Analyzing Eclipse Applications with Trace Compass

MARC-ANDRÉ LAPERLE,
ERICSSON

PRESENTER

- 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)

AGENDA

1. About Tracing
2. Trace Compass Overview
3. An Example: Analyzing Trace Compass with Trace Compass!
4. Trace Compass Integrations
5. Q&A

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)

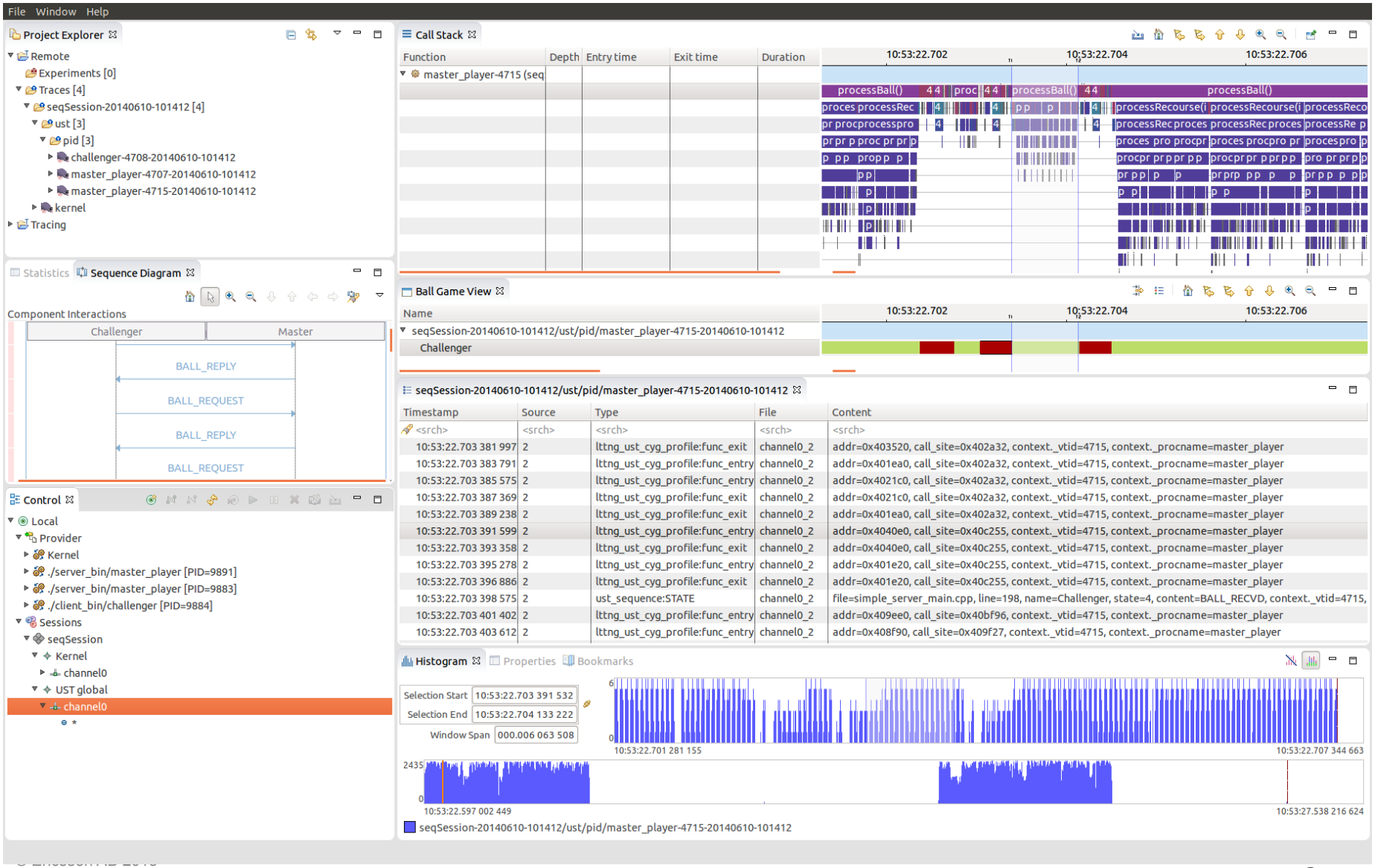
› New project Trace Compass at Eclipse Foundation
  – To increase community and collaboration in open-source
  – Larger scope, beyond LTTng, Linux

http://www.eclipse.org/tracecompass

TRACE COMPASS

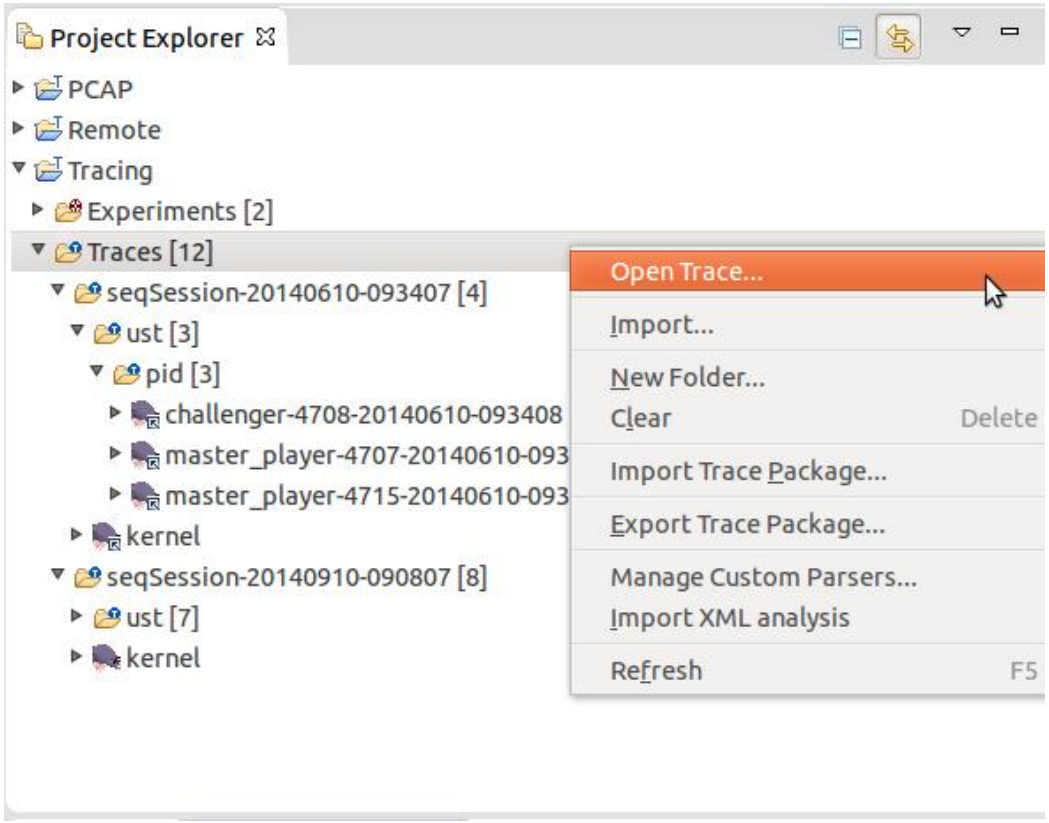
- Framework to build trace visualization and analysis tools
- Scalable: handle traces exceeding memory
- Extensible for any trace or log format
  - Binary, text, XML etc.
- Reusable views and widgets
- Available as standalone application or set of plug-ins

