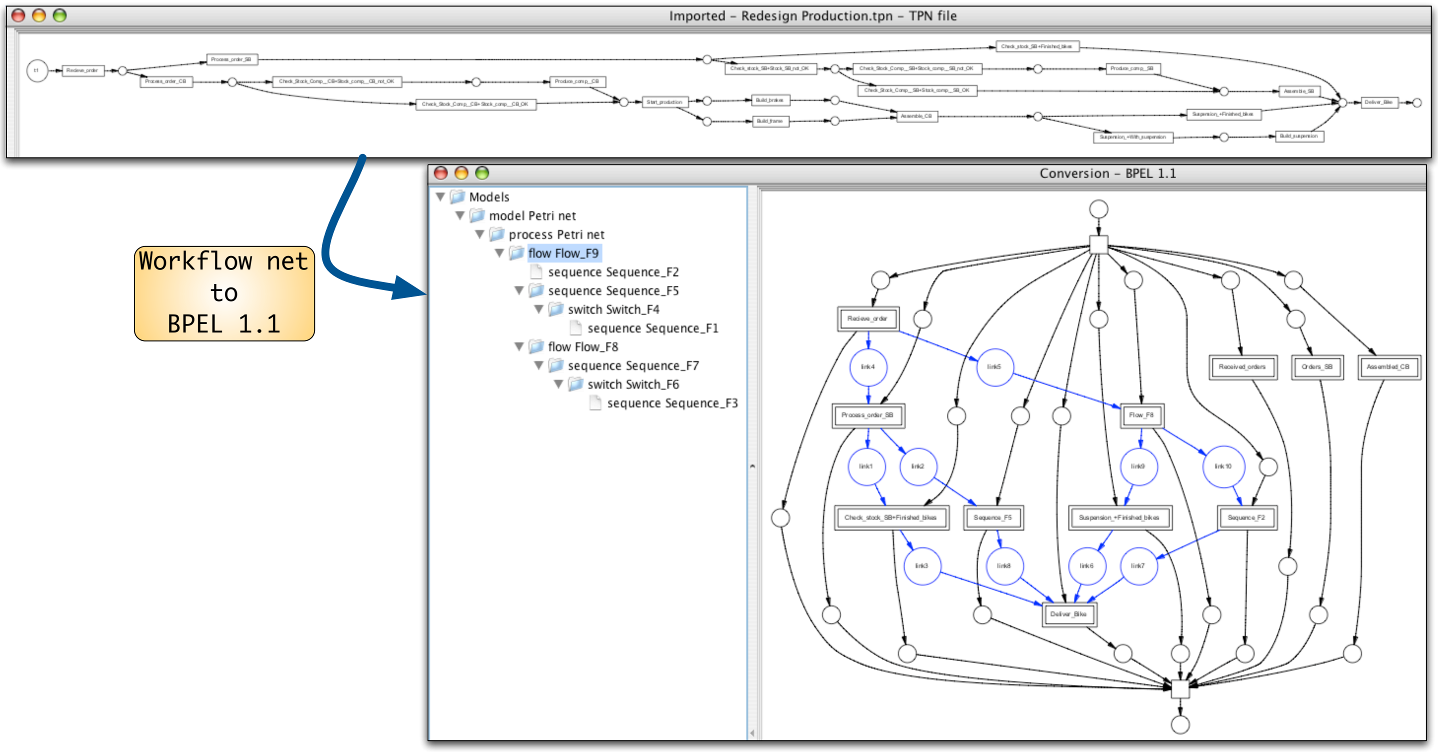
CONVERTING WORKFLOW NETS TO BPEL 1.1 IN PROM



Manual for Converting Workflow Nets to BPEL 1.1 in ProM

Kristian Bisgaard Lassen, M.Sc.
Department of Computer Science, University of Aarhus
IT-parken, Aabogade 34
DK-8200, Aarhus N, Denmark
Marts 2007

Introduction

In this manual I will show how you can translate your Workflow nets into BPEL 1.1 in ProM. This manual is structured as follows: In the first part, I will explain what ProM is, and how you may obtain it; in the second part, I will show how you may obtain a Workflow net in ProM so you can use the conversion plug-in; in the third part, we will look into how exactly the conversion plug-in should be used; finally, I will show what you can do with the generated BPEL 1.1 model. The plug-in in this paper build on the work of Wil M.P. van der Aalst and Kristian Bisgaard Lassen on translating Workflow nets to BPEL 1.1.

References

- Online version of the paper *"Translating Unstructured Workflow Processes to Readable BPEL: Theory and Implementation"*, can be found here.

