ÉCOLE POLYTECHNIQUE MONTRÉAL

Low disturbance multi-level observation and production of enhanced data

Presented by : Florian Wininger

DORSAL Lab

Supervisor : Professor Michel Dagenais

Introduction

- Objectives :
  - Capture and merge observations made anywhere in the system (alerts, events, and states) by many surveillance systems (AV, HIDS, performance monitor systems, software tracers…)
  - Convert this data in the most appropriate format for detection analysis.

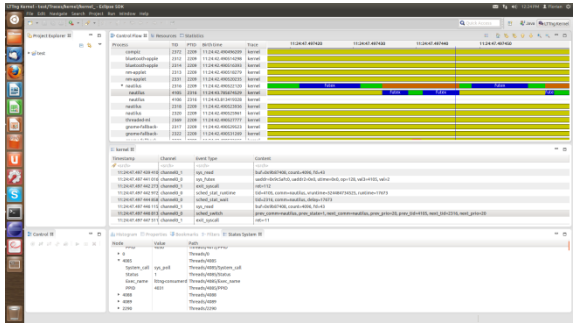
- Proposed solution :
  - Formal expressions to use the state-system in TMF
    - Allow the creation of personalized State System
    - Allow a new analysis with virtual states (filter)

Motivation

- Improve the use of the State System

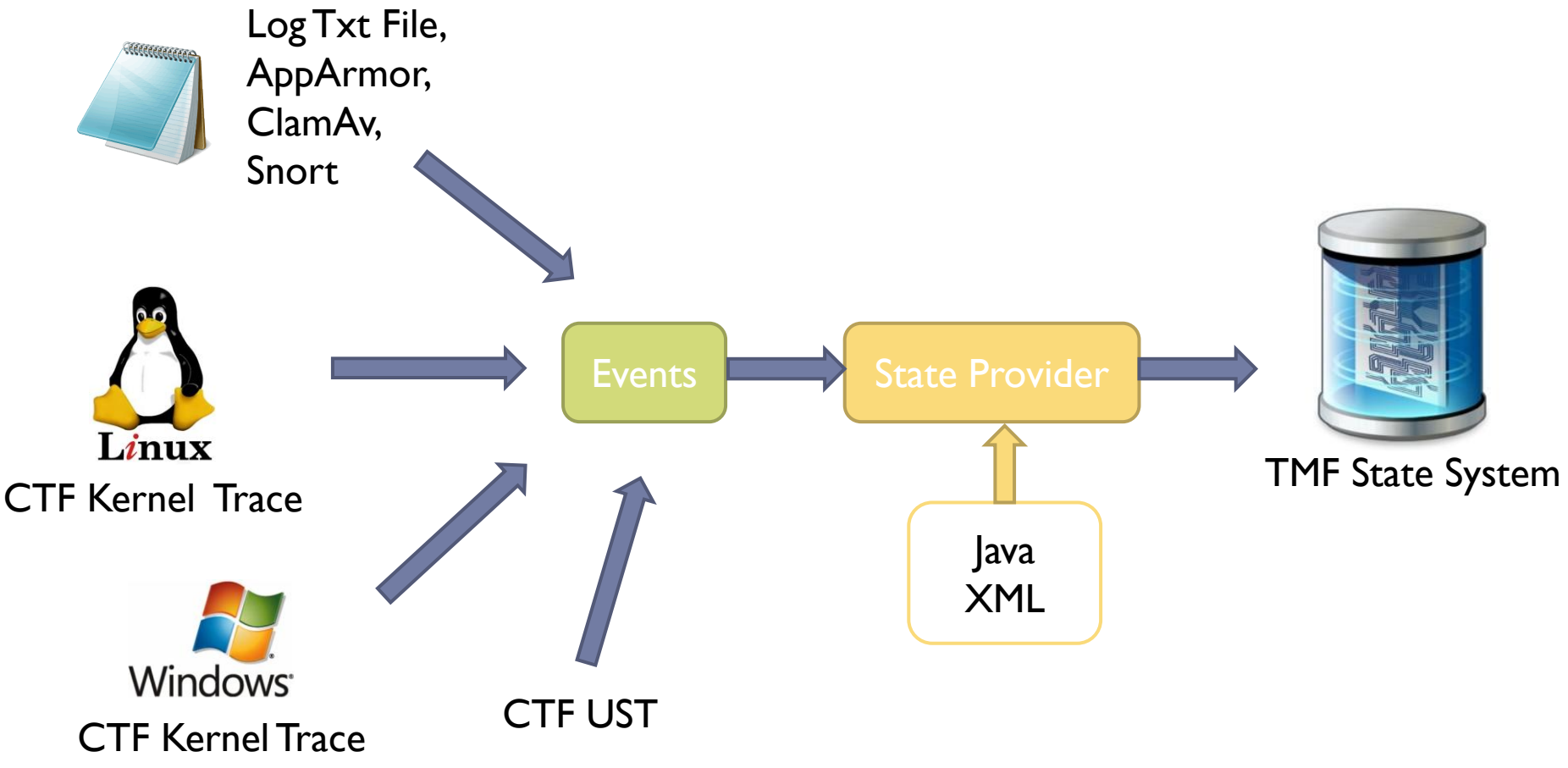
CTF (Linux)
Kernel Trace

State System



TMF Control Flow View

Motivation

