Everything in Sight: Kieker's WebGUI in Action – Tutorial Paper –

Nils Christian Ehmke
Department of Computer Science, Kiel University, Kiel, Germany
nie@informatik.uni-kiel.de

Abstract: In order to process monitoring data, the Kieker framework provides an API to define and execute pipe-and-filter oriented analysis networks. We develop a multi-user web application, which uses this API to allow graphical assembling and execution of Kieker analysis projects. Besides supporting programmers with the configuration of pipe-and-filter networks, the application helps to assemble so-called cockpits. Cockpits visualize results from running analyses and provide a live view on software systems within the web application using widgets. Such cockpits can be a useful tool for administrators to observe a system.

In this paper, we outline Kieker's pipe-and-filter API and introduce the web application. We put a main focus on its features, current development state, and planned future. Using an exemplary analysis, we show the tool and its potentials in action.

1 Introduction

Kieker[1] [vHWH12] is a Java-based framework which provides functionality to monitor and analyze, amongst others, Java, .NET, and COM applications. Various interception technologies, such as AspectJ[2] and Spring[3] AOP, can be used to weave monitoring code into the software. Pipe-and-filter analysis networks can be created, configured, and executed using a Java API.

Larger analysis networks, consisting of tens or even hundreds of plugins, can be difficult to handle though. Using the API to define and configure such networks is possible, but is usually cumbersome and error-prone. Especially maintenance tasks and later modification requests require a suitable graphical editor. Our approach provides a web application for this issue.

However, the assembly and configuration of analysis networks is only one part of the motivation for the web application. The other part is the live visualization of running analyses using widgets, which we designate as displays. Using the application, administrators can observe the behavior of software systems. In combination with anomaly detection [CBK09] filters, it is possible to detect and diagnose failures (e.g., long response times) within a system early.

[1] http://kieker-monitoring.net/
[2] http://eclipse.org/aspectj/
[3] http://spring.io/

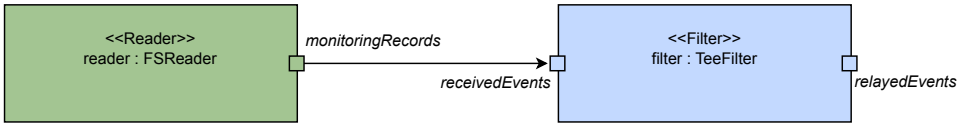

Figure 1: An exemplary network using a file system reader and a filter to print the records

The rest of this paper is structured as follows. In Section 2, we describe the pipe-and-filter architecture of Kieker and the API for analyses. Section 3 contains an introduction to the web application with an exemplary analysis project. Section 4, finally, contains a summary and the outlook.

2 Kieker's Pipes and Filters Architecture

Kieker's pipe-and-filter analysis networks consist of three component-types: readers, filters, and repositories. Readers read monitoring data from a file system, a network, or simply create them on demand. Filters are used to process and create new data within the networks. Repositories are usually shared storages for the filters and readers. They are used, for example, to store a reconstructed system model, which is enriched by various filters.

All of these components can be connected using named input, output, and repository ports. Furthermore, they provide static property keys, which can be used to configure them at initialization time.

Besides the possibility of creating, configuring, and executing analysis networks, Kieker's API allows also to save and load the analyses in specific xml configuration files. It is possible to assemble an analysis, save it as a file, and execute it later or import it for further editing into the web application.

An example, consisting of a file system reader and a filter, printing the records on the terminal, is shown in Figure 1. It can be seen that the creating and connecting of only two components using the Java API (Listing 1) requires already a lot of code to write and maintain. The xml representation, which is used by Kieker to save and load analysis networks, is shown in Listing 2. Unlike the Java code, it can be edited and used without recompilation. However, modifying a xml file without a suitable editor is again very cumbersome and error-prone.

