Design of flexible analysis in TMF

Florian Wininger
Presented by Geneviève Bastien

Laboratoire DORSAL
Département de génie informatique

POLYTECHNIQUE MONTRÉAL
AFFILIÉE À L'UNIVERSITÉ DE MONTRÉAL

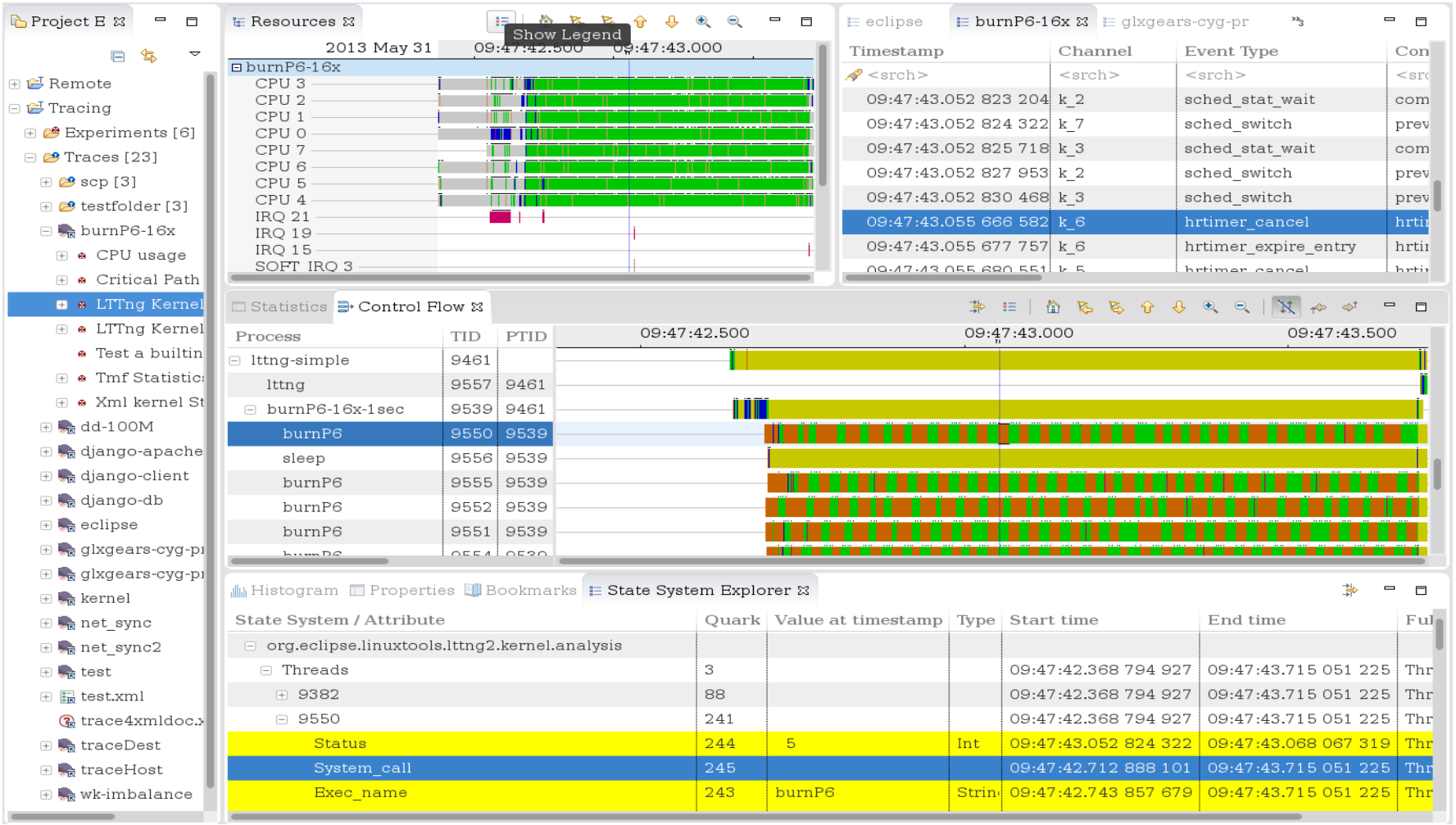
Outline

- Summary of Florian's work
- Results
- Status of the development in TMF
- Future steps

Objectives

How to build custom analyses for various trace types without having to write a single line of code?

Descriptive language to do so



Goals

- **Expressiveness**: Replace actual use cases and extend to new ones
- **Usability**: Make it easy for users to create new analyses and views
- **Performance**: Preserve or improve the actual TMF performances.

