Kieker– Roadmap

Matthias Rohr, André van Hoorn, Nina S. Marwede

November 18, 2008

http://www.matthias-rohr.com/kieker/roadmap.pdf

This document describes the changes in current and old releases and outlines plans for future releases of Kieker's monitoring component *Tpmon*. Kieker is an open source project for the monitoring and visualization of Java applications. It's major target are Java (Web) applications. Details can be found in the tutorial[1].

Current and past releases overview:

Version number	Intended date	Major feature
0.91	11/2008	Completely re-engineered and re-documented (tutorial) release. Aspects are now in @Aspect-style and use ThreadLocal. Performance was also improved. Support for monitoring in the spring framework was extended.
0.2	01/2008	Improved dependability
0.1	11/2007	Initial release with basic command line functionality

Future releases:

Version number	Intended date	Major feature
0.98	01/2009	Database writer tests; usability improvements; graphical user interface; server-side runtime model; support for cloud computing.

1 Changes in current release

1.1 Release 0.91

1. The data type of threadid was changed to long. This heavily reduces the amount of memory and computation required for post-processing.

[1] http://www.matthias-rohr.com/kieker/KiekerTutorial.pdf

2. From now on, Tpmon uses @Aspect-style for the aspects (cross-cutting code) instead of the old-style .aj AspectJ files. The advantage of the @Aspect-style is that the aspect files can be directly handled by Java IDEs and Compilers.

3. The data structure KiekerExecutionRecord now is used all the way from the measurement point, through the controller to the selected monitoring data writer and queues of writers that use producer-consumer pattern. This avoids type-casting and improves consistency through the architecture.

4. A new property allows the user to select the monitoring writer:
   `monitoringDataWriter=SyncFS|AsyncFS|SyncDB|AsyncDB`
   The value of this property can be either one of the constants shown below or a full-qualified classname of a class (implementing kieker.tpmon.IMonitoringDataWriter) loaded dynamically during runtime (must be in the runtime classpath). Classes loaded by the classname are constructed by calling the default constructor followed by a called to the method init(String monitoringDataWriterInitString) with the value of the below-listed property monitoringDataWriterInitString.

   Existing constants are:

   - SyncFS (Synchronous File System Writer)
   - AsyncFS (Asynchronous File System Writer)
   - SyncDB (Synchronous Database Writer)
   - AsyncDB (Asynchronous Database Writer)

   The old properties to activate asynchr. FS and DB respectively are removed. Switching to *Tpmon* 0.91 requires to replace old tpmon.properties wit a new one based on tpmon.properties.examples of 0.91.

5. The experimental JMS Writer prototype is not anymore part of the SourceForge distribution of *Tpmon*.

2 Changes in old releases

2.1 Release 0.2 - ”Improved Dependability for Tpmon” 01/2008

1. Input
   a) Asynchronous access to Tpmon database (schema 0.7+)
   b) On-the-fly conversion into Message traces

2. Quality
   - Additional test cases
   - One month test run

2.2 Release 0.1 – “Initial release” 11/2007

1. Visualization
   a) UML Sequence Diagrams for single Message Trace (generates code for pic2plot)
   b) Markov Chain from Collection of Message Traces (generates GraphViz code)
   c) Dependency Graphs from Collections of Message Traces (generates code for GraphViz)
2. Output (Code generation for external tools)
   a) Storing visualization code in /tmp (or other path configured)
   b) Showing in console command to create image file from visualization code
3. Input
   a) Synchronous access to Tpmon database (schema 0.7+)
   b) Access traces by experimentid, traceid, tin, operation, sessionid
4. Quality
   a) Complete basic JavaDoc
   b) Testing
      i. Datainput
      ii. Several unit tests for the core of Tpan (creation of message traces and Execution Traces)
      iii. One unit test for each plugin
      iv. One end2end unit test testing the 4 core plugins and dataimport
   c) Handle fault scenario ”no database driver”
   d) Handle fault scenario ”database username or password wrong?”
5. User documentation
   a) Tutorial Tpmon
   b) Tutorial Tpan
   c) Manual Tpmon
   d) Use cases
   e) Alternative configurations
   f) Manual Tpan
6. Developer Documentation
   a) Developer manual Tpmon
   b) Developer manual Tpan
   c) Roadmap