Listing 1: Java code to assemble and execute the network from Figure 1

```
// Prepare the controller for the analysis network
final IAnalysisController controller = new AnalysisController();

// Create and configure the file system reader
final Configuration rConfig = new Configuration();
rConfig.setProperty(FSReader.CONFIG_PROPERTY_NAME_INPUTDIRS,
  "home/nie/monitoring-logs/log-2013-10-03-12-00-00");
```

```
final FSReader reader = new FSReader(rConfig, controller);

// Create and configure the tee filter for the printing
final Configuration fConfig = new Configuration();
final TeeFilter filter = new TeeFilter(fConfig, controller);

// Connect the reader to the filter
controller.connect(reader, FSReader.OUTPUT_PORT_NAME_RECORDS,
  filter, TeeFilter.INPUT_PORT_NAME_EVENTS);

// Execute the analysis
controller.run();
```

Listing 2: A simplified excerpt from the xml (kax) file describing the network from Figure 1

```
<plugins xsi:type="Reader" name="reader"
    classname="kieker.analysis.plugin.reader.filesystem.FSReader">
  <properties name="inputDirs"
    value="home/nie/monitoring-logs/log-2013-10-03-12-00-00"/>
  <outputPorts name="monitoringRecords" id="2" subscribers="3"/>
</plugins>
<plugins xsi:type="Filter" name="filter"
    classname="kieker.analysis.plugin.filter.forward.TeeFilter">
  <outputPorts name="relayedEvents" id="5"/>
  <inputPorts name="receivedEvents" id="3"/>
</plugins>
```

Note that more detailed examples and instructions for the development of own components can be found in [Kie13].

3 Kieker's WebGUI

Kieker's WebGUI[4] has been developed since 2011 and has been part of the Kieker release since version 1.6. It influenced the development of Kieker's analysis part, which resulted in the described pipe-and-filter architecture. It has specifically been designed for the analysis and not for the monitoring part. The web application is based on JSF and Primefaces[5] technologies.

The WebGUI allows to manage multiple projects with separated Java libraries. Analysis networks and cockpit views can be assembled using graphical editors. In the current version, we provide the possibility to visually observe running analyses by displaying results given by the plugins. Furthermore, we implemented a simple access control, including a user management system.

In the following, we use an example to show the application in action. We assemble and

[4] http://kieker-monitoring.net/features/webgui/
[5] http://primefaces.org/

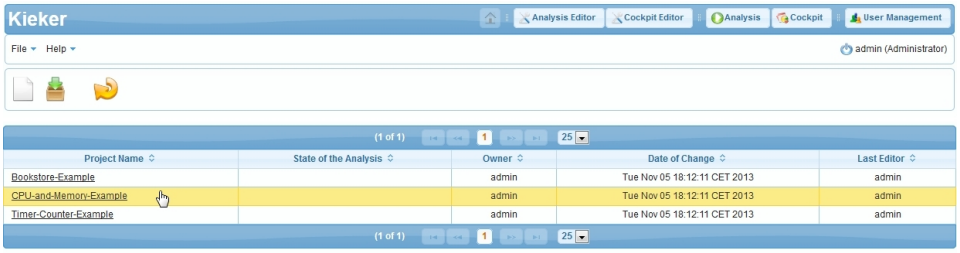

Figure 2: An overview of the available projects

configure the example, create some cockpit views, execute the example, and observe the results. Finally, we present the user management page and detail the user hierarchy.

For the example, we use a reader to read prepared CPU and memory/swap utilization records from the file system. The records are delayed in real-time, based on their corresponding time stamps. Three different filters are responsible for visualizing the monitored data. Note that it would be possible to use, for example, a JMS or a JMX reader instead, to receive online monitoring data from a running software system.

3.1 Managing Projects

An overview page allows to view and manage the analysis projects. A view on this page is shown in Figure 2. It is possible to perform usual operations, like, for example, creating, copying, deleting, renaming, and importing (uploading) projects. The page also shows the last editor, the owner, the last date of change, and the current execution state of each project.

For our example, we simply create a new analysis project and name it *CPU-and-Memory-Example*. The web application creates the necessary files and shows the new project in the project overview list.

3.2 Assembling a Kieker Analysis Project

The analysis editor page, shown in Figure 3, allows to edit a Kieker project in various ways. A visual graph editor is the centerpiece of this page. It can be used to assemble and modify an analysis network. The layout of the network is automatically generated by a graph layouter from Kieler[6], a research project from the real-time and embedded systems group at Kiel University. The plugins from the Kieker framework are already available in the toolbox on the left side of the page. However, it is possible to upload further Java libraries in order to provide additional plugins.

