Visually representing data-driven analysis using state diagrams

Simon Delisle

Michel Dagenais

Progress Report Meeting
Dec 2014

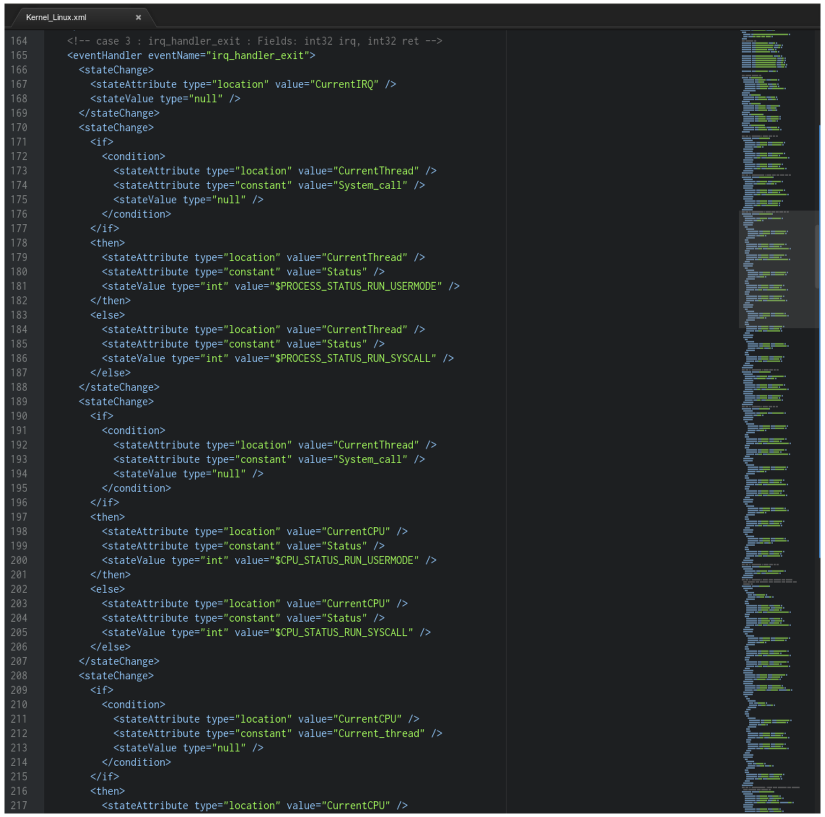
POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

Presentation plan

Introduction

- Java state provider -> XML state provider
- It might be difficult for some users to deal directly with the XML
- We need a simple UI to define those things



```xml
<!-- case 3 : irq_handler_exit : Fields: int32 irq, int32 ret -->
<eventHandler eventName="irq_handler_exit">
  <stateChange>
    <stateAttribute type="location" value="CurrentIRQ" />
    <stateValue type="null" />
  </stateChange>
  <stateChange>
    <if>
      <condition>
        <stateAttribute type="location" value="CurrentThread" />
        <stateAttribute type="constant" value="System_call" />
        <stateValue type="null" />
      </condition>
    </if>
    <then>
      <stateAttribute type="location" value="CurrentThread" />
      <stateAttribute type="constant" value="Status" />
      <stateValue type="int" value="$PROCESS_STATUS_RUN_USERMODE" />
    </then>
    <else>
      <stateAttribute type="location" value="CurrentThread" />
      <stateAttribute type="constant" value="Status" />
      <stateValue type="int" value="$PROCESS_STATUS_RUN_SYSCALL" />
    </else>
  </stateChange>
  <stateChange>
    <if>
      <condition>
        <stateAttribute type="location" value="CurrentThread" />
        <stateAttribute type="constant" value="System_call" />
        <stateValue type="null" />
      </condition>
    </if>
    <then>
      <stateAttribute type="location" value="CurrentCPU" />
      <stateAttribute type="constant" value="Status" />
      <stateValue type="int" value="$CPU_STATUS_RUN_USERMODE" />
    </then>
    <else>
      <stateAttribute type="location" value="CurrentCPU" />
      <stateAttribute type="constant" value="Status" />
      <stateValue type="int" value="$CPU_STATUS_RUN_SYSCALL" />
    </else>
  </stateChange>
  <stateChange>
    <if>
      <condition>
        <stateAttribute type="location" value="CurrentCPU" />
        <stateAttribute type="constant" value="Current_thread" />
        <stateValue type="null" />
      </condition>
    </if>
    <then>
      <stateAttribute type="location" value="CurrentCPU" />
```

