UNIVERSITÀ DEGLI STUDI DI PADOVA

PROCESS MINING

Lab Session 2

Petri-Net Process Modelling and Verification with the WoPeD Tool

Content

Laboratory Session

WoPeD is a tool for creating Petri Nets models, running the model, and performing soundness verification. The tool is available for every modern operating system: MacOS, Windows, and Linux. It can be downloaded from https://WoPeD.dhbw-karlsruhe.de/.

WoPeD supports the *Petri Net Markup Language (PNML)*, an XML-based standard ISO/IEC 15909 to store specification files of Petri Nets. The Petri-net specification files used in this course are always given in this format.

While installing Woped, if you receive such an error message as in Figure 1, you should check whether

- you have Java 11 or higher
- you define the environment variable you should check whether you have

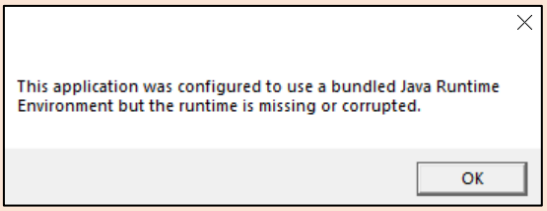

Figure 1. JRE error

- Download Java's last version from the link and install it for the first point.

https://www.oracle.com/java/technologies/downloads/

- For defining the environment variable, after installing Java, you should copy the Java folder. It may be "*C:\Program Files\Java\jdk-19*". Find "*Edit environment variables for your account*" by searching "environment variables" in your folders. Click New under the User variable for onurb (your username). Then set the variable name as *JAVA_HOME* and the variable value as the Java folder path you have already copied, as shown in Figure 2. And click *OK*.

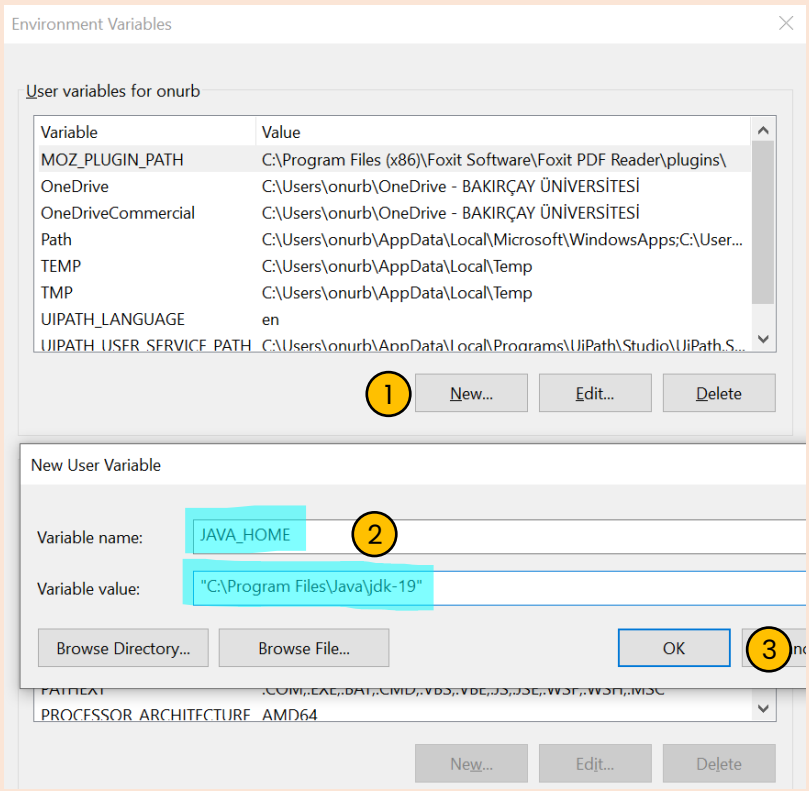

Figure 2. Defining environment variables

1. Modelling via WoPeD

Example 1: Handling of Insurance

- Model the following business process as a Petri Nets in the WoPeD Tool, specifying the initial and the final marking.
- Try to run the process modelled via the Petri net, using the "Token Game" functionality of WoPed.
- Use the functionalities in WoPeD to verify the soundness of the Petri-net model. If the model is not sound, fix the model to ensure soundness. If you have problems to find the issues that prevent soundness, you can be helped by "replaying" the model via the Token Game.

Consider the handling of insurance claims at Sunny Side Corp., Australia. Sunny Side distinguishes simple claims and complex claims. The type of the claim is determined in the first step.

For simple claims, Sunny Side carries out two steps independently: it checks the insurance policy of the insured party for validity and retrieves the statement of a local authority. When both results are available, Sunny Side checks the statement against the policy in the next step. If the result