TRACE COMPASS



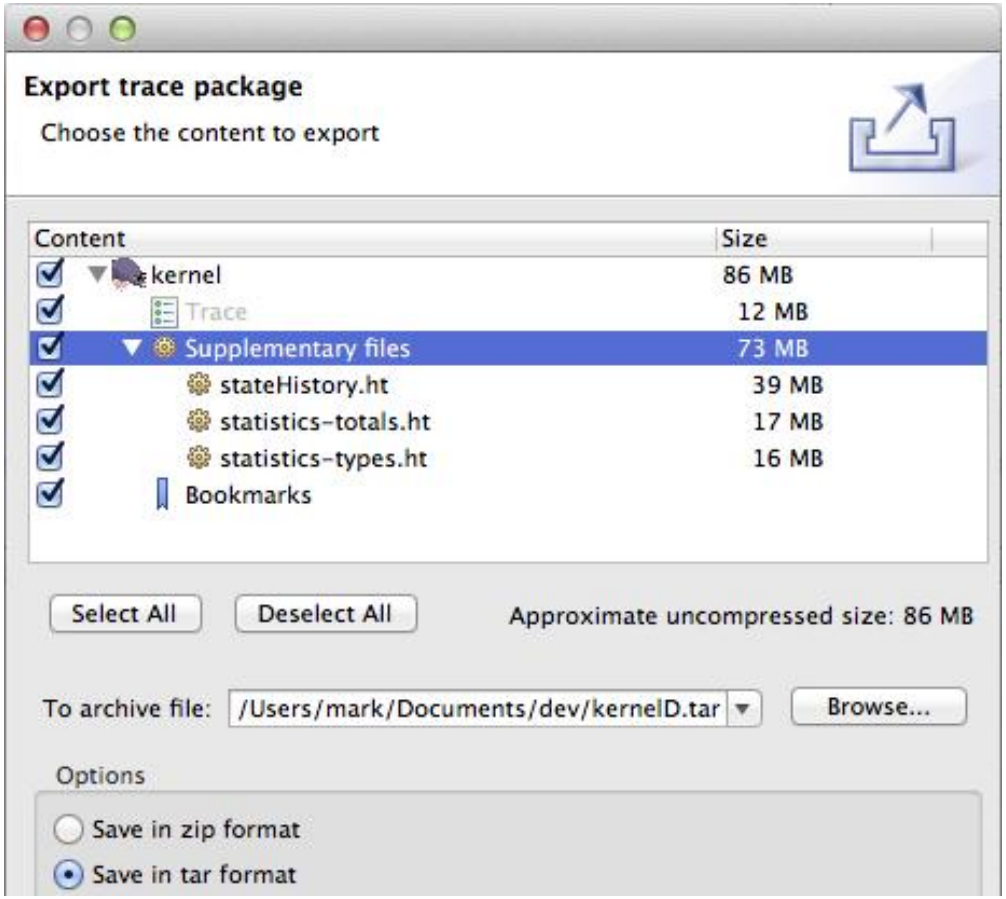
COMMON FEATURES

› Management of traces, trace formats and experiments



COMMON FEATURES

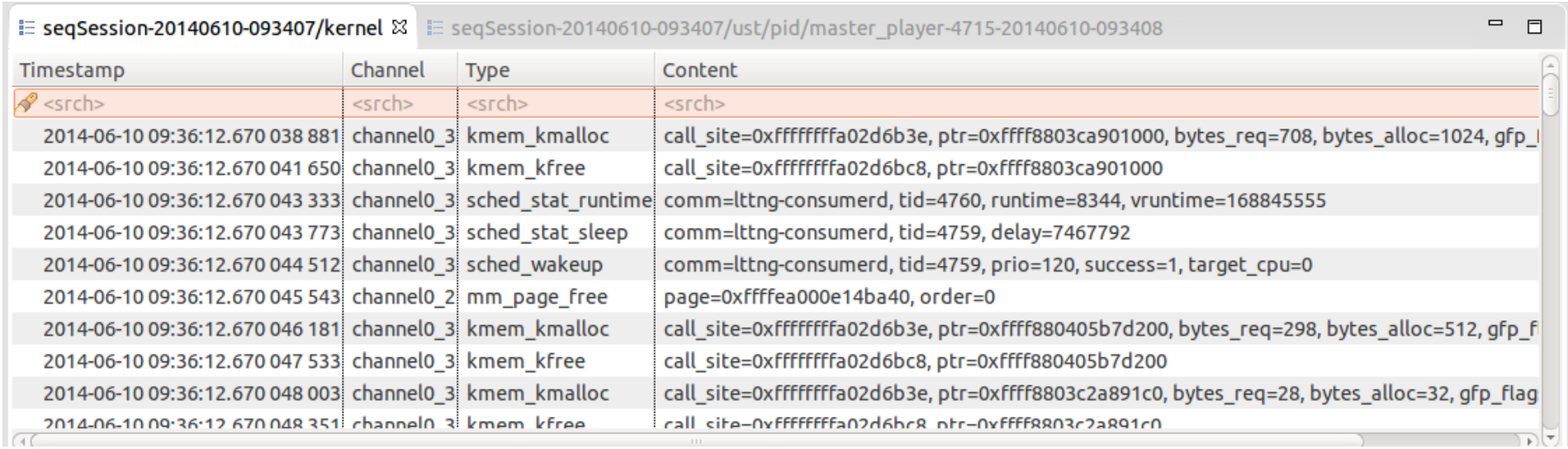
› Package export and import



COMMON FEATURES

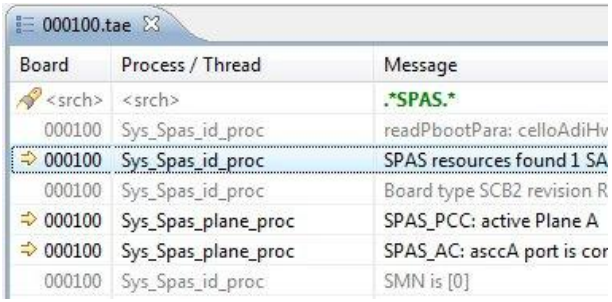
› Events Table

Implemented as an Eclipse “editor”

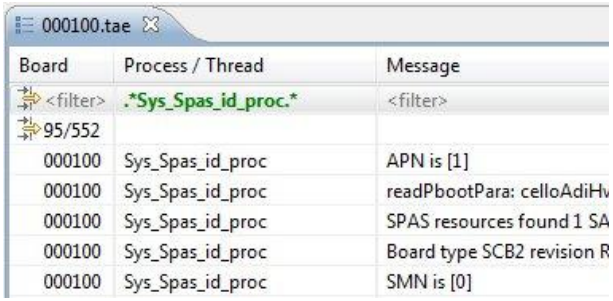


Timestamp	Channel	Type	Content
<srch>	<srch>	<srch>	<srch>
2014-06-10 09:36:12.670 038 881	channel0_3	kmem_kmalloc	call_site=0xffffffffa02d6b3e, ptr=0xffff8803ca901000, bytes_req=708, bytes_alloc=1024, gfp_
2014-06-10 09:36:12.670 041 650	channel0_3	kmem_kfree	call_site=0xffffffffa02d6bc8, ptr=0xffff8803ca901000
2014-06-10 09:36:12.670 043 333	channel0_3	sched_stat_runtime	comm=lttng-consumerd, tid=4760, runtime=8344, vruntime=168845555
2014-06-10 09:36:12.670 043 773	channel0_3	sched_stat_sleep	comm=lttng-consumerd, tid=4759, delay=7467792
2014-06-10 09:36:12.670 044 512	channel0_3	sched_wakeup	comm=lttng-consumerd, tid=4759, prio=120, success=1, target_cpu=0
2014-06-10 09:36:12.670 045 543	channel0_2	mm_page_free	page=0xffffea000e14ba40, order=0
2014-06-10 09:36:12.670 046 181	channel0_3	kmem_kmalloc	call_site=0xffffffffa02d6b3e, ptr=0xffff880405b7d200, bytes_req=298, bytes_alloc=512, gfp_f
2014-06-10 09:36:12.670 047 533	channel0_3	kmem_kfree	call_site=0xffffffffa02d6bc8, ptr=0xffff880405b7d200
2014-06-10 09:36:12.670 048 003	channel0_3	kmem_kmalloc	call_site=0xffffffffa02d6b3e, ptr=0xffff8803c2a891c0, bytes_req=28, bytes_alloc=32, gfp_flag
2014-06-10 09:36:12.670 048 351	channel0_3	kmem_kfree	call_site=0xffffffffa02d6bc8, ptr=0xffff8803c2a891c0

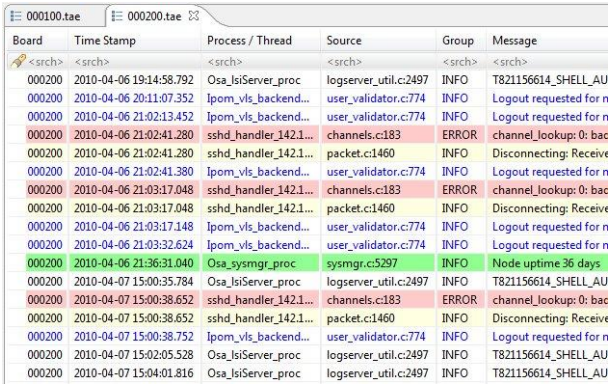
COMMON FEATURES



Searching



Filtering



Highlighting

COMMON FEATURES

› Bookmarks

seqSession-20140610-093407/kernel | seqSession-20140610-093407/ust/pid/master_player-4715-20140610-093408

Timestamp	Source	Type	File	Content
<srch>	<srch>	<srch>	<srch>	<srch>
2014-06-10 09:36:12.972 364 274	1	lttng_ust_cyg_profile:func_entry	channel0_1	addr=0x4040e0, call_site=0x40d51c, context._vtid=4715, context._procna
2014-06-10 09:36:12.972 364 782	1	lttng_ust_cyg_profile:func_exit	channel0_1	addr=0x4040e0, call_site=0x40d51c, context._vtid=4715, context._procna
2014-06-10 09:36:12.972 365 527	1	lttng_ust_cyg_profile:func_entry	channel0_1	addr=0x401e20, call_site=0x40d51c, context._vtid=4715, context._procna
2014-06-10 09:36:12.972 366 004	1	lttng_ust_cyg_profile:func_exit	channel0_1	addr=0x401e20, call_site=0x40d51c, context._vtid=4715, context._procna
2014-06-10 09:36:12.972 366 439	1	ust_sequence:RECEIVE	channel0_1	file=simple_server_main.cpp, line=173, sender=Challenger, receiver=Mast
2014-06-10 09:36:12.972 367 424	1	lttng_ust_cyg_profile:func_entry	channel0_1	addr=0x4031c0, call_site=0x40b484, context._vtid=4715, context._procna
2014-06-10 09:36:12.972 368 007	1	lttng_ust_cyg_profile:func_exit	channel0_1	addr=0x4031c0, call_site=0x40b484, context._vtid=4715, context._procna
2014-06-10 09:36:12.972 368 501	1	lttng_ust_cyg_profile:func_entry	channel0_1	addr=0x403520, call_site=0x402a32, context._vtid=4715, context._procna

Bookmarks | Histogram | Properties | State System Explorer | Stream List | Progress

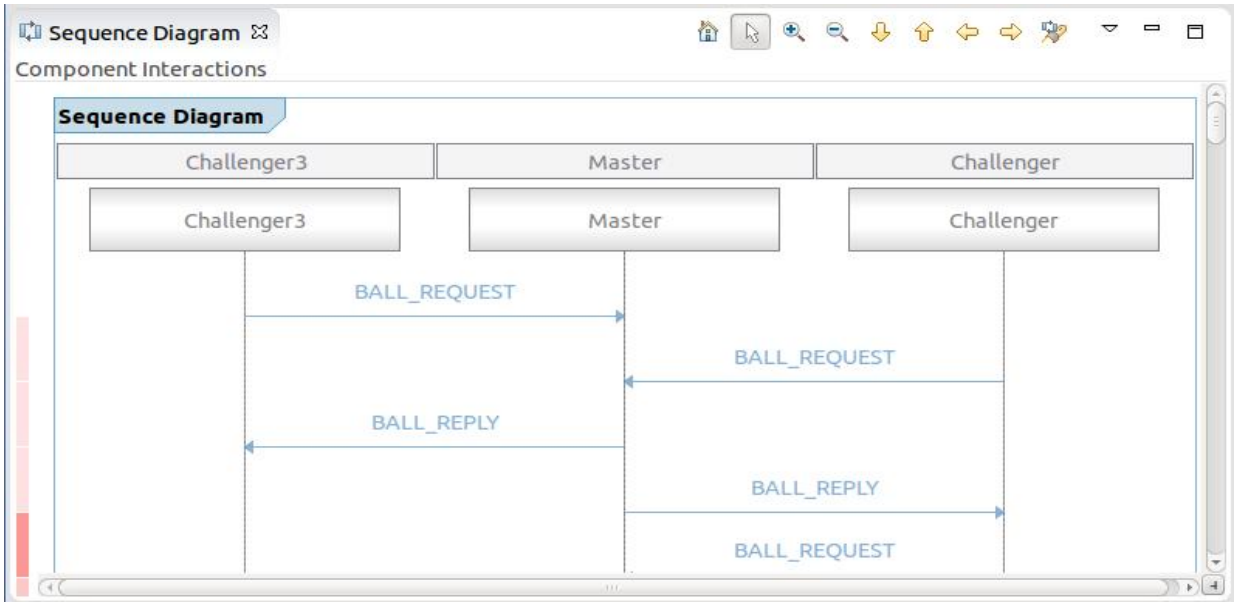
2 items

Description	Resource	Path	Location
Error trace	master_player	/Tracing/Traces/seqSes	249997
Started	master_player	/Tracing/Traces/seqSes	129

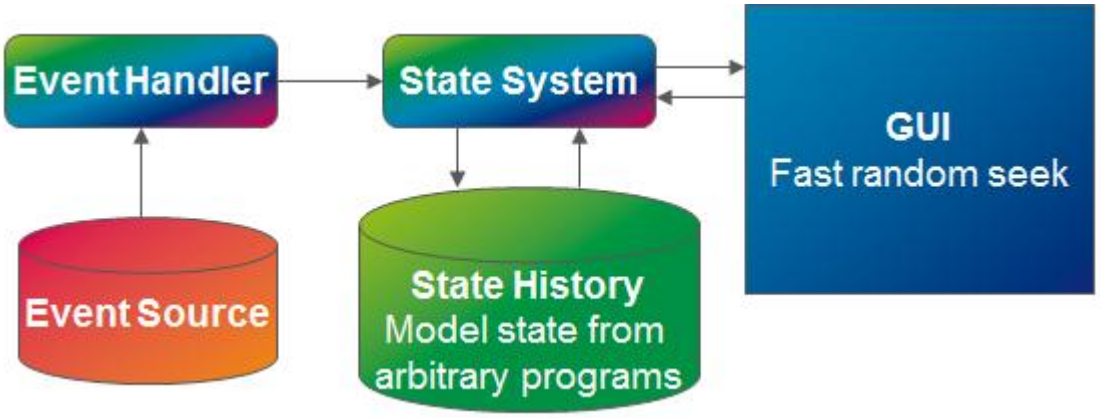
COMMON FEATURES

› Sequence Diagrams

- Translates events to sequence diagram transaction
- Extensions can define their own model



