28. November 2014 @ Stuttgart, Germany

Continuous Integration in Kieker
(Experience Report)

Nils Christian Ehmke, Christian Wulf, and Wilhelm Hasselbring
Software Engineering Group, Kiel University, Germany
A Short Outline

1. An Introduction to Continuous Integration

2. Continuous Integration in Kieker

3. Conclusion
A Short Outline

1. An Introduction to Continuous Integration

2. Continuous Integration in Kieker

3. Conclusion
The Meaning of CI

- "Continuous Integration is a software development practice where members of a team integrate their work frequently [...]" (Martin Fowler)
- Origins in Extreme Programming (XP)

- Goals:
  - Increase and monitor software quality
  - Reduce effort for developers
  - Increase communication

- Realization:
  - Automate as much as possible (build, tests, ...)

Diagram:
- Writing Code
- Performing Automated Build
- Performing Automated Test
- Publishing Reports
The Meaning of CI (Cont’d)

• Some Practices of CI [Based on Fowler and Foemmel 2006] => CI Server (e.g., Jenkins)
  – Maintain a single source repository => VCS (e.g., Git)
  – Automate the build => Build tools (e.g., Ant, Gradle)
  – Make your build self-testing => Testing (e.g., JUnit)
  – Keep the build fast
  – Make it easy […] to get the latest executable => Snapshots
  – Everyone can see what's happening => Reports

• Task Management (e.g., Trac, Jira)
The Impact of CI

• „Given all of the above, teams moving to a CI driven process can expect to achieve at least a 40% reduction in check-in overhead when compared to a check-in process that maintains the same level of code base and product quality.” (Ade Miller, 2008)

• 2013: 68% use CI\(^1\)

• 2014: 78.5% use CI\(^2\)

A Short Outline

1. An Introduction to Continuous Integration

2. Continuous Integration in Kieker

3. Conclusion
CI in Kieker

“Maintain a single source repository”
Cl in Kieker
“Automate the build”
CI in Kieker

“Automate the build (cont’d)”

Jenkins

<table>
<thead>
<tr>
<th>Name</th>
<th>Last Success</th>
<th>Last Failure</th>
<th>Last Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>kieker</td>
<td>3 hr 10 min</td>
<td>1 mo 9 days</td>
<td>9 min 4 sec</td>
</tr>
<tr>
<td>kieker-nightly-releases</td>
<td>9 hr 12 min</td>
<td>4 days 9 hr</td>
<td>46 min</td>
</tr>
<tr>
<td>kieker-webui</td>
<td>14 hr</td>
<td>10 mo</td>
<td>1 min 59 sec</td>
</tr>
<tr>
<td>kieker-webui-nightly-releases</td>
<td>14 hr</td>
<td>10 mo</td>
<td>2 min 6 sec</td>
</tr>
<tr>
<td>kiekerform-nightly-releases</td>
<td>11 hr</td>
<td>1 yr 11 mo</td>
<td>39 sec</td>
</tr>
</tbody>
</table>
CI in Kieker

“Make your build self-testing”

• Unit- and Integration-Tests (JUnit)
• Static Code Analysis (PMD, Checkstyle, Findbugs)
• Regression Benchmarking (MooBench)

Jan Waller, Florian Fittkau and Wilhelm Hasselbring

Application Performance Monitoring: Trade-Off between Overhead Reduction and Maintainability
## CI in Kieker

### “Make your build self-testing (cont’d)”

<table>
<thead>
<tr>
<th>Package</th>
<th>Fehlaufrufe</th>
<th>Fehlaufrufe (diff.)</th>
<th>Übersprüngen</th>
<th>Übersprüngen (diff.)</th>
<th>Pass</th>
<th>Pass (diff.)</th>
<th>Summe</th>
<th>Summe (diff.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>kieker/test-analysis/unit-configuration</td>
<td>0.37 Sekunden</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-analysis/unit-uiopen</td>
<td>8 Sekunden</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-analysis/unit-uiopen_filterflow</td>
<td>81 ms</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-analysis/unit-uiopen_filterforward</td>
<td>3.6 Sekunden</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-analysis/unit-uiopen_assertrate</td>
<td>55 Sekunden</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-analysis/unit-uiopen_visualizer</td>
<td>0.11 Sekunden</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-analysis/unit-uiopen_filtertrace</td>
<td>68 ms</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-analysis/unit-uiopen_filtervisualization</td>
<td>0.31 Sekunden</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-analysis/unit-uiopen_reader</td>
<td>0.29 Sekunden</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-analysis/unit-uiopen_reader/flowSystem</td>
<td>0.2 Sekunden</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-analysis/unit-uiopen_reader/namedRecordFlow</td>
<td>63 ms</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-analysis/unit-uiopen_reader/collector</td>
<td>28 Sekunden</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-common/unit</td>
<td>3.6 Sekunden</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-common/unit-configuration</td>
<td>7 ms</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-common/unit-record</td>
<td>2.1 Sekunden</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-common/unit-record-controlflow</td>
<td>25 ms</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-common/unit-record-flow-trace</td>
<td>39 ms</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-common/unit-record-flow-trace-concurrency</td>
<td>34 ms</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-common/unit-record-flow-trace-concurrency-monitor</td>
<td>0.1 Sekunden</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-common/unit-record-flow-trace-operation</td>
<td>0.13 Sekunden</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-common/unit-record-flow-trace-operation-processor</td>
<td>81 ms</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-common/unit-record-flow-trace-operation-object</td>
<td>74 ms</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kieker/test-common/unit-xml</td>
<td>16 ms</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
“Make your build self-testing (cont’d)”