What is ProM?

ProM (**Pro**cess **M**ining) is a framework used for process mining and related topics. It is design as a plug-able architecture so it is easy for contributors in the field to extend the functionality of ProM with their own tools. Currently more than 130 plug-ins have been added. There are five categories of plug-ins:

1. **Mining plug-ins** which implement some mining algorithm, e.g., mining algorithms that construct a Petri net based on some event log.
2. **Export plugins** which implement some "save as" functionality for some objects (such as graphs). For example, there are plugins to save EPCs, Petri nets, spreadsheets, etc.
3. **Import plug-ins** which implement an "open" functionality for exported objects, e.g., load instance-EPCs from ARIS PPM.
4. **Analysis plug-ins** which typically implement some property analysis on some mining result. For example, for Petri nets there is a plugin which constructs place invariants, transition invariants, and a coverability graph.
5. **Conversion plug-ins** which implement conversions between different data formats, e.g., from EPCs to Petri nets and from Petri nets to YAWL.

The plug-in described in this manual falls into the category of conversion plug-ins. Notice that since many conversion plug-ins have been added it is often the case that one can convert one model type to practically any other. For example, one may convert and EPC to a Petri net, and then convert this Petri net to YAWL, if the goal was to convert an EPC model to a YAWL model.

ProM is written in Java and will therefore work on Windows, Linux, and Mac OSX.

References

- ProM can be downloaded from prom.sourgeforge.net.
- More background information on ProM, and process mining in general can be found at processmining.org.

How Do I Obtain a Workflow Net?

The Workflow net to BPEL conversion plug-in in ProM it do actually not work on Workflow net objects, only Petri nets. It will, however, test the Petri net for the simple static properties of Workflow net, i.e.:

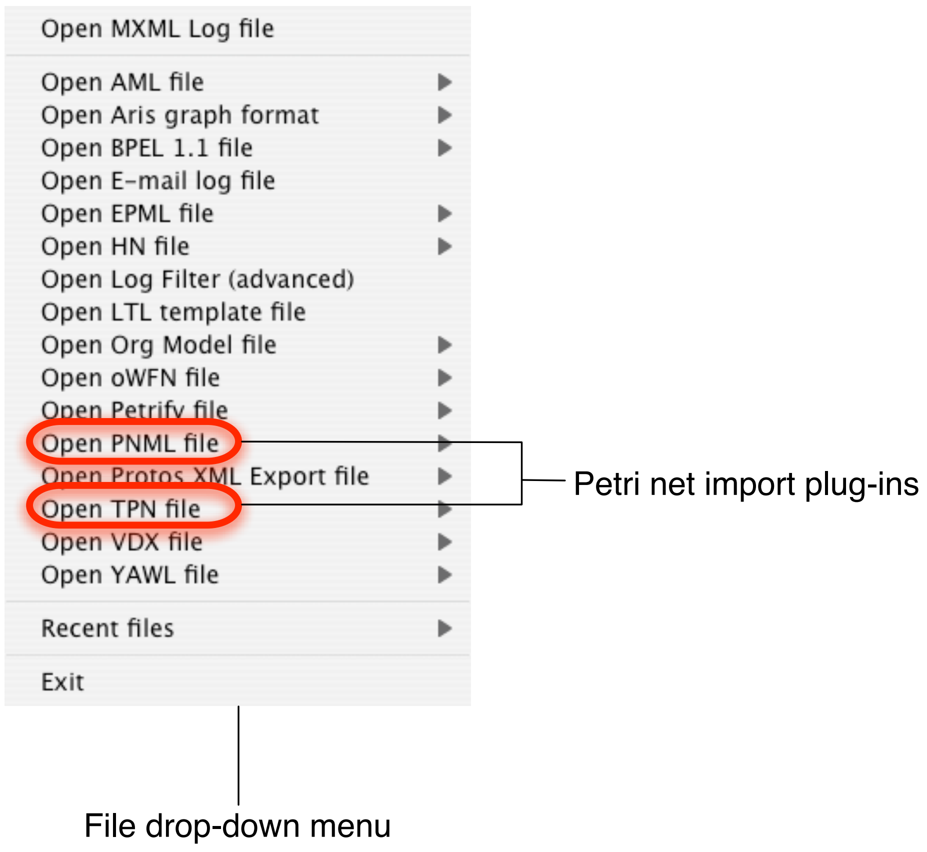
- One input place i.
- One output place o.
- All nodes are on a path from i to o.

