User Manual

“Privacy Infringement Severity”

ProM Plug-in

Author: Sebastian Banescu
Version: 0.1

Table of Contents

1. Introduction

This document describes how the “Privacy Infringement Severity” ProM plug-in, hereafter called just “the plug-in”, can be installed and used. The plug-in was developed as an aid for privacy auditors to help with the prioritization of privacy infringements depending on their severity. Privacy infringements are detected by replaying an audit log over a business process model. A more detailed description of the technique was presented by Banescu, Petknovic and Zannone [2].

1.1 Implementation Details

We implemented the proposed privacy infringement identification and quantification technique as a plug-in in the ProM 6 framework[1]. ProM is an extensible framework that supports a broad range of process mining techniques in the form of plug-ins. The plug-in requires two inputs (an event log and a CPN model) and gives two outputs (an XML file describing all deviations of the event log from the process model and a scatter plot illustrating the severity of the privacy infringements).

The plug-in uses the Access/CPN framework [3] to interact with the CPN Tools simulator. Access/CPN provides two interfaces: one written in the Standard ML and another written in the Java programming language. In the implementation we used the latter interface since it can be easily integrated with the ProM framework, which is also written in Java. The Java interface of Access/CPN provides object-oriented abstractions for all the building blocks of CPN models (i.e., transitions, places, arcs, etc). The interface also facilitates loading a CPN model from a file created using CPN Tools and provides an object-oriented interface to the CPN model simulator.

2. Installation

This section describes the steps needed to install the plug-in. ProM contains many process mining related plug-ins, however, the “Privacy Infringement Severity” plug-in is not part of the official ProM release. Therefore, the plug-in has to be used as an Eclipse project via SVN. Here is a step-by-step tutorial for plug-in installation:

1. Download and install Eclipse from http://www.eclipse.org/
2. Install the Subclipse, SVN plug-in for Eclipse from http://subclipse.tigris.org/
3. From your Eclipse environment, make a new SVN project. Right-click on the Package Explorer tab, select *New -> Other…* and then choose “Checkout Projects from SVN” under SVN folder and then click “Next”
4. Create a new repository location using the following URL: https://subversion.assembla.com/svn/privacy-infringement-severity/
5. Select the repository you just created and click “Next”
6. Choose the “trunk” folder from the repository and click “Finish” and wait for the download to finish

[1] http://www.processmining.org/prom

7. Now you should have a new project called “Privacy” inside your “Package Explorer”
8. To run the package, right click on the “ProM with UITopia (Privacy).launch” inside the newly created project, and select *Run As… -> ProM with UITopia*
9. The first time ProM starts you need to be really patient until it initializes, it may take a few minutes.

3. Plug-in inputs

This section describes the format of all inputs used by the plugin. It also provides some guidelines on how some of these inputs can be created.

3.1 Event Logs

The input event logs need to follow the extensible event stream (XES) format[1]. ProM offers native support for log files that comply with the XES standard. The plug-in uses the OpenXES[2] library for processing event logs.

The particular XML schema that needs to be implemented by the event log files is:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://www.xes-standard.org/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="log">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="trace">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="string">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute type="xs:string" name="key"/>
                      <xs:attribute type="xs:string" name="value"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
              <xs:element name="event" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="string" maxOccurs="unbounded" minOccurs="0">
                      <xs:complexType>
                        <xs:simpleContent>
                          <xs:extension base="xs:string">
                            <xs:attribute type="xs:string" name="key" use="optional"/>
                            <xs:attribute type="xs:string" name="value"
use="optional"/>
                          </xs:extension>
                        </xs:simpleContent>
```

[1] http://www.xes-standard.org
[2] http://code.deckfour.org/xes

```
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

A very short example of a log file containing one trace with one event is:

```
<?xml version="1.0" encoding="UTF-8" ?>
<log>
  <trace>
    <string key="concept:name" value="Run 1"/>
    <event>
      <string key="concept:instance" value="Top.Alice.Authenticate_Patient{ demogr =
&amp;quot;Smith&amp;quot; }"/>
      <string key="org:resource" value="Alice"/>
      <string key="org:role" value="Receptionist"/>
      <string key="concept:name" value="Authenticate Patient"/>
    </event>
  </trace>
</log>
```

Note that a log may contain several traces and a trace may contain several events. Moreover, all the attributes listed in the example are used by the plug-in and have to be set in any log file that is provided as an input.

