Kieker in Eclipse

A Plug-in for Application Performance Monitoring and Dynamic Analysis in Eclipse

Florian Echternkamp
Software Engineering Group
Kiel University
fec@informatik.uni-kiel.de

Christian Wulf
Software Engineering Group
Kiel University
chw@informatik.uni-kiel.de

Abstract

The Kieker framework offers features to monitor and to analyse the runtime behaviour of software systems. In this paper, we propose an associated Eclipse plug-in to ease the usage of Kieker for novice users and to enable the profiling in the Eclipse environment.

The monitoring part of the plug-in enables the automatic integration and configuration of the monitoring via an integrated UI. In this way, an Eclipse project can be seamlessly monitored from within the Eclipse IDE.

The analysis part of the plug-in provides predefined analyses and associated views for reasoning about monitored traces. It allows to sort, to filter, and to search for specific operation calls and traces. Moreover, it seamlessly integrates into Eclipse and thus enables to jump directly to the code of a selected operations call.

1 Introduction

Kieker [1] is an application performance monitoring and dynamic analysis framework, which is widely used for the past 10 years. Besides the fundamental monitoring and analysis components, it provides several tools which provide both a graphical user interface (GUI) and a command line interface. However, none[1] of these tools integrate with an integrated development environment (IDE), such as Eclipse[2]. Furthermore, novice users require unnecessarily much knowledge about configuration and technology details to start Kieker for the first time.

To fill this gap, we propose an Eclipse plug-in [5] for the Kieker framework in this paper. We chose Eclipse as an IDE due to its high market share and popularity. The plug-in is able to monitor an Eclipse project and to analyse the resulting monitoring logs both from within Eclipse. It abstracts from several configuration details and provides multiple built-in analysis views which allow to reason about the monitored application. We describe the monitoring part in Section 2 and the analysis part in Section 3. In Section 4, we conclude this paper and present our future work.

[1] Excluding Kieker's instrumentation record language [2]
[2] https://eclipse.org

2 Monitoring

To monitor an application with Kieker, the application needs to be instrumented with monitoring probes. The recommend way for this purpose is to use the aspect-oriented programming (AOP) framework AspectJ[3] since it is non-intrusive to the application. Usually, the AOP XML file needs to be adapted in a text editor to define the desired pointcuts and aspects. With our new Kieker plug-in, we now provide an additional GUI called Probes Configuration View (shown in Figure 1) which abstracts from the XML notation.

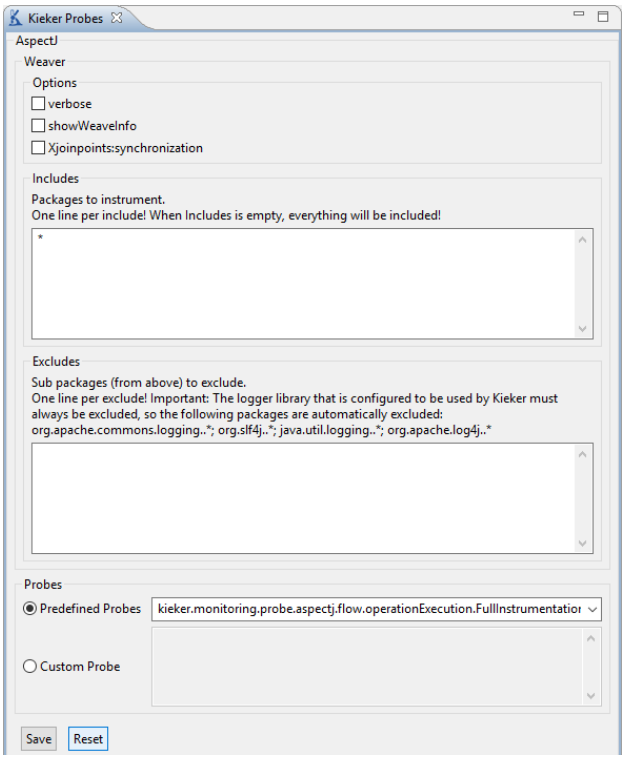

Figure 1: Probes Configuration View

Moreover, the user does not need to know anymore where the AOP XML file should be dropped so that the weaver is able to find it. At the top of

[3] https://eclipse.org/aspectj

the view the weaver can be configured. The options provided by the weaver can be easily activated through checkboxes. Furthermore it can be specified which classes should be included or excluded from the weaver. The logging classes used by Kieker must be always excluded and are automatically added to the excludes by the plug-in. Thus they could not be inadvertently forgotten by the user. At the bottom of the view the probes can be configured. Either one of the predefined probes can be used or a custom probe may be declared. Currently, the GUI is restricted to define custom probes of the type `(..).flow.operationExecution.AbstractAspect` with one single pointcut. To define multiple pointcuts the expressions must be concatenating by the logical "or". Independent of the GUI, it is still possible to let the plug-in create the AOP XML file and edit it manually in the editor.