STATE SYSTEM

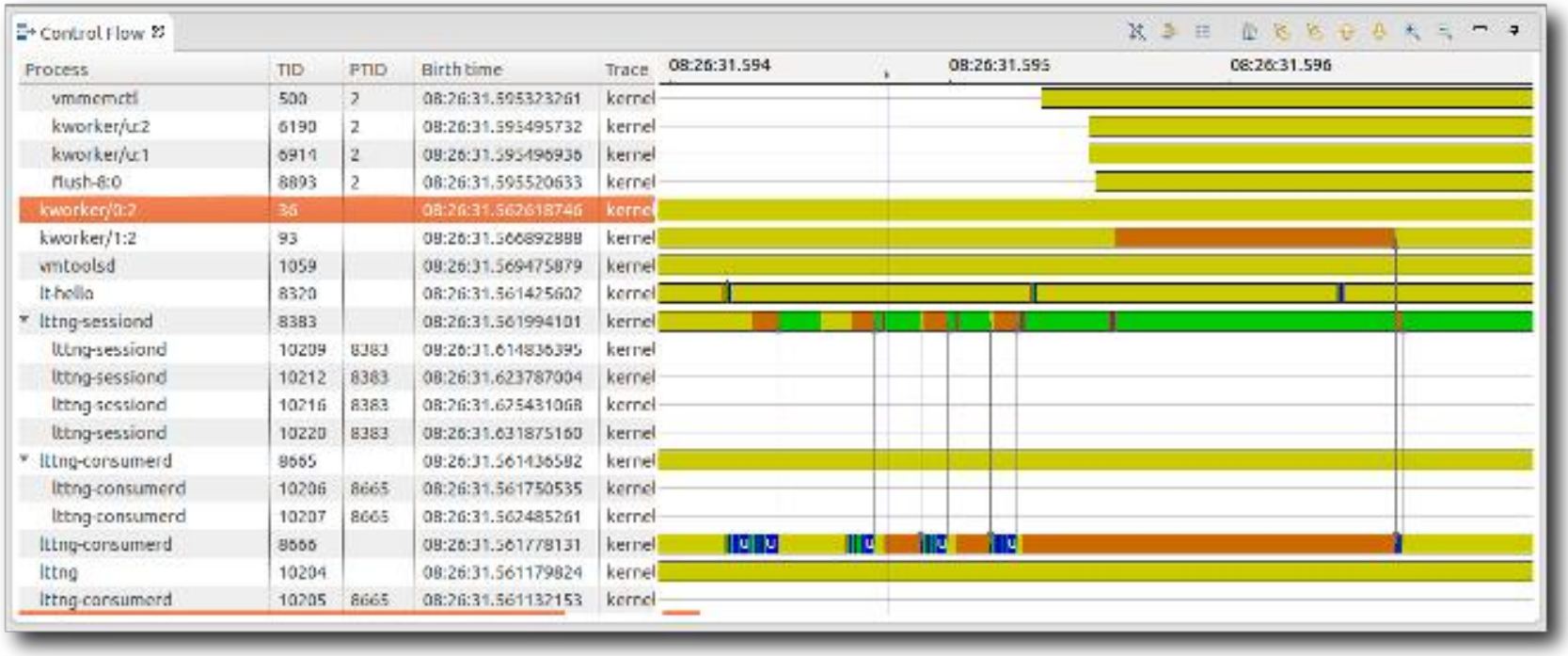


› State system abstracts events, analyses traces and creates models to be displayed

CONTROL FLOW VIEW

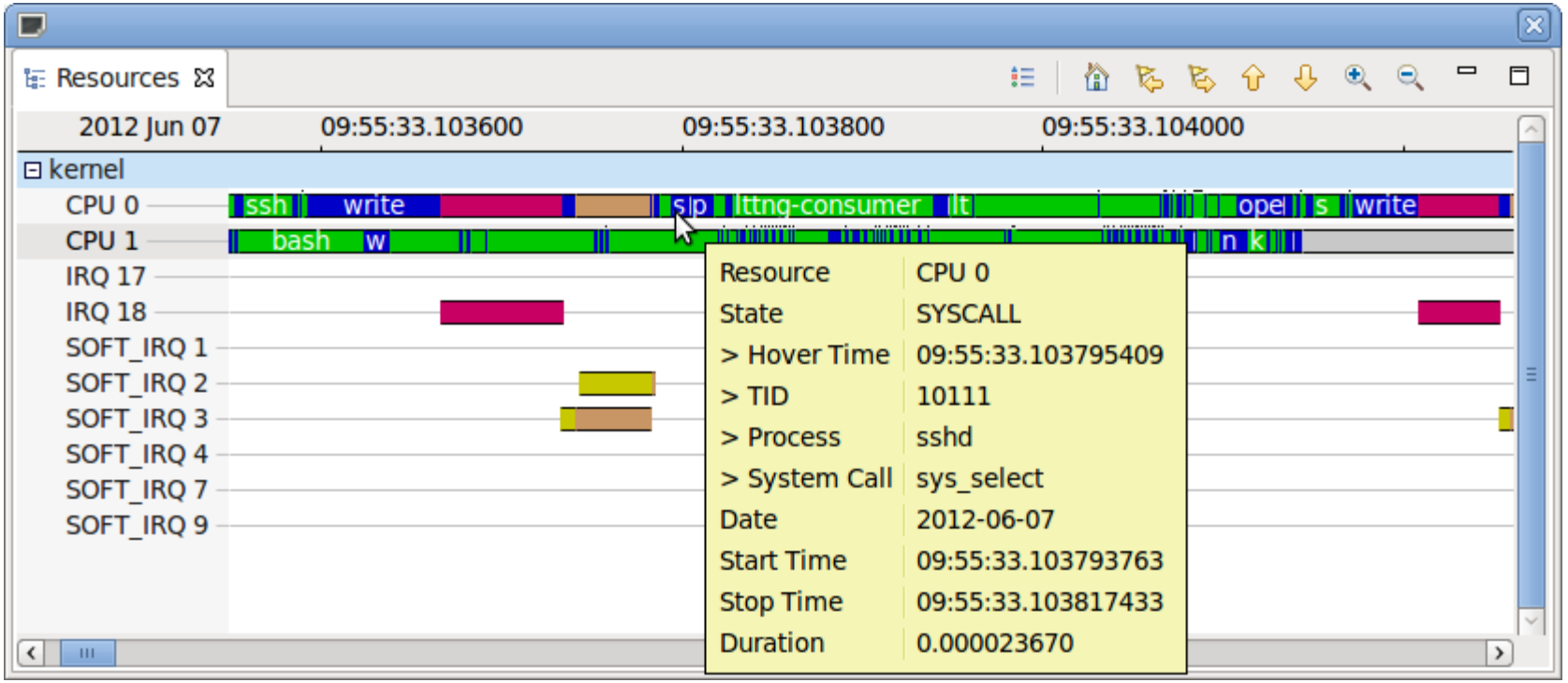
› Displays processes state changes (color-coded) over time

USERMODE, SYSCALL, INTERRUPED, WAIT_FOR_CPU, etc



RESOURCES VIEW

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

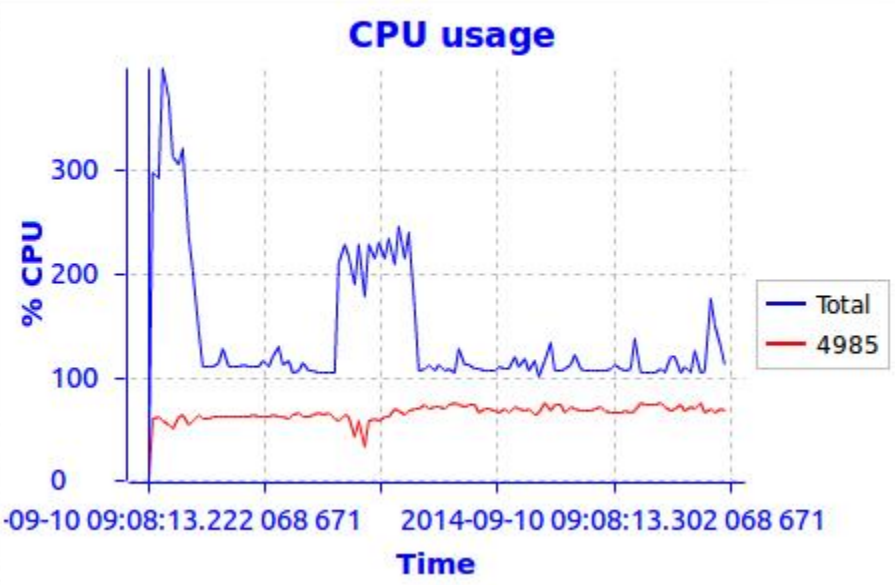


CPU USAGE VIEW

› Displays % of CPU used per thread over time

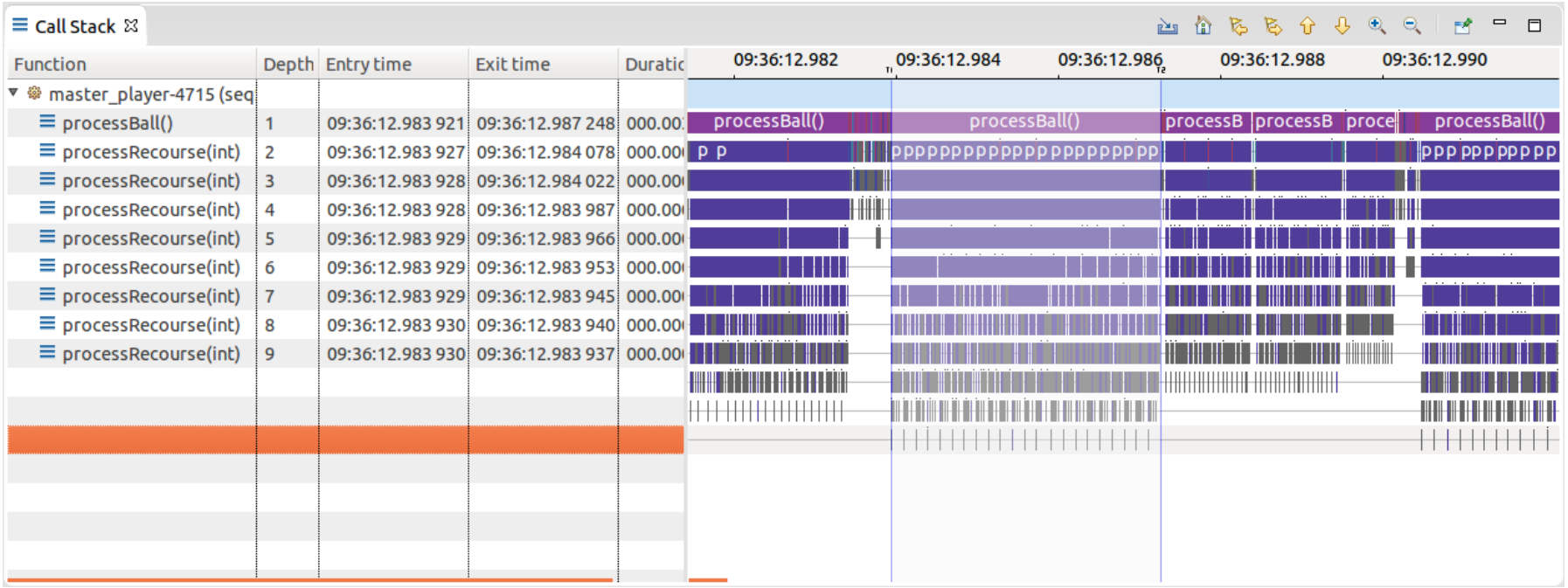
CPU Usage

TID	Process	%	Time
4983	master_player	0.508 %	507549 ns
5079	java	0.172 %	172449 ns
4980	challenger	0.469 %	469351 ns
4986	master_player	0.508 %	507759 ns
4985	master_player	66.658 %	66657523 ns
5074	sh	0.041 %	41494 ns
5077	java	0.138 %	137739 ns
5076	java	0.013 %	13099 ns
4847	java	0.020 %	20163 ns
4988	kworker/3:1	0.595 %	595313 ns
3837	gtk-window-deco	0.074 %	73961 ns
5045	java	0.029 %	28705 ns