The plug-in does *not* check for soundness properties.

There are basically two ways you can obtain a Petri net in ProM: Either you have a Petri net on disk, or you have a Petri net that you obtained through mining or conversion plug-ins.

Loading a Petri net From Disk

ProM have import plug-ins for loading a variety of Petri net formats from disk. When choosing File you should see the following drop-down menu.



As you can see there are two different ways of loading a Petri net, depending on what sort of format it is represented by. Most notable is that you may import PNML, but also import plug-ins for the more proprietary format TPN.

References

- PNML (Petri Net Markup Language) homepage: www.informatik.hu-berlin.de/top/pnml.

Obtaining a Petri Net From a Plug-in

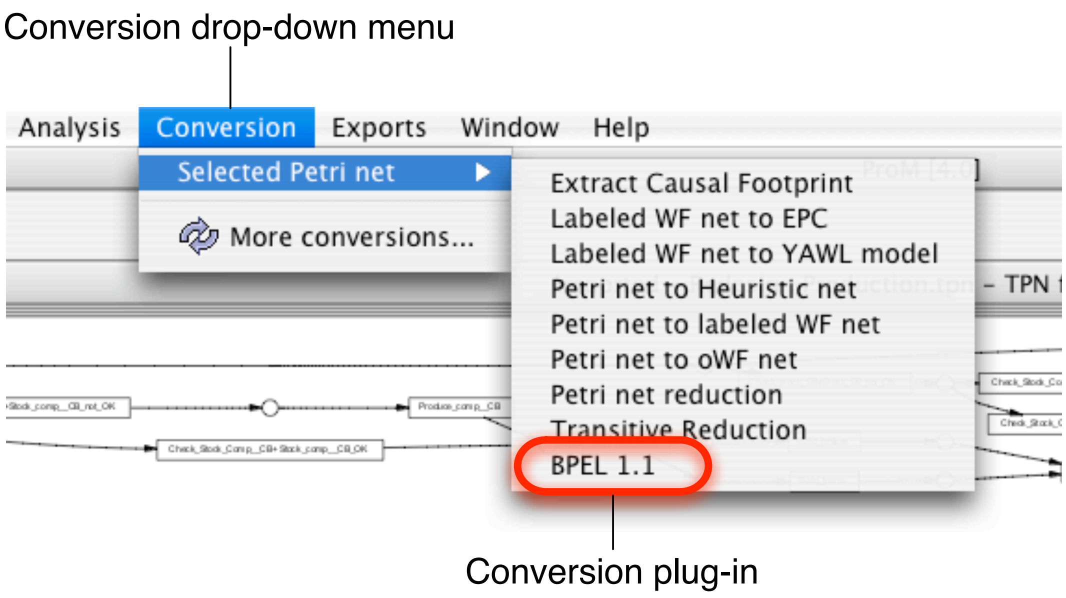
It is also possible to obtain a Petri net through various plug-ins. I will not describe each plug-in that can generate a Petri net, since there are so many ways this can be done, but simply give examples of how one would typically get a Petri net object in ProM:

- You have just mined an MXML file using, e.g., the alpha algorithm mining plug-in, or another mining plug-in that discover Petri net models.
- You have just converted an EPC to a Petri net using the EPC to Petri net conversion plug-in. Other conversion plug-ins exists that takes model languages, e.g. a region, as argument and produces a Petri net.

Using the BPEL 1.1 Conversion Plug-in

Now that you have a Petri net that you know is a Workflow net how do you convert it to BPEL 1.1? This is what I will show you now in the following.

The Workflow net to BPEL 1.1 conversion plug-in can be found in the conversion drop-down menu when the Petri net is selected; see the figure below.

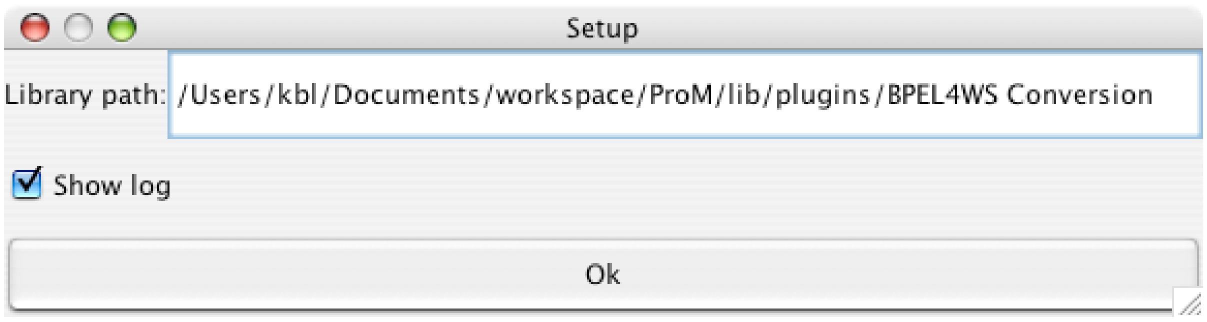


Once you have selected this you are the plug-in ask you for two pieces of information:

