Live Trace Visualization
for System and Program Comprehension
in Large Software Landscapes

Seminar @ Lugano
Kiel University, Software Engineering Group
Florian Fittkau — July 1, 2015
Motivation

- Increasing code complexity on application level
- Likewise increase of systems and communications (e.g., microservices)
- Systems often form a large software landscape
- Knowledge about usage and communication of each system often gets lost

Live trace visualization of those landscapes for comprehension of systems and applications

Selected Challenges:
- Possible huge monitoring data amount (performance/cost efficiency)
- Finding abstractions to understand huge landscapes but also application-level details
- Live visualization of thousands or even millions of traces

[Fittkau et al. 2013]
Approach & Contributions
The ExplorViz Method

Live Trace Visualization for System and Program Comprehension in Large Software Landscapes

[Fittkau et al. 2013a]
Monitoring & Trace Processing

Monitoring

- Monitoring Probe 1
- Monitoring Probe n
- Monitored Code
- Byte Buffer
- Monitoring Controller
- Ring Buffer
- << Thread >> Monitoring Writer

Analysis

- << Thread >> Record Construction
- Ring Buffer
- Trace Reconstruction
- Ring Buffer
- < Thread >> Trace Reduction
- Ring Buffer
- Trace
- << Thread >> Repository

[Fittkau et al. 2013b, Beye 2013, Matthiessen 2014, Weißenfels 2014]
Scalable Trace Processing

Live Trace Visualization for System and Program Comprehension in Large Software Landscapes

[Fittkau et al. 2014b, Koppenhagen 2013, Stelzer 2014]
Landscape Meta-Model

System
- name
nodeGroup
- name
- ipAddress
- cpuUtilization
- freeRAM
- usedRAM

Node
- name
- ipAddress
- cpuUtilization
- freeRAM
- usedRAM

Communication
- requests
- technology
- averageResponseTime

Application
- name
- image
- lastUsage
- database
- programmingLanguage

Component
- name
- fullQualifiedName
- synthetic

Clazz
- name
- fullQualifiedName
- instanceCount
- objectIds

CommunicationClazz
- methodName
- attribute

RuntimeInformation
- calledTimes
- overallTraceDuration
- requests
- averageResponseTime
- orderIndexes

Sole representation of landscape information
ExplorViz Visualization

Landscape-Level Perspective

[Fittkau et al. 2013a]
ExplorViz Visualization

Application-Level Perspective

[Fittkau et al. 2013a]
Selected ExplorViz Features

Clustering

Performance Analysis

Architecture Modeling & Conformance Checking
[Barzel 2014] [Jähde 2015]

Trace Replayer

Interactive Tutorial

Database Monitoring
[Finke 2014]

[Barzel 2014] [Fittkau et al. 2014b, Simolka 2015]

[Finke 2014] [Zirkelbach 2015]
Physical 3D Models
Virtual Reality

[Krause 2015]
Virtual Reality

[Krause 2015]
Evaluations
Monitoring & Trace Processing

Kieker 1.8 with Analysis:
**0.5k** traces per second

ExplorViz with Analysis:
**112.6k** traces per second (limited by network speed)

Speedup of about **factor 250**

Kieker 1.8:
**16.6k** traces per second

ExplorViz:
**141.2k** traces per second

Speedup of about **factor 9**

[Fittkau et al. 2013b]

[Waller et al. 2014]
Scalable Trace Processing

Processing Capabilities:
- Cost efficient
- Scalable to millions of monitored methods

[Fittkau et al. 2014b, Stelzer 2014]
Application Perspective

First Controlled Experiment
**Visualization Capabilities:**

- Usable for solving program comprehension tasks
- Interactive concept for method calls works
- More efficient and effective than competitor (for the chosen tasks)
Physical 3D Model

VS

112 participants solved program comprehension tasks in pairs

---

Live Trace Visualization for System and Program Comprehension in Large Software Landscapes

Florian Fittkau — June 11, 2015
## Visualization Capabilities:

- Usable for system comprehension
- Abstractions provide a valuable addition

### Table: Time Spent and Correctness

<table>
<thead>
<tr>
<th></th>
<th>Flat</th>
<th>Hierarchical</th>
<th>Flat</th>
<th>Hierarchical</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>23.49</td>
<td>23.45</td>
<td>17.07</td>
<td>19.3</td>
</tr>
<tr>
<td>difference</td>
<td>23.49-17.07 = 6.42</td>
<td>23.45-19.3 = 4.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sd</td>
<td>3.87</td>
<td>3.29</td>
<td>3.27</td>
<td>3.33</td>
</tr>
<tr>
<td>min</td>
<td>15.03</td>
<td>15.93</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>median</td>
<td>24.64</td>
<td>23.14</td>
<td>17.25</td>
<td>20.5</td>
</tr>
<tr>
<td>max</td>
<td>29.68</td>
<td>33.16</td>
<td>22</td>
<td>22</td>
</tr>
</tbody>
</table>