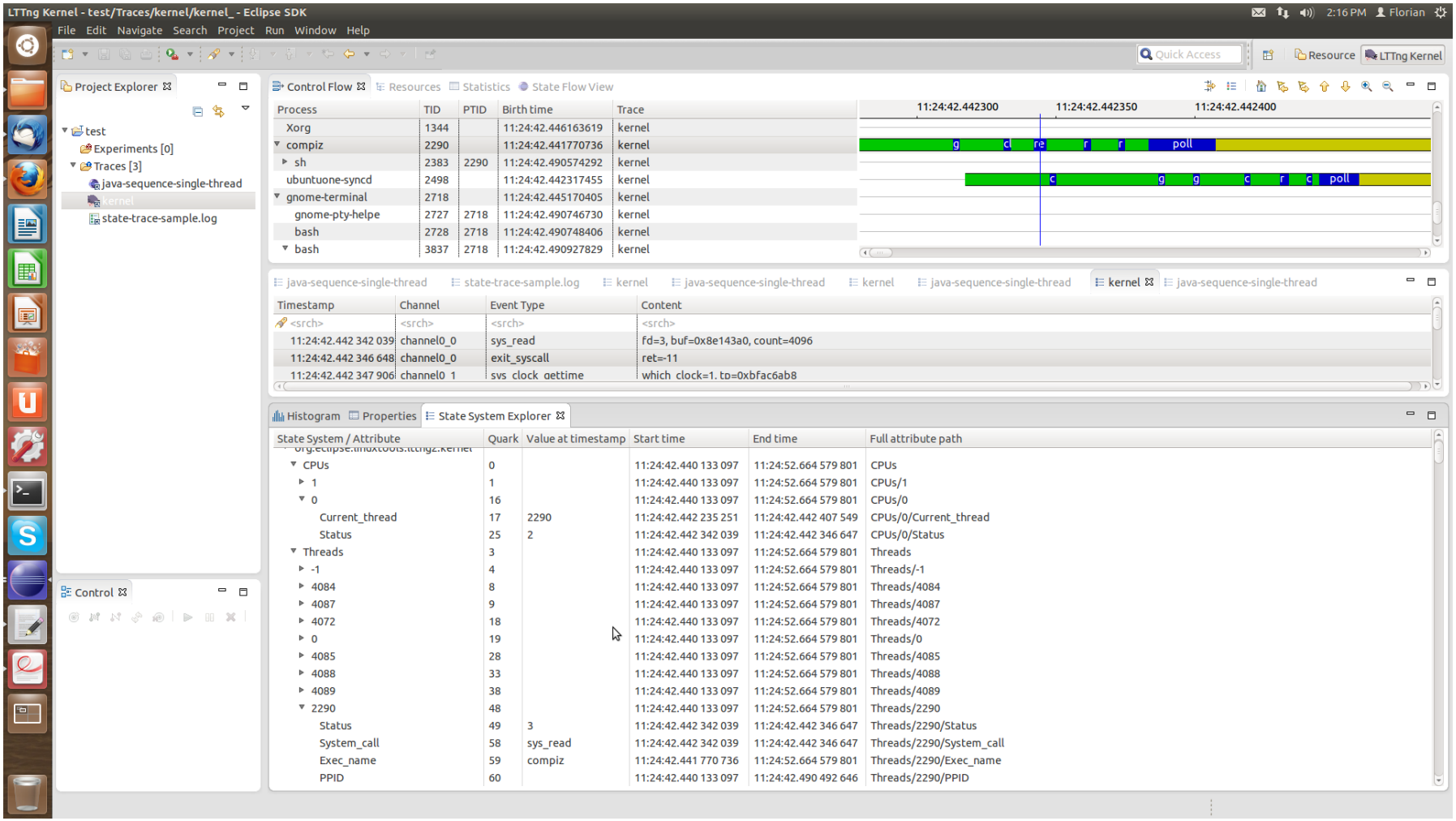

Log Txt File,
AppArmor,
ClamAv,
Snort
Linux
CTF Kernel Trace
Windows
CTF Kernel Trace
CTF UST
Events
State Provider
Java
XML
TMF State System

Motivation

- Currently the way to do the state system is hardcoded in a java file : CtfKernelStateInput.java with a big switch/case

- The idea is to transcribe it into a XML style sheet (at first) and then in the CTF metadata with an expressive language.