1. Where is the component library? Since the translation algorithm is pragmatic, in the sense it does not try to translate everything, you need to specify a place where you store manually translated components. Later on I will describe how these components are added.

2. Do you wish to see a log that describe the translation process? Since many things happen during the translation process, it may be interesting for you to see e.g. which components where chosen, or if a library component was used, to see the isomorphism. All this and more is shown in the log. I will show how a log may look like later on.

The dialog asking you for this information is shown in the figure below.



Library path: /Users/kbl/Documents/workspace/ProM/lib/plugins/BPEL4WS Conversion

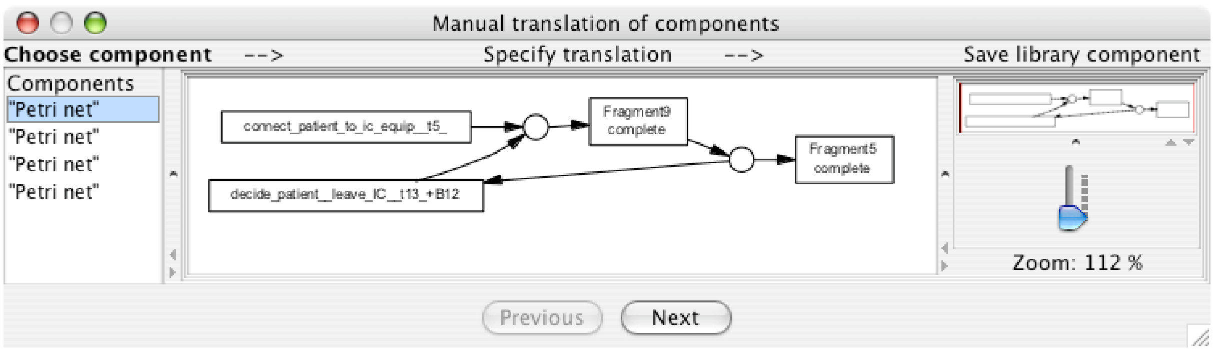
☑ Show log

The library path will initially be set to a default path library in your ProM installation but you may change to another if necessary. Please note that the two files *matching-order.xml* and *matching-order.xsd* should always be present in the directory that you point to since they are used to determine the content of the component library.

Handling Irreducible Components

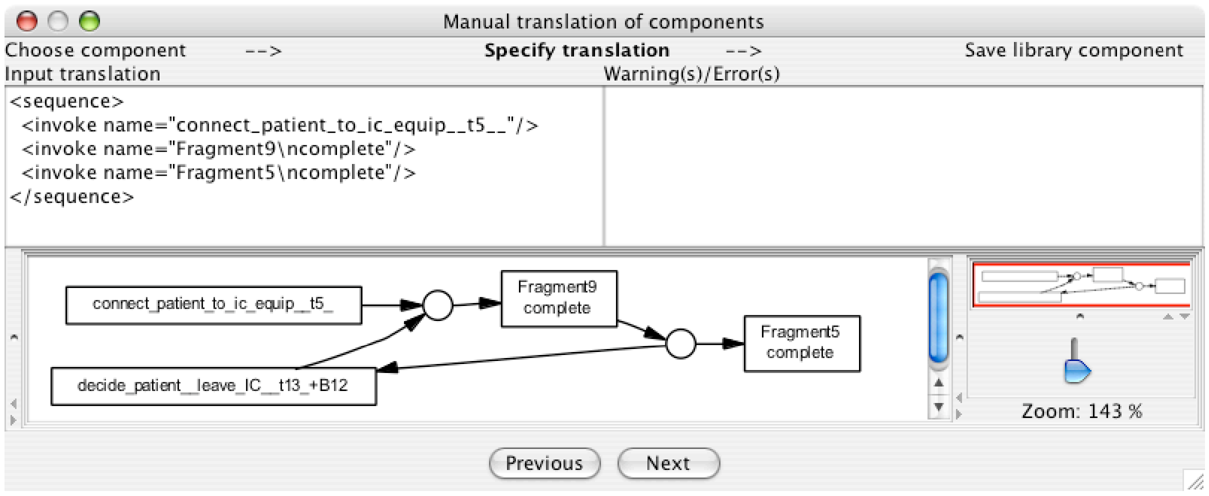
The net you are trying to convert to BPEL 1.1 may not be structured in the sense of the paper of Aalst et al. which this plug-in implements. In this situation it is necessary to guide the conversion by using the component library. If a component in the library is isomorphic with an irreducible component in the net, the conversion algorithm will use the translation of that component and continue its work. Otherwise, you, the user, need to specify a translation. In the following I will show how this is done.

