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 1 extension definition

Software extension definition	
Name	Software
Prefix	software
Extension URI	http://www.xes-standard.org/software.xesext
XML Representation	<extension name="Software" prefix="software" uri="http://www.xes-standard.org/software.xesext"/>

The extension defines the following attributes:

Table 2 attributes definition

Attribute Level	Key	XES Data Type	Description
Event	calledClass	string	Class name of the called method

Event	calledPackage	string	Package name of the called method
Event	calledComponent	string	Component name of the called method
Event	callingMethod	string	Method name of the calling method
Event	callingClass	string	Class name of the calling method
Event	callingClassObject	string	Class object that invokes the calling method
Event	callingPackage	string	Package name of the calling method
Event	callingComponent	string	Component name of the calling method
Event	nanos	int	The additional nanoseconds, to be added to the value of the "*time:timestamp*" attribute to obtain the timestamp in nanosecond precision

XES extension

```
<xesextension name="Software" prefix="software" uri="http://www.xes-standard.org/software.xesext">
	<event>
		<string key="calledClass">
			<alias mapping="EN" name="class name of the called method"/>
		</string>
		<string key="calledPackage">
			<alias mapping="EN" name="package name the called method"/>
		</string>
		<string key="calledComponent">
			<alias mapping="EN" name="component name the called method"/>
		</string>
		<string key="callingMethod">
			<alias mapping="EN" name="method name of the calling method"/>
		</string>
		<string key="callingClass">
			<alias mapping="EN" name="class name of the calling method"/>
		</string>
		<string key="callingClassObject ">
			<alias mapping="EN" name="class object of the calling method"/>
		</string>
		<string key="callingPackage">
			<alias mapping="EN" name="package name of the calling method"/>
		</string>
		<int key="nanos">
			<alias mapping="EN" name="additional nanoseconds for the timestamp"/>
		</int>
	</event>
</xesextension>
```

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].

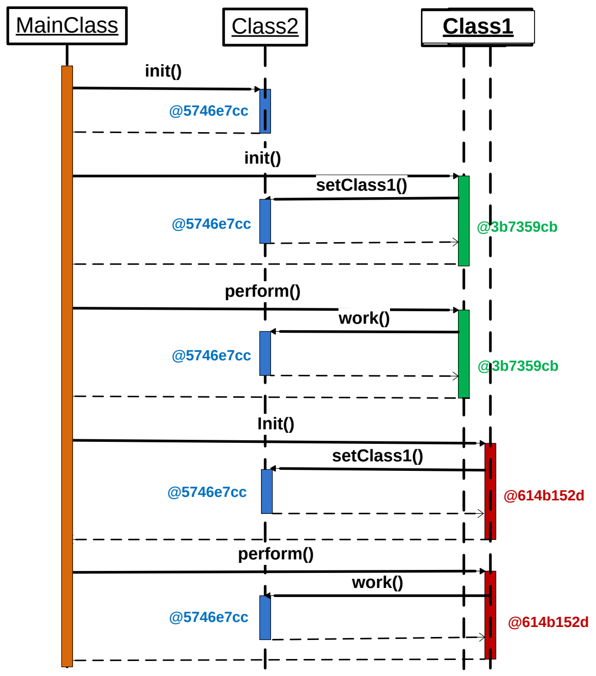

Figure 1 Sequence diagram of the example software

A fragment of the software event log (with method calls *Class1.init*() and *Class2.setClass1*()) would look like as follows:

```
<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"/>
```

```
				<string key="software:callingPackage" value="BE"/>
				<string key="software:callingComponent" value="C1"/>
				<int key="software:nanos" value="674916"/>
			</event>
			<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>
			<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>
			<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

[1] Wil van der Aalst. 2015. Big software on the run: in vivo software analytics based on process mining (keynote). In *Proceedings of the 2015 International Conference on Software and System Process* (ICSSP 2015). ACM, New York, NY, USA, 1-5.

[2] Cong Liu, Boudewijn van Dongen, Nour Assy, Wil van der Aalst, “Software component Behavior Discovery from execution data” 2016 IEEE Symposium on Computational Intelligence and Data Mining, Athens, Greece.