EPrints performance analysis using Kieker

Christian Zirkelbach, B.Sc.

Kiel University
Department of Computer Science
czi@informatik.uni-kiel.de

University of Southampton, May 8th 2014

Agenda

Introduction

Performance analysis of EPrints using Kieker

- Performance evaluation (e.g., bottleneck detection)
- Software maintenance, reverse engineering, modernization
- Different instrumentation levels are possible
- Perl module based on bachelor thesis of Nis Wechselberg [1]

Results

It's all about results...

Results

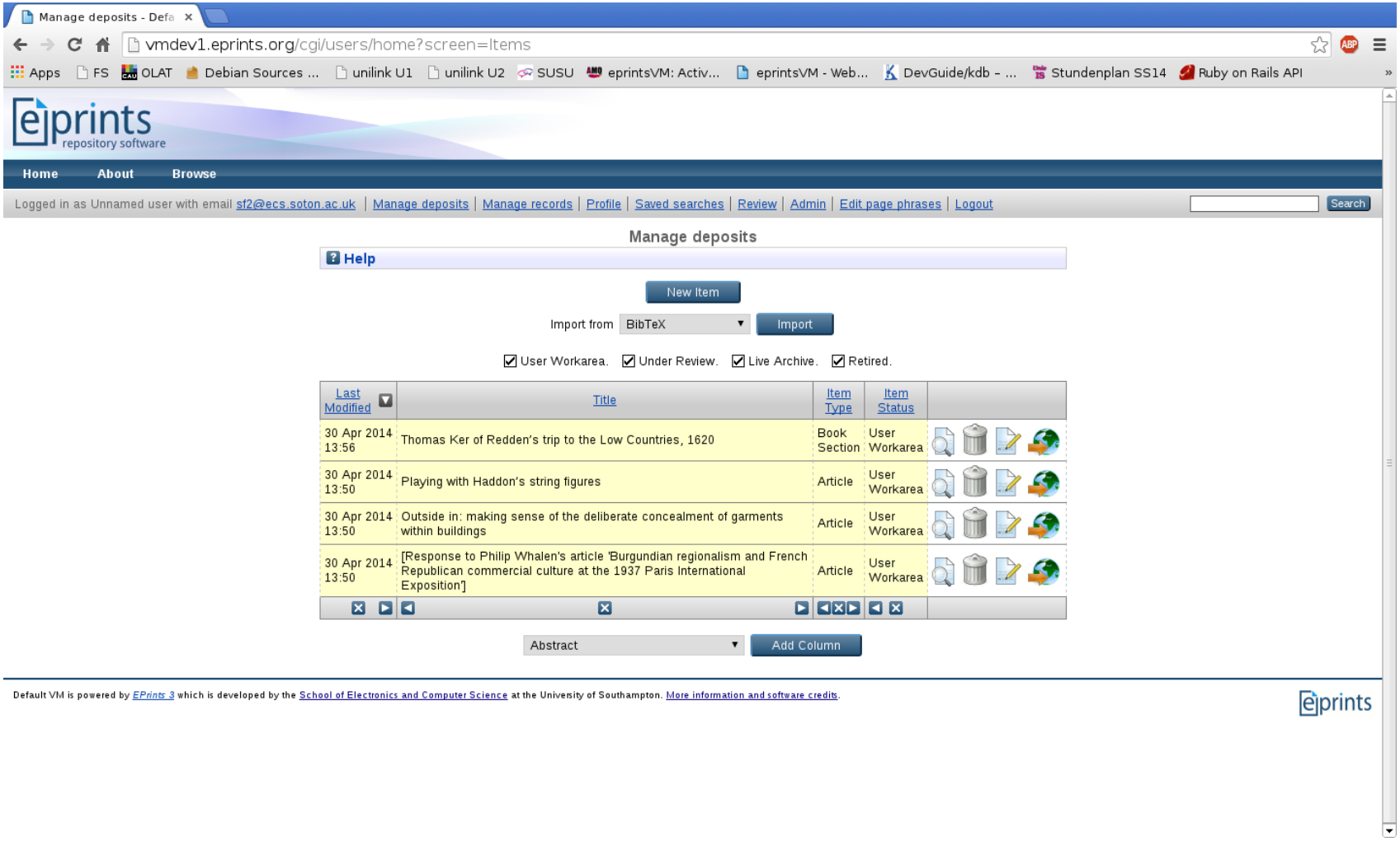

Figure 1: Request: cgi/users/home?screen=Items

First try: full instrumentation

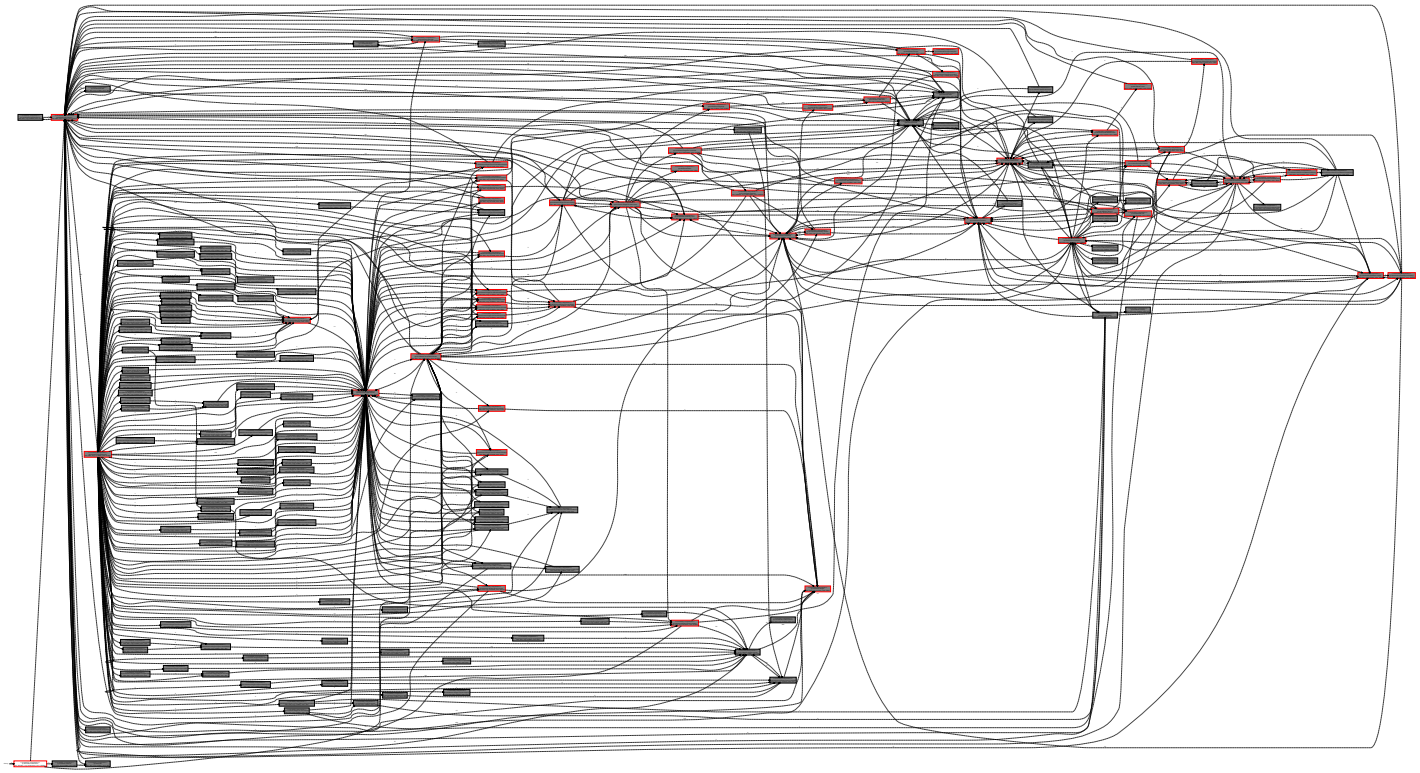

Figure 2: Component dependency graph for EPrints::*

System Architecture Level

Is it possible to identify an architecture?