Goals

- **Expressiveness**: Replace actual use cases and extend to new ones
- **Usability**: Make it easy for users to create new analyses and views
- **Performance**: Preserve or improve the actual TMF performances.

==> Choice of XML for the syntax: extensible, widely-used, easily integrates with Eclipse and TMF

XML Syntax

- Define state changes caused by events to generate a state system

```
<stateProvider version="0" id="my.test.state.provider">
    <head>
        <traceType id="org.eclipse.linuxtools.tmf.core.development" />
        <label value="My test state provider" />
    </head>

    <definedValue name="RUNNING" value="100" />

    <eventHandler eventName="start">
        <stateChange>
            <stateAttribute type="constant" value="Tasks" />
            <stateAttribute type="eventField" value="number" />
            <stateValue type="int" value="$RUNNING" />
        </stateChange>
    </eventHandler>
</stateProvider>
```

- Define how to visualize it in a time graph view

```
<timeGraphView id="org.eclipse.linuxtools.tmf.analysis.xml.ui.views.statesystem">
    <head>
        <analysis id="my.test.state.provider" />
        <label value="My Sample XML View" />
    </head>
    <!-- StateValues -->
    <definedValue name="The process is running" value="100" color="#118811" />

    <!-- Control Flow View -->
    <entry path="Tasks/*">
        <display type="self" />
    </entry>
</timeGraphView>
```

- Further processing of events or state system for specific use cases through filters.

```
<filter name="filter_1">
    <if>
        <attribute location="App_Thread" />
        <attribute constant="Status" />
        <value int="$STATUS_WAIT_FOR_CPU" />
    </if>
    <then>
        <attribute location="Filter" />
        <attribute constant="Blocked" />
        <value int="$BLOCKED">
    </then>
    <else>
        <attribute location="Filter" />
        <attribute constant="Blocked" />
        <value int="$UNBLOCKED" />
    </else>
</filter>
```

Results (expressiveness)

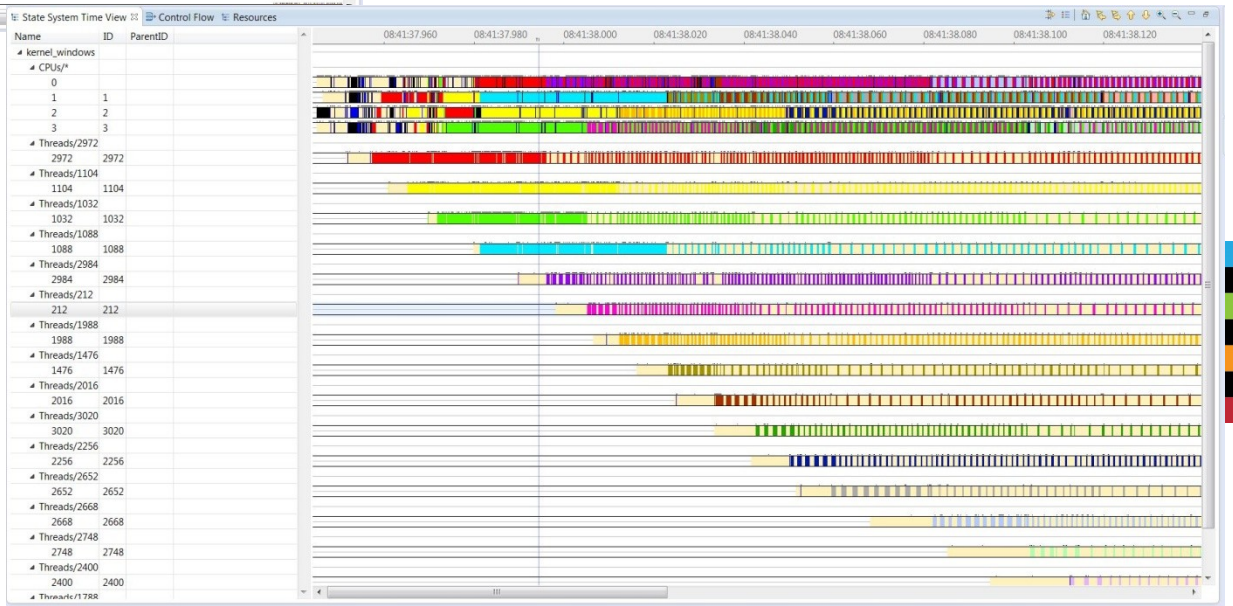
- Use a single model to compare 2 different operating systems