Description of attributes:

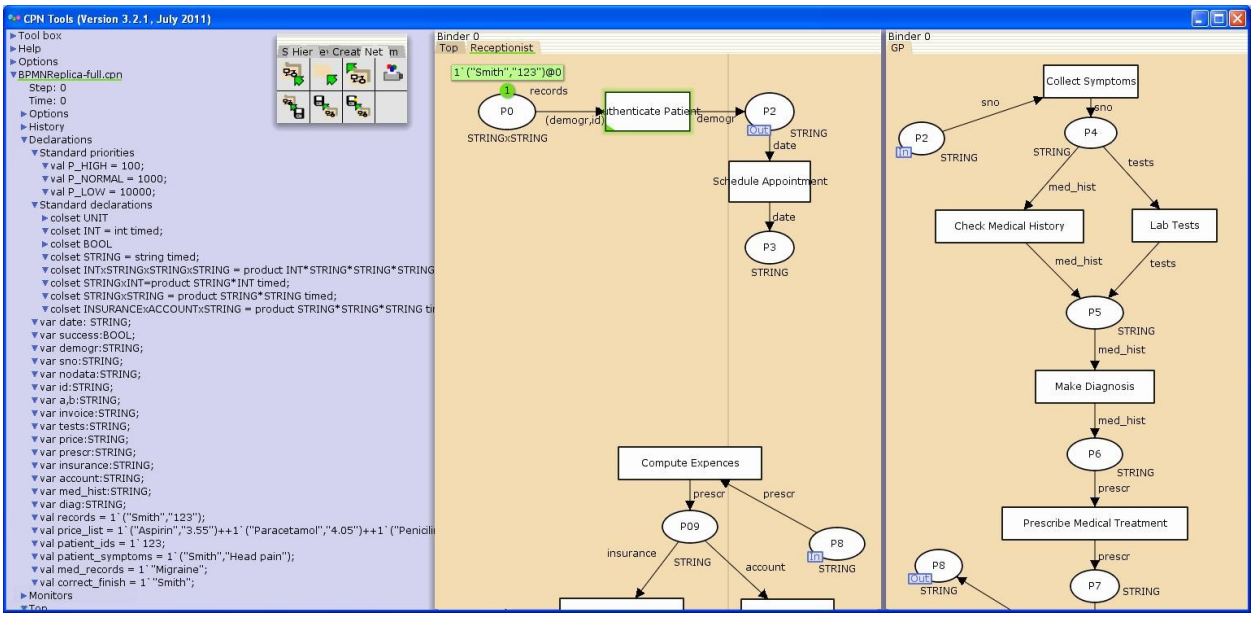
- concept:instance - represents the binding recorded by CPN Tools during simulation; it includes the hierarchical data regarding the location of the executed transition and the data items used during the execution.
- org:resource - represents the user name
- org:role - the role held by the user during execution
- concept:name - the name of the transition

3.2 CPN Models

CPN Tools[1] provides a mature environment for constructing, simulating, and performing analysis of CPN models. A tutorial on building CPN models using CPN Tools is out of the scope of this document. The reader is referred to the CPN Tools website for such purposes.

A screen shot of CPN Tools with a CPN model containing multiple modules on different pages is presented in the figure below.

[1] http://cpntools.org/



The guideline for designing CPN models are the following:

- Add all the transitions belonging to a role on a single page labeled with the corresponding role name
- The communication between all pages representing user roles has to be done through input/output ports via a page labeled “Top”.
- The “Top” page will contain only substitution transitions corresponding to the roles, where each such substitution transition has to be labeled with a user name.

3.3 Privacy Weights

The file containing information about the sensitivity of data items must follow a simple text format. Every line from this file must contain the identifier of the data item and a positive integer number indicating its sensitivity, separated by a blank space. An example of such a file is presented next:

```
id 3
demogr 2
med_hist 10
diag 8
date 1
email 2
sno 7
prescr 6
insurance 2
account 10
nodata 0
```

Note that the same identifier names have to be used on the CPN model arcs and for the `concept:instance` attribute in the log files.