State System Explorer



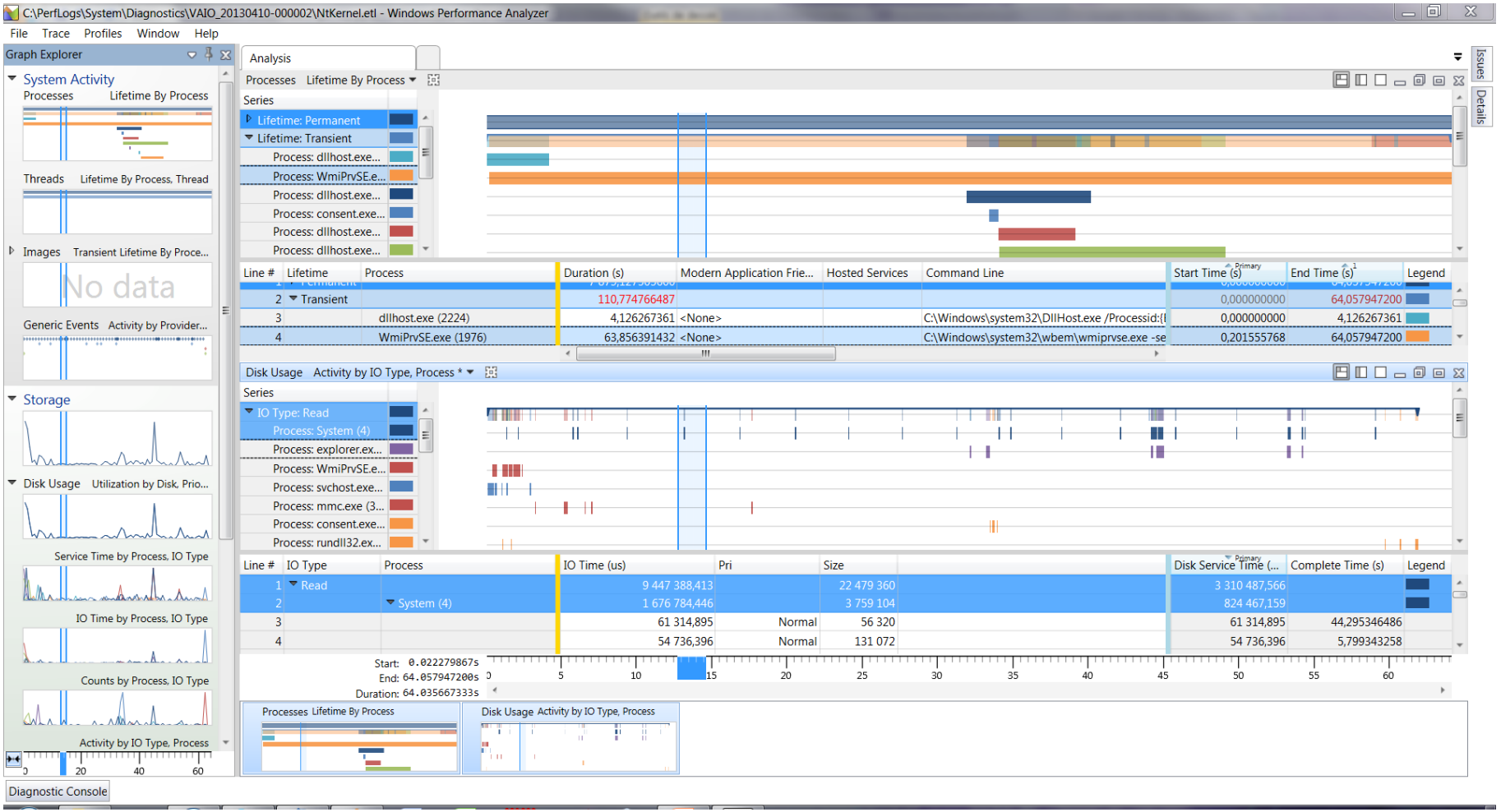
Motivation

- Windows Kernel Trace
  - ETW : Event Trace for Windows
  - .etl : etl format

- Google want to optimize Chrome
  - Convert ETW trace to CTF

- Windows Performance Toolkit

Windows Performance Toolkit



Motivation

- Automated Fault Identification,
  - Béchir Ktari, Hashem Waly, Université de Laval
  - Automating the detection in lttng traces of malicious behaviours, performance degradation, and software bugs.
  - They provide a descriptive language for scenarios
    - State machine
  - Source of inspiration to the second part: analysis with virtual states (filter)

Work

- Expression language to model the state transitions caused by an event, and filter from the state system

  /*Threads*/4/state == RUNNING

  /*Threads*/${/event/tid}/files/${/event/fd}/pathname = name

Work

- State Provider

- Java Code :

```
/* Put the process' status back to user mode */
quark = ss.getQuarkRelativeAndAdd(currentThreadNode, Attributes.STATUS);
value = TmfStateValue.newValueInt(StateValues.PROCESS_STATUS_RUN_USERMODE);
ss.modifyAttribute(ts, value, quark);
```

- Expression code :

```
/* Put the process' status back to user mode */

/Threads/[${/CPUs/[${/event/Cpu}]/Current_thread}]/Status
                                        = PROCESS_STATUS_RUN_USERMODE
```

Work

- State Provider
- Expression :

  /* Put the process' status back to user mode */
  /Threads/[${/CPUs/[${/event/Cpu}]/Current_thread}]/Status
  = PROCESS_STATUS_RUN_USERMODE