Introduction - Goal

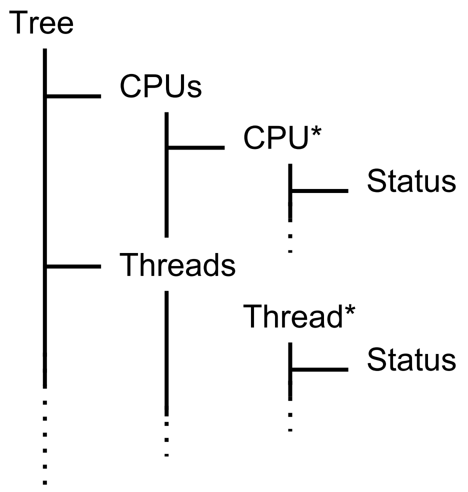
- Capturing in a convivial way all the information related to data-driven trace analysis

Related work

- State provider
- Attribute tree
- Graphiti and Eclipse Modeling Framework (EMF)

Attribute tree

- Each attribute contains a state value
- Each attribute node represents a system resource



Graphiti and EMF

- Graphiti is a graphic framework
- Editor for domain models like EMF
- We could have also used another graphic framework like Sirius

Solution

- Generate the actual XML with a modeling tool
- Develop a state machine model
  - Adapted to tracing
  - Based on UML
- Use Graphiti to manipulate this model
- Define an attribute tree for easier modeling