System Architecture Level: extract of full instrumentation

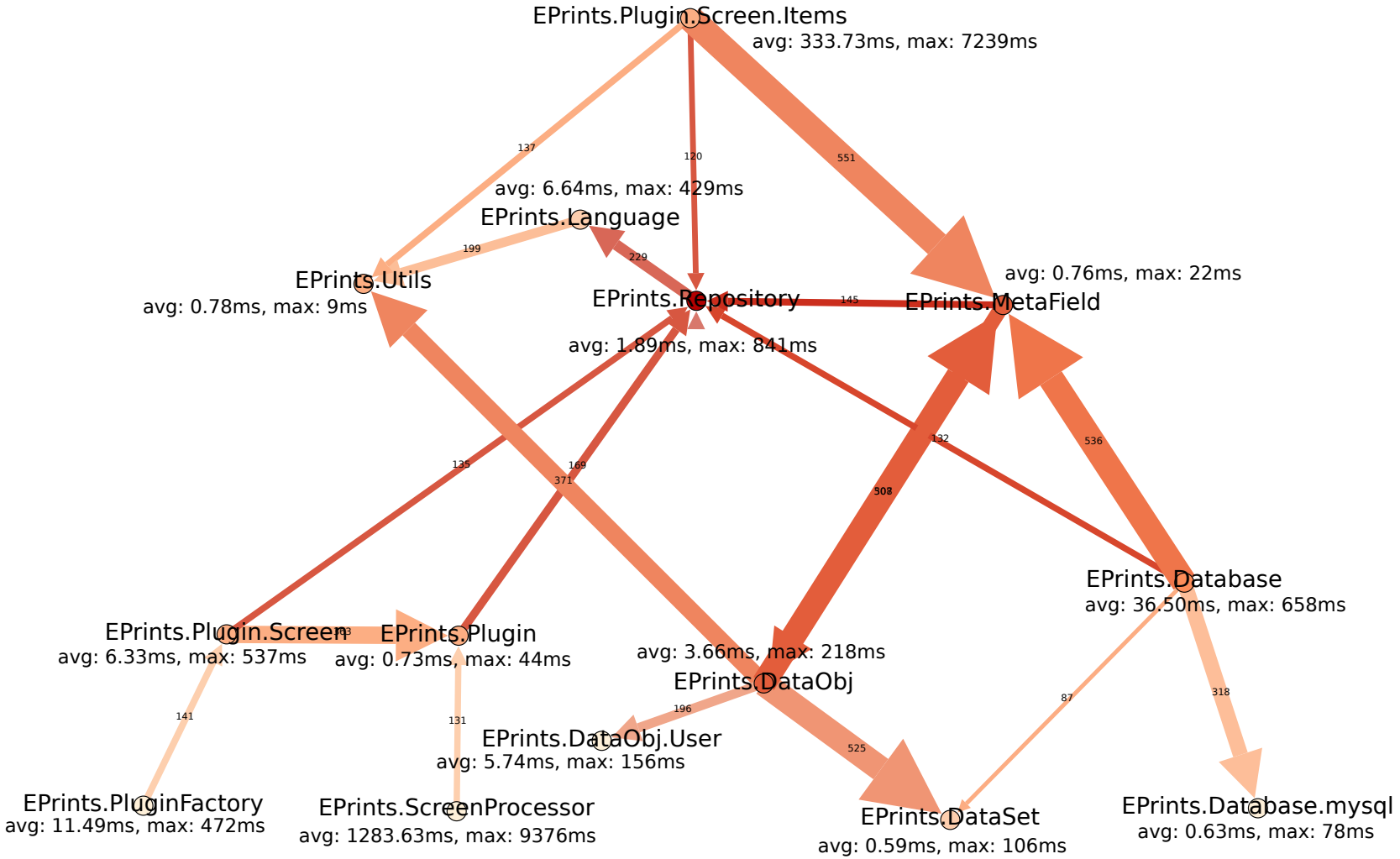

Figure 3: Reduced component dependency graph using gephi [8]

System Architecture Level II

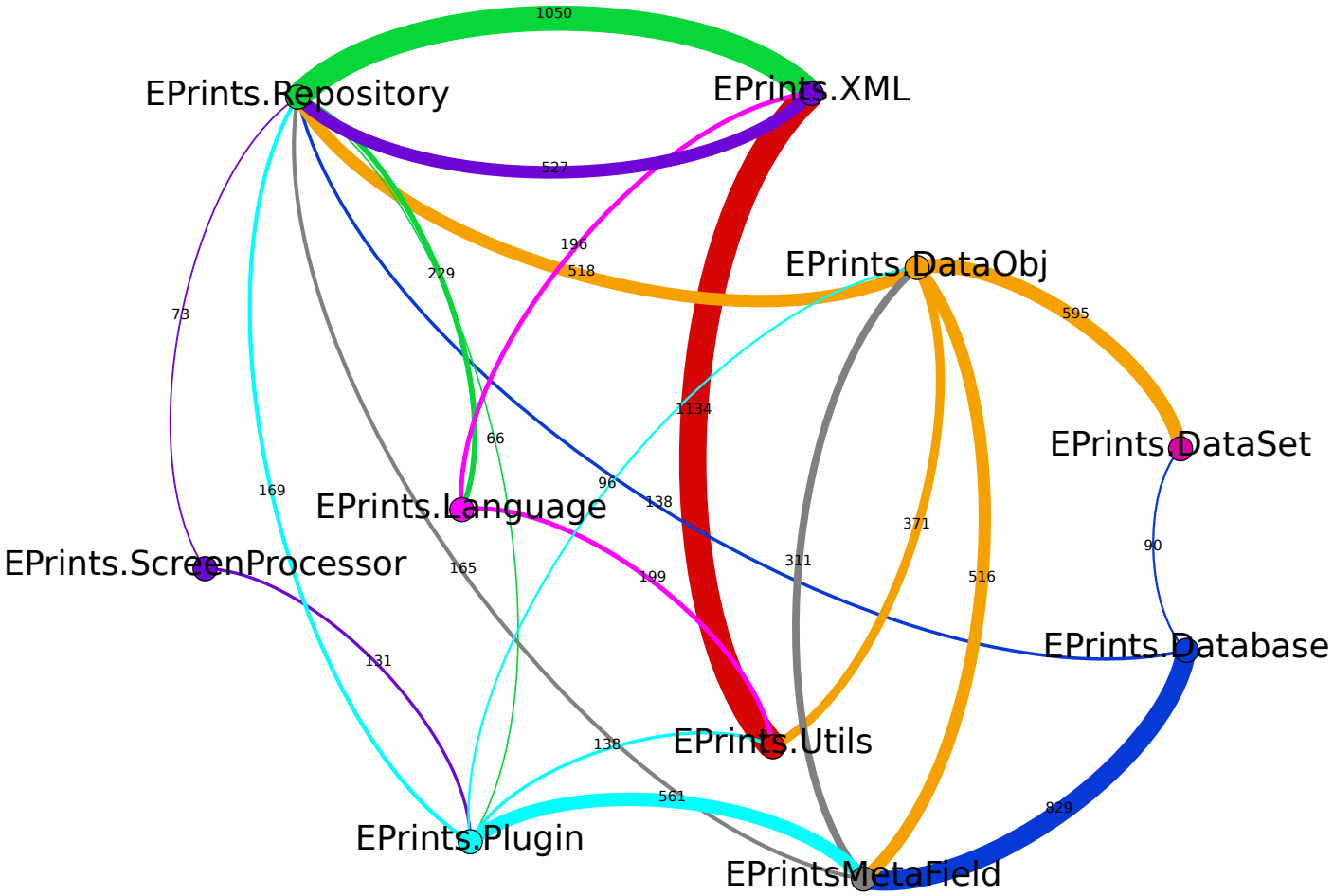

Figure 4: Reduced component dependency graph using gephi [8]

