Design of flexible analysis in TMF

Florian Wininger
Presented by Geneviève Bastien
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

- Define how to visualize it in a time graph view

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

<table>
<thead>
<tr>
<th>State providers</th>
<th>Java</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>State provider class + analysis class + plugin.xml</td>
<td>XML state provider element</td>
</tr>
<tr>
<td>Development environment</td>
<td>Eclipse SDK + TMF development environment</td>
<td>XML (text) editor, one will come with TMF.</td>
</tr>
<tr>
<td>Testing and debugging</td>
<td>Compile + execute (2\textsuperscript{nd} Eclipse instance) + state system explorer</td>
<td>Import modified XML file + state system explorer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Views</th>
<th>Java</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>View class + presentation provider class + entry class + plugin.xml</td>
<td>XML view element (~10 lines for a simple view)</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Fully customizable</td>
<td>Within limits of implemented features</td>
</tr>
</tbody>
</table>
## Results (performance)

- **Build time of the model**

<table>
<thead>
<tr>
<th>Trace 100 MB</th>
<th>Java</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average time (s)</td>
<td>49.359</td>
<td>50.025</td>
</tr>
<tr>
<td>Standard deviation (s)</td>
<td>1.034</td>
<td>1.140</td>
</tr>
<tr>
<td>Min (s)</td>
<td>47.054</td>
<td>44.325</td>
</tr>
<tr>
<td>Max (s)</td>
<td>52.670</td>
<td>52.427</td>
</tr>
</tbody>
</table>
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