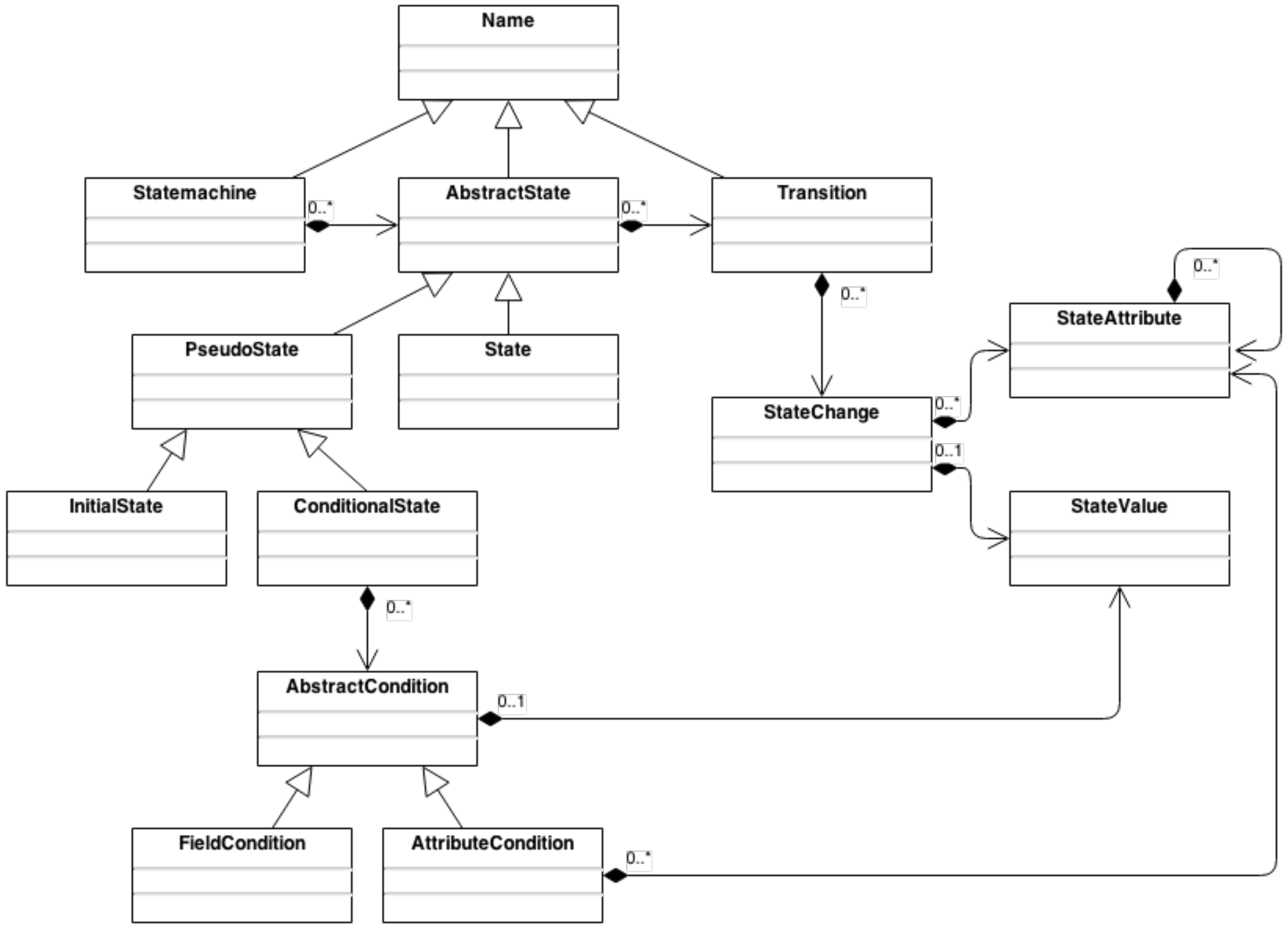
Workflow

1. Build the corresponding attribute tree for the type of trace to analyse
2. Build the state machine that represents your analysis
3. Use the generated file to execute your analysis