3.4 User Reputation

The file containing information about the user reputation levels follows a similar format to the previous file, except that each line contains a user identifier and a real number between 0 and 1 indicating that user's reputation level. An example of such a file is presented next:

```
Alice 0.5
Bob 0.9
Charlie 0.6
Mallory 0.1
none 1
```

3.5 Task and Role Distance

The file containing information about the similarity distance of the tasks also follows a simple text format, however, its structure is different. The first line consists of a list *L* of all the task identifiers separated by blank spaces surrounded by brackets. The following |*L*| lines form an |*L*| *x* |*L*| matrix of real numbers. Each element of the matrix $m_{i,j}$ indicates the similarity between tasks L_i and L_j. An example of a file containing a 4 by 4 matrix is presented next:

```
(Authenticate Patient) (Schedule Appointment) (Compute Expences) (.)
0.0 0.8 1.0 1.0
0.8 0.0 0.8 1.0
1.0 0.8 0.0 1.0
1.0 1.0 1.0 1.0
```

The file containing information about the similarity of roles has the same format as the previous file. An example of a file containing a 4 by 4 matrix is presented next:

```
Receptionist GP LabTechnician .
0.0 0.8 0.6 1.0
0.8 0.0 0.5 1.0
0.6 0.5 0.0 1.0
1.0 1.0 1.0 1.0
```

Note that for the second files the role names are not surrounded by brackets. However, role names composed of several words are written together. For experimental purposes the Matrix comparison tool provided on the Colorado University Boulder website[1] was used.

3.6 Configuration file

The configuration file (“config.txt”) has to be placed in the working directory and it contains 5 constant values that are used by the privacy metric and 4 paths to default input files.

The privacy metric which this plug-in is based on was developed by Banescu and Zannone [1] and its analytical formula is presented next:

$$\Phi(a,e) = (c_1 + c_2\rho)[(1 + c_3 s_R)(1 + c_4 s_T)(1 + c_5\Omega) - 1]$$

where:

- a - represents the activity that should have been executed according to the business process specification
- e - represents the log event that indicates the activity that was actually executed by the user

[1] http://lsa.colorado.edu/

- $c_i \in R^+, i \in \{1,2,...,5\}$- are constants that are used to tune the metric (these constants can be set in the configuration file "config.txt")
- ρ - represents the reputation of the user
- s_R - represents the similarity of the role held by the user and the role that should have executed a
- s_T - represents the similarity of the task that was executed to the task that should have been executed
- Ω - represents the privacy penalty due to unauthorized access to data items. It is actually the sum of privacy weights corresponding to the data items that were accessed and should not have been accessed by the user during the execution of the activity from event e.

The rationale of this metric is that the role distance, task distance and sum of privacy weights determine the severity of the infringement. The relevance of these factors on the severity is determined by constants c_3, c_4 and c_5 , respectively. The reputation of the user performing the activity scales the effects of all other inputs. The constraint $c_1 > c_2$ is needed to capture the infringement when a user with high reputation (ρ = 1) deviates from the specification. Intuitively, the reason for including all combinations between the privacy factors in the previous equation, prevents the quantification from being nullified in case one of the factors is zero.