When the plug-in halts because no components could be reduced it will present you with a dialog as the one shown below.



As the dialog indicate there are three steps left before the translation has been fully specified. First we need to select what irreducible component that we wish to translate by selecting one of the Petri nets in list at the left. The Petri nets are order by how many nodes they contain so the first is the net with the smallest number of nodes. After you have decided what component to translate press *Next*.

The second thing you need to do is to specify the BPEL code that the component should be translated to. This is shown in the figure below. As you can see it is up to you to give a sensible translation.



```
<sequence>
  <invoke name="connect_patient_to_ic_equip__t5__"/>
  <invoke name="Fragment9\ncomplete"/>
  <invoke name="Fragment5\ncomplete"/>
</sequence>
```

At the bottom of the dialog you can see the component that you choose. In the top left corner you specify the BPEL process in XML, and the top right show errors that are found when parsing the BPEL that you are writing. During editing of the BPEL you may see several errors that indicate that the XML is malformed and similar things, but this is because it continuously checks if the XML is ready to be used as BPEL, so you should not worry about these before you complete editing the BPEL.

The names that you choose for the different components are important in the sense that when I write the XML expression `<invoke name=”Fragment5\ncomplete”/>` it actually have a very special meaning. A BPEL element with a name that reoccur on the label of a transition, in the component that was chosen, will be substituted with the annotation of that transition. So if the transition `Fragment5\ncomplete` has, e.g., the annotation

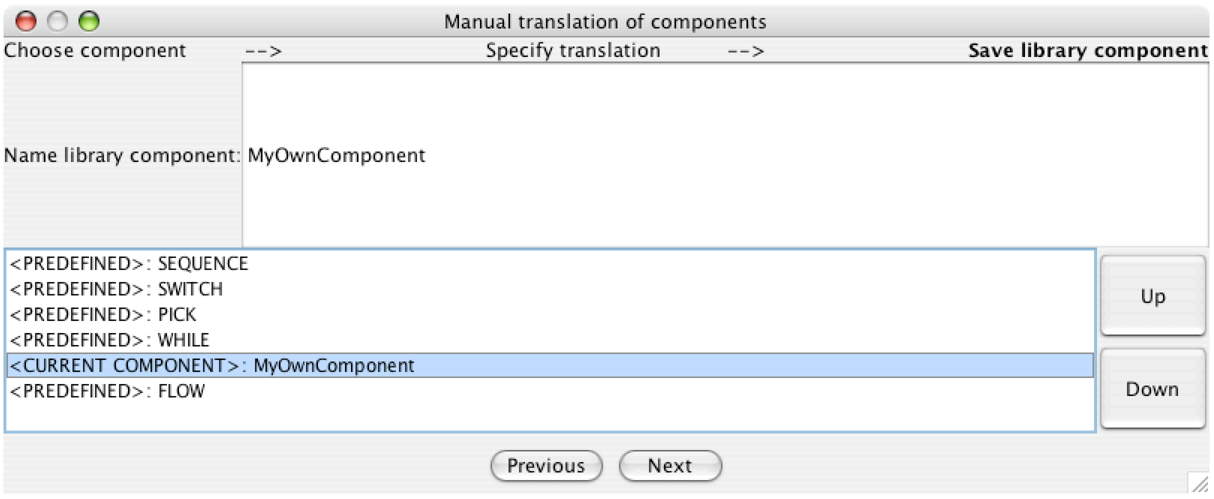
```
<while condition=”?”>
    <invoke name=”X”/>
</while>
```

then the XML expression that you specified will be replaced by the BPEL while.

From now on, every time the component you specified is used in a reduction, this information is checked in the following way: Assume that a component that is used for a reduction has a transition T. In the BPEL translation of the component in the library, T may be expressed by some BPEL activity B with the same name as T. Now let us assume that T' is the transition of the Petri net being reduced that T is isomorphic to, and it has the annotation B'. The conversion plug-in will now take B and replace every element with the same name as T, with B'.