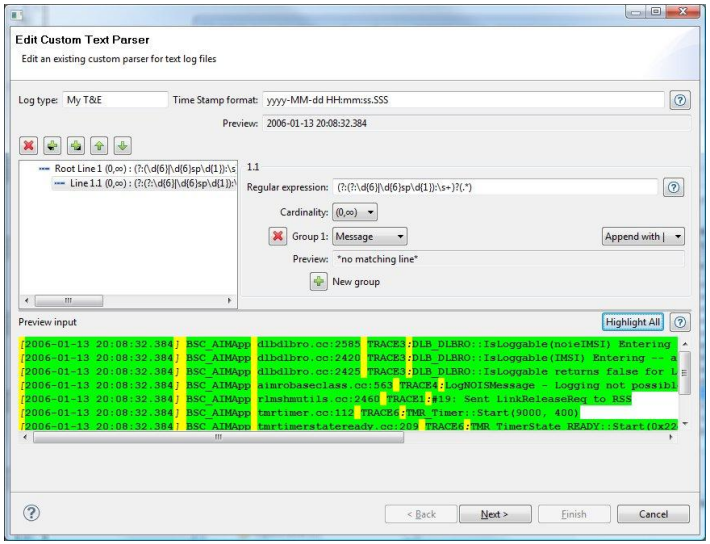


CALL STACK VIEW

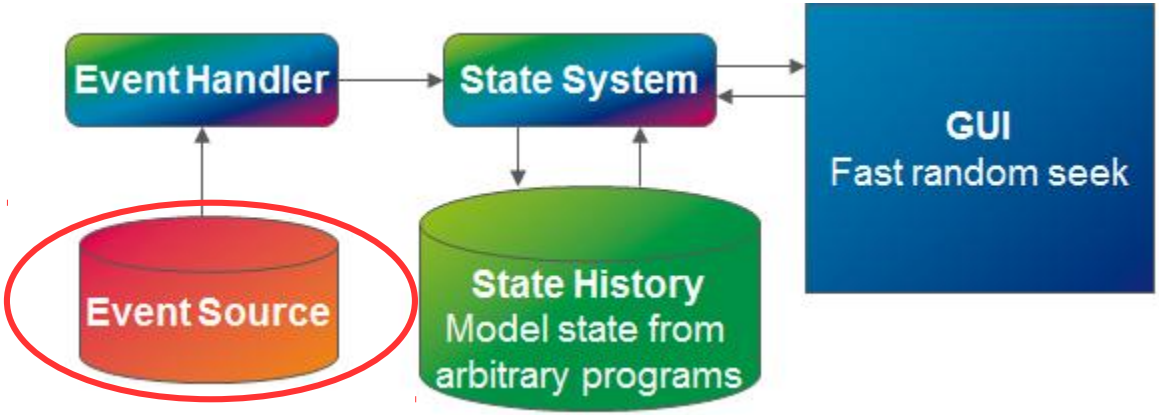
› Shows the stack trace at any point during execution



COMMON FEATURES



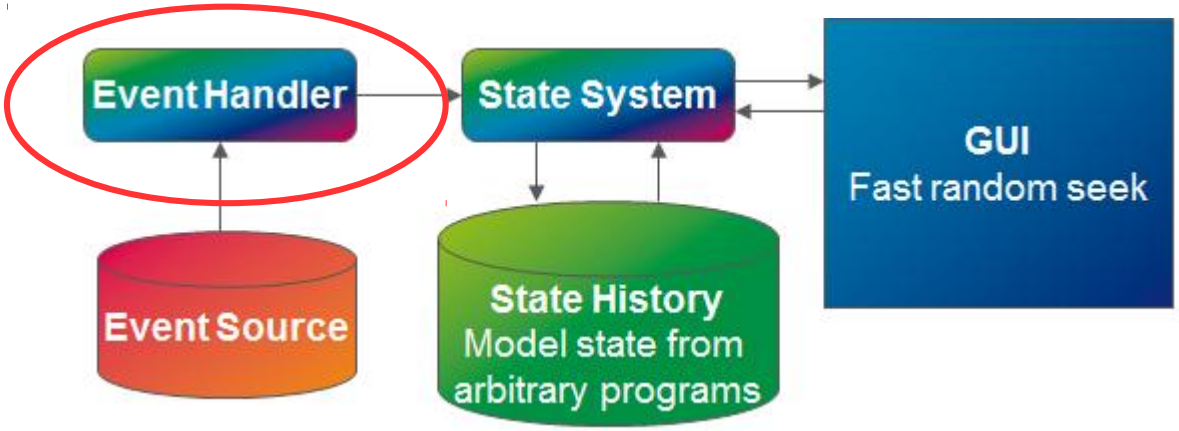
- Custom Text and XML Parsers
  - Line based parser with regex defined in a wizard
  - XML based extracting data from XML elements and their attributes



COMMON FEATURES

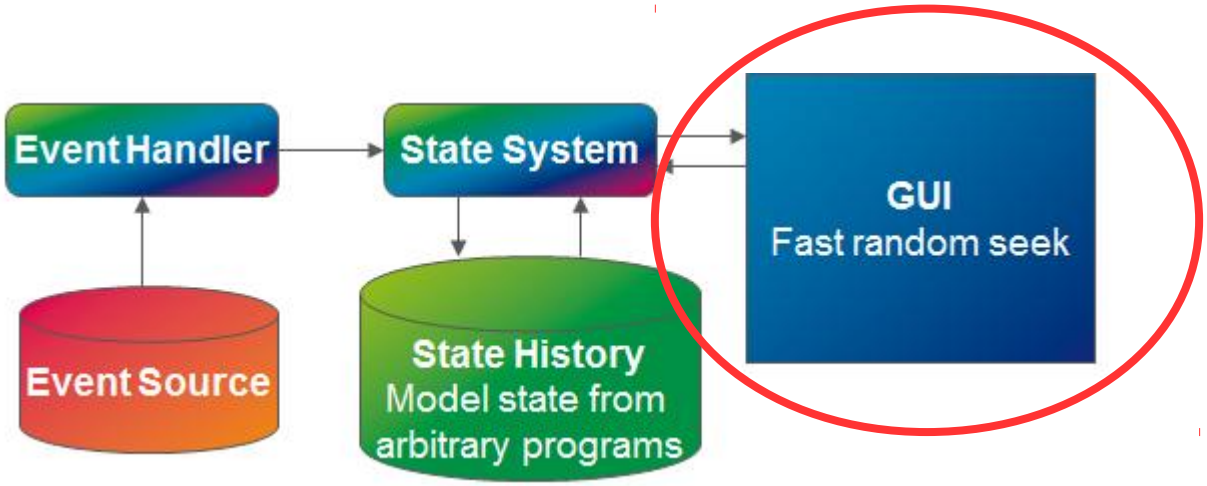
› Data-driven state provider

- XML description of state changes to convert trace events to states
- Can be created without changing source code or recompiling



