Analyzing Eclipse Applications with Trace Compass

MARC-ANDRÉ LAPERLE, ERICSSON
Marc-André Laperle
- Software Developer at Ericsson since 2013
- Worked in various software development positions
- Eclipse Committer for Trace Compass, CDT and Linux Tools
- Occasional contributor to other projects (Platform UI, SWT, EGit, Mylyn, PDE)
ABOUT TRACING

› Tracing records information about program’s execution

› Useful for debugging, troubleshooting, understanding a system

› Static instrumentation
  - inserted before compile-time, activated at run-time

› Dynamic instrumentation
  - inserted and activated at run-time
TRACING USE CASES

› Finding cause of
  - Failures, crashes
  - Concurrency issues
  - Performance issues

› Live monitoring of system in production
  - Raising alarms, warnings
  - Resource usage (e.g. CPU load)
  - Overload protection

› System-wide troubleshooting
  - Multi-core, multi-processor, multiple nodes, multiple layers, etc.
TRACE COMPASS

› Used to be part of Eclipse Linux Tools Project (LTTng)

› New project Trace Compass at Eclipse Foundation
  - To increase community and collaboration in open-source
  - Larger scope, beyond LTTng, Linux

http://www.eclipse.org/tracecompass
TRACE COMPASS

› Framework to build trace visualization and analysis tools

› **Scalable**: handle traces exceeding memory

› **Extensible** for any trace or log format
  - Binary, text, XML etc.

› **Reusable** views and widgets

› Available as **standalone application** or set of plug-ins
COMMON FEATURES

› Management of traces, trace formats and experiments
COMMON FEATURES

› Package export and import
COMMON FEATURES

› Events Table

 Implemented as an Eclipse “editor”
COMMON FEATURES

Searching

Filtering

Highlighting
COMMON FEATURES

› Bookmarks
COMMON FEATURES

› **Sequence Diagrams**
  - Translates events to sequence diagram transaction
  - Extensions can define their own model
State system abstracts events, analyses traces and creates models to be displayed.
CONTROL FLOW VIEW

› Displays processes state changes (color-coded) over time
  
  USERMODE, SYSCALL, INTERRUPTED, WAIT FOR CPU, etc
RESOURCES VIEW

› Displays system resource states (color-coded) over time
CPU USAGE VIEW

› Displays % of CPU used per thread over time
CALL STACK VIEW

› Shows the stack trace at any point during execution
COMMON FEATURES

› Custom Text and XML Parsers
  - Line based parser with regex defined in a wizard
  - XML based extracting data from XML elements and their attributes
COMMON FEATURES

› Data-driven state provider
  - XML description of state changes to convert trace events to states
  - Can be created **without changing source code** or recompiling
COMMON FEATURES

› Data-driven state system based view
  - XML description of view representation of the computed state system
  - Can be created without changing source code or recompiling
COMMON FEATURES

› For example: 50 lines of XML created the view below
COMMON FEATURES

› A graphical editor for defining state providers is in the works
AN EXAMPLE

› A Trace Compass event(s) request
AN EXAMPLE

When many concurrent requests...
An Example

Trace points

```java
String message = "CREATED 
+ " Type=" + type + " Index=" + getIndex()
+ " NbReq=" + getNbRequested()
+ " Range=" + getRange()
+ " DataType=" + getDataType().getSimpleName();
TmfCoreTracer.traceRequest(fRequestId, message);
```

[1418416987.948] [TID=037] [REQ] Req=6 CREATED (FG) Type=TmfEventsCache
Index=0 NbReq=2000 Range=TmfTimeRange [fStartTime=19:12:43.145 224 192,
fEndTime=19:47:16.854 775 807] DataType=ITmfEvent
AN EXAMPLE

› A bit more digestible:

[1418416987.948] [TID=037] [REQ] Req=6 CREATED (FG)
Type=TmfEventsCache Index=0 NbReq=2000 Range=TmfTimeRange
DataType=ITmfEvent
AN EXAMPLE

Creating the Custom Text parser
AN EXAMPLE

› Event requests have states

events: CREATED SENT COMPLETED

states: CREATED RUNNING COMPLETED

Event Handler ➔ State System ➔ GUI
Event Source ➔ State History
Model state from arbitrary programs

GUI
Fast random seek

© Ericsson AB 2015
AN EXAMPLE

› A XML Event Handler

```xml
<stateProvider version="0" id="org.eclipse.tracecompass.request.analysis">
  <head>
    <traceType id="o.e.tracecompass.CustomTxtTrace:TMF:TmfRequest" />
    <label value="Request Analysis" />
  </head>

  <definedValue name="CREATED" value="0" />
  <definedValue name="COALESCE" value="1" />
  <definedValue name="SUSPENDED" value="2" />
  <definedValue name="RUNNING" value="3" />
  <definedValue name="COMPLETED" value="4" />
</stateProvider>
```
AN EXAMPLE