Linux: trace obtained with LTTng, same as the Lttng Kernel analysis in TMF

Results (expressiveness)

- Use a single model to compare 2 different operating systems

Linux: trace obtained with LTTng, same as the Lttng Kernel analysis in TMF



Windows: trace obtained with ETW and converted to CTF using ETW2CTF converter

https://github.com/fwininger/ETW2CTF/

Results (usability)

State providers	Java	XML
Development	State provider class + analysis class + plugin.xml	XML state provider element
Development environment	Eclipse SDK + TMF development environment	XML (text) editor, one will come with TMF.
Testing and debugging	Compile + execute (2nd Eclipse instance) + state system explorer	Import modified XML file + state system explorer

Views	Java	XML
Development	View class + presentation provider class + entry class + plugin.xml	XML view element (~10 lines for a simple view)
Flexibility	Fully customizable	Within limits of implemented features

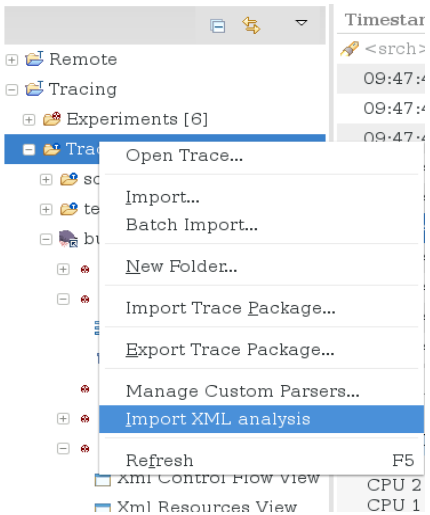
Results (performance)

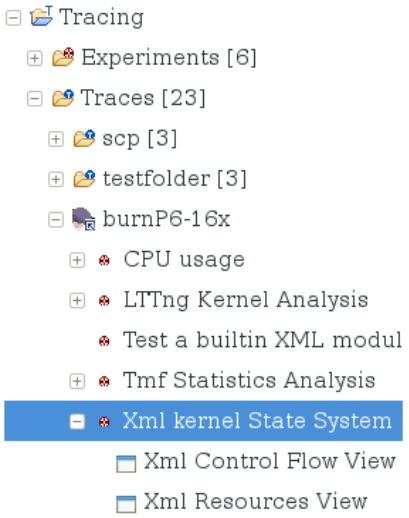
- Build time of the model

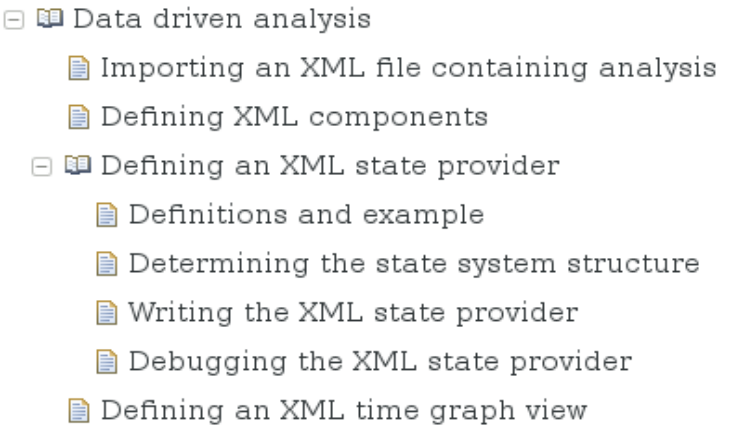
Trace 100 MB	Java	XML
Average time (s)	49.359	50.025
Standard deviation (s)	1.034	1.140
Min (s)	47.054	44.325
Max (s)	52.670	52.427