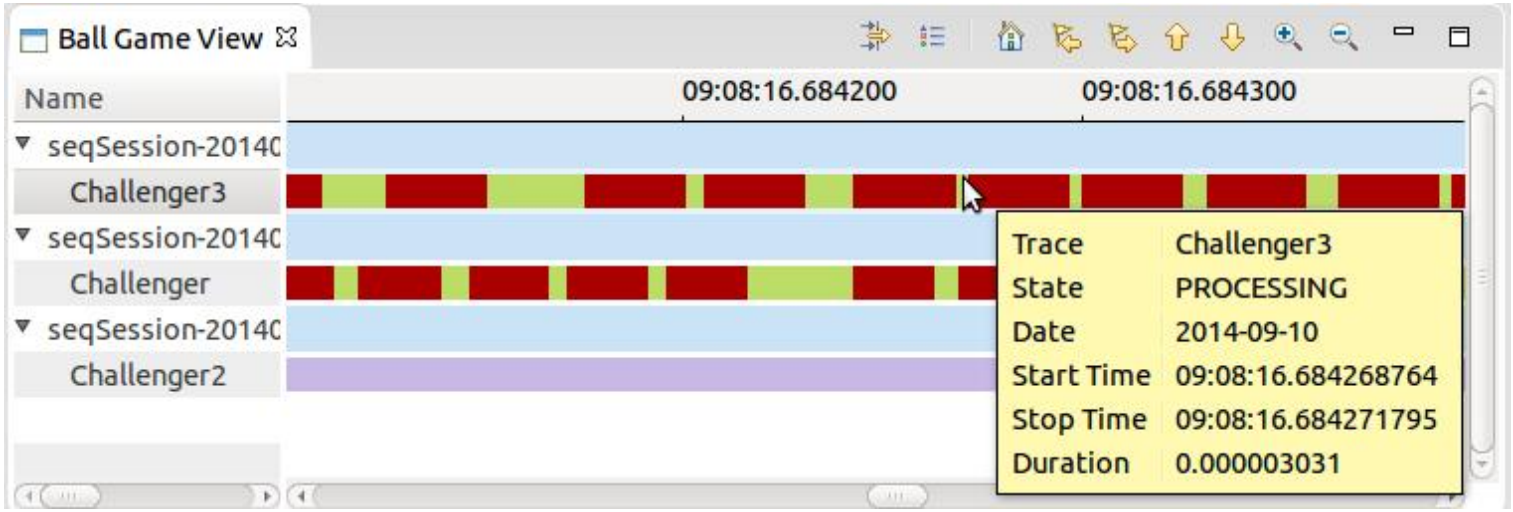
COMMON FEATURES

- 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



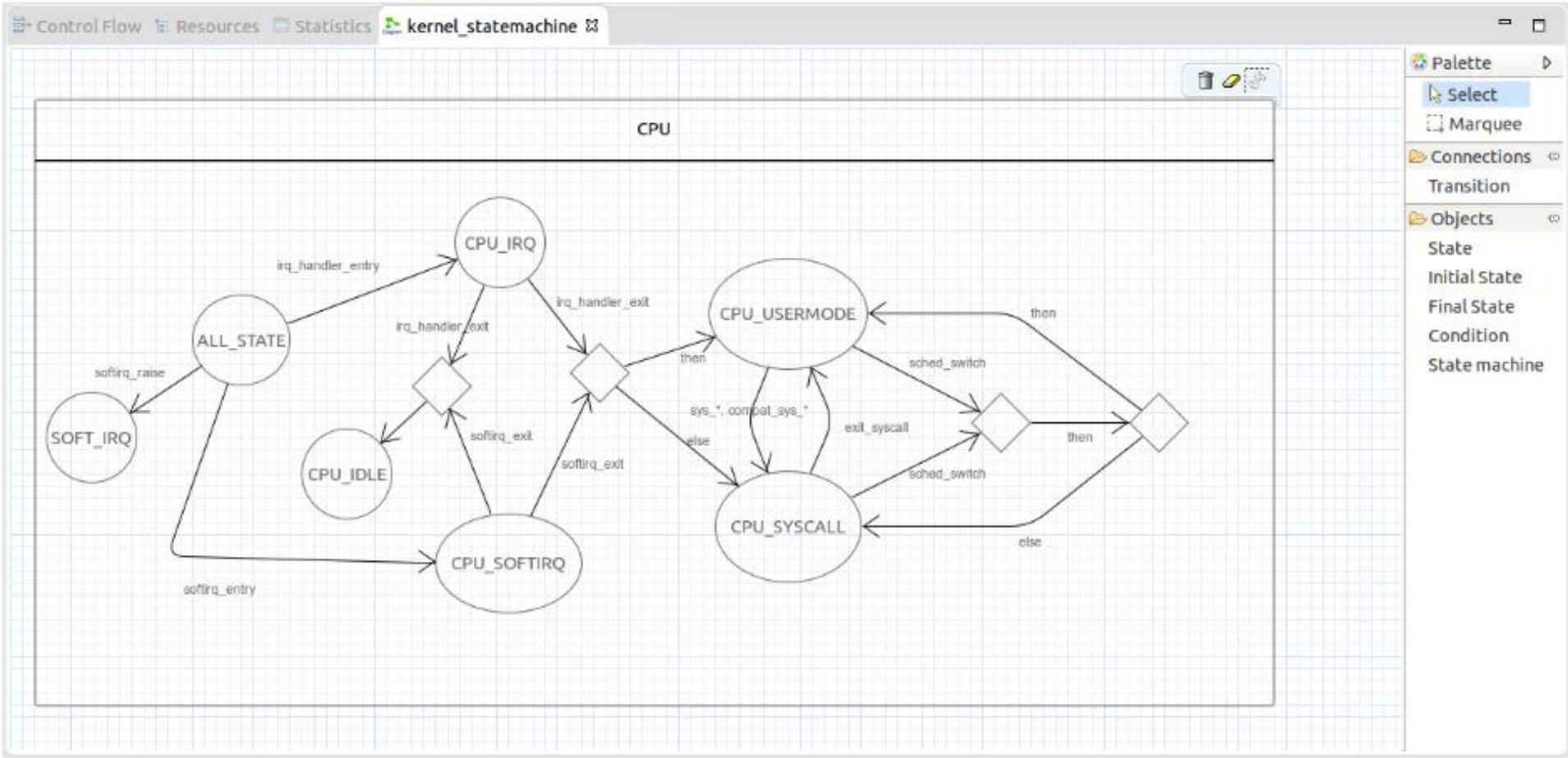
COMMON FEATURES

- For example: 50 lines of XML created the view below



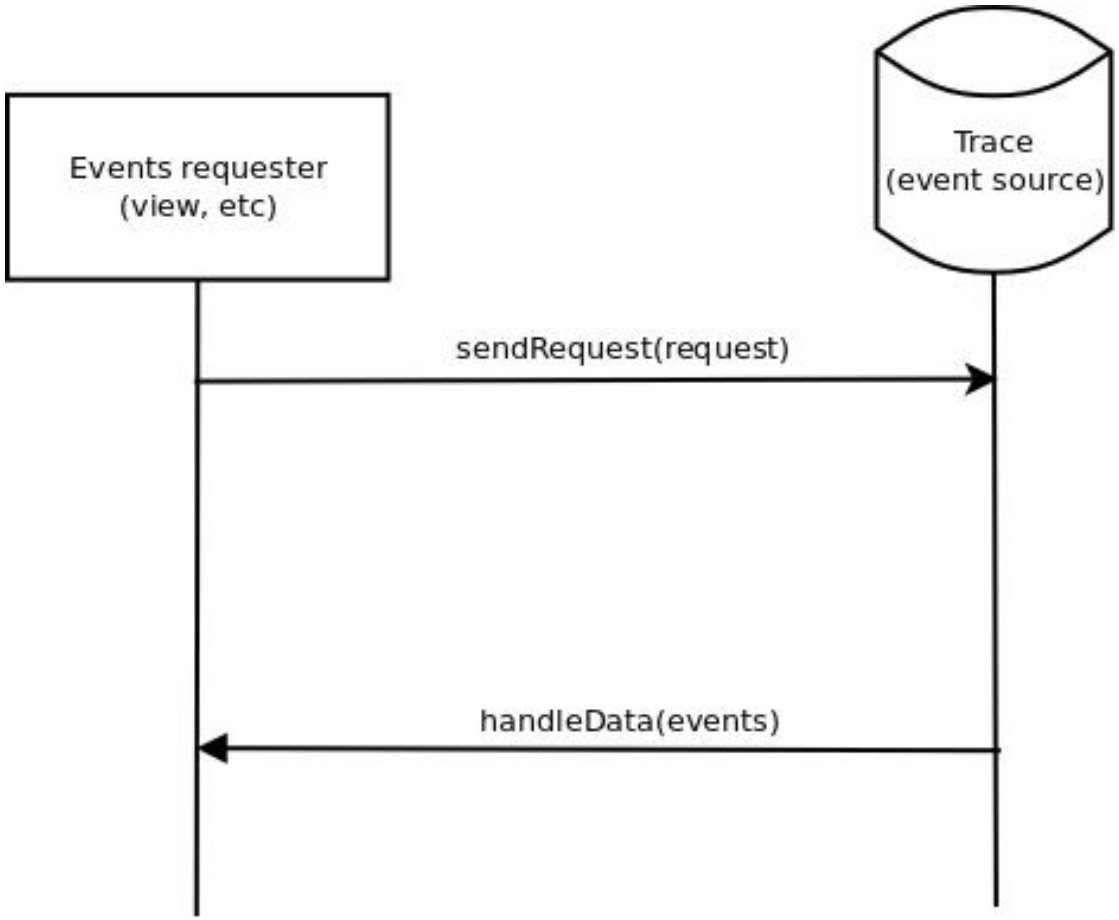
COMMON FEATURES

- A graphical editor for defining state providers is in the works

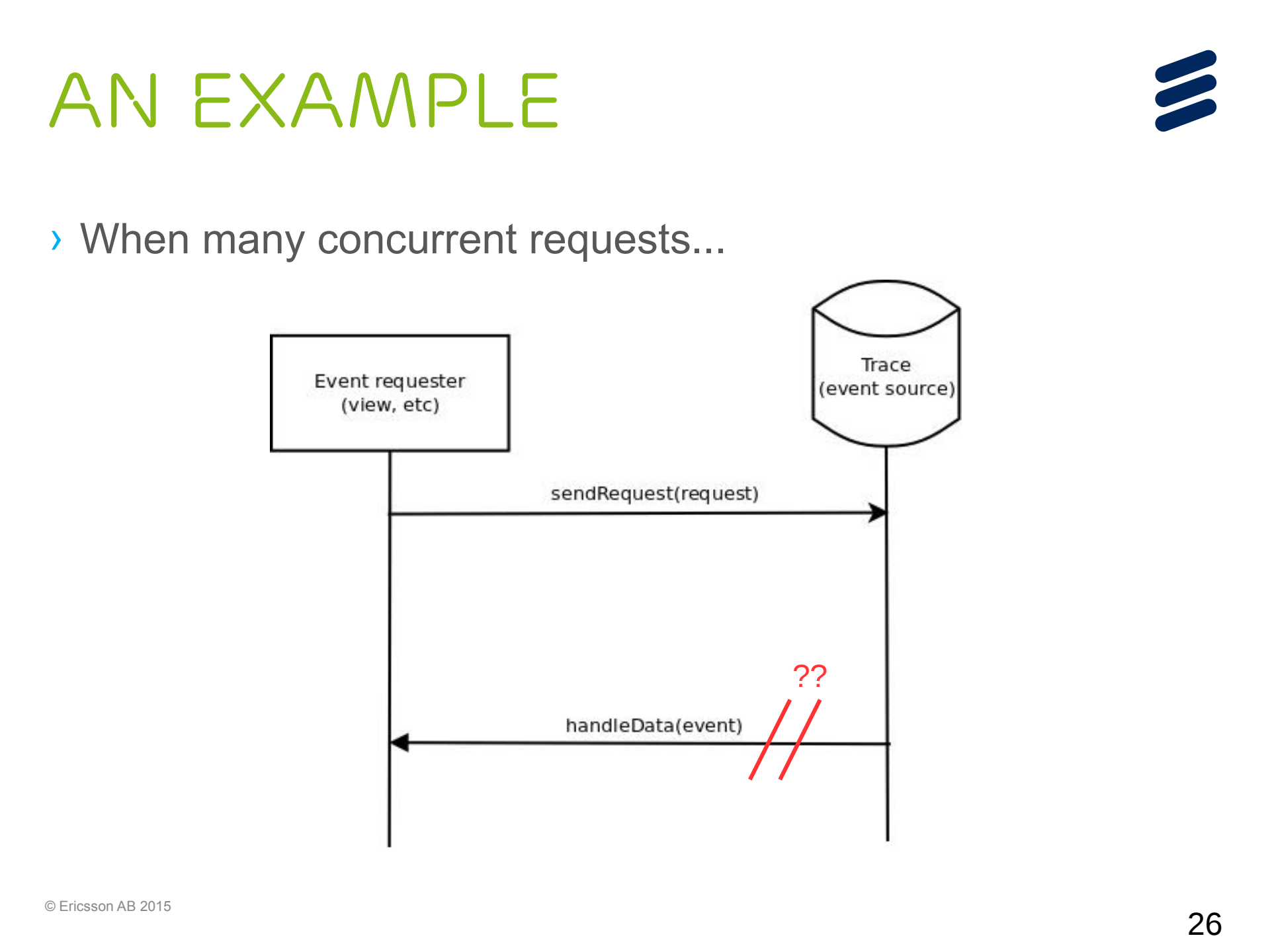


AN EXAMPLE

- A Trace Compass event(s) request



AN EXAMPLE

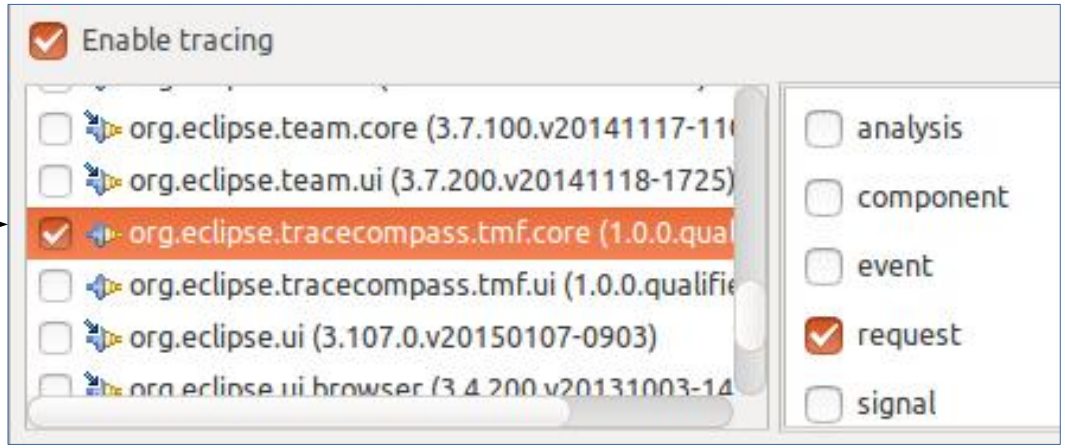


- When many concurrent requests...

