Simplifying Software System Monitoring through Application Discovery with ExplorViz

SSP ’18
9th Symposium on Software Performance 2018

Alexander Krause, Christian Zirkelbach, Wilhelm Hasselbring

Talk by Sören Henning
– Kiel University

November 08, 2018
ExplorViz in a Nutshell

Live trace visualization of large software landscapes for comprehension of systems and applications [1]

Selected Challenges
- Finding abstractions to understand huge landscapes but also application-level details
- Live visualization of thousands or even millions of traces
The ExplorViz Method

Legend:
A1: Monitoring
A2: Preprocessing
A3: Aggregation
A4: Transformation
A5: Navigation
Let’s start to visualize
Let’s start to visualize
Let’s start to visualize

Kieker 1.13 User Guide

Kieker Project

October 4, 2017

http://kieker-monitoring.net
Let’s start to visualize
Let’s start to visualize
Let’s start to visualize … with

ADAMMS

(Application Discovery and Monitoring Management System)
Design – Application Discovery

Application Discovery (Single Iteration)

1. Obtain OSJPL
2. Obtain Working Directory
3. Apply Recognition Strategies
4. Analyze and Merge

current OSJPL
enriched OSJPL (1)
enriched OSJPL (2)
old OSJPL

OSJPL – Operating System Java Process List
Design – Application Discovery

Application Discovery (Single Iteration)

1. Obtain OSJPL
2. Obtain Working Directory
3. Apply Recognition Strategies
4. Analyze and Merge

OSJPL – Operating System Java Process List

<<Interface>>

RecognitionStrategy
+isDesiredApplication() : boolean
+applyStrategy() : void
Design – Monitoring Management

Monitor Instruction

- Kill process
- Insert identifier
- Update model
- Internal model
- Insert Monitoring
- Find process
- Get new OSJPL
- Run exec cmd
- OSJPL

ADA
MM
S

Monitoring Instruction

Design – Monitoring Management

Monitor Instruction

- Kill process
- Insert identifier
- Update model
- Internal model
- Insert Monitoring
- Find process
- Get new OSJPL
- Run exec cmd
- OSJPL

ADA
MM
S

Monitoring Instruction
Design – Monitoring Management

- Find process
- Get new OSJPL
- Run exec cmd
- Internal model
- Update model
- Kill process
- Insert identifier
- Insert Monitoring

```
java -javaagent:kieker.jar -Dexplorviz.agent.model.id=2 -cp . -jar /kiekerSampleApp/sampleApplication.jar
```
Design – Monitoring Management

1. Find process
2. Internal model
3. Update model
4. Kill process
5. Insert identifier
6. Insert Monitoring
7. Run exec cmd
8. Get new OSJPL
9. OSJPL

```
java -javaagent:kieker.jar -Dexplorviz.agent.model.id=2 -cp . -jar /kiekerSampleApp/sampleApplication.jar
```
Implementation – Software Stack

Agent

```
Jersey

Jersey Client
```

HTTP (REST) 1..*

Backend

```
Jersey

Jersey Client
```

HTTP (REST) 1..*

Frontend

```
ember

Cytoscape.js
```
Implementation – Overview Page
Implementation – Process Page
First Pilot Study

Usability Evaluation

Setup

Results & Discussion

Goals

• Overall usability regarding setup and operation
First Pilot Study

Usability Evaluation

Setup

Results & Discussion

Goals
- Overall usability regarding setup and operation

Experiment
- Proband solved tasks
- Conductor notated issues
- Pluralistic walkthrough [2]
First Pilot Study

Goals
- Overall usability regarding setup and operation

Experiment
- Proband solved tasks
- Conductor notated issues
- Pluralistic walkthrough [2]

Structural Interview
- Perceived usability
- Enhancements
First Pilot Study

Software Landscape
- 8 running applications
  - Tomcat with JPetStore 6
  - kiekerSampleApplication¹

<table>
<thead>
<tr>
<th>Name</th>
<th>Quantity</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>node2, node3</td>
<td>2</td>
<td>CPU: 2x Intel Xeon E5-2650 (2.8GHz, 8 cores)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ram: 128 GB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OS: Debian</td>
</tr>
<tr>
<td>lp1</td>
<td>1</td>
<td>CPU: Intel Core i7-6700HQ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ram: 32 GB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OS: Windows 10</td>
</tr>
<tr>
<td>lp2</td>
<td>1</td>
<td>CPU: Intel Core i5-4278U</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ram: 16 GB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OS: macOS High Sierra</td>
</tr>
</tbody>
</table>

¹ https://github.com/czirkelbach/kiekerSampleApplication
First Pilot Study

Software Landscape
- 9 running applications
  - Tomcat with JPetStore 6
  - kiekerSampleApplication

<table>
<thead>
<tr>
<th>Name</th>
<th>Quantity</th>
<th>Setup</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>node2, node3</td>
<td>2</td>
<td>CPU: 2x Intel Xeon E5-2650 (2.8GHz, 8 cores)</td>
<td>Ram: 128 GB, OS: Debian</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lp1</td>
<td>1</td>
<td>CPU: Intel Core i7-6700HQ</td>
<td>Ram: 32 GB, OS: Windows 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lp2</td>
<td>1</td>
<td>CPU: Intel Core i5-4278U</td>
<td>Ram: 16 GB, OS: macOS High Sierra</td>
</tr>
</tbody>
</table>

DOI: 10.5281/zenodo.1471447

1 https://github.com/czirkelbach/kiekerSampleApplication
First Pilot Study

Usability Evaluation

Setup

Results & Discussion

Learnability
- Setup requirements are manageable
- Monitoring Management is easily accessible…
- … but requires knowledge about program internals

Efficiency
- Faster execution of repeating tasks

Memorability
- Workflow seemed to be memorable …
- … but casual users might show another result

Low error rate
- System helped to resolve errors during operation…
- … but did not prevent flaws

Satisfaction
- Graph design is pleasant to use
- Management dialogs require more work
Conclusions & Future Work

- Observations indicate good usability
  
- (Simple) Monitoring Management works
  - Prevent errors
  - Provide templates
  - Refactor GUI forms

- Application Discovery method works
  - JDK 9 usage?
  - Cryptic execution commands?

- More research required
  - Wider range of probands
Live Demo
References