Results - Model

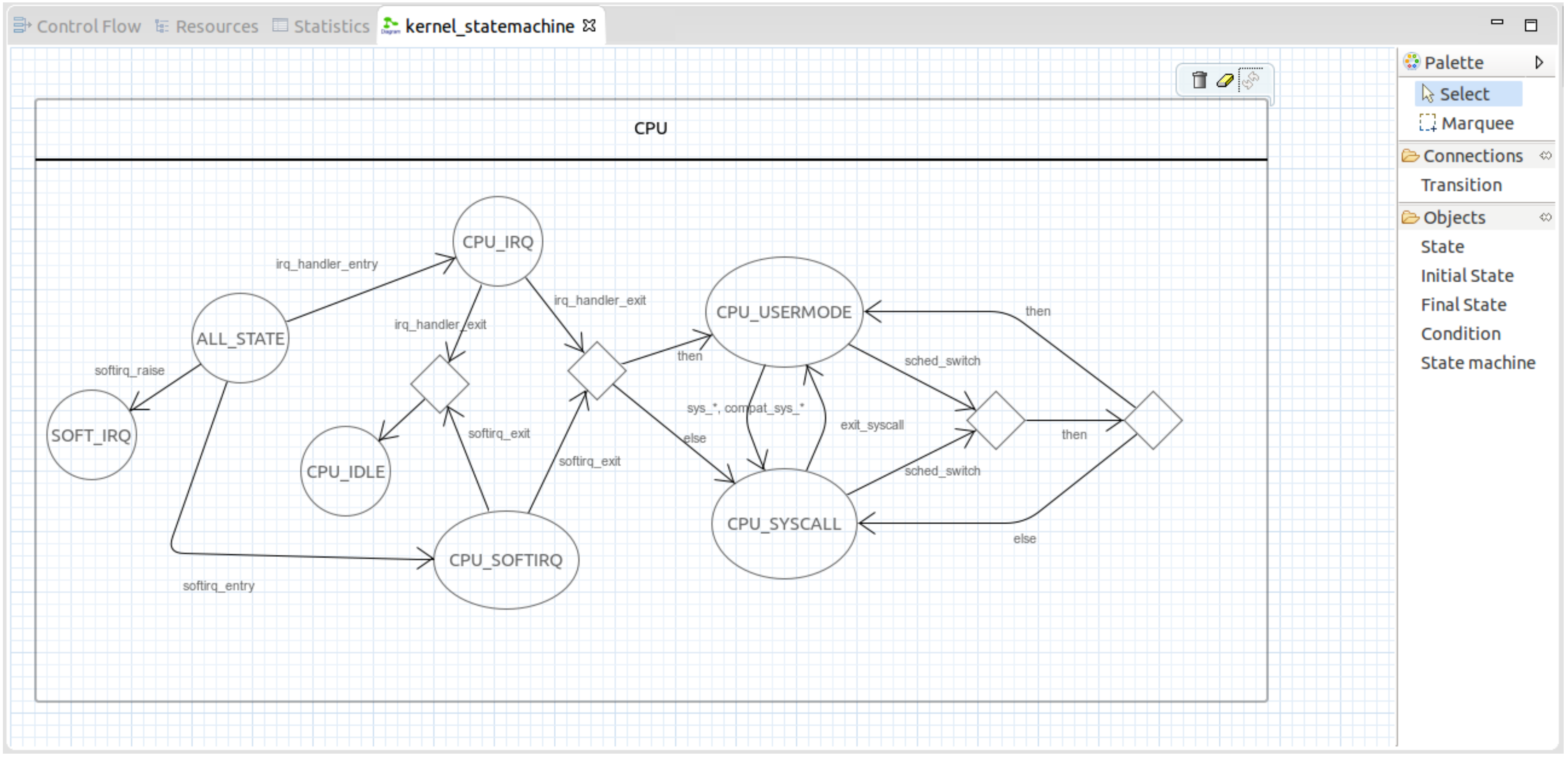
- Transition behavior -> State change
- State changes are defined as an attribute and its updated value
- The “Choice” pseudostate from UML -> Condition

Results - Model



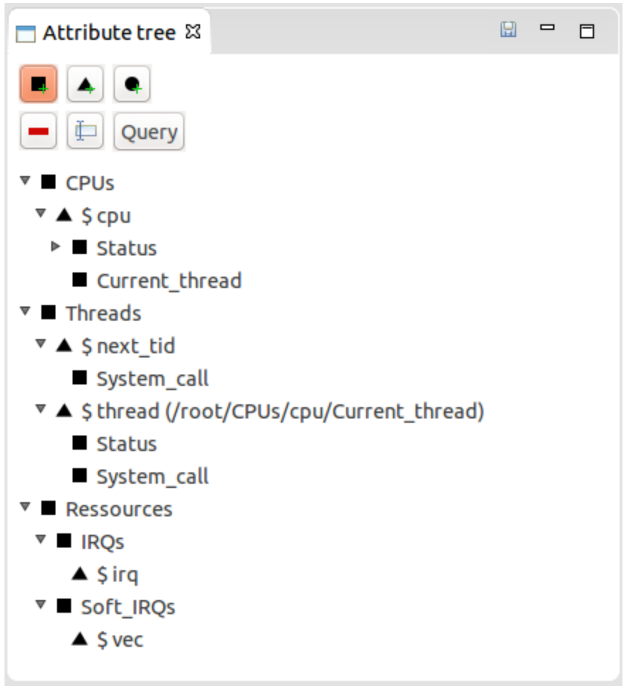
Results - Graphiti editor

- Trace analysis of the Linux kernel
- Easy to build and intuitive



Results - Attribute tree

- Define your attribute tree once beforehand
- Simply select the defined attributes when building the state machine

