Micro Event Extension

Background

Many current Business Process Management Suites (BPMS) can define ***nested activities*** in order to provide a summary view and an abstract flow by hiding the nested activities. The nested activities may also be called ***sub activities***, ***micro activities, sub process***, and so on. Several BPMSs support nested activities [1-3]. Examples of nested activities include executions of web services, web pages, and manual processes. For example, in IBM BPM [1], a process designer first defines an abstract flow of activities. Then he selects one of the activities, moves to the nested activity view, and defines web pages that actually conduct the selected parent activity.

In order to correctly analyse the behaviour of such nested activities, the execution of every nested activity must be recorded in an event log. For this reason we use the XES *micro* extension. The micro extension assumes that every event has an *id* through the *identity* extension. As such, the micro extension *requires* the identity extension. The micro extension then assigns a required level and an optional parent ID to each event. These attributes and their values must be interpreted as follows:

- If the parent ID is present, then the event has the event with that ID as parent. The event must be at the next-higher level than this parent, that is, if the parent is at level X, then the event must be at level $X + 1$.
- If the parent ID is not present, then the level must be 1, that is, the lowest level. Only events at the lowest level have no parent.

For sake of convenience, we also introduce an optional length attribute, which holds the number of child events, that is the number of events that have this event as parent event.

Definition

The micro event extension defines a nesting level, a nesting parent, and the number of nested children for events within a log.

The extension is defined as shown in the table below:

Extension definition	
Name	Micro
Prefix	micro
Extension URI	http://www.xes-standard.org/micro.xesext
XML representation	`<extension name="Micro" prefix="micro" uri="http://www.xes-standard.org/micro.xesext"/>`

The extension defines the following attributes:

No	XES Attribute key	Definition	XES datatype	Occ.	Allowed values, examples, other constraints
1	level	Event level	int	1	A positive integer.
2	parentId	Id of parent event	id	0-1	May not be present if level equals 1. Must be present for all other levels. If present, then this event must have a level that is one higher than the event with the specified id.

No	XES Attribute key	Definition	XES datatype	Occ.	Allowed values, examples, other constraints
3	length	Number of child events	int	0-1	The number of child events, that is, the number of events that have this event as parent event. If not present, the number of child events must be retrieved from the event log.

XES extension

```
<xesextension
    name="Micro"
    prefix="micro"
    uri="http://www.xes-standard.org/micro.xesext">
  <event>
    <int key="level">
      <alias mapping="EN" name="Micro level of this event"/>
    </int>
    <id key="parentId">
      <alias mapping="EN" name="Id of parent event of this event"/>
    </int>
    <int key="length">
      <alias mapping="EN" name="Number of child events for this event"/>
    </int>
  </event>
</xesextension>
```

Example

Assume that a top level activity 'Register' consists of three nested activities, 'input_customer_info', 'input_insurance_app', and 'input_etc'. The execution of these micro events are stored as separate events with the top level activity as parent. The parent event of a micro event can define attributes that are valid for all micro events. For example, the total number of micro events for event 'Register' is 3.

```
<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="Micro"
    prefix="micro"
    uri="http://www.xes-standard.org/micro.xesext"/>
  <trace>
    <event>
      <id key=”identity:id” value=”3d2aa460-98dc-11e5-805c-0002a5d5c51b” />
      <string key=”concept:name” value=”Register” />
      <string key=”concept:instance” value=”My case” />
      <string key=”lifecycle:transition” value=”started” />
      <date key=”time:timestamp” value=”2015-04-13T14:02:30.287Z” />
      <int key=”micro:level” value=”1” />
      <int key=”micro:length” value=”3” />
    </event>
    <event>
```

```
      <string key="concept:name" value="input_customer_info" />
      <date key="time:timestamp" value="2015-04-13T14:02:31.199Z" />
      <int key="micro:level" value="2" />
      <int key="micro:parentId" value="3d2aa460-98dc-11e5-805c-
0002a5d5c51b" />
    </event>
    <event>
      <string key="concept:name" value=" input_insurance_app" />
      <date key="time:timestamp" value="2015-04-13T14:02:41.032Z" />
      <int key="micro:level" value="2" />
      <int key="micro:parentId" value="3d2aa460-98dc-11e5-805c-
0002a5d5c51b" />
    </event>
    <event>
      <string key="concept:name" value="input_etc" />
      <date key="time:timestamp" value="2015-04-13T14:02:49.729Z" />
      <int key="micro:level" value="2" />
      <int key="micro:parentId" value="3d2aa460-98dc-11e5-805c-
0002a5d5c51b" />
    </event>
    <event>
      <string key="concept:name" value="Register" />
      <string key="concept:instance" value="My case" />
      <string key="lifecycle:transition" value="completed" />
      <date key="time:timestamp" value="2015-04-13T14:02:51.453Z" />
      <int key="micro:level" value="1" />
    </event>
  </trace>
</log>
```

