Combining Kieker with Gephi for Performance Analysis and Interactive Trace Visualization

Christian Zirkelbach, Wilhelm Hasselbring, and Leslie Carr
Performance issues regarding established Perl software

- Often difficult to identify potential bottlenecks
- Architectural discovery
- Program comprehension
- Performance analysis

→ Visualization of results

→ Kieker Monitoring Framework
Approach
Perl Instrumentation using Kieker

Approach

```
1 use Sub::WrapPackages;
2     packages = > [qw(EPrints EPrints::* )],
3     pre = > sub {
4         use Kieker;
5         my $kieker = Kieker -> new ();
6         my $packageName = $_[0];
7         $packageName =~ s /:://. g ;
8         $packageName =~ /^(.*) \..*? $ /;
9         $kieker -> EntryEvent ($packageName, $1);
10    },
11    post = > sub {
12        use Kieker;
13        my $kieker = Kieker -> new ();
14        my $packageName = $_[0];
15        $packageName =~ s /:://. g ;
16        $packageName =~ /^(.*) \..*? $ /;
17        $kieker -> ExitEvent ($packageName, $1);
18    };
```
Instrumentation Components

Approach

Perl Instrumentation
- Monitoring Writer
  - Monitoring Controller
  - Monitoring Probe

Kieker Data Bridge
- JMS Client
  - Service Container
  - Monitoring Controller
  - Monitoring Writer

Kieker.Analysis
- Monitoring Reader
  - Analysis Controller
  - Analysis / Visualization Plugin
Performance Analysis Workflow

Approach

Perl Instrumentation → Monitoring via Kieker

Appropriate coverage?

- no → Batch Visualization via Graphviz
- yes → Interactive Visualization via Gephi

finished?
Approach

1. Manipulating the generated dependency call graph (Kieker)
   - Architecture Discovery & Program Comprehension
     - Aggregation based on the package hierarchy
   - Performance Analysis
     - Number of edges (calls) in combination with response times

2. Visualizing the manipulated graph
Further analysis and visualization via Gephi

Approach

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>wishes_to_export() min: 0ms, avg: 0.00ms, max: 0ms</td>
<td>depNode_117</td>
</tr>
<tr>
<td>who_filter() min: 0ms, avg: 0.00ms, max: 0ms</td>
<td>depNode_128</td>
</tr>
<tr>
<td>show_columns() min: 0ms, avg: 0.00ms, max: 0ms</td>
<td>depNode_106</td>
</tr>
<tr>
<td>render_title() min: 3ms, avg: 3.00ms, max: 3ms</td>
<td>depNode_169</td>
</tr>
<tr>
<td>render_title() min: 1ms, avg: 2.50ms, max: 4ms</td>
<td>depNode_149</td>
</tr>
<tr>
<td>render_title() min: 0ms, avg: 0.00ms, max: 0ms</td>
<td>depNode_150</td>
</tr>
<tr>
<td>render_tab_title() min: 0ms, avg: 0.00ms, max: 0ms</td>
<td>depNode_36</td>
</tr>
<tr>
<td>render_tab_title() min: 0ms, avg: 0.00ms, max: 0ms</td>
<td>depNode_64</td>
</tr>
<tr>
<td>render_items() min: 505ms, avg: 505.00ms, max: 505ms</td>
<td>depNode_135</td>
</tr>
<tr>
<td>render_import_bar() min: 21ms, avg: 21.00ms, max: 21ms</td>
<td>depNode_132</td>
</tr>
</tbody>
</table>
Further analysis and visualization via Gephi

### Approach

#### Data Table

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Type</th>
<th>Id</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>depNode_1</td>
<td>depNode_11/</td>
<td>Directed</td>
<td>1/5</td>
<td>1</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_125</td>
<td>Directed</td>
<td>1789</td>
<td>2</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_111</td>
<td>Directed</td>
<td>1792</td>
<td>1</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_116</td>
<td>Directed</td>
<td>1804</td>
<td>1</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_170</td>
<td>Directed</td>
<td>1807</td>
<td>6</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_101</td>
<td>Directed</td>
<td>1818</td>
<td>131</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_118</td>
<td>Directed</td>
<td>1826</td>
<td>1</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_2</td>
<td>Directed</td>
<td>1838</td>
<td>10</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_115</td>
<td>Directed</td>
<td>1886</td>
<td>1</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_155</td>
<td>Directed</td>
<td>1870</td>
<td>1</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_173</td>
<td>Directed</td>
<td>1797</td>
<td>1</td>
</tr>
</tbody>
</table>
Case Study

Let’s evaluate our approach!
Performance Analysis of EPrints

Case Study

Performance analysis of EPrints 3.3.12 using Kieker

- Potential bottleneck detection for upcoming release version 4
- Evaluation in collaboration with the development team
- Software maintenance & modernization
- Different instrumentation levels are possible
- Combined visualization: GraphViz and Gephi
Evaluation Goals

Case Study

- Is it possible to identify an architecture? (Architectural Discovery)
- Can we support other developers by providing information about the software? (Program Comprehension)
- Can we detect potential bottlenecks? (Performance Analysis)
Case Study

Architecture Discovery
Batch Visualization via Graphviz

Case Study ▶ Architecture Discovery
Gephi: System Architecture Level

Case Study ▶ Architecture Discovery

avg: 333.73ms, max: 7239ms

EPrints.Language
avg: 6.64ms, max: 429ms

EPrints.Repositoty
avg: 1.89ms, max: 841ms

EPrints.PluginScreen
avg: 6.33ms, max: 537ms

EPrints.Plugin
avg: 0.73ms, max: 44ms

EPrints.DataObject
avg: 3.66ms, max: 218ms

EPrints.DataObject.User
avg: 5.74ms, max: 156ms

EPrints.DataSet
avg: 0.59ms, max: 106ms

EPrints.Database
avg: 36.50ms, max: 658ms

EPrints.Database.MySql
avg: 0.63ms, max: 78ms

Zirkelbach, Hasselbring, and Carr

Combining Kieker with Gephi

2015-11-06   16 / 28
Case Study

Performance Bottleneck Identification
Case Study ▶ Performance Bottleneck Identification

\[ \text{Request}_1() \]
min: 559ms, avg: 559.00ms, max: 559ms
Case Study ▶ Performance Bottleneck Identification

Potential bottleneck: database-oriented operations

Zirkelbach, Hasselbring, and Carr

Combining Kieker with Gephi

2015-11-06 20 / 28
Case Study  ▶  Performance Bottleneck Identification

Potential bottleneck: database-oriented operations

Case Study ▶ Performance Bottleneck Identification

- get_sql_names()
- value_from_sql_row()
- render_name()
- repository()
- get_name()
- name()
- is_virtual()
- render_items()
- get_value()
- get_property()
Case Study ▸ Performance Bottleneck Identification

Potential bottleneck: database-oriented operations
Related Work

Combining Kieker with Gephi

Zirkelbach, Hasselbring, and Carr
Summary

- Well-structured workflow for a performance analysis
- Verification through a case study of an established application
- Detection of performance issues → basis for recommendations

→ Continuous integration
What happened after our Case Study?
Application-level Perspective in ExplorViz

Outlook
Physical 3D Model based on ExplorViz

Outlook