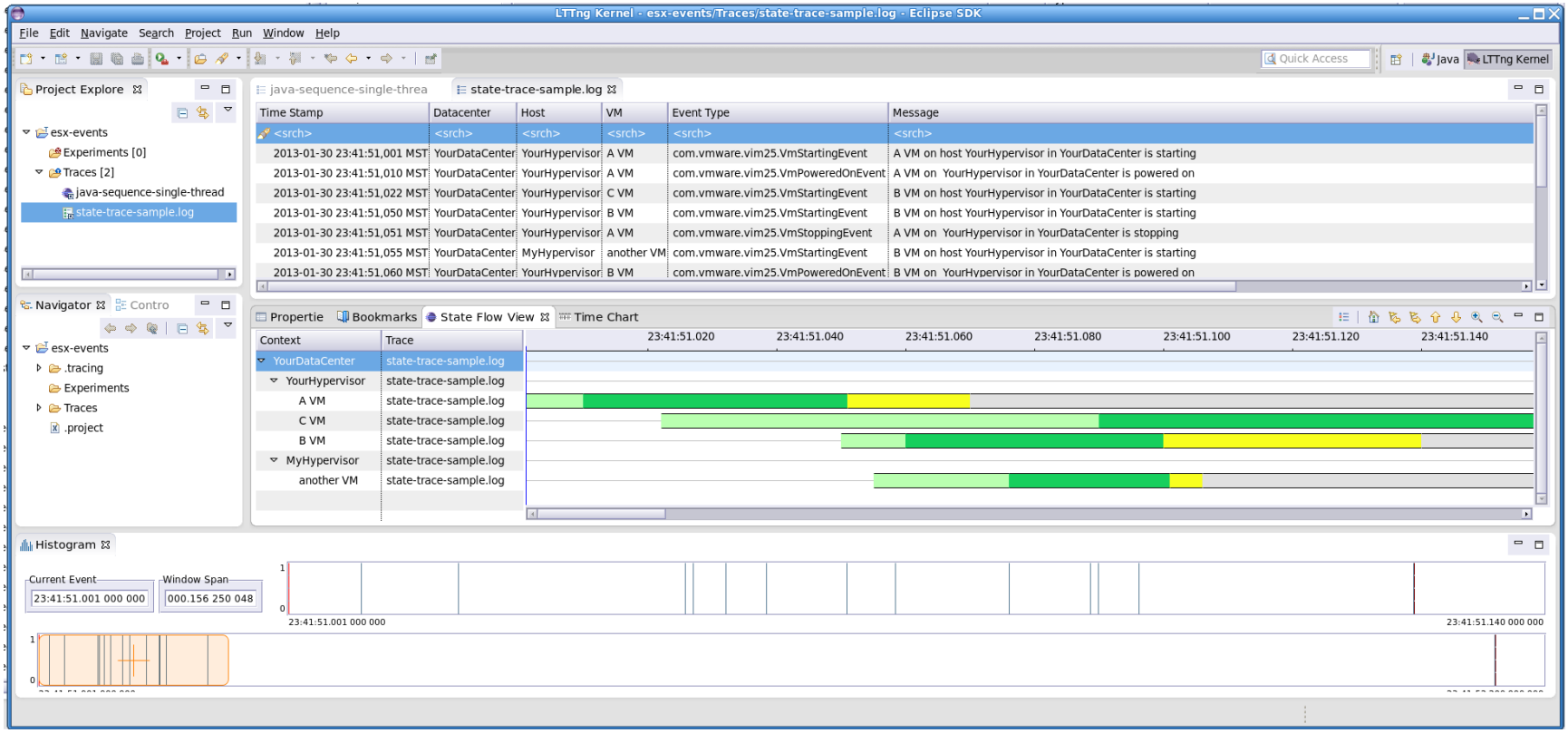
- XML :

```
<stateprovider>
   <location id="CurentThread">
       <attribute type="hardcoded" name="Threads" />
       <attribute type="query">
           <attribute type="hardcoded" name="CPUs" />
           <attribute type="field" fieldValue="cpu" />
           <attribute type="hardcoded" name="Current_thread" />
       </attribute>
  </location>

  <eventHandler eventName="exit_syscall">
     <stateChange type="modify">
       <attribute type="location" id="CurentThread">
       <attribute type="hardcoded" name="Status" />
       <value type="int" name=" PROCESS_STATUS_RUN_USERMODE "/>
     </stateChange>
   </eventHandler>
</stateprovider>
```

