Karlsruhe Institute of Technology

Software Design and Quality

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 ii

1 Architectural Overview 1
1.1 Overview of the entire system 1
1.2 Components’ interaction 1
1.3 Communication between Beagle and external tools 1

2 Component: Beagle Core 3
2.1 The most important classes 3
2.2 Reasons for chosen design 3
2.3 Chosen design patterns 3
2.4 Evaluable Expressions 3
2.5 Conversion from and to Palladio 3

3 Component: Beagle GUI 5
3.1 The most important classes 5
3.2 Reasons for chosen design 5
3.3 Chosen design patterns 5

4 Component: Measurement Tool 7
4.1 Reasons for chosen design 7
4.2 Adapter to Kieker 7

5 Component: Result Analyser 9
5.1 Reasons for chosen design 9

6 Component: Final Judge 11
6.1 Reasons for chosen design 11
6.2 “Averaging” Final Judge 11

List of Figures

2.1 Controller classes 4

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 The most important classes

The Controller classes. The classes `Beagle Controller` and `Measurment Controller` manage, when which measurement or result analyser component is working.

There is always only one measurement tool, one result analyser or the final judge working.

The `Beagle Controller` starts by asking the `Measurment Controller` weather it wants to measure something now, which will be usually the case, when there is something unmeasured, and if so invoking to . The `Measurment Controller` now decides, which tool to run. Usually it will tell each tool to measure, when there is anything left to measure.

After that the `Beagle Controller` checks for each result analyser, weather it can contribute to the blackboard and if so, let it work.

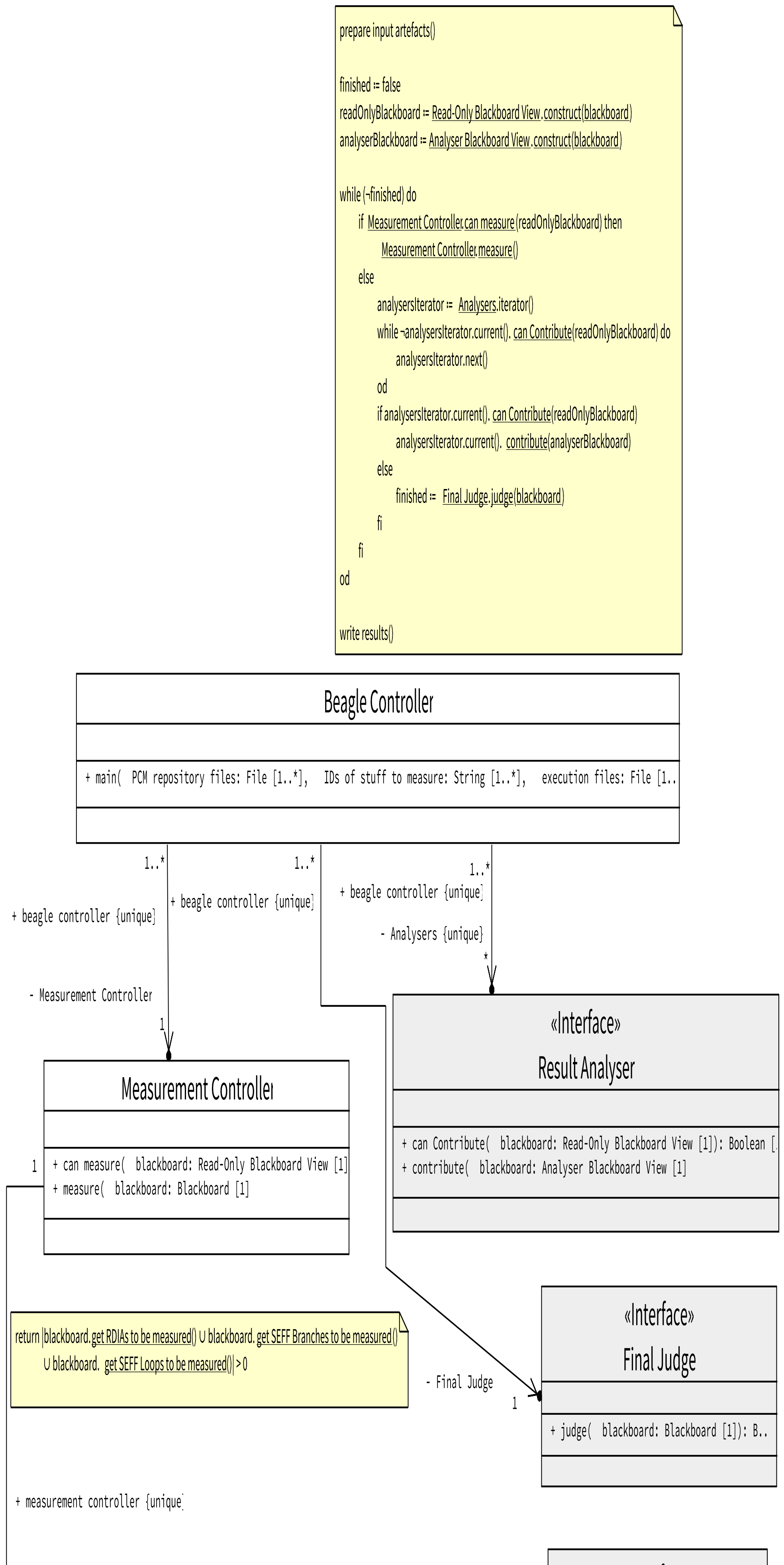
Finally a special result analyser, the final judge, always works. It decides wether enough information has been collected and Beagle can terminate. If so, it also creates or selects the final result for each measurment, which will be added in the PCM.

2.2 Reasons for chosen design

2.3 Chosen design patterns

2.4 Evaluable Expressions

2.5 Conversion from and to Palladio



```
prepare input artefacts()

finished := false
readOnlyBlackboard := Read-Only Blackboard View.construct(blackboard)
analyserBlackboard := Analyser Blackboard View.construct(blackboard)

while (¬finished) do
    if Measurement Controller.can measure(readOnlyBlackboard) then
        Measurement Controller.measure()
    else
        analysersIterator := Analysers.iterator()
        while ¬analysersIterator.current(). can Contribute(readOnlyBlackboard) do
            analysersIterator.next()
        od
        if analysersIterator.current(). can Contribute(readOnlyBlackboard)
            analysersIterator.current(). contribute(analyserBlackboard)
        else
            finished := Final Judge.judge(blackboard)
        fi
    fi
od

write results()
```

```
return |blackboard.get RDIAs to be measured() ∪ blackboard. get SEFF Branches to be measured()
    ∪ blackboard. get SEFF Loops to be measured()| > 0
```

Beagle Controller
+ main(PCM repository files: File [1..*], IDs of stuff to measure: String [1..*], execution files: File [1..
1..*
1..*
1..*
+ beagle controller {unique}
+ beagle controller {unique}
+ beagle controller {unique}
- Analysers {unique}
*
- Measurement Controller
1
Measurement Controller
+ can measure(blackboard: Read-Only Blackboard View [1]
+ measure(blackboard: Blackboard [1]
1
«Interface»
Result Analyser
+ can Contribute(blackboard: Read-Only Blackboard View [1]): Boolean [
+ contribute(blackboard: Analyser Blackboard View [1]
«Interface»
Final Judge
+ judge(blackboard: Blackboard [1]): B..
- Final Judge
1
+ measurement controller {unique}

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