› A XML Event Handler

```
<eventHandler eventName="TmfRequest">
  <stateChange>
    <if>
      <condition>
        <field name="Msg" />
        <stateValue type="string" value="CREATED"/>
      </condition>
    </if>
    <then>
      <stateAttribute type="constant" value="Request" />
      <stateAttribute type="eventField" value="Req ID" />
      <stateAttribute type="constant" value="value" />
      <stateValue type="int" value="\$CREATED" />
    </then>
  </then>
  <else>
    ...
  </else>
</eventHandler>
```

Request/0/value = 0 (CREATED)
AN EXAMPLE

› A XML state system based view
An example

A XML state system based view

```xml
<timeGraphView id="org.eclipse.tracecompass.request.view.xml">
  <head>
    <analysis id="org.eclipse.tracecompass.request.analysis" />
    <label value="Request Analysis View" /> 
  </head>

  <definedValue name="CREATED" value="0" color="#888888" /> 
  <definedValue name="COALESCED" value="1" color="#95bc5f" /> 
  <definedValue name="SUSPENDED" value="2" color="#CCCCCC" /> 
  <definedValue name="RUNNING" value="3" color="#bcdd68" /> 
  <definedValue name="COMPLETED" value="4" color="#b8e4e6" /> 
  <entry path="Request/*">
    <display type="constant" value="value" /> 
    <name type="self" /> 
  </entry>
</timeGraphView>
```

15 lines !!
AN EXAMPLE

› The result:
Going further

- We can add “validation” to the state transition

```xml
<stateProvider version="0" id="org.eclipse.tracecompass.request.analysis">
...
<definedValue name="BAD" value="5"/>
</stateProvider>

<eventHandler eventName="TmfRequest">
  <stateChange>
    <if>
      <and>
        <condition>
          <stateAttribute type="constant" value="Request"/>
          <stateAttribute type="eventField" value="Req ID"/>
          <stateAttribute type="constant" value="value"/>
          <stateValue type="null"/>
        </condition>
      </and>
    </if>
  </stateChange>
</eventHandler>
```
... 
... 
<else>
  <stateAttribute type="constant" value="Request" />
  <stateAttribute type="eventField" value="Req ID" />
  <stateAttribute type="constant" value="value" />
  <stateValue type="int" value="$BAD" />
</else>

<timeGraphView>
... 
  <definedValue name="BAD" value="5" color="#FF0000" />
...
Why is event request 0 going **bad**?
AN EXAMPLE

Fixing the bug

Was request 0 was really created twice?

Maybe two requests shared the same ID?
Fixing the bug

```java
public TmfEventRequest(...) {
    fRequestId = fRequestNumber++;
    fDataType = datatype;
    fIndex = index;
    fNbRequested = nbRequested;
    fExecType = priority;
    fRange = range;
    fNbRead = 0;
}
```

fRequestNumber is static. The assignment of fRequestId is not thread-safe!
public TmfEventRequest(..) {
    synchronized (TmfEventRequest.class) {
        fRequestId = fRequestNumber++;  
    }
    fDataType = dataType;
    fIndex = index;
    fNbRequested = nbRequested;
    fExecType = priority;
    fRange = range;
    fNbRead = 0;
}
INTEGRATIONS

- LTTng (UST, Kernel)
- Text Logs (custom parsers)
- Common Trace Format (Application, kernel, HW, Bare metal)
- Packet Capture
- BTF (Best Trace Format)
- GDB Trace Points
LT TNG ECLIPSE INTEGRATION

› Reference implementations for
  – a plug-in extension to Trace Compass
  – various trace analyses
  – several visualization views

› Analysis of LT Tng Kernel and UST Traces

› LT Tng Remote Tracer Control
IP NETWORK TRACE ANALYSIS

› Plug-in extension to Trace Compass

› Packet Capture (libPcap) Parser

› Ethernet, IPv4, TCP and UDP

› Stream analysis (conversations)

› Correlation of network traces with application traces
GDB TRACEPOINT ANALYSIS
COLLABORATIONS

› Trace Research Project
   - Academia: Polytechnique Montreal, Concordia, others
   - Industry: Ericsson, EfficiOS, others
   - Government: NSERC, DRDC

› Contributors for Trace Compass
   - Kalray in France
   - CEA in France (Papyrus – Modeling integration)
   - And more!

› Upcoming contributions
   - Dependency analysis
   - Critical Path
REFERENCES

› Project pages
  - [http://www.eclipse.org/tracecompass](http://www.eclipse.org/tracecompass)
  - [https://dev.eclipse.org/mailman/listinfo/tracecompass-dev](https://dev.eclipse.org/mailman/listinfo/tracecompass-dev)
  - [http://lttng.org/](http://lttng.org/)
  - [https://www.polarsys.org/](https://www.polarsys.org/)
  - [http://www.diamon.org/](http://www.diamon.org/)
  - [http://tracingsummit.org/](http://tracingsummit.org/)

› Documentations
  - [Trace Compass User Guide](http://istmffastyet.dorsal.polymtl.ca/)
  - [Trace Compass Developer Guide](http://lttng.org/)
Q&A