Jenkins

Project kieker

Permalinks

- Last build (#2450), 3 hr 11 min ago
- Last stable build (#2450), 3 hr 11 min ago
- Last successful build (#2450), 3 hr 11 min ago
- Last unstable build (#2441), 6 days 21 hr ago
- Last successful build (#2441), 6 days 21 hr ago

CI in Kieker

keeps an eye on your software

Nils Christian Ehmke, Christian Wulf, and Wilhelm Hasselbring — Continuous Integration in Kieker
CI in Kieker

“Make your build self-testing (cont’d)”

Mean Overhead of Kieker

![Graph showing mean overhead of Kieker with different time points and build numbers. The graph includes lines for No Probe, Deactivated Probe, Collecting Data, Writer (ASCII), Writer (Bir), and Writer (TCP).]
"Make your build self-testing (cont’d)"

<table>
<thead>
<tr>
<th>Date</th>
<th>Base</th>
<th>Instrumentation</th>
<th>Collecting Data</th>
<th>Writer (ASCII)</th>
<th>Confidence Intervals (95%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013-03-05</td>
<td>4.15</td>
<td>0.84</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2013-03-06</td>
<td>4.10</td>
<td>0.87</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2013-03-07</td>
<td>4.00</td>
<td>4.00</td>
<td>3.86</td>
<td>3.94</td>
<td></td>
</tr>
<tr>
<td>2013-03-08</td>
<td>3.94</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2013-05-26</td>
<td>3.88</td>
<td>0.76</td>
<td>4.02</td>
<td>7.20</td>
<td></td>
</tr>
<tr>
<td>2013-05-27</td>
<td>7.20</td>
<td>0.76</td>
<td></td>
<td>7.25</td>
<td></td>
</tr>
<tr>
<td>2013-05-28</td>
<td>7.25</td>
<td>0.76</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CI in Kieker

“Everyone can see what’s happening”
“Make it easy [...] to get the latest executable”
#1126 new feature

Adaptive monitoring based on dynamic instrumentation

<table>
<thead>
<tr>
<th>Created by:</th>
<th>avh</th>
<th>Responsible:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>major</td>
<td>Meierstein: Release 1.11</td>
</tr>
<tr>
<td>Component:</td>
<td>software-monitoring</td>
<td>Version:</td>
</tr>
<tr>
<td>Keywords:</td>
<td>thesis</td>
<td>Watchers: jwa</td>
</tr>
</tbody>
</table>

**Description**

Adaptive monitoring in Kieker is currently not based on dynamic instrumentation. However, this would be desirable, e.g., based on JavaAssist and/or runtime class redefinitions. @inspectIT has support for this.

**Attachments** (0)

**Change History** (4)

- Watcher jwa added
- Keywords thesis, discussInMeeting added; thesis deleted

A student from Stuttgart will write a Bachelor's thesis on this topic (currently at the beginning of the proposal phase). For the evaluation, which is essentially a comparison of the overhead of the current implementation with the new one, we were thinking about using/adopting MooBench.

@Jan: Maybe we can briefly talk about this on Monday.
A Short Outline

1. An Introduction to Continuous Integration

2. Continuous Integration in Kieker

3. Conclusion
Conclusion

• Benefits:
  – Early warnings and feedbacks for the developers
  – Avoid corruption of anything without noticing
  – Automated a lot of jobs
  – More relaxing releases

• Challenges:
  – CI does need some initial effort
  – Keep CI in synchronisation with the project
  – Complex system for new developers
Bibliography

• Martin Fowler and Matthew Foemmel; Continuous Integration; 2006
• RebelLabs; Developer Productivity Report 2013; 2013
• RebelLabs; Java Tools and Technologies Landscape for 2014; 2014
• Ade Miller; A Hundred Days of Continuous Integration; 2008; AGILE ‘08 Conference; Pages 289-293