[6]http://www.informatik.uni-kiel.de/rtsys/kieler/

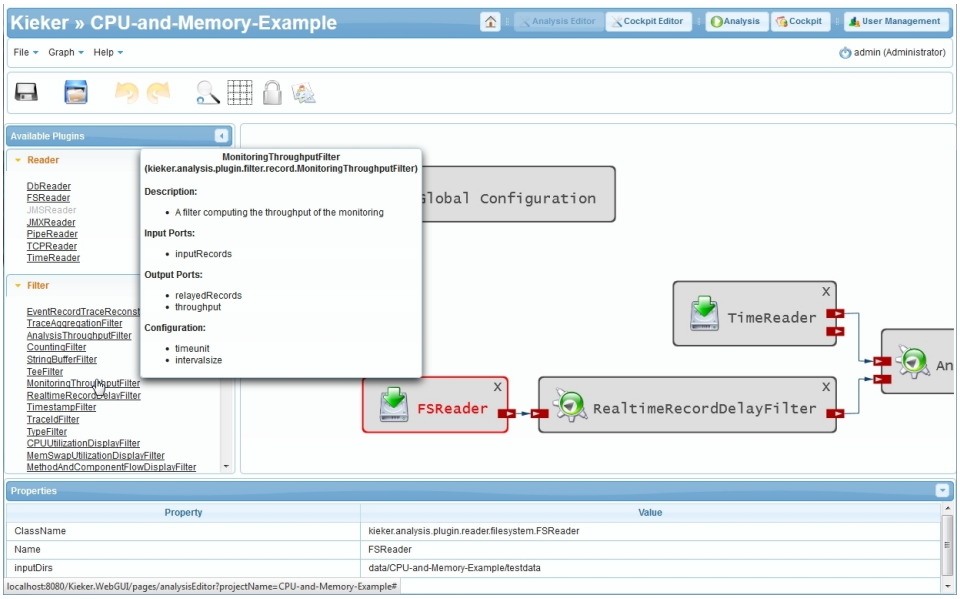

Figure 3: A view on the analysis editor

For our example, we add the necessary plugins by clicking on the corresponding links in the toolbox. We add a file system reader and a filter to delay the records in real-time. Additional filters measure the analysis throughput and provide visualizations for the CPU and memory/swap utilizations. A type filter makes sure that the read records are sent to the correct visualization filters. As the analysis throughput filter visualizes the throughput per time unit, we also add a time reader, which delivers time stamps in regular, configurable intervals.

Some of the plugins have to be configured. For example, we have to configure the reader to read prepared example monitoring log data from the file system. With a click on the reader, an editor for the properties shows up. It can be used to set the input directory of the reader. The remaining plugins can be configured in a similar manner.

As a last step, we have to connect the plugins with each other. Clicking a plugin's port reveals an arrow, which can be linked to the port of another plugin. A click on an existing connection can be used to modify or delete the link.

The result of the created and configured example is a *kax*[7] file, which can already be used by Kieker's analysis API and the provided tools to execute the analysis network.

