Software Event Extension v3.0

Background

During the execution of software, execution data can be recorded. With the development of process mining techniques on the one hand, and the growing availability of software execution data on the other hand, a new form of software analytics comes into reach, i.e., applying process mining techniques to analyze software execution data. This inter-disciplinary research area is called Software Process Mining [1]. To enabling process mining for software, a software event log is required. A software event log can be recorded at method call level during software execution. Normally, a method call records software-specific information, including the called method name, the called class name, the called class object that invokes this method, the called package name, the called component name, the start time (in nanosecond precision), complete time (in nanosecond precision), the calling method name, the calling class name, the calling class object name, the calling package name and the calling component name [2].

The called method name can be conveniently stored using the standard “concept:name” attribute (that is, the name attribute of the concept extension), and the called class object using the standard “concept:instance” attribute. As a result of the latter, all calls to this object can be nicely related in the log. Furthermore, the start and completion times can be stored using the standard “time:timestamp” attribute, but this captures the times only in millisecond precision and not in nanosecond precision. Finally, we can use the standard “lifecycle:transition” attribute to store whether the event corresponds to starting the call (“start”) or completing the call (“complete”). As a result, we need additional attributes for the called class name, the called package name, the called component name, the calling method name, the calling class name, the calling class object, the calling package name, the calling component name, and the additional nanoseconds for the time as captured using the standard “time:timestamp” attribute. For this reason, we introduce the XES software extension as defined below.

Definition

The software extension defines the called class name, the called package name, the called component name, the calling method name, the calling class name, the calling class object, the calling package name, the calling component name, and the additional nanoseconds for the time as captured using the “time:timestamp” attribute for software events within a log.

<table>
<thead>
<tr>
<th>Table 1 extension definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Software extension definition</strong></td>
</tr>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td><strong>Prefix</strong></td>
</tr>
<tr>
<td><strong>Extension URI</strong></td>
</tr>
<tr>
<td><strong>XML Representation</strong></td>
</tr>
</tbody>
</table>

The extension defines the following attributes:

<table>
<thead>
<tr>
<th>Table 2 attributes definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attribute Level</strong></td>
</tr>
<tr>
<td>Event</td>
</tr>
</tbody>
</table>
Recall the method name of called method is stored using the standard “concept:name” attribute, the class object of the called method is stored using the standard “concept:instance” attribute, the time in millisecond precision is stored using the standard “time:timestamp” attribute, and that we use the standard “lifecycle:transition” attribute to specify whether the event corresponds to starting the call or completing the call.

Example

Give a simple software example. It includes three classes (Class1, Class2 and MainClass) and the sequence diagram of one execution is shown in Fig. 1. Note that the component information of a piece of software
can be obtained directly from the development documents or automatically identified using cohesion and coupling metrics. For this example software, we simply assume that MainClass belongs to component C1 and Class1 and Class2 belongs to component C2. For more discussion of component and its identification, please refer to [2].

Figure 1 Sequence diagram of the example software

A fragment of the software event log (with method calls Class1.init() and Class2.createClass1()) would look like as follows:

```xml
<log>
  <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
  <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
  <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
  <extension name="Software" prefix="software" uri="http://www.xes-standard.org/software.xesext"/>
  ...
  <trace>
    ...
    <event>
      <string key="concept:name" value="init"/>
      <string key="concept:instance" value="@3b7359cb"/>
      <string key="lifecycle:transition" value="start"/>
      <string key="time:timestamp" value="2016-4-14T18:28:07.574+02:00"/>
      <string key="software:calledClass" value="Class1"/>
      <string key="software:calledPackage" value="BE"/>
      <string key="software:calledComponent" value="C2"/>
      <string key="software:callingMethod" value="main"/>
      <string key="software:callingClass" value="MainClass"/>
      <string key="software:callingClassObject" value="@main"/>
    </event>
  </trace>
  ...
</log>
```
<string key="software:callingPackage" value="BE"/>
<string key="software:callingComponent" value="C1"/>
<int key="software:nanos" value="674916"/>
</event>

<string key="concept:name" value="setClass1"/>
<string key="concept:instance" value="@5746e7cc"/>
<string key="lifecycle:transition" value="start"/>
<string key="time:timestamp" value="2016-4-14T18:28:07.574+02:00"/>
<string key="software:calledClass" value="Class2"/>
<string key="software:calledPackage" value="BE"/>
<string key="software:calledComponent" value="C2"/>
<string key="software:callingMethod" value="init"/>
<string key="software:callingClass" value="Class1"/>
<string key="software:callingClassObject" value="@3b7359cb"/>
<string key="software:callingPackage" value="BE"/>
<string key="software:callingComponent" value="C2"/>
<int key="software:nanos" value="695155"/>
</event>

<string key="concept:name" value="setClass1"/>
<string key="concept:instance" value="@5746e7cc"/>
<string key="lifecycle:transition" value="complete"/>
<string key="time:timestamp" value="2016-4-14T18:28:07.574+02:00"/>
<string key="software:calledClass" value="Class2"/>
<string key="software:calledPackage" value="BE"/>
<string key="software:calledComponent" value="C2"/>
<string key="software:callingMethod" value="init"/>
<string key="software:callingClass" value="Class1"/>
<string key="software:callingClassObject" value="@3b7359cb"/>
<string key="software:callingPackage" value="BE"/>
<string key="software:callingComponent" value="C2"/>
<int key="software:nanos" value="723807"/>
</event>

<string key="concept:name" value="init"/>
<string key="concept:instance" value="@3b7359cb"/>
<string key="lifecycle:transition" value="complete"/>
<string key="time:timestamp" value="2016-4-14T18:28:07.574+02:00"/>
<string key="software:calledClass" value="Class1"/>
<string key="software:calledPackage" value="BE"/>
<string key="software:calledComponent" value="C2"/>
<string key="software:callingMethod" value="main"/>
<string key="software:callingClass" value="MainClass"/>
<string key="software:callingClassObject" value="@main"/>
<string key="software:callingPackage" value="BE"/>
<string key="software:callingComponent" value="C1"/>
<int key="software:nanos" value="730650"/>
</event>

…
</trace>
…
</log>

References