After you completed entering the BPEL translation, press *Next*.

The third, and final step is about selecting the right name and determining what order the component should be matched. The figure below show the final part of the manual translation dialog.



Manual translation of components
Choose component --> Specify translation --> **Save library component**
Up
Down
Previous
Next

Name library component: MyOwnComponent

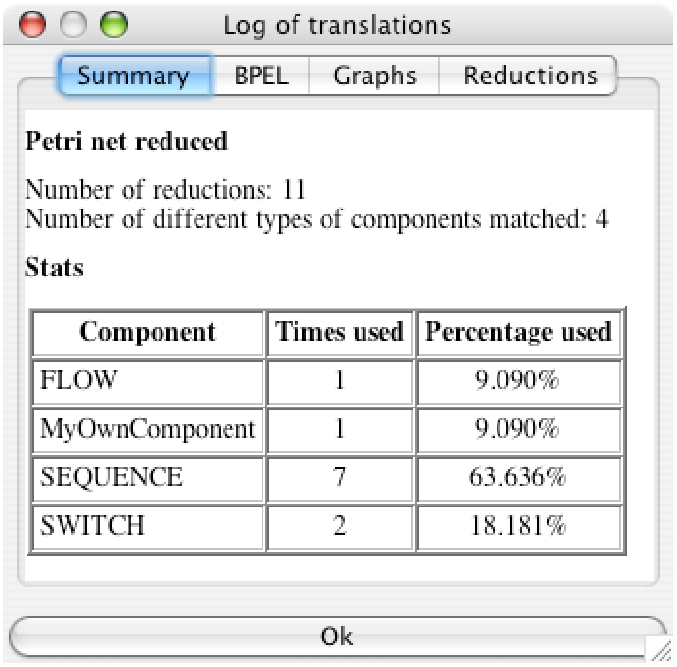
```
<PREDEFINED>: SEQUENCE
<PREDEFINED>: SWITCH
<PREDEFINED>: PICK
<PREDEFINED>: WHILE
<CURRENT COMPONENT>: MyOwnComponent
<PREDEFINED>: FLOW
```

This final part is easy. You simply type in the name for the component, and use the *Up* and *Down* buttons to tell conversion plug-in, in what order it should match components in future conversions using the component library. After doing these two things, press *Next*.

If more irreducible components are found you have to repeat the three steps just described.

Reading the Translation Log

If you choose to see the translation log, you will see a figure similar to the one shown below.



Petri net reduced

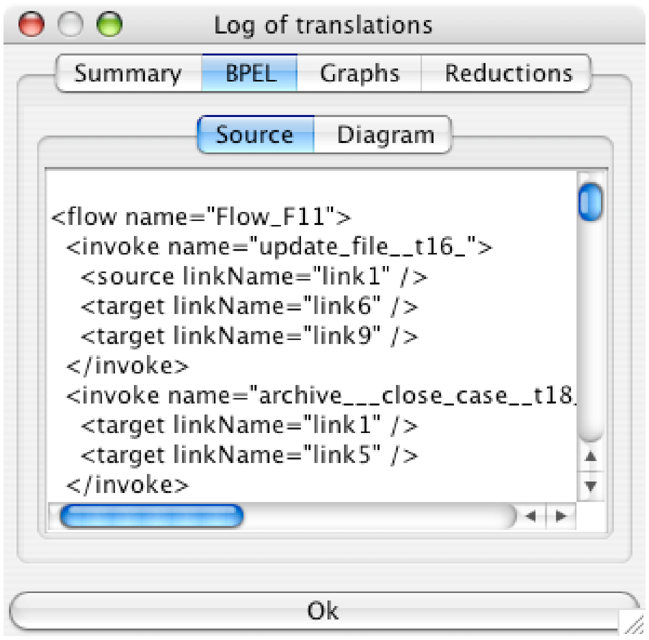
Number of reductions: 11
Number of different types of components matched: 4

Stats

Component	Times used	Percentage used
FLOW	1	9.090%
MyOwnComponent	1	9.090%
SEQUENCE	7	63.636%
SWITCH	2	18.181%