System Architecture Level III

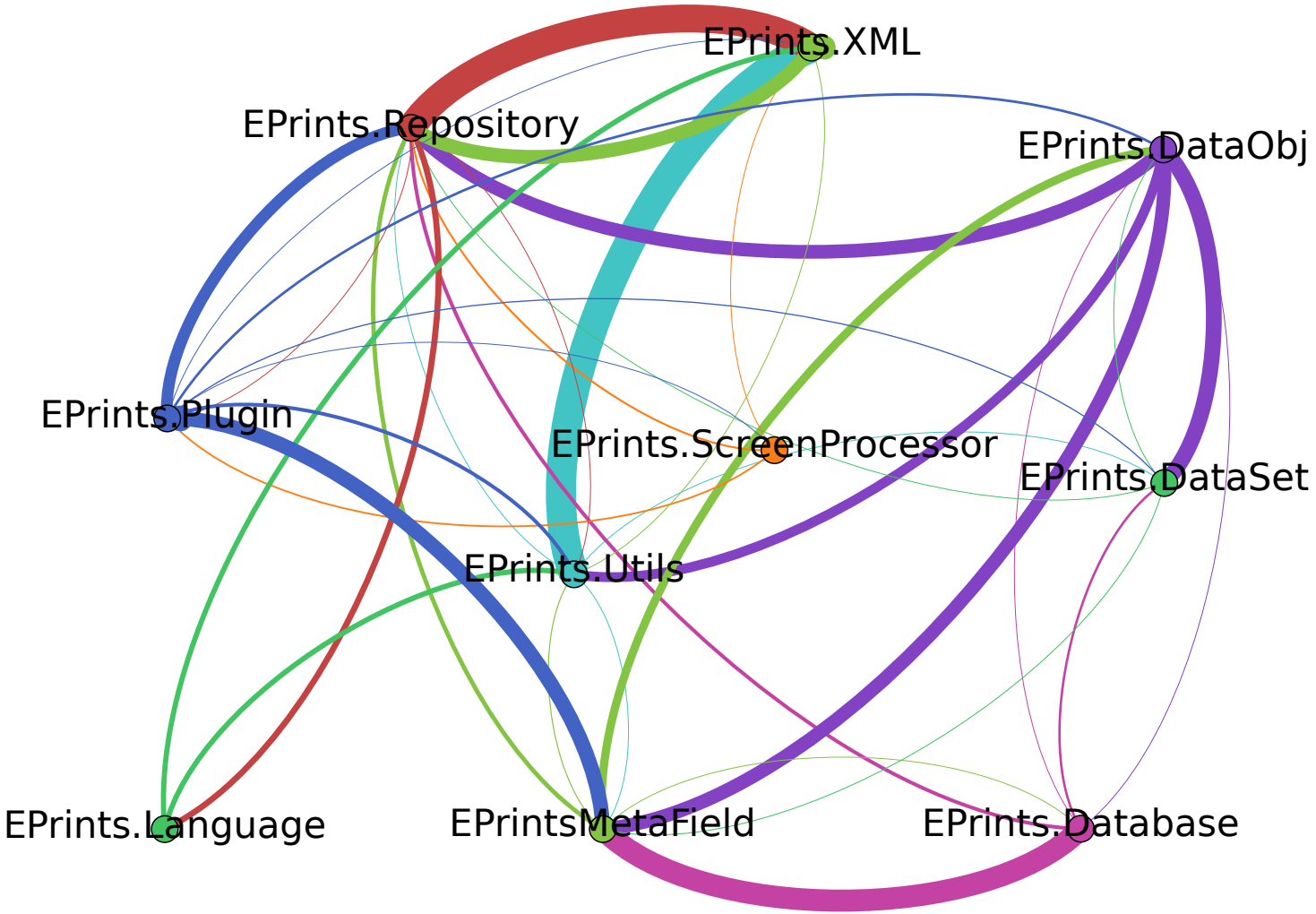

Figure 5: Reduced component dependency graph using gephi [8]

Detailed instrumentation

Let's take a deeper look at Screen::Items::*...