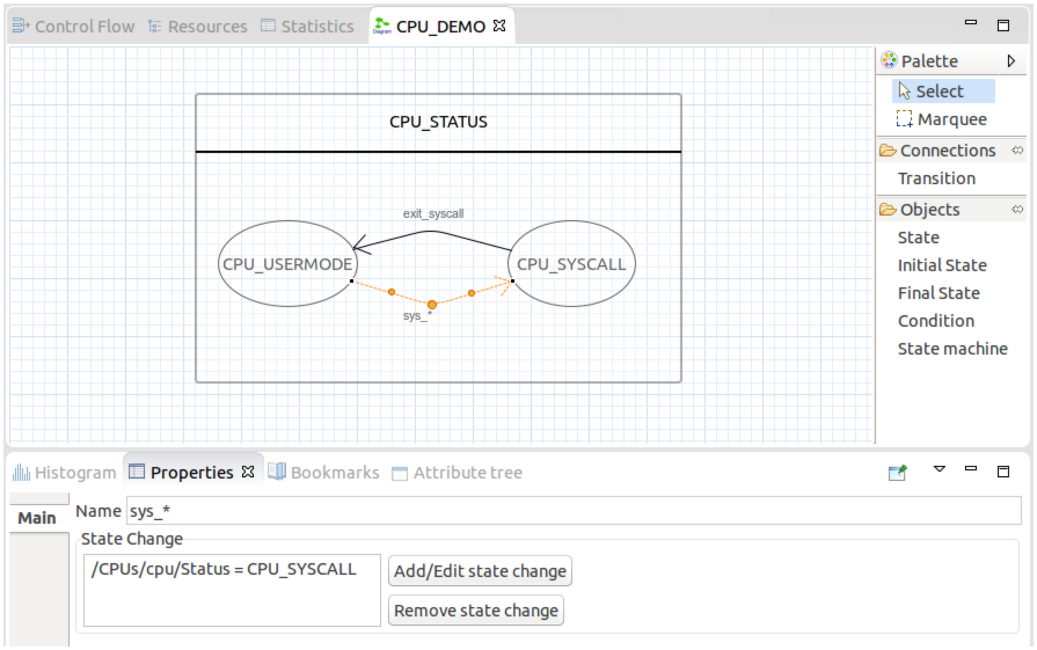

Attribute tree

Query

```
▾ ■ CPUs
  ▾ ▲ $ cpu
    ▸ ■ Status
      ■ Current_thread
▾ ■ Threads
  ▾ ▲ $ next_tid
      ■ System_call
  ▾ ▲ $ thread (/root/CPUs/cpu/Current_thread)
      ■ Status
      ■ System_call
▾ ■ Ressources
  ▾ ■ IRQs
      ▲ $ irq
  ▾ ■ Soft_IRQs
      ▲ $ vec
```

Results - State machine

- Specify the attribute that will be changed with your state machine
- Automatic state change