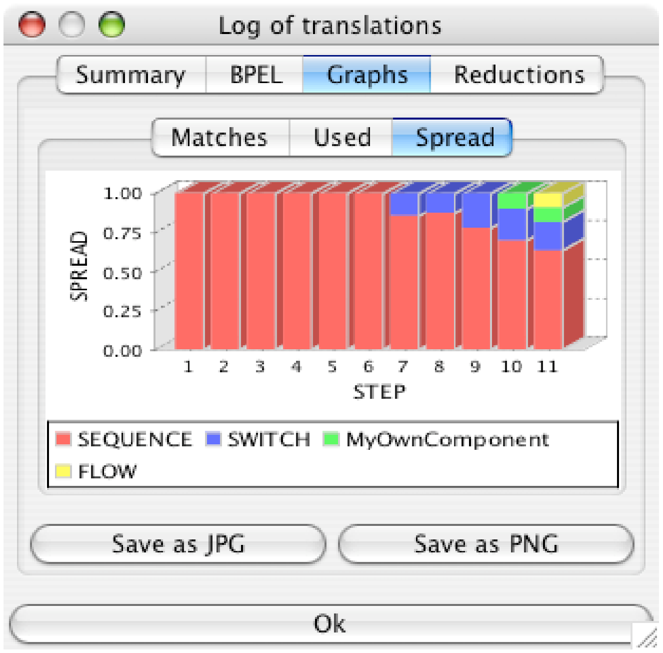
The first page is a summary of the translation process. It states wether or not the conversion was successful. In the above case, it was indeed so, and we are told that the Petri net was reduced. Other statistics are also shown, such as the component match.

In the *BPEL* tab you can see the BPEL, both as source and diagram.

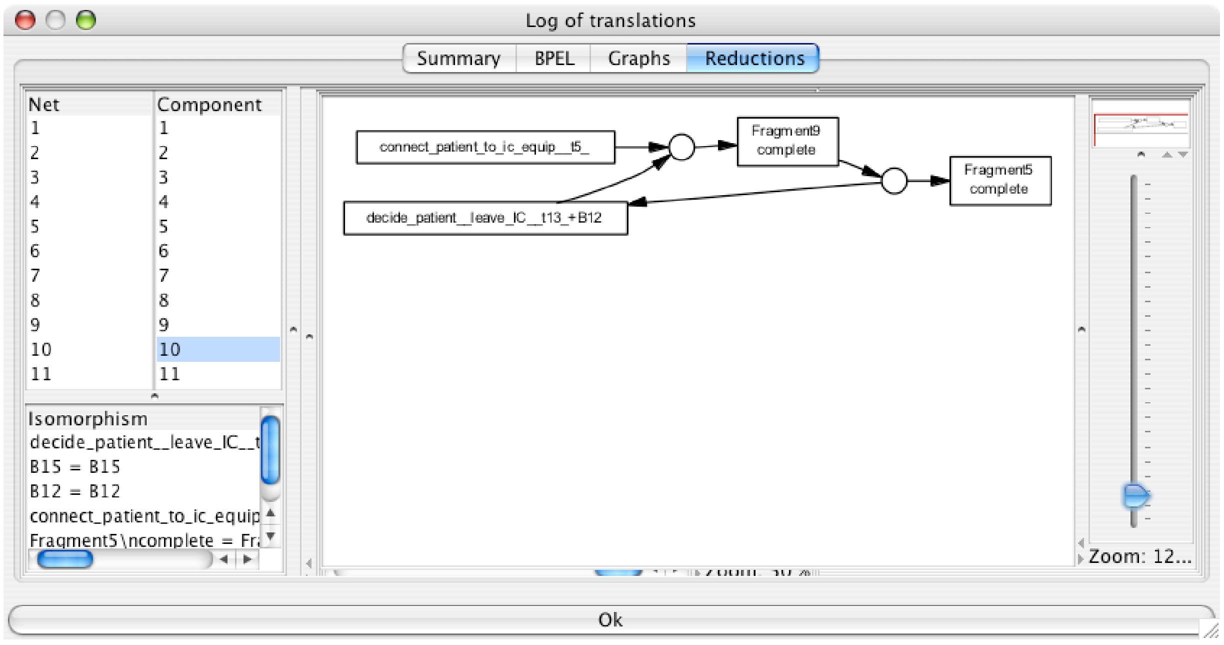


```
<flow name="Flow_F11">
 <invoke name="update_file__t16_">
  <source linkName="link1" />
  <target linkName="link6" />
  <target linkName="link9" />
 </invoke>
 <invoke name="archive___close_case__t18
  <target linkName="link1" />
  <target linkName="link5" />
 </invoke>
```

In the *Graphs* tab there are several different graphs showing different statistics on the conversion process.



In the last tab, *Reductions*, you can see how the net looks like after each reduction step and what component was matched. Notice that for library component you can also see the found isomorphism.