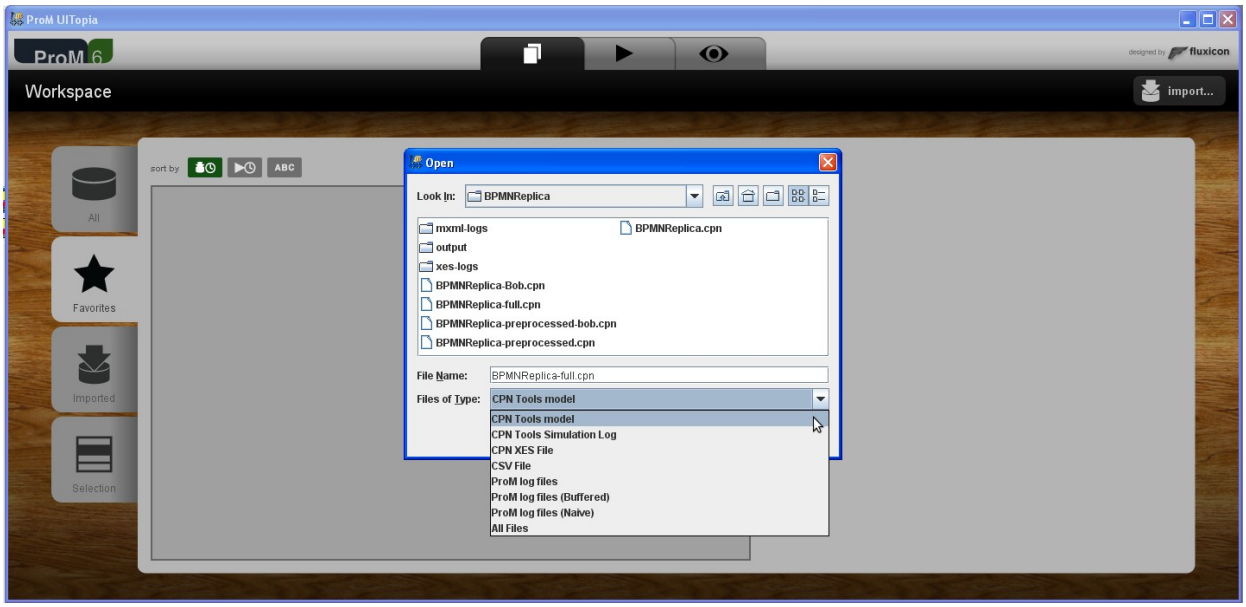
Indicating the privacy weights, user reputation, task and role distances each time you run the plug-in may be tedious because most likely you will use the same files for several different logs and CPN models. That is why the default values for these 4 files can be set in the configuration file. An example of such a file is presented next:

```
1.1
1.0
1.0
1.0
1.0
taskSimilarity.txt
roleSimilarity.txt
reputation.txt
privacyWeights.txt
```

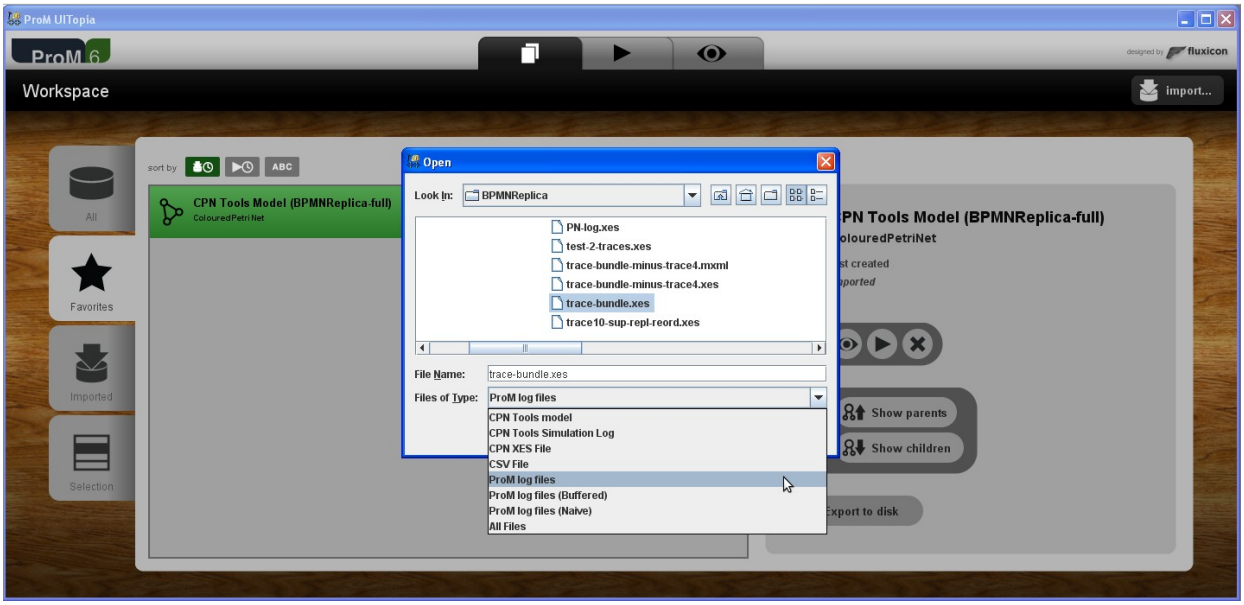
Note that the first five lines of the file contain the values for constants c_1, c_2, c_3, c_4 respectively c_5. The following 4 lines contain the file paths relative to the eclipse project directory (absolute path may also be given) for the task similarity, role similarity, user reputation respectively privacy weights input files.