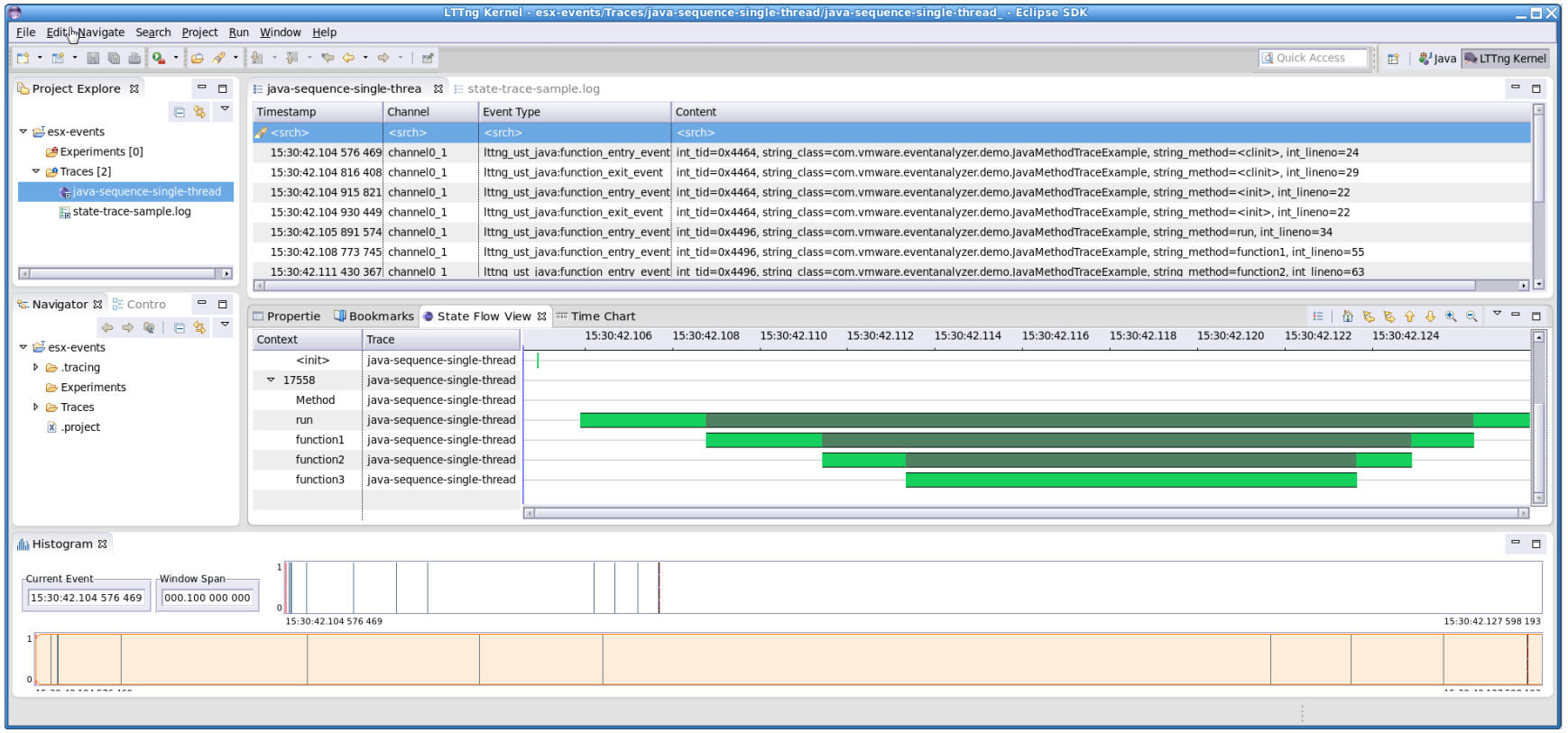
Work

- XML : Aaron Spear



Work

- XML : Aaron Spear



Work

- Context (view parameter)

```
<defineContext id="Process"   parentId=""         hasState="true"/>
<defineContext id="Thread"   parentId="Process"   hasState="true"/>
<defineContext id="Method"   parentId="Thread"    hasState="true"/>
```

- State Declaration

```
<stateDeclaration>
    <state name="Running" rgb="23,207,93">
     <value>1</value>
    </state>
</stateDeclaration>
```

- Switch Context (State Change)

```
<switchContext  eventType="lttng_ust_java:function_entry_event">
      <valueContext id="Thread"  field="int_tid"       fieldRegex="(.*)"/>
</switchContext>
```

Next Move

- Validate the syntax (expressiveness!)
- Make the Kernel State Provider for Windows with it
  - Analyze and prove it work with new architecture
- Test performance compared to Java code
- Convert the syntax to use with the CTF Metadata
  - Independence of the viewer
- Make a Gui to use virtual state to have filter with the State System.

Conclusion

- Formal expressions to have a more efficient use of the state-system in TMF
- Good way to easily use external data and make specific and flexible analysis
- More users, more contributors

Questions

Sources

- Automated Fault Identification, Kernel Trace Analysis, Mémoire de Hashem Waly, Université de Laval, 2011
- Windows ADK : http://msdn.microsoft.com/en-us/performance/cc825801.aspx
  - TraceLog
  - Windows Performance Toolkit