Operation level: Screen::Items

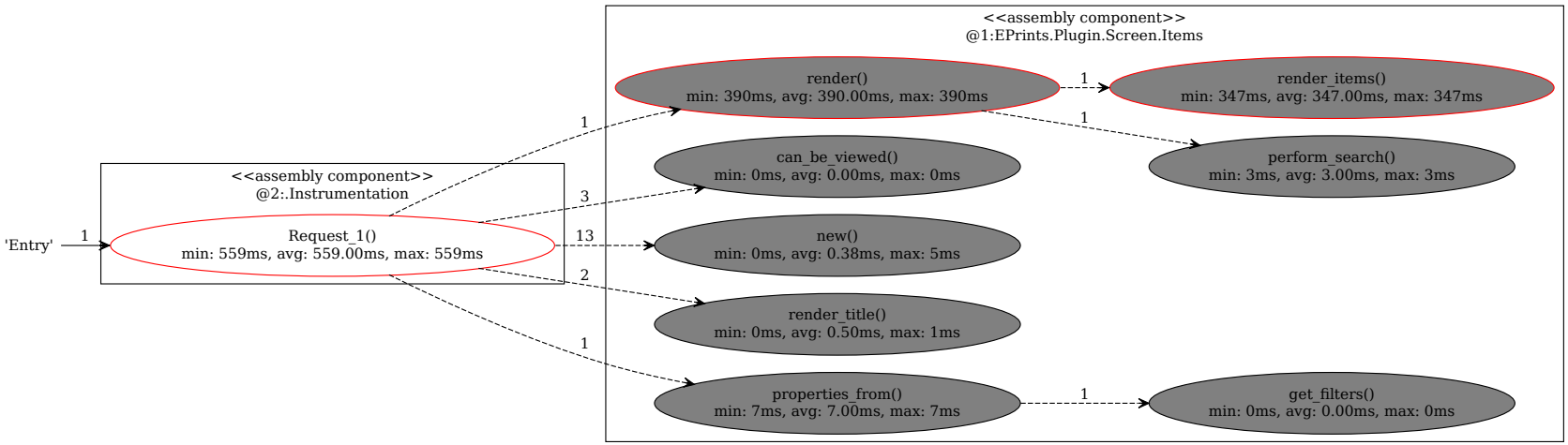

Figure 6: Operation dependency graph for Screen::Items::*

Operation level: Screen::Items II

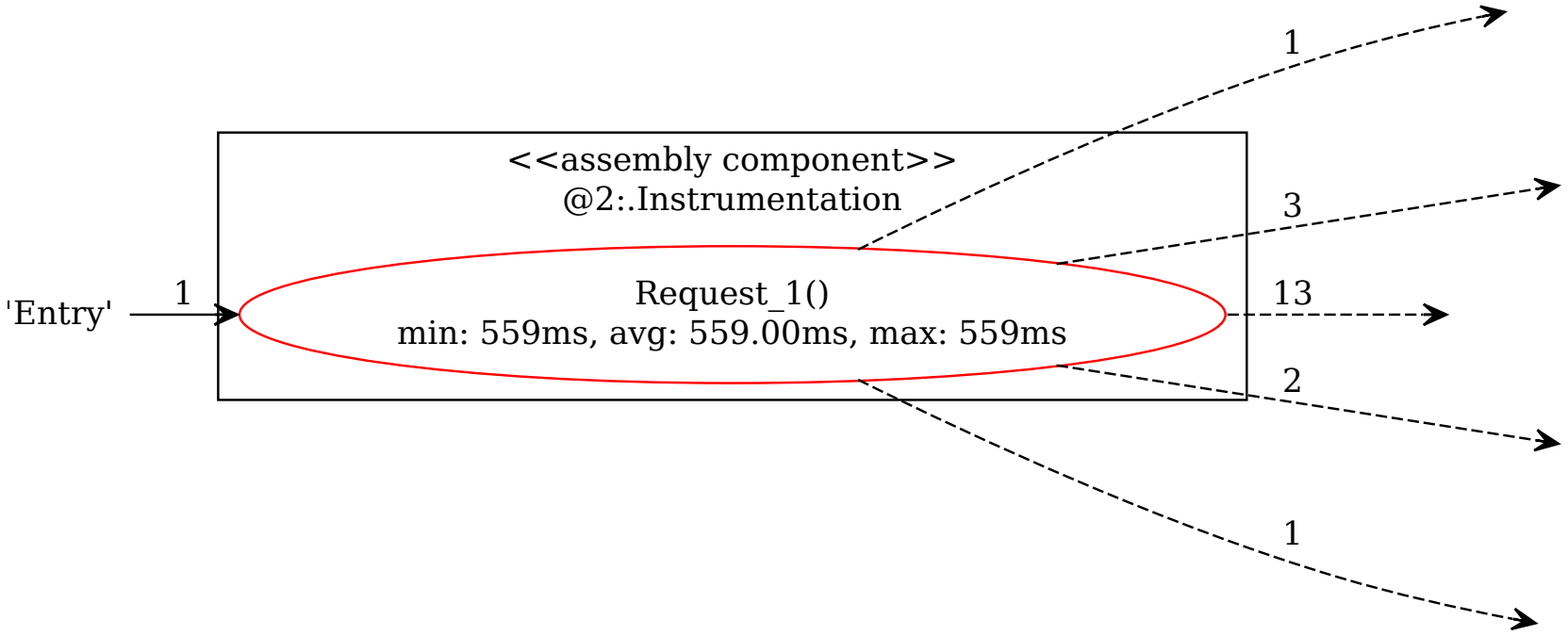

Figure 7: Operation dependency graph for Screen::Items::* left sector