4. Step-by-step tutorial

This section describes and illustrates how to load the inputs and run the plug-in. When you start ProM and go to the "Workspace" tab (left-most tab in the center), you will be able to see an "Import" button in the top-right corner of the window. You will have to click that button and import the event log and the CPN model that will be used by the plug-in. The following figure shows how to load the CPN Model:

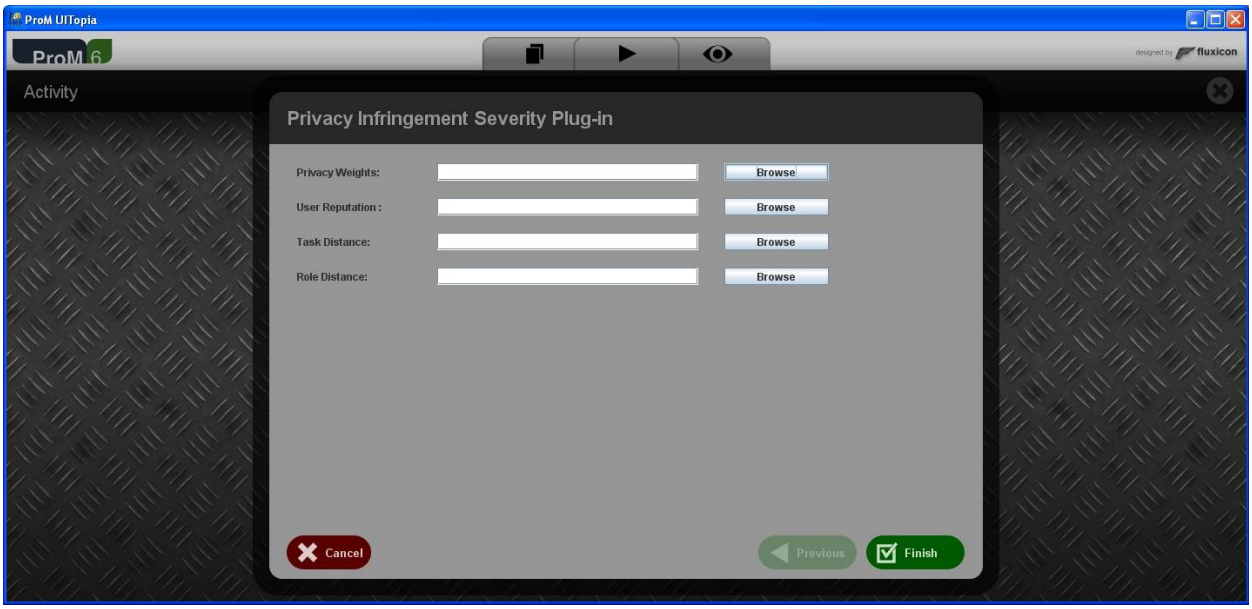


Note that the file type has to be “CPN Tools model”, otherwise the plug-in does not work. The next figure shows how to import the log file:



Note that the file format can be any of the XES native formats supported by ProM. However it needs to follow the event log schema and guidelines presented in the previous section. The next step is to move to the “Actions” panel of ProM (the middle tab at the top) and find the “Privacy Infringement Severity” plug-in in the list of actions. And then select the two previously imported items in the left side-bar called “Inputs”. Selection is done by double-clicking an input placeholder and then double-clicking the desired resource from the pop-up window. The next figure shows the plug-in with the two inputs already selected:

The next step is to run the plug-in by clicking the “Start” button on the lower-right side of the action list. This will trigger an input dialog to be displayed as shown in the following figure:



This dialog is used to provide the paths to the 4 text files containing the:

- Privacy weights;
- User reputation levels;
- Task distances;
- Role distances.