### Statistical Tests:

- Shapiro-Wilk W: 0.9232, 0.9605
- Levene F: 2.1048, 1.2307
- Student’s t-test: df = 27, t = 2.4416, p-value = 0.02303

29 participants solved system comprehension tasks

140 applications
Related Work & Outlook
Closely Related Work

Software Visualizations

[Image of a 3D visualization of software components]

APM Tools

[Image of a screenshot of APM tools]

Monitoring

[Image of the Kieker logo]

Live Trace Visualization for System and Program Comprehension in Large Software Landscapes

Florian Fittkau — June 11, 2015
Publications & Theses

- 20 publications in 3.5 years
  (15 peer-reviewed)

Selected Publications

- **ICPC 2015**: Fittkau, Finke, Hasselbring, and Waller, “Comparing Trace Visualizations for Program Comprehension through Controlled Experiments” (Acceptance Rate: 31.5%)
- **ECIS 2015**: Fittkau, Roth, and Hasselbring, ExplorViz: Visual Runtime Behavior Analysis of Enterprise Application Landscapes (Acceptance Rate: 31%)
- **VISSOFT 2013**: Fittkau, Waller, Wulf, and Hasselbring, Live Trace Visualization for Comprehending Large Software Landscapes: The ExplorViz Approach
- **ICSE 2013**: Frey, Fittkau, and Hasselbring, Search-Based Genetic Optimization for Deployment and Reconfiguration of Software in the Cloud

15 Student’s Theses in the Context of ExplorViz

Live trace visualization for large software landscapes available as **open-source software** (Apache License 2.0)

**All results available online**
- Raw results, R scripts, code, ratings, ...
- ExplorViz versions used in the experiments
- All screen and camera recordings about 160 hours material
- Long-time archival on Zenodo.org

**Future Work:**
- More controlled experiments (e.g., comparison with more visualization metaphors; professionals as subjects)
- Layout of the Application-Level Perspective  
  [Barbie 2014]
Bibliography


[Barbie 2014] A. Barbie. Stable 3D City Layout in ExplorViz, Bachelor thesis, Kiel University, Kiel, Germany, 93 pp


[Fittkau 2013] F. Fittkau. Live Trace Visualization for System and Program Comprehension in Large Software Landscapes, TR 1310. Department of Computer Science, Kiel University, Kiel, Germany.