Note that because the level attribute is mandatory, it is straightforward to filter in (or out) all events at certain levels. As an example, it we only want to have the top level events, we can filter in only those events that have level 1. For this filtering, existing attribute-based filters can be used.

Relation with existing XES extensions

The micro extension requires the **identity extension** to be able to assign identities to events.

The micro extension also has a relation with the **concept and lifecycle extensions**. In a way, the concept extension (concept:name) and certain values for the lifecycle extension (lifecycle:transition, like started and completed) can be used to capture events on level 1, whereas the other values for the lifecycle extension (like input_customer_info, input_insurance_app, and input_etc) can be used to capture events on level 2. As before, the concept:instance attribute is then used to correlate different events. The example would look like 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"/>
  <string key="lifecycle:model" value="My model" />
  <trace>
    <event>
```

```
      <string key="concept:name" value="Register" />
      <string key="concept:instance" value="My case" />
      <string key="lifecycle:transition" value="started" />
      <date key="time:timestamp" value="2015-04-13T14:02:30.287Z" />
    </event>
    <event>
      <string key="concept:name" value="Register" />
      <string key="concept:instance" value="My case" />
      <string key="lifecycle:transition" value="input_customer_info" />
      <date key="time:timestamp" value="2015-04-13T14:02:31.199Z" />
    </event>
    <event>
      <string key="concept:name" value="Register" />
      <string key="concept:instance" value="My case" />
      <string key="lifecycle:transition" value="input_insurance_app" />
      <date key="time:timestamp" value="2015-04-13T14:02:41.032Z" />
    </event>
    <event>
      <string key="concept:name" value="Register" />
      <string key="concept:instance" value="My case" />
      <string key="lifecycle:transition" value="input_etc" />
      <date key="time:timestamp" value="2015-04-13T14:02:49.729Z" />
    </event>
    <event>
      <string key="concept:name" value="Register" />
      <string key="lifecycle:transition" value="completed" />
      <string key="concept:instance" value="My case" />
      <date key="time:timestamp" value="2015-04-13T14:02:51.453Z" />
    </event>
  </trace>
</log>
```

This requires knowledge on a very specific lifecycle transition model. Filtering the events then needs to take this model into account. For example, the equivalent of filtering in only the top level events requires the knowledge that, in the example, only the started and completed transitions need to be filtered in. This knowledge, and hence all possible lifecycle models, then need to be encoded in the lifecycle extension. As such, this approach is less flexible than using the micro extension.

References

[1] IBM BPM, “Modeling your business processes with IBM WebSphere Lombardi Edition, Part 3: Advanced modeling”
http://www.ibm.com/developerworks/websphere/library/techarticles/1112_wang/1112_wang.html
[2] Bizagi , “Creating a sub-process”
http://help.bizagi.com/processmodeler/en/index.html?creating_a_sub_process.htm
[3] Oracle BPM, “Using Subprocesses to Organize Your Process”
http://docs.oracle.com/cd/E25178_01/doc.1111/e15176/model_bus_procs_bpmpd.htm#CJADJBHC