[7] **k**ieker **a**nalysis **x**ml configuration file

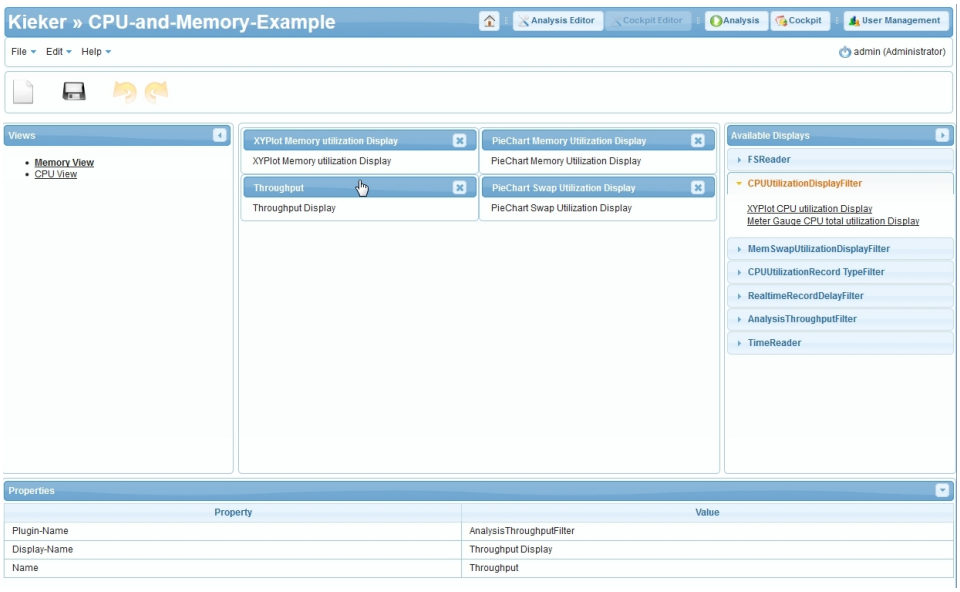

Figure 4: A view on the cockpit editor

3.3 Assembling a Cockpit View

A view on the cockpit editor is shown in Figure 4. Similar to the project overview page, we can create, copy, and delete named cockpit views. Cockpit views are filled with displays. They represent the interfaces between plugins and the web application in order to provide visualizations.

For this example we create two views, *Memory View* and *CPU View*. Once selected, it is possible to add the available displays from the filters with a click and to arrange them in a two-columned grid layout.

For the view on the Memory/Swap utilization, we add four displays: one line chart and two pie charts showing the memory and swap utilization, and one rudimentary text display for the analysis throughput. For the view on the CPU utilization, we use two displays: one line chart and a meter gauge.

3.4 Controlling the Analysis

The analysis network control is shown in Figure 5. On this page, the analysis can be initialized, started, and stopped. It shows the current state (e.g., running or terminated) of the network and presents the corresponding log of the underlying analysis controller. This allows to show, for example, exception stack traces from errors within the analysis controller, which occurred during the initialization. It is very likely that this page will

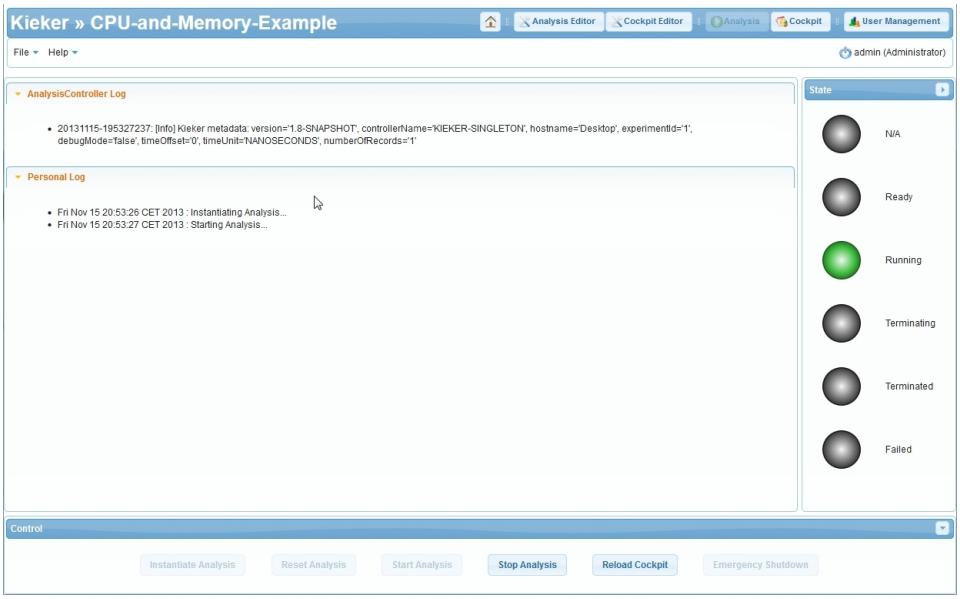

Figure 5: Analysis control view

contain additional information or controls in later versions of the web application.

In the context of our example, we simply use this page to initialize and start the analysis network. Note that it is not possible to instantiate multiple instances of the same analysis project.

3.5 Observing a Running Analysis

The cockpit page is shown in Figure 6. It presents the available views and, once one of the cockpit views has been selected, shows the corresponding visualizations. The four added displays of the Memory/Swap utilization display filter can be seen in the center of the page.

The chart displays can also be configured during runtime by each user. For example, the line chart can be configured to hide some of the shown series or to stack them.