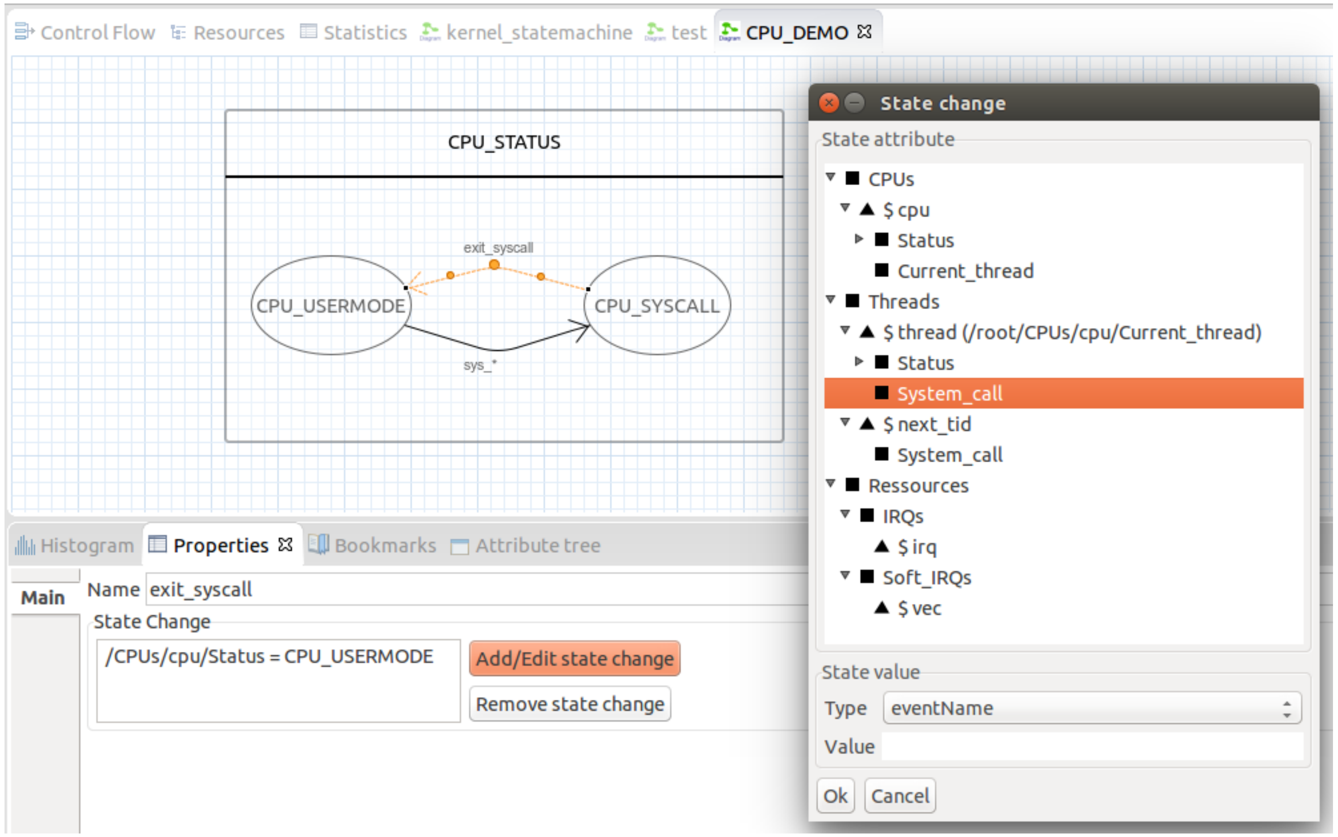
Name sys_*

State Change

/CPUs/cpu/Status = CPU_SYSCALL

Results - State machine

- Add additional information



Name exit_syscall

State Change

/CPUs/cpu/Status = CPU_USERMODE

State attribute

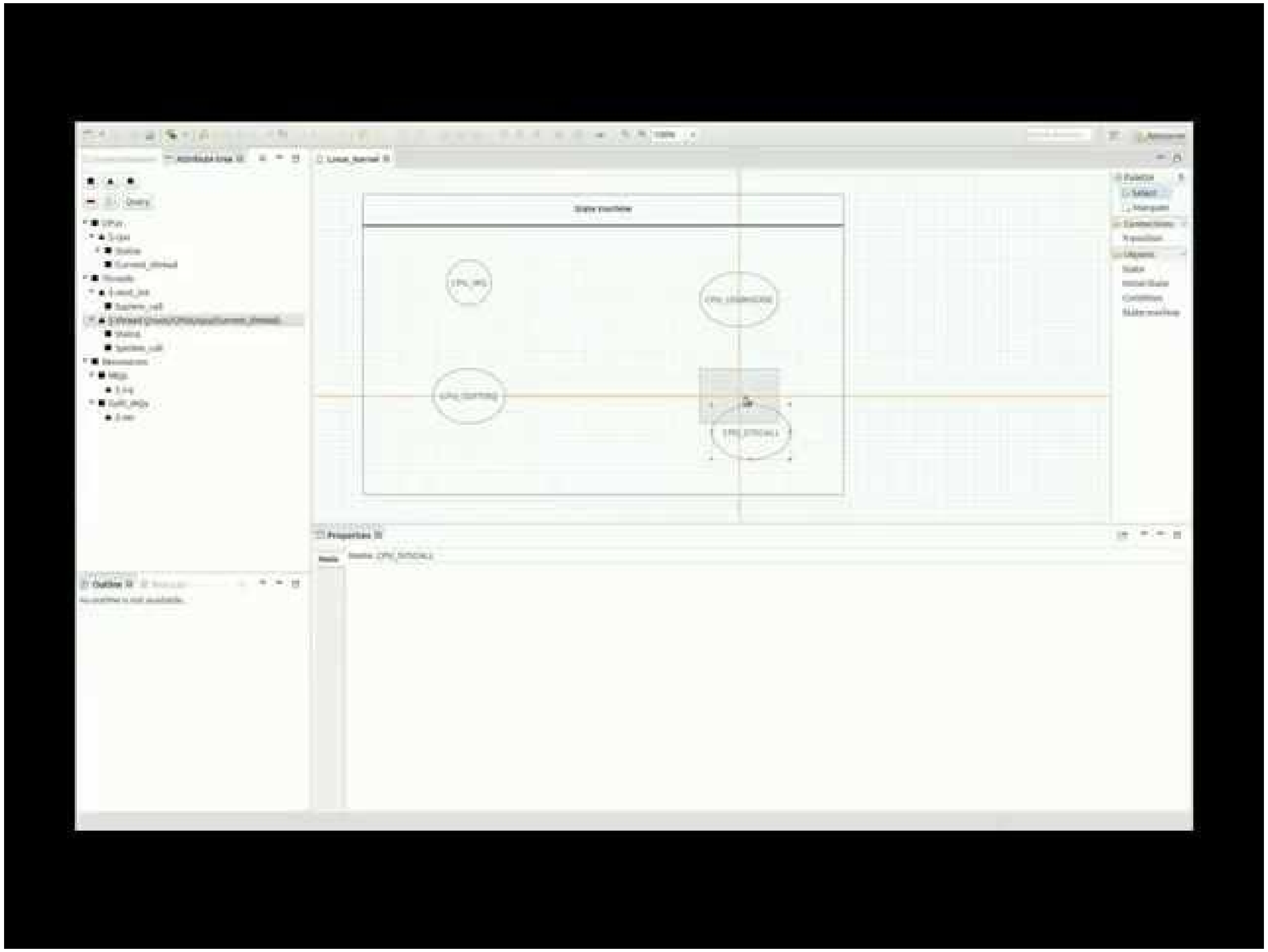
- CPUs
  - $ cpu
    - Status
    - Current_thread
- Threads
  - $ thread (/root/CPUs/cpu/Current_thread)
    - Status
    - System_call
  - $ next_tid
    - System_call
- Ressources
  - IRQs
    - $ irq
  - Soft_IRQs
    - $ vec

State value

Type eventName

Value

Demo



Results - Extract information

- We need to convert the diagram to the actual XML
- Extract information from the model that is generated
- Organize it and write the XML

Road ahead

- Filter and pattern support
- Specify views by adding information on the state diagram
- Better integration with Trace Compass
  - Synchronise views with the editor and vice versa

Conclusion

- We have an editor to capture all the information related to trace analysis
- A more efficient way to make the XML state provider