The Probes Configuration View is accessible through the Package Explorer's context menu (shown in Figure 2) under the *Kieker Monitoring* menu item. The *Kieker Monitoring* menu is only visible if the Kieker Monitoring Nature is added to the Eclipse project via *Configure* → *Convert to Kieker Monitoring Project*.

Besides the Probes Configuration View, our Kieker plug-in provides a quick access to the Kieker properties file (also shown in Figure 2) by opening it with the system's default editor. If the Kieker properties file doesn't exist in the project yet, an initial version is created before opening it. In this way, the user can for example configure the output location of the monitoring logs without having to know whether and where the properties file has been created.

To execute an application enabled with Kieker's monitoring capability, an additional launch configuration must be created in Eclipse. Instead of the standard *Java Application* launch configuration a *Kieker-monitored Java Application* launch configuration is used, which is provided by the plug-in. This custom launch configuration extends the *Java Application* one and adds a Kieker Monitoring tab (shown in Figure 3). Inside the tab, the user can enable the monitoring and declare the Kieker version which should be used. The plug-in expects the AspectJ variant of the Kieker JAR file and checks for the specified version in the project. If the corresponding Kieker file is not available, the user cannot enable the monitoring. Finally, when launching a valid configuration, the plug-in automatically adds Kieker as java agent and appends the AOP XML file to the classpath. In this way, the user does not need to know the launch parameters and the location of the AOP XML file anymore.

3 Analysis

To provide analysis features directly in Eclipse, the tabular views known from the standalone Kieker

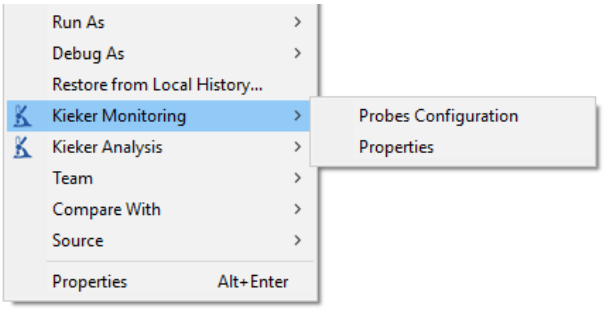

Figure 2: The new monitoring context menu entry provides a GUI for configuring probes and writers

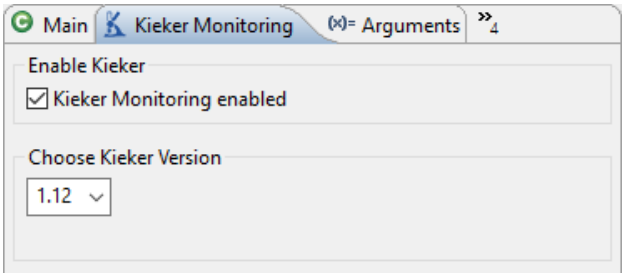

Figure 3: The new custom launch configuration "Kieker-monitored Java Application"

Trace Diagnosis[4] tool were integrated in the Kieker Analysis View (shown in Figure 4). The Analysis View is accessible through the Package Explorer's context menu (shown in Figure 5) under the *Kieker Analysis* menu item. Like the Kieker Trace Diagnosis tool, the analyses are based on the Pipe-and-Filter framework TeeTime [3, 4].

Tabs for Traces, aggregated Traces, Operation Calls, aggregated Operation Calls and Monitoring Log statistics are provided. The tables can be filtered and the table columns can be sorted and rearranged in order by drag and drop. The clarity is improved by only displaying the operation name and an icon for

[4] `https://build.se.informatik.uni-kiel.de/kieker/kieker-trace-diagnosis`

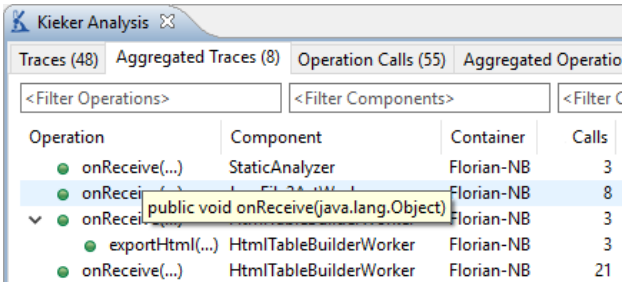

Figure 4: The tab "Aggregated Traces" of the analysis view

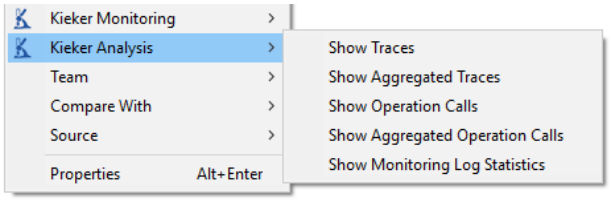

Figure 5: The new analysis context menu entry provides analyses and visualizations within Eclipse