Backup Slides
<table>
<thead>
<tr>
<th>ID</th>
<th>Category</th>
<th>Description</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>A{4,8}</td>
<td><strong>Context: Identifying refactoring opportunities</strong>&lt;br&gt;Name three classes (from different packages) that have high fan-in (at least 4 incoming communications) and almost no fan-out (outgoing communication).</td>
<td>3</td>
</tr>
<tr>
<td>T2.1</td>
<td>A{3,4,5}</td>
<td><strong>Context: Understanding the checking process</strong>&lt;br&gt;Write down all constructor/method calls between <code>RuleChain</code> and <code>JavaRuleChainVisitor</code>.</td>
<td>3</td>
</tr>
<tr>
<td>T2.2</td>
<td>A{1,2,5,6}</td>
<td>In general terms, describe the lifecycle of <code>GodClassRule: Who creates it, what does it do (on a high level)?</code></td>
<td>3</td>
</tr>
<tr>
<td>T3.1</td>
<td>A{1,5}</td>
<td><strong>Context: Understanding the violation reporting process</strong>&lt;br&gt;Which rules are violated by the input file using the design rule set? Hint: Due to dynamic analysis the violation object is created only for those cases.</td>
<td>2</td>
</tr>
<tr>
<td>T3.2</td>
<td>A{1,3}</td>
<td><strong>Context: Understanding the violation reporting process</strong>&lt;br&gt;How does the reporting of rule violations work? Where does a rule violation originate and how is it communicated to the user? Write down the classes directly involved in the process. Hint: The output format is set to HTML.</td>
<td>4</td>
</tr>
<tr>
<td>T4</td>
<td>A{1,7,9}</td>
<td><strong>Context: Gaining a general understanding</strong>&lt;br&gt;Starting from the Mainclass PMD – On high level, what are the main abstract steps that are conducted during a PMD checking run. Stick to a maximum of five main steps. Hint: This is an exploration task to get an overview of the system. One strategy is to follow the communication between classes/packages. Keep the handout of PMD in mind.</td>
<td>5</td>
</tr>
<tr>
<td>ID</td>
<td>Category</td>
<td>Description</td>
<td>Score</td>
</tr>
<tr>
<td>-----</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>RT1</td>
<td>A{4,8}</td>
<td><strong>Context: Identifying refactoring opportunities</strong> Name three classes that have high fan-in (at least 3 incoming communications) and almost no fan-out (outgoing communication).</td>
<td>3</td>
</tr>
<tr>
<td>RT2.1</td>
<td>A{3,4,5}</td>
<td>Write down all constructor/method calls between gui.MainActivity and comm.Sync.</td>
<td>3</td>
</tr>
<tr>
<td>RT2.2</td>
<td>A{1,2,5,6}</td>
<td>In general terms, describe the lifecycle of data.User: Who creates it, how is it used? Write down the method calls.</td>
<td>3</td>
</tr>
<tr>
<td>RT3</td>
<td>A{1,3}</td>
<td><strong>Context: Understanding the antibiotics display process</strong> How does the display of antibiotics work? Where and how are they created? Write down the classes directly involved in the process.</td>
<td>6</td>
</tr>
<tr>
<td>RT4</td>
<td>A{1,7,9}</td>
<td><strong>Context: Gaining a general understanding</strong> Starting from the Mainclass gui.MainActivity - What are the user actions (e.g., Login and Logout) that are performed during this run of Babsi. Write down the classes of the activities/fragment for each user action. Stick to a maximum of seven main steps (excluding Login and Logout). Hint: This is an exploration task to get an overview of the system. One strategy is to follow the communication between classes.</td>
<td>7</td>
</tr>
<tr>
<td>ID</td>
<td>Category</td>
<td>Description</td>
<td>Score</td>
</tr>
<tr>
<td>----</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>T1</td>
<td>A{3,5,6,8}</td>
<td><em>Context: Metric-Based Analysis</em> Find the package containing the one class having the most instances in the application. How is the package named? How many classes (and subpackages if existing) does it contain? Please write down the full package path.</td>
<td>2</td>
</tr>
<tr>
<td>T2</td>
<td>A{6,8}</td>
<td><em>Context: Structural Understanding</em> What are the names of the three packages directly containing the most classes (without their subpackages)? Please order your answer by beginning with the package containing the most classes and write down the full path.</td>
<td>4</td>
</tr>
<tr>
<td>T3</td>
<td>A{1,3,7}</td>
<td><em>Context: Concept Location</em> Assuming a good design, which package could contain the <em>Main</em> class of the application? Give reasons for your answer.</td>
<td>2</td>
</tr>
<tr>
<td>T4</td>
<td>A{3,4}</td>
<td><em>Context: Structural Understanding</em> Which package name occurs the most in the application? In addition, shortly describe the distribution of these packages in the system. Hint: Have a look at the different levels of the packages. There are exactly two types of distribution.</td>
<td>3</td>
</tr>
<tr>
<td>T5</td>
<td>A{1,2,3,9}</td>
<td><em>Context: Design Understanding</em> What is the purpose of the <em>lang</em> package and what can you say about its content regarding PMD? Are there any special packages? Do they differ by size? Ignore the <em>xpath</em> and <em>dfa</em> packages and name three facts in your answer. <em>Hint:</em> Remember the received paper about the introduction to PMD.</td>
<td>3</td>
</tr>
<tr>
<td>ID</td>
<td>Description</td>
<td>Score</td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>Name three applications that have a high fan-in (at least two incoming communication lines). The two incoming communication lines should be on one node and not distributed over multiple nodes.</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>Name the Top 3 communications with the highest request count in descending order. Write down the start application and the end application.</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>Which applications are duplicated on multiple nodes? The answer should contain all 8 duplicated applications which are all named differently. Hint: The hostname of the nodes, where the applications are running, are numbered, e.g., Server 1, Server 2,...</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td>What is the purpose of the WWWPRINT application in your opinion? How does the process might work to achieve the functionality for the user?</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>What are the consequences of a failure of the LDAP application? Name all affected applications and briefly describe their purposes. Hint: Remember the received paper about the introduction to the university landscape.</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>
Extensibility (Control Center)

Symptoms

Diagnosis

Planning

[Fittkau et al. 2014a, Gill 2015, Michaelis 2015, Mannstedt 2015]
Meta-Monitoring

Live Trace Visualization for System and Program Comprehension in Large Software Landscapes

Florian Fittkau — June 11, 2015