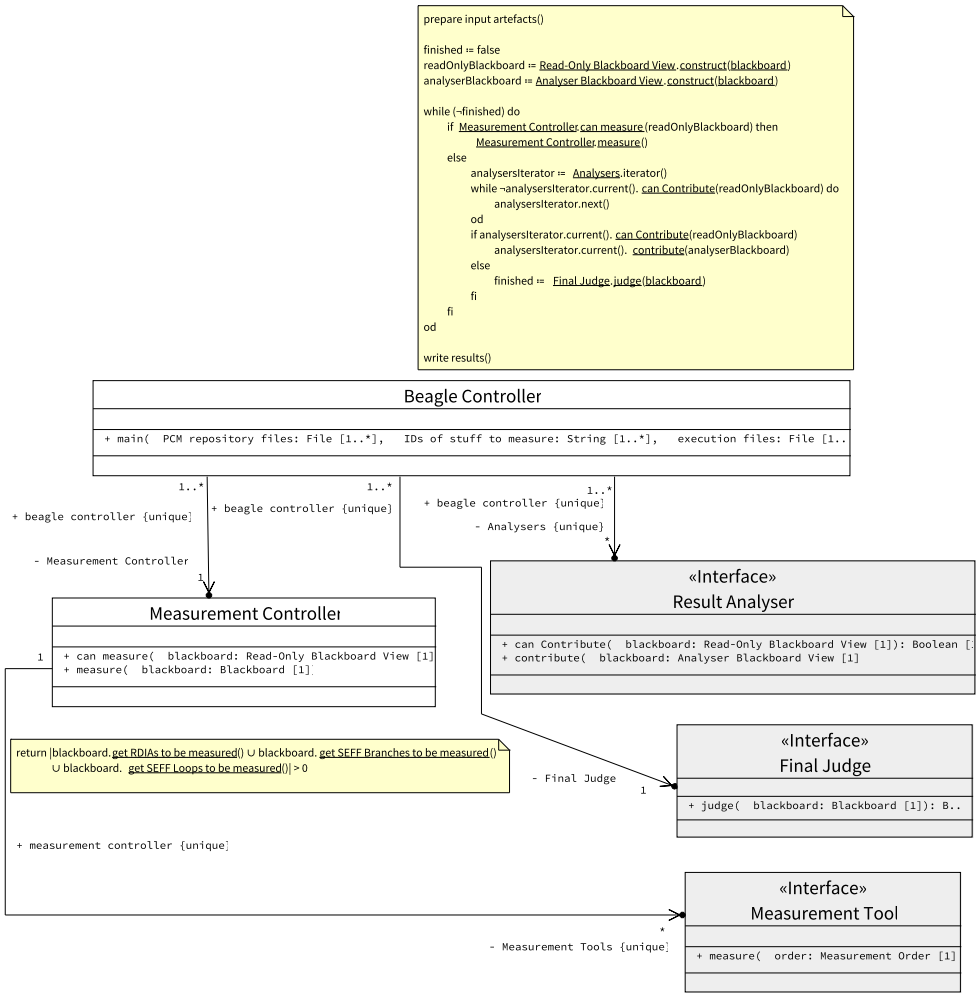

Figure 2.1: UML class diagram of the controller classes.

2.2 Reasons for chosen design

2.3 Chosen design patterns

2.4 Evaluable Expressions

2.5 Conversion from and to Palladio

3 Component: Beagle GUI

3.1 The most important classes

3.2 Reasons for chosen design

3.3 Chosen design patterns

4 Component: Measurement Tool

4.1 Reasons for chosen design

4.2 Adapter to Kieker

5 Component: Result Analyser

5.1 Reasons for chosen design

6 Component: Final Judge

6.1 Reasons for chosen design

6.2 “Averaging” Final Judge