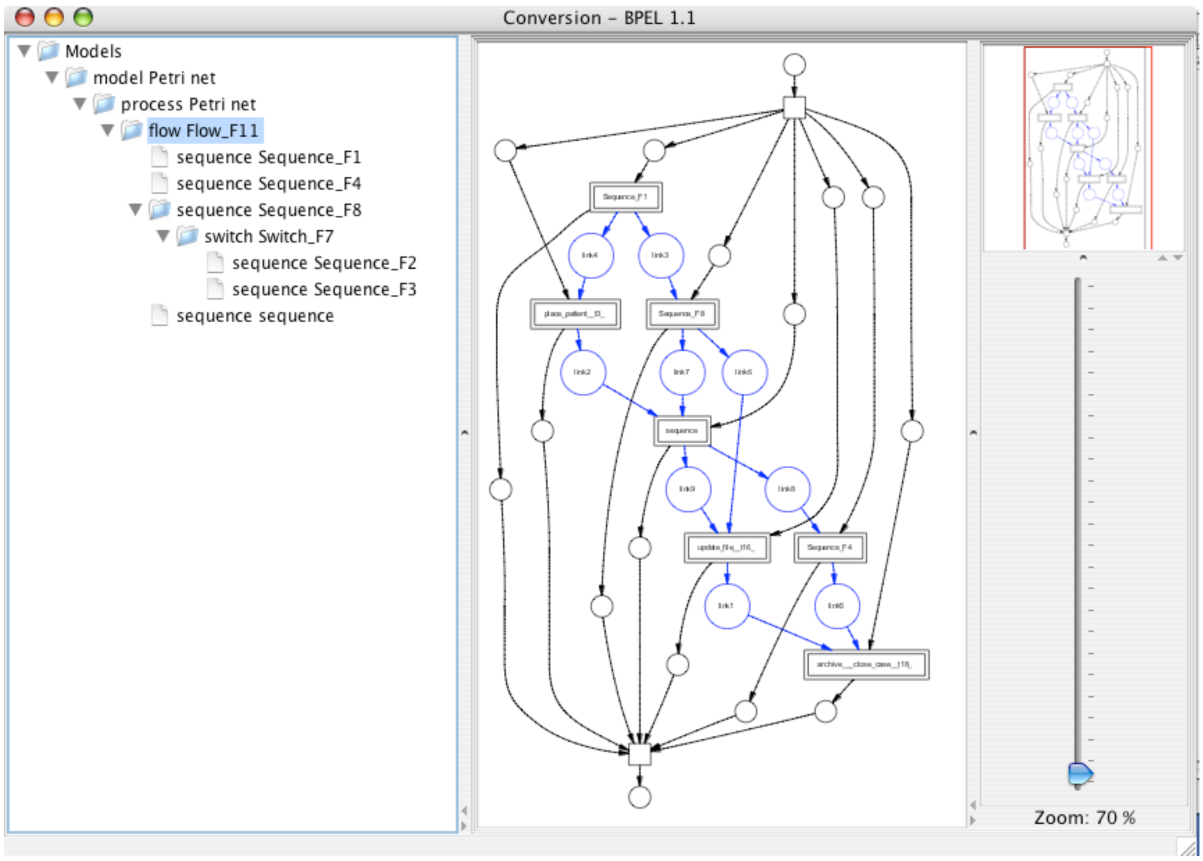


Net	Component
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11

```
Isomorphism
decide_patient__leave_IC__t
B15 = B15
B12 = B12
connect_patient_to_ic_equip
Fragment5\ncomplete = Fr
```

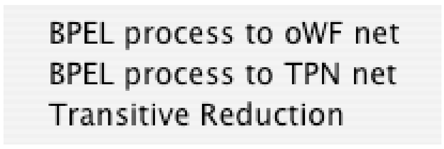
The BPEL 1.1 Result

After you successfully converted the net you should see something like the figure below.

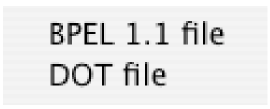


What can I do with the BPEL 1.1 model?

The BPEL 1.1 can either be converted again, using the conversion menu. Currently, it is possible to translate the BPEL to an open Workflow net, TPN, or do a transitive reduction on the process.



Another possibility is to use the export plug-ins that are available for the BPEL. There are currently two very useful plug-ins, one for exporting the BPEL to a BPEL XML file, or to export the BPEL to a DOT file that you can convert to, e.g., EPS or PDF using Graphviz.


BPEL 1.1 file
DOT file