AN EXAMPLE

- Trace points

```
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:

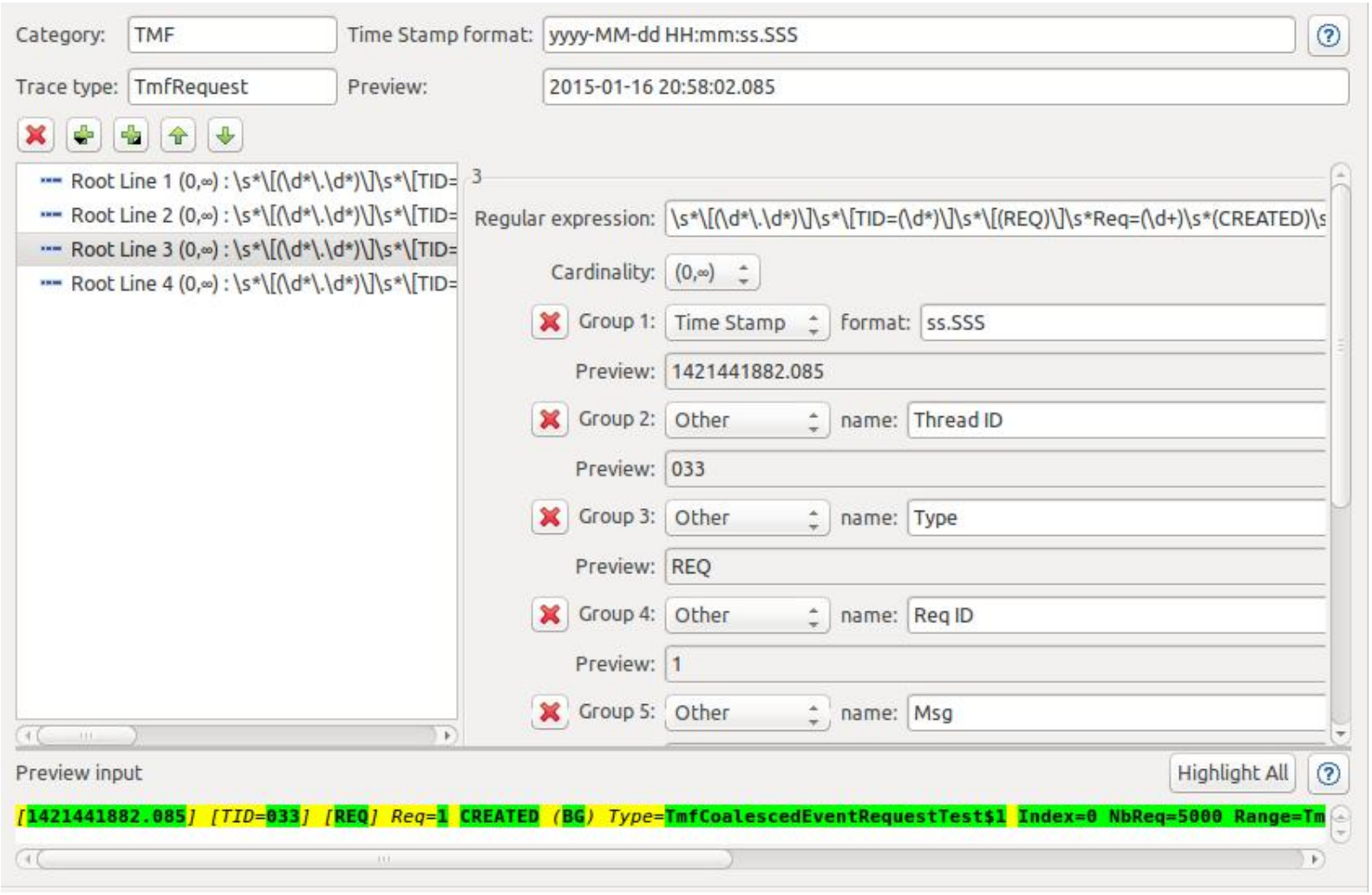
```
[1418416987.948] [TID=037] [REQ] Req=6 CREATED (FG)
Type=TmfEventsCache Index=0 NbReq=2000 Range=TmfTimeRange
[fStartTime=19:12:43.145 224 192, fEndTime=19:47:16.854 775 807]
DataType=ITmfEvent
```

TmfTrace.log(2)

Time Stamp	Thread ID	Req ID	Msg	Info	Priority	Type	Req Type	Message
<srch>	<srch>	<srch>	<srch>	<srch>	<srch>	<srch>	<srch>	<srch>
2014-12-12 20:43:07.938	033	5	COALESCED			REQ		with 3
2014-12-12 20:43:07.938	033	3		now contains		REQ		[2, 4, 5]
2014-12-12 20:43:07.948	037	6	CREATED		FG	REQ	TmfEventsCache$1$1	Index=0 NbReq=2000 Range=TmfTimeF
2014-12-12 20:43:07.949	037	6	SENT	to provider		REQ		
2014-12-12 20:43:07.949	037	6	QUEUED			REQ		

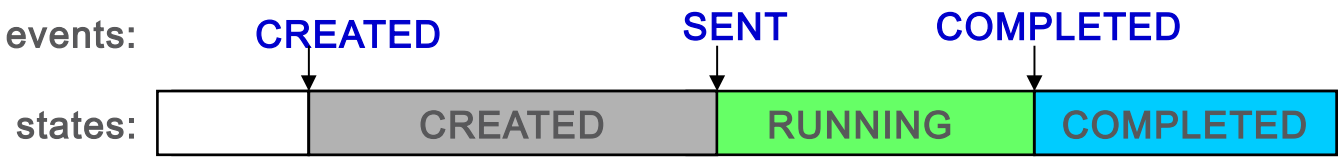
AN EXAMPLE

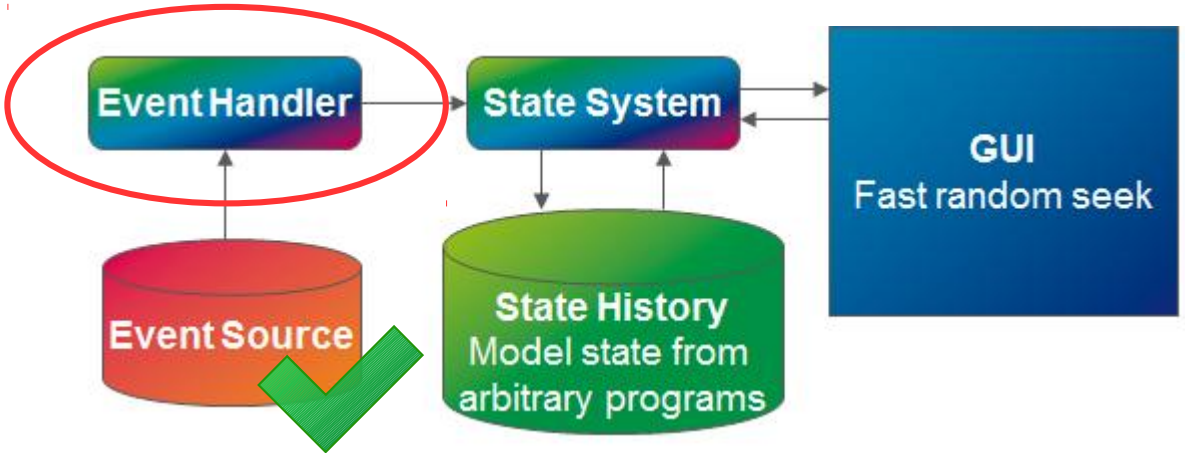
- Creating the Custom Text parser



AN EXAMPLE

- Event requests have states





AN EXAMPLE

› A XML Event Handler

```
<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="COALESCED" value="1" />
    <definedValue name="SUSPENDED" value="2" />
    <definedValue name="RUNNING" value="3" />
    <definedValue name="COMPLETED" value="4" />
```

AN EXAMPLE

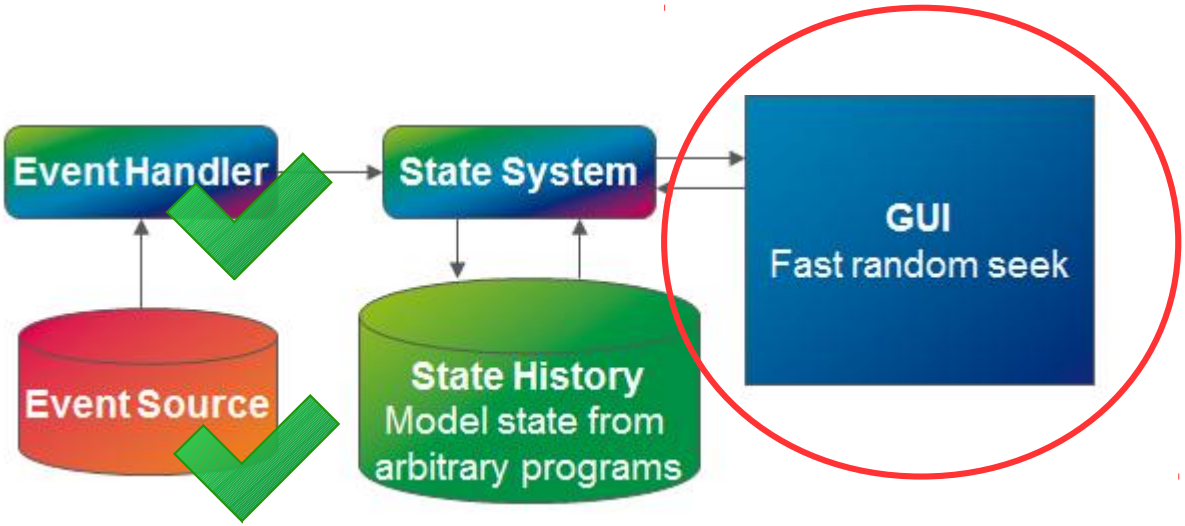
› A XML Event Handler