After providing the paths to these files click the “Finish” button in the lower-right corner to trigger the execution of the plug-in.

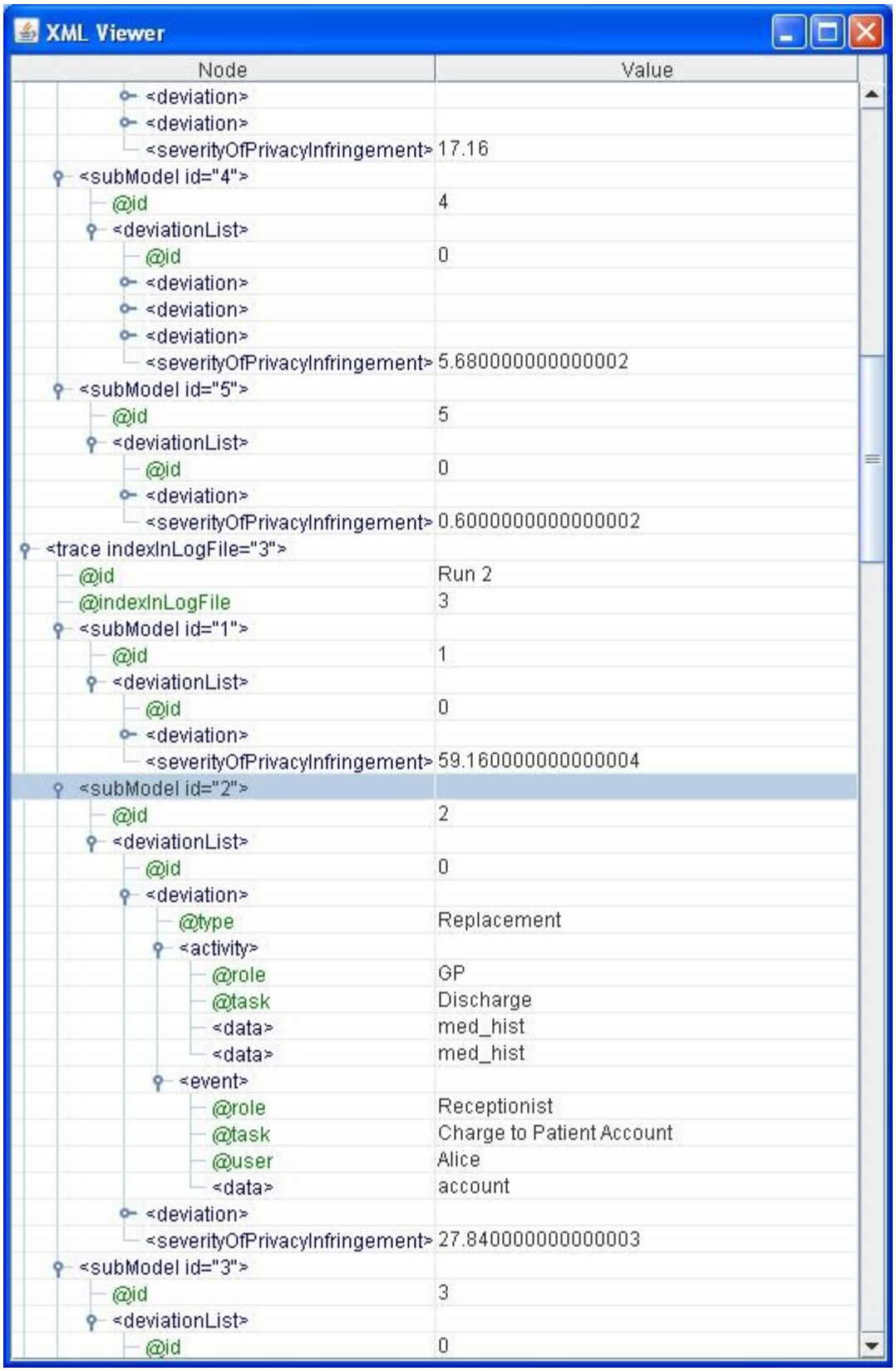
5. Plug-in outputs

The execution time of the plug-in varies with the size of the CPN model and specifically the number of XOR-gates in the model. Naturally the size of the event log is also a very important factor that determines the execution time. After execution is finished two outputs will be given:

- an XML file describing all deviations of the event log from the process model;
- a scatter plot illustrating the severity of the privacy infringements.