Operation level: Screen::Items III

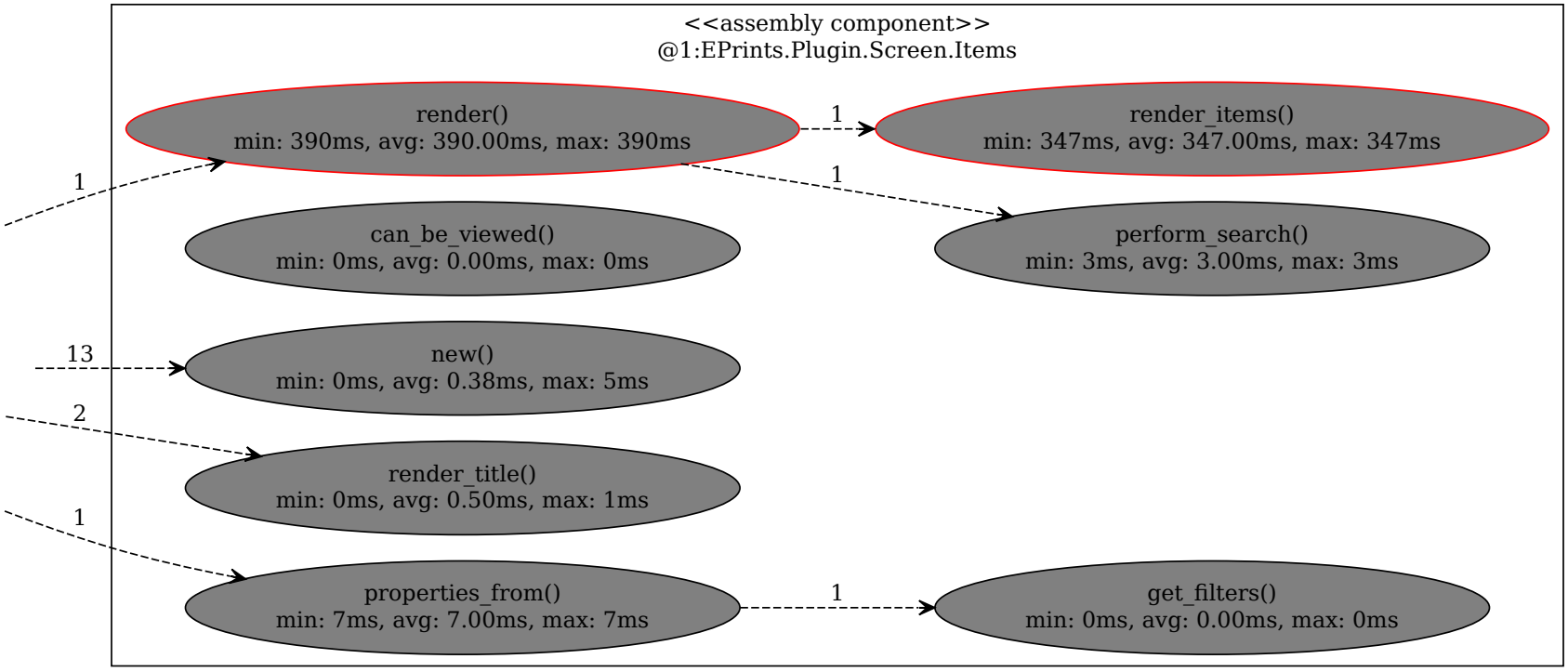

Figure 8: Operation dependency graph for Screen::Items::* right sector

Operation dependencies

What's causing these high response times?