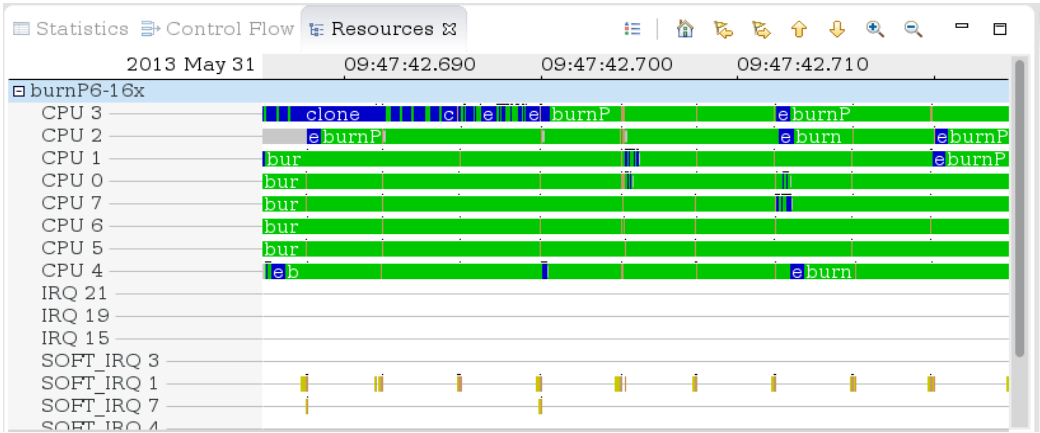
Status in TMF

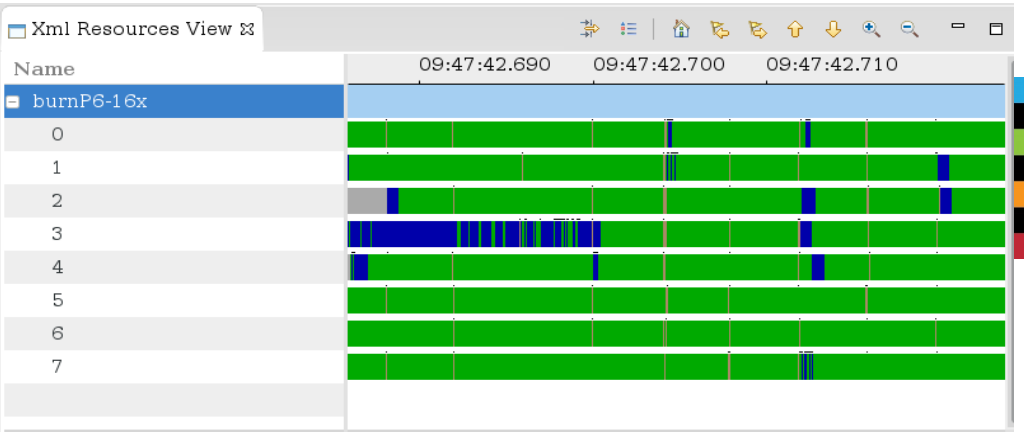
- State provider and Time graph views are in TMF master





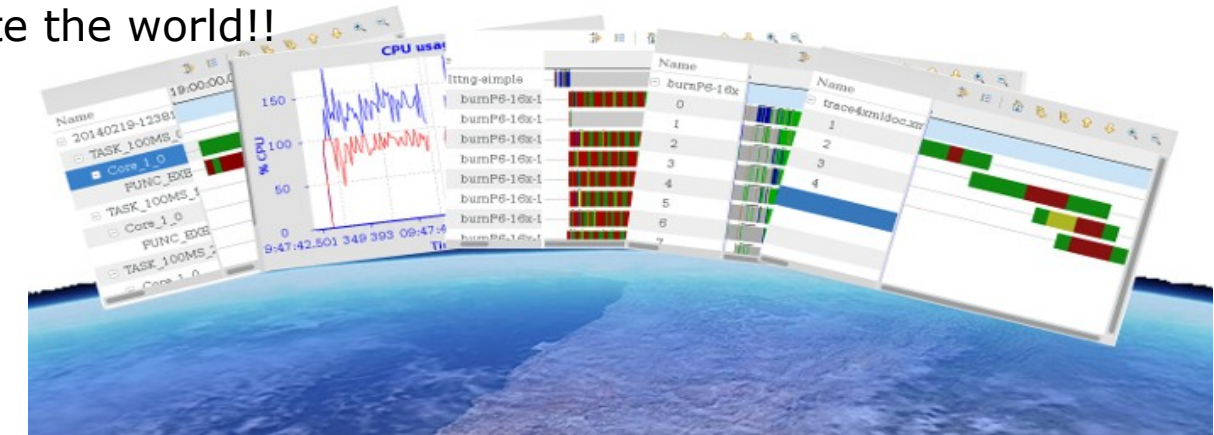






Future steps

- Short term:
  - Views: support tooltips and texts in states, XY charts
  - State providers: Add expressiveness (operations on string, regexes, save values for later use, etc.)
  - Make sure it works for experiments with different trace types
- Medium term:
  - Implement filters described by Florian
  - Generate XML state providers from UML state diagrams
- Long term:
  - Multiply analyses and dominate the world!!



Questions