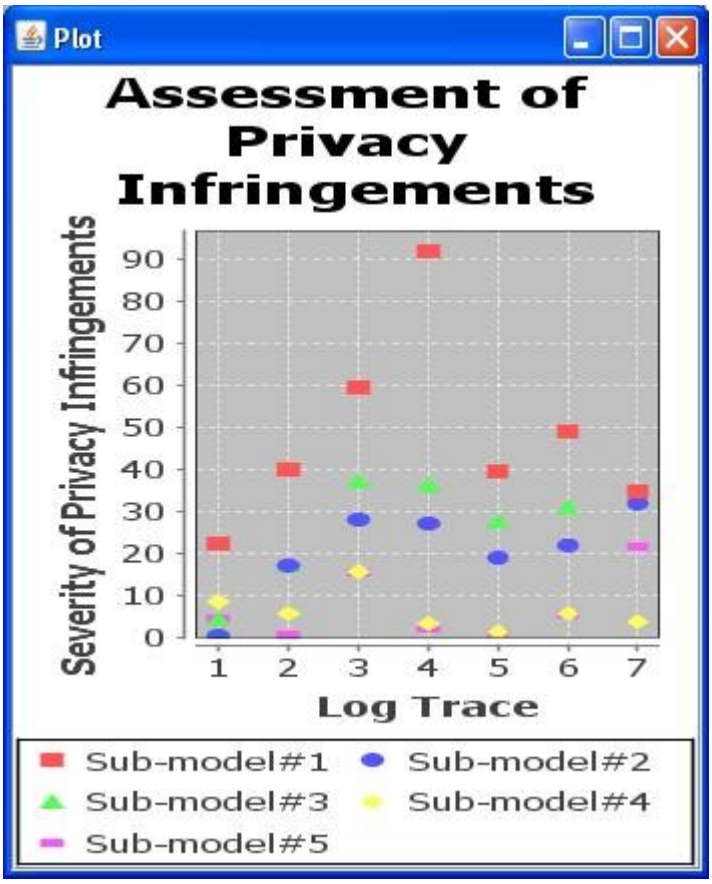
5.1 XML File

One output of the plug-in consists of several sets of deviation lists for each trace in the event log. Each log trace is analyzed by the privacy conformance checking algorithm against each sub-model output by the algorithm. Therefore, one set of deviation lists is associated to each pair of log trace and sub-model. A set of deviation lists consists of one or more deviation lists. Each deviation list comprises of zero or more deviations and the quantification of the privacy infringement. A deviation is described by its type, activities and events it involves. The quantification of the privacy infringement is a positive real number computed per deviation list by summing up the quantification of all the individual deviations in that list. The format of the output follows an XML tree-structure which is visualized using the JTree GUI widget from the Java Swing toolkit. A screenshot of the visualization is presented in the following figure:



5.2 Scatter Plot

Another output of the plug-in consists of a scatter plot as illustrated in the following figure:



The horizontal axis of the plot represents the traces from the event log. The vertical axis represents the severity of the privacy infringement. Every sub-model output by the pre-processing phase is associated with a data series represented by a distinctly colored mark. Each data series consists of k values, where k is the number of traces in the event log. Every value from a data series is computed by taking the mean of the privacy infringement quantifications of all deviation lists from the deviation list set associated with a trace and sub-model pair. For the implementation of the scatter plot visualization the JFreeChart[1] Java library was used.

[1] http://www.jfree.org/jfreechart

Bibliography

1. Banescu, Sebastian, and Nicola Zannone. "Measuring privacy compliance with process specifications." *Security Measurements and Metrics (Metrisec), 2011 Third International Workshop on* 21 Sep. 2011: 41-50.
2. Banescu, Sebastian, Milan Petković, and Nicola Zannone. "Measuring Privacy Compliance Using Fitness Metrics." *Business Process Management* (2012): 114-119.
3. Westergaard, Michael, and Lars Kristensen. "The access/CPN framework: a tool for interacting with the CPN tools simulator." *Applications and Theory of Petri Nets* (2009): 313-322.