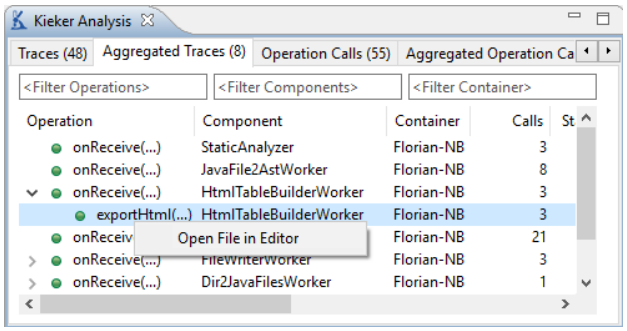

Figure 6: The "Open File in Editor" feature from the analysis context menu

the operation visibility. For a better recognition, the Eclipse iconography for the visibility (private, public, etc.) is used. The full operation identifier is displayed through a mouse hover tooltip. However, it can also be displayed permanently in the plug-in preferences. The tabs are partially connected with each other. Currently, it is possible to jump to the Operation Calls tab and set the filters based on a selected aggregated operation call. Moreover, the user can jump to the Traces tab based on a selected operation call and highlight this call in the trace. The integration of Kieker Analysis into Eclipse allows to directly interact with the IDE. Thus the *Open File in Editor* context menu item, provided by almost all tabs, lets the user automatically open the corresponding Java file inside the Eclipse project and mark the code area of the selected operation (shown in Figure 6 and Figure 7).

4 Conclusion

In this paper, we propose an Eclipse plug-in for the Kieker framework. Its main purpose is to ease the usage of Kieker for novice users and to enable the close-to-code profiling in the Eclipse environment.

As future work, we plan to enhance the "Probes Configuration View" to support custom probes of arbitrary types, not just of the `(..).flow.operationExecution.AbstractAspect` type. Moreover, we plan to ease the definition of multiple pointcuts without the use of logical "or"s. For users who do not use the build tools Maven or

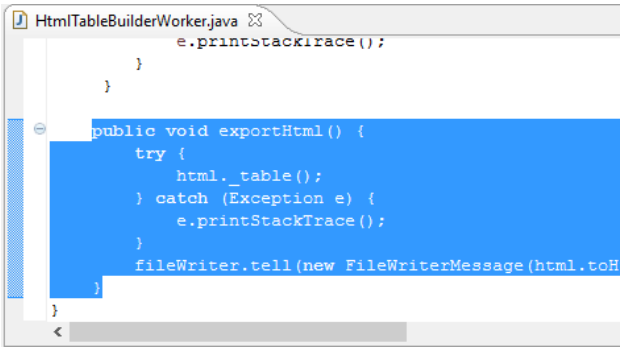

Figure 7: The code area of the selected operation call in Figure 6

Gradle, we plan to add a download button which triggers the download of the selected Kieker version from the central Maven repository.

Inside the analysis views, it is currently possible to select an operation call and to jump to the corresponding lines of code in a Java editor view. To directly provide the performance information, the Java editor could be extended with overlay annotations for each monitored operation.

Complementary to the tabular representation of the analysis views, we plan to add a graph visualization for the trace analysis. In the long term, we intend to adapt the plug-in in a way so that it is able to perform arbitrary analyses with any associated views.

References

[1] A. Van Hoorn, J. Waller, and W. Hasselbring. "Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis". In: *Proc. of the ICPE*. 2012.

[2] R. Jung. *An Instrumentation Record Language for Kieker*. Tech. rep. Kiel University, Aug. 2013.

[3] C. Wulf, N. C. Ehmke, and W. Hasselbring. "Toward a Generic and Concurrency-Aware Pipes & Filters Framework". In: *Symposium on Software Performance 2014: Joint Descartes/Kieker/Palladio Days*. Nov. 2014.

[4] C. Wulf, C. C. Wiechmann, and W. Hasselbring. "Increasing the Throughput of Pipe-and-Filter Architectures by Integrating the Task Farm Parallelization Pattern". In: *Proceedings of the 19th International ACM SIGSOFT Symposium on Component-Based Software Engineering (CBSE 2016)*. IEEE, Apr. 2016, pp. 13–22.

[5] F. Echternkamp. *Kieker Eclipse Plug-in*. URL: `https://build.se.informatik.uni-kiel.de/kieker/kieker-eclipse-plugin`.