Operation level: Dependencies to EPrints::Database::*

Overall request response time: 1128ms

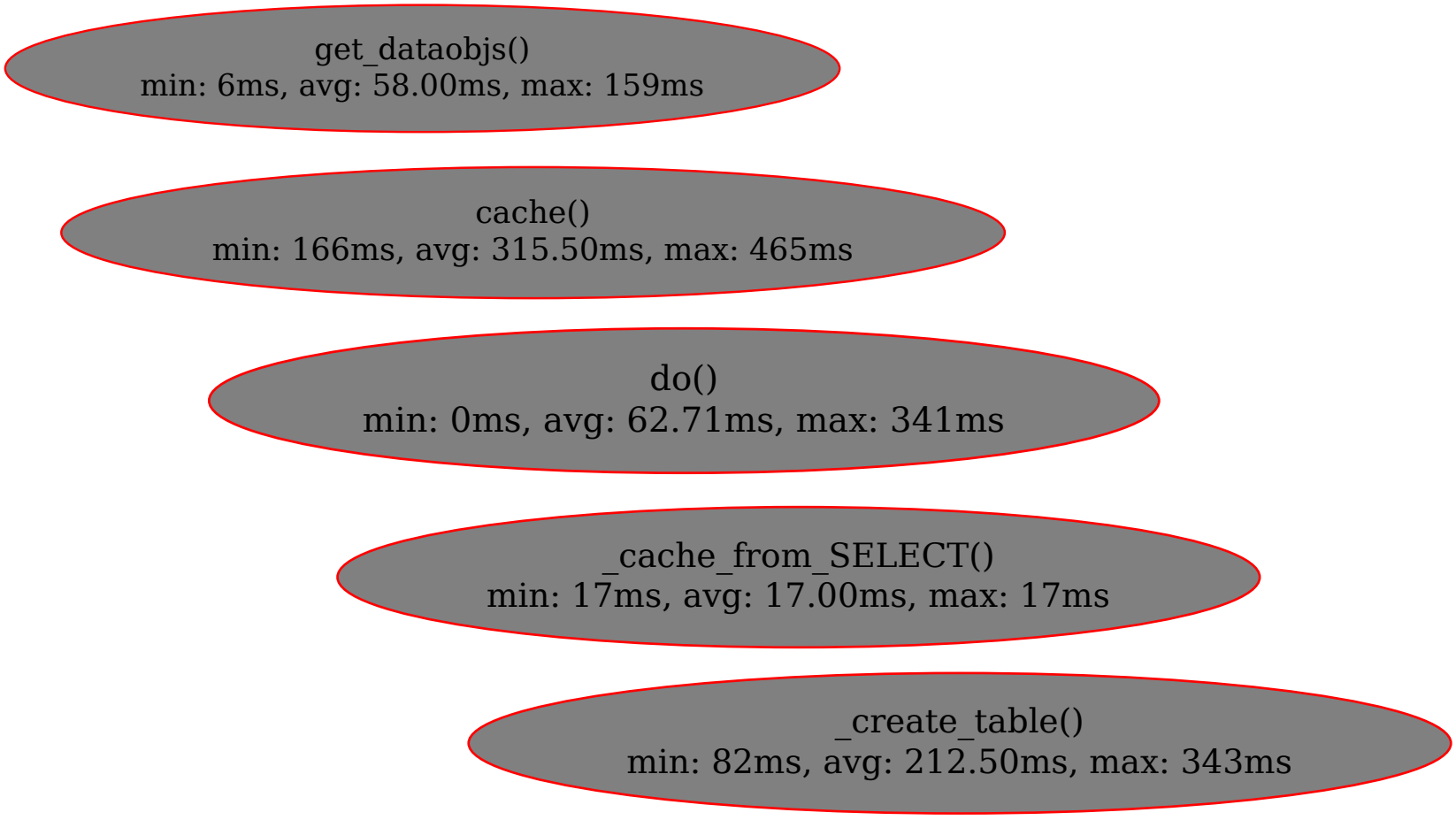

Figure 9: Operation dependencies with focus on Database packages

Operation level: Dependencies to EPrints::MetaField::*

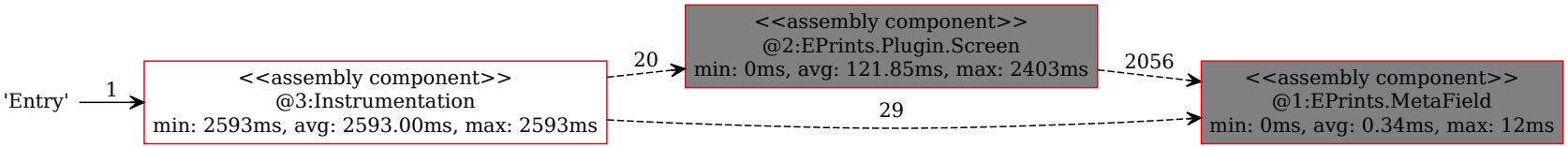

Figure 10: Operation dependencies with focus on MetaField packages

Operation level: Dependencies to EPrints::MetaField::*

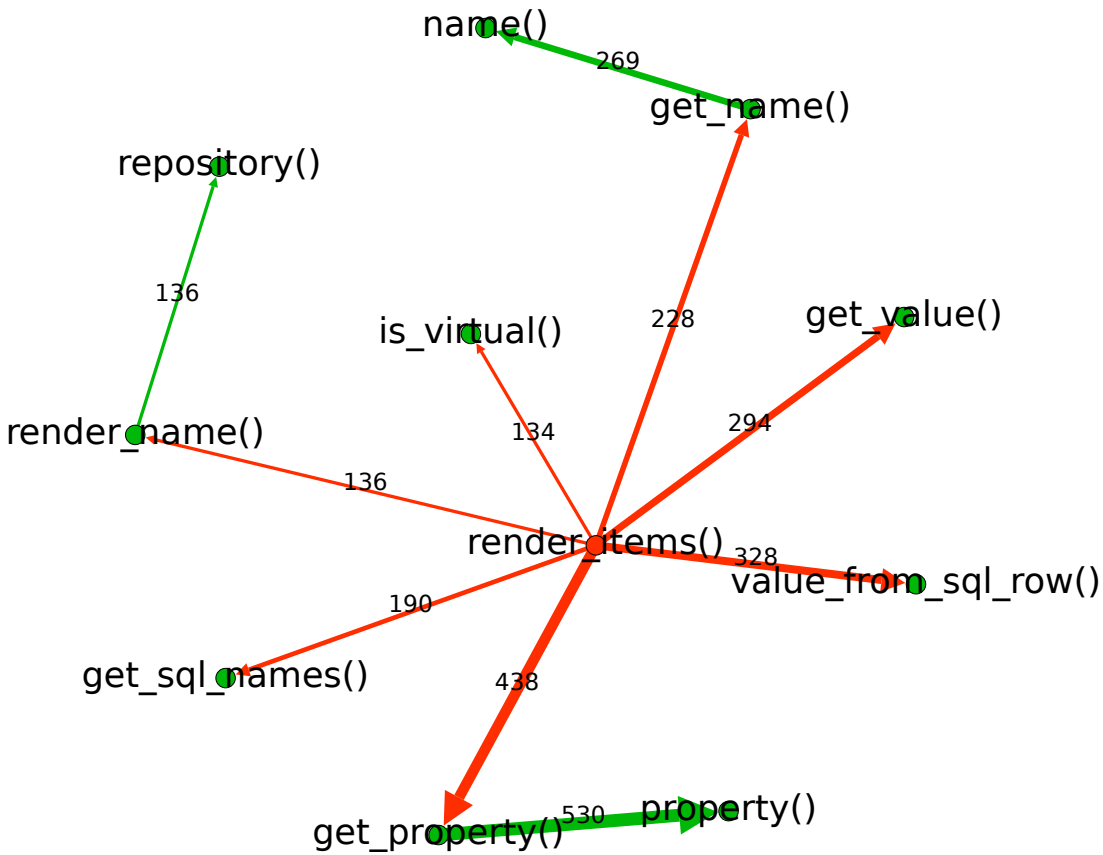

Figure 11: Reduced operation dependency graph using gephi [8]

Operation level: Dependencies to EPrints::MetaField::*

Table 1: MetaField package calling numbers and response times in ms

#calls	operation	response time		
		avg.	max.	sum. (avg)
136	render_name()	0.54	5.00	73.44
136	repository()	0.04	5.00	5.44
140	is_virtual()	0.09	8.00	12.60
197	get_sql_names()	0.26	11.00	51.22
269	get_name()	0.42	6.00	112.98
303	get_value()	0.19	7.00	57.57
334	value_from_sql_row()	0.33	12.0	110.22
530	get_property()	0.40	8.00	212.00

Instrumentation

What needs to be done?