3.6 Managing Users

The web application distinguishes between three types of users: guests, users, and administrators. Each type of user has different authorizations. This allows, for example, to give a guest access to the cockpits, without the possibility for him to control the running

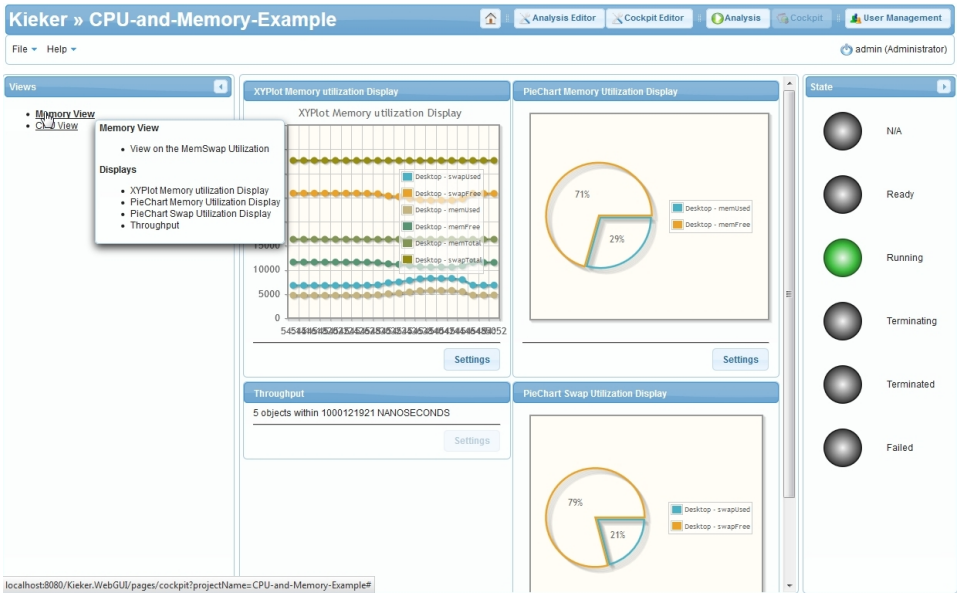

Figure 6: The cockpit of the running analysis

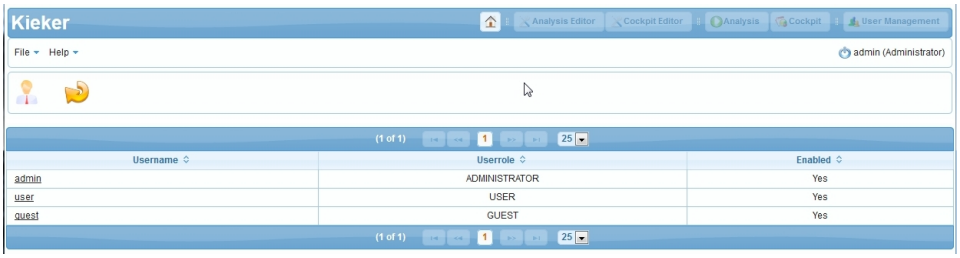

Figure 7: An overview of the available users

analyses. Guests can view the available analysis projects, the analysis networks, and they can access the cockpit views to observe an analysis. In addition to this, users can create, modify, and control the analysis projects at will. Administrators, finally, can furthermore manage the users within the web application.

Figure 7 shows the page for the user management. The page lists all available users and can be used to create new users or edit the existing ones.

4 Conclusion and Outlook

In this paper, we outlined the analysis framework of Kieker and presented the WebGUI project. The WebGUI is a graphical tool to comfortably assemble and configure Kieker analysis networks. It can be used to prepare cockpit views and visualize the live results of running analyses. As it is a web application, it can easily be used by multiple users and can be set up for remote access.

Although the web application is still in development, it already provides a number of features and potentials. Furthermore, the development cycle is synchronized with the development of Kieker. The WebGUI is therefore included in the Kieker releases.

Future development of the WebGUI comprises usability, performance, and stability. As the current display API within Kieker is experimental, we will perform further research in this direction as well. We will provide additional visualization types and add new displays to existing analysis components.

References

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15:1–15:58, July 2009.

[Kie13] Kieker Project. Kieker User Guide 1.8, October 2013.

[vHWH12] André van Hoorn, Jan Waller, and Wilhelm Hasselbring. Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis. In *Proceedings of the 3rd joint ACM/SPEC International Conference on Performance Engineering (ICPE 2012)*, pages 247–248. ACM, April 2012.