Kieker Data Bridge and Instrumentation Language

Kieker Workshop

Reiner Jung

Christian-Albrechts-Universität zu Kiel
Institut für Informatik

06.03.2013

Motivation

- Kieker
  - Primarily supports Java
  - Special solutions for some languages
- Every new languages have to implement
  - Monitoring records & probes
  - Record translation
  - Record transmission
  - Weaving mechanism

Present Solutions

- Kieker.4com VisualBasic 6
- Kieker.4net C#
- Cobol-Dialects

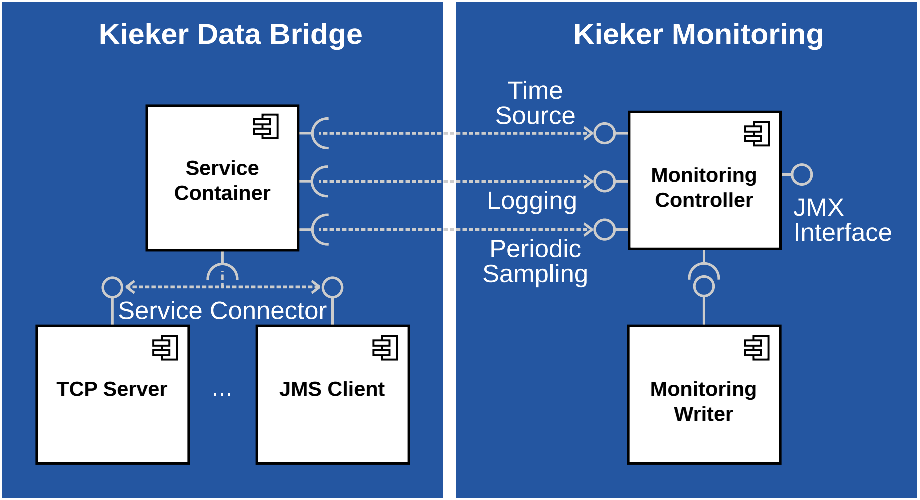
Goal

Goal Establish a standard way to add new languages and platforms

Solution

- Kieker Data Bridge
- Instrumentation (Record) Language
- Weaver Collection

Kieker Data Bridge



Service Connectors

TCP Client Connects to a remote service on startup

TCP Single Server Listens for one client

TCP Multi Server Handles multiple clients

JMS Client Connects to a JMS queue

JMS Embedded Start a JMS service and connects to it

Service Container

Input

- Kieker Configuration
- Service Connector

Main Loop

1. Setup Kieker
2. Setup service connector
3. Get record
4. goto 3 if not terminated
5. Close service connector
6. Shutdown Kieker

Service Container

Other Features

- Connector respawn
- Progress monitor support
- Load record types at startup
- Embeddable container

Service Implementations

CLI Server

- Command line application
- Read class id mapping from ASCII file
- Can run as deamon

Eclipse Plugin

- Eclipse job & run configuration
- Class mapping setup in run configuration

Serialization Format

General Structure

- First value **type id** (int32)
- Other values in order of declaration

Kieker	fields expressed in TYPES
Other	reflection API (non static fields)

References

- Id only
  - First byte = 0
  - Second value **type id** (int32)
  - Unique object run-time id
- Containment
  - First byte = 1
  - Second value **type id** (int32)
  - Other values in order of declaration (Java only)

Serialization Format

Binary Format

- Based on **Java base-types**
- Byte order **big endien** (network byte order)
- String composed of

length	32bit signed integer (int)
data	variable length byte vector

Text Format

- Semicolon separated value list

Service Connector API

```
public interface IServiceConnector {

/** setup connector */
void setup() throws Exception;

/** close connector */
void close() throws Exception;

/** get next record */
IMonitoringRecord deserialize() throws Exception;
}
```

Service Connector API

```
public interface IServiceConnector {

/** setup connector */
void setup() throws Exception;

/** close connector */
void close() throws Exception;

/** get next record */
IMonitoringRecord deserialize() throws Exception;
}
```

Service Connector API

```
public interface IServiceConnector {

/** setup connector */
void setup() throws Exception;

/** close connector */
void close() throws Exception;

/** get next record */
IMonitoringRecord deserialize() throws Exception;
}
```

Service Connector API

```
public interface IServiceConnector {

/** setup connector */
void setup() throws Exception;

/** close connector */
void close() throws Exception;

/** get next record */
IMonitoringRecord deserialize() throws Exception;
}
```

Instrumentation Language

Goals

- Language independent record notation
- Annotate nodes of arbitrary models/ASTs

Instrumentation Language

Goals

- Language independent record notation
- Annotate nodes of arbitrary models/ASTs

Requirements

- Source language meta model independent
- Define probes for meta-model classes (nodes)
- Define annotations (like AspectJ)

Language Features

Generation of

- Type compatible record types across languages
- Serialization functions

Supports

- Java (example generator, run-time environment present)
- C (example probe code)
- Perl (example probe code)

Language Independent Record Notation

```
package kieker.common

record OperationExecutionRecord {
    default string NO_SESSION_ID = "<no−session−id>"
    default long NO_TRACEID = −1
    default long NO_HOSTNAME = "<default−host>"
    default long NO_TIMESTAMP = −1
    default int NO_EOI_ESS = −1

    string operationSignature
    string sessionId = NO_SESSION_ID
    long traceId = NO_TRACEID
    long tin
    long tout
    string hostname = NO_HOSTNAME
    int eoi = NO_EOI_ESS
    int ess = NO_EOI_ESS
}
```

Language Independent Probe Notation

```
package kieker.common

model java "http://www.eclipse.org/JvmTypes"

import kieker.common.OperationExecutionRecord

probe OperationExecutionProbe : java::MethodDeclaration {
    use OperationExecutionRecord
}
```

Weaver

Weaver Technologies

- AspectJ
- Perl-Weaver (Nis)
- AspectC or other C weaver

Question Do we need a generic weaving language?

Summary

- Kieker Data Bridge
  - Multi protocol support
  - Serialization method
  - Extendable record library
  - Two use cases in Perl and C
- Instrumentation Language
  - Platform independent record notation
  - Generator for Java (experimental)

To Do List

- Kieker Data Bridge
  - Improve documentation
  - Refactor to meet Kieker package naming
  - Integrate into Kieker distribution
  - Support for adaptive monitoring
  - Support for AJAX/HTTP connection
- Instrumentation Language
  - Finalize grammar (checks and type evaluation)
  - Generator for Perl & C
  - Finalize generator for Java
- Kieker
  - C run-time library and instrumentation (thesis)
  - Perl run-rime package