Work flow

- instrument the interesting code
  - (manually) set up some monitoring probes
- perform the instrumentation
- use the Kieker Data Bridge to convert the records
- analyse the recorded data

Environmental set-up

- set up a virtual machine with EPrints
- integrate Kieker into EPrints architecture
- configure an JMS provider like Apache ActiveMQ

Instrumenting EPrints

```
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        };
```

Figure 12: Perl code snippet: weaving Kieker into EPrints

Converting the records using the Kieker Data Bridge

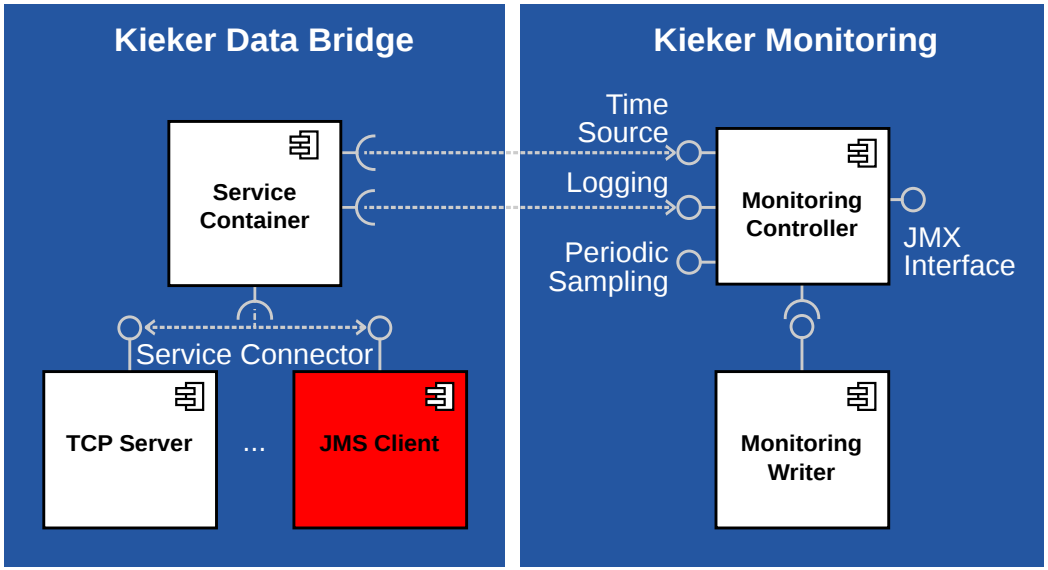

Figure 13: Component diagram of the Kieker Data Bridge [7]

Analysis

Analysing is the most important step...

Analysis

Kieker.TraceAnalysis

- reconstructs and visualizes architectural models based on recorded data
- Software architectural diagrams
  - Dependency graphs (component- and operation-level)

Conclusion

- some bottlenecks are detected
- some are already known, but not all of them
- interesting practical case to test and modify the Perl module

Outlook

What further options are existing?

Outlook

- further performance analysis may be useful
  - especially other requests
- use it as a kick-off to perform
  - Continuous Monitoring
  - Continuous Integration
- integrate into development process

References I

Wechselberg, Nis Börge.

Monitoring von Perl-basierten Webanwendungen mittels Kieker.
http://eprints.uni-kiel.de/21141, Bachelorthesis, April 2013, Kiel University.
[Online; accessed 07-May-2014].

Hasselbring, Wilhelm.

Application Performance Monitoring and Architecture Discovery with Kieker.
http://eprints.uni-kiel.de/24382, April 23th, 2014, in WAIS Seminar, University of Southampton.
[Online; accessed 07-May-2014].

André van Hoorn, Jan Waller, and Wilhelm Hasselbring.

Kieker: A framework for application performance monitoring and dynamic software analysis.
In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE 2012)*, pages 247–248. ACM, April 2012.

André van Hoorn, Matthias Rohr, Wilhelm Hasselbring, Jan Waller, Jens Ehlers, Sören Frey, and Dennis Kieselhorst.

Continuous monitoring of software services: Design and application of the Kieker framework.
Technical Report TR-0921, Department of Computer Science, Kiel University, Germany, November 2009.

References II

Kieker Project.
Kieker web site.
http://kieker-monitoring.net, 2014.
[Online; accessed 07-May-2014].

Kieker Project.
Kieker user guide.
http://eprints.uni-kiel.de/16537/, April 2014.
[Online; accessed 07-May-2014].

Kieker Project.
Kieker DevGuide, Kieker Data Bridge web site.
http://kieker.uni-kiel.de/trac/wiki/DevGuide/kdb.
[Online; accessed 07-May-2014].

Gephi Consortium.
Gephi quickstart guide.
http://gephi.org/tutorials/gephi-tutorial-quick_start.pdf.
[Online; accessed 02-May-2014].

Modifications by the EPrints team

What modifications have been developed?