```
<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>
            <else>
                ...
```

Request/0/value = 0 (CREATED)

AN EXAMPLE

› A XML state system based view

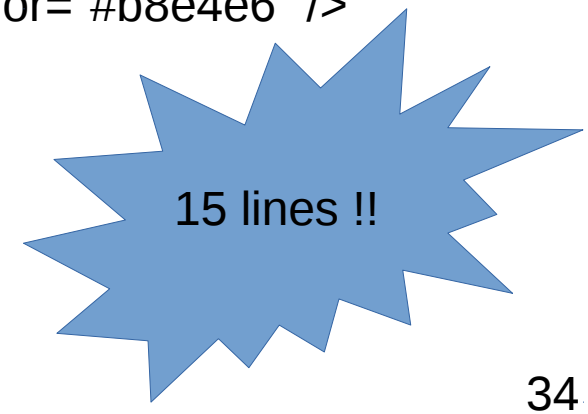


AN EXAMPLE

› A XML state system based view

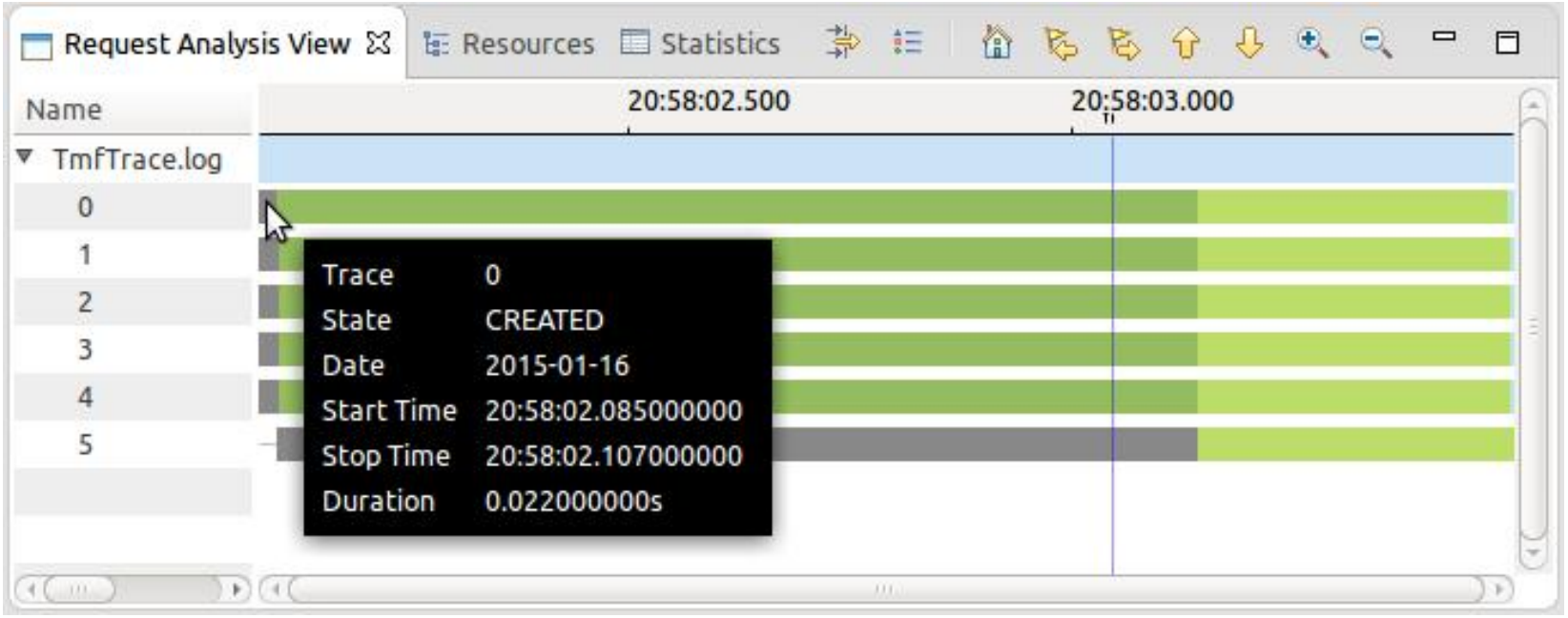
```
<timeGraphView id="org.eclipse.tracecompass.request.view.xml">
    <head>
        <analysis id="org.eclipse.tracecompass.request.analysis" />
        <label value="Request Analysis View" />
    </head>

    <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>
```



AN EXAMPLE

› The result:



AN EXAMPLE

› Going further

- We can add “validation” to the state transition

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

<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>
```

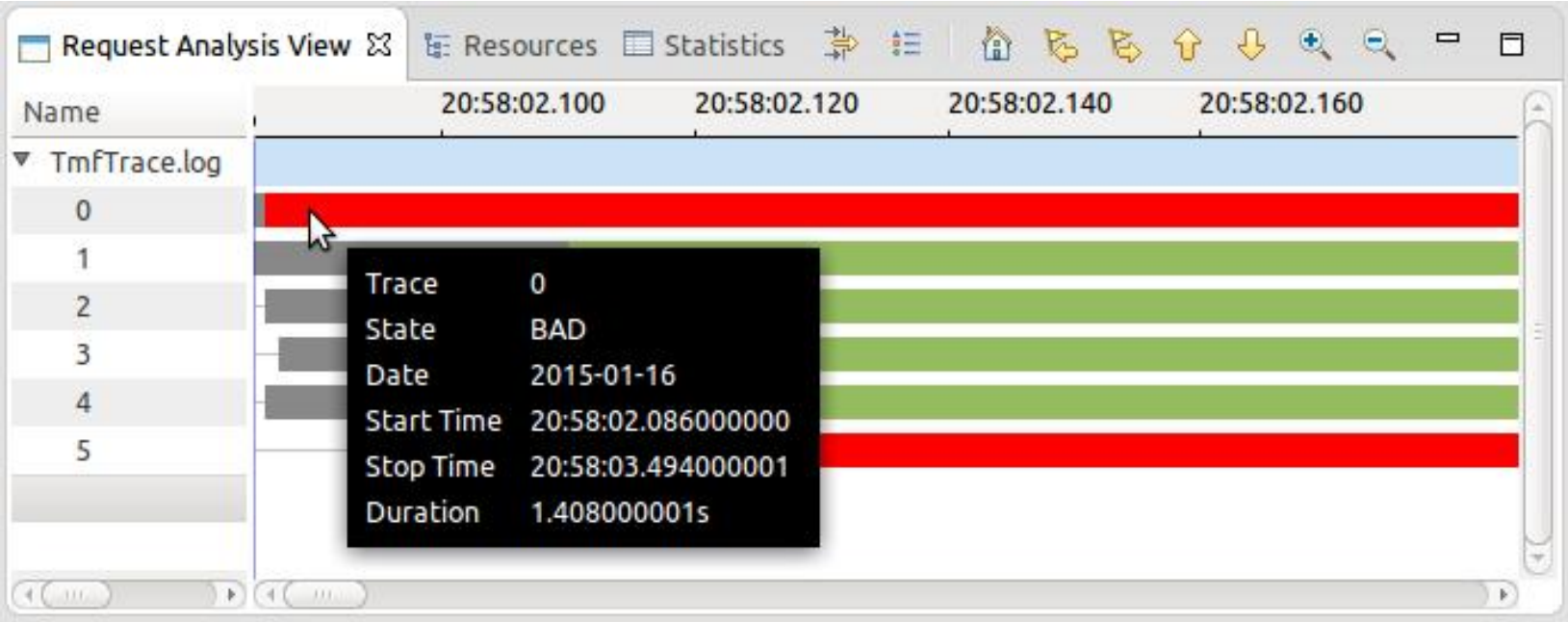
AN EXAMPLE

```
...
...
<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" />
...
```

AN EXAMPLE

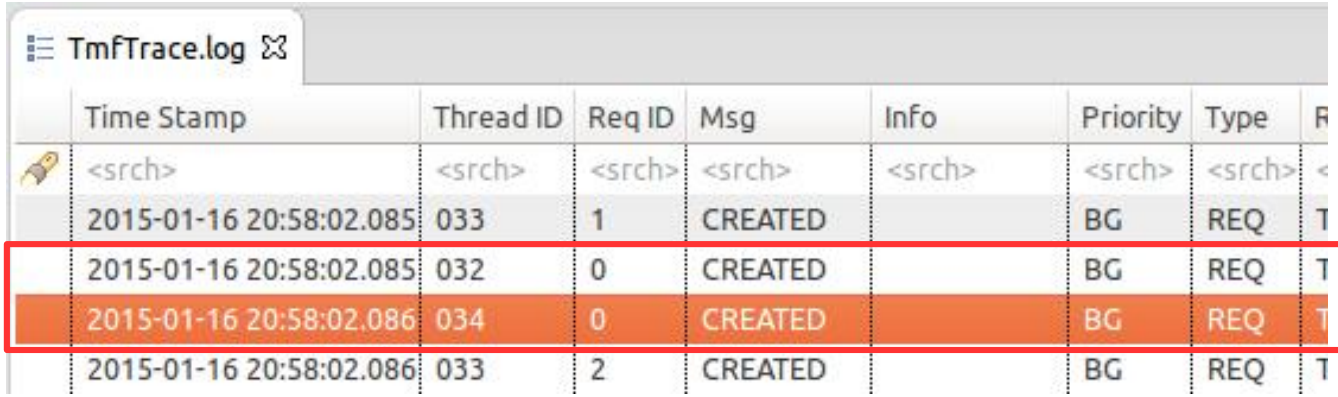
› The result:



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?

AN EXAMPLE

› Fixing the bug

```
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!

AN EXAMPLE

› Bug fixed!

```
public TmfEventRequest(..) {

    synchronized (TmfEventRequest.class) {
        fRequestId = fRequestNumber++;
    }
    fDataType = dataType;
    fIndex = index;
    fNbRequested = nbRequested;
    fExecType = priority;
    fRange = range;
    fNbRead = 0;
```

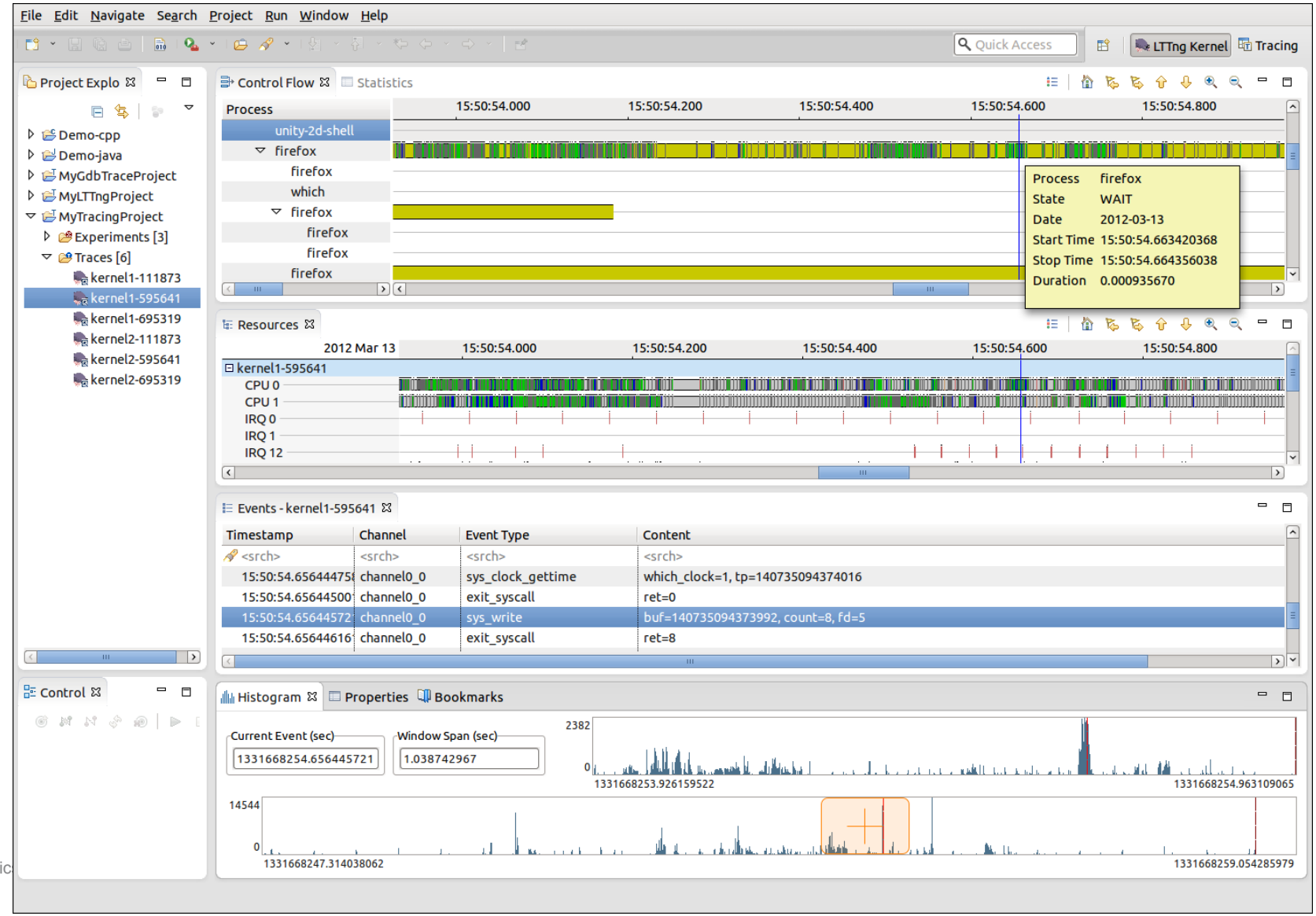
INTEGRATIONS

- LTTng (UST, Kernel)
- Text Logs (custom parsers)
- Common Trace Format (Application, kernel, HW, Bare metal)
- **P**acket **Cap**ture
- BTF (Best Trace Format)
- GDB Trace Points

LTTNG ECLIPSE INTEGRATION

- Reference implementations for
  - a plug-in extension to Trace Compass
  - various trace analyses
  - several visualization views
- Analysis of LTTng Kernel and UST Traces
- LTTng Remote Tracer Control

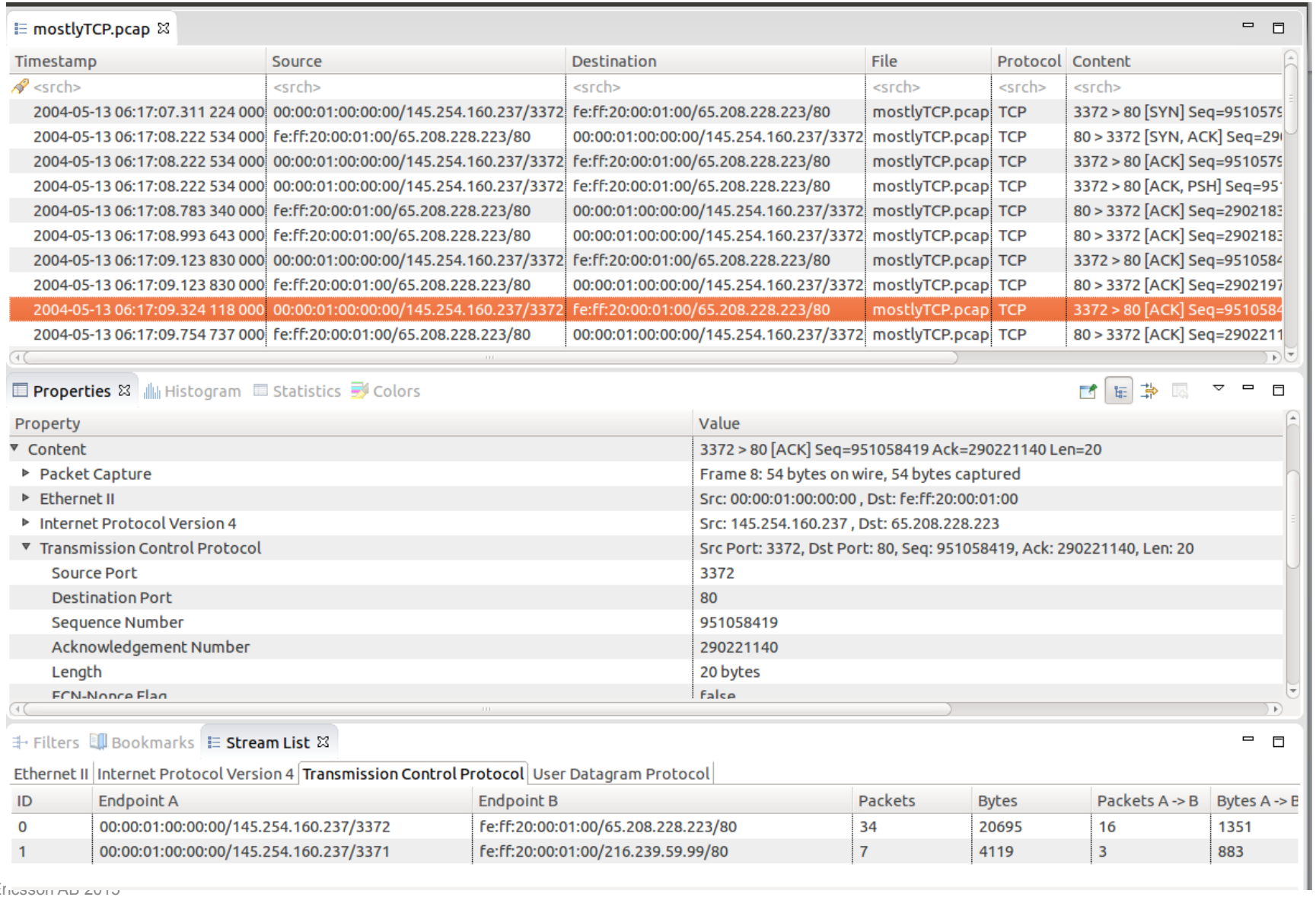
LTTNG ECLIPSE INTEGRATION



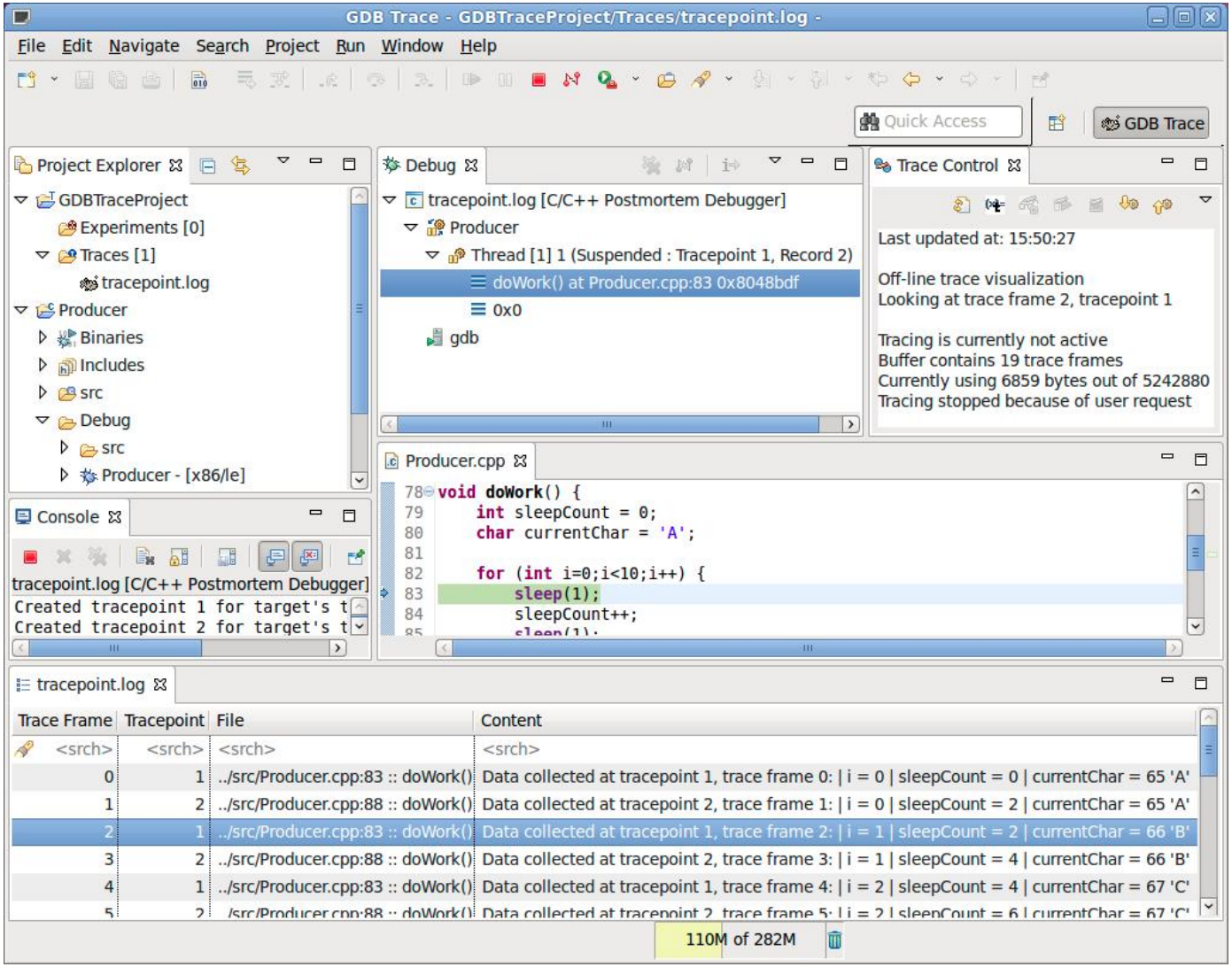
IP NETWORK TRACE ANALYSIS

- 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

IP NETWORK TRACE ANALYSIS



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
  - https://dev.eclipse.org/mailman/listinfo/tracecompass-dev
  - Is Trace Compass Fast Yet? http://istmffastyet.dorsal.polymtl.ca/
  - http://lttng.org/
  - https://www.polarsys.org/
  - http://www.diamon.org/
  - http://tracingsummit.org/

- Documentations
  - Trace Compass User Guide
  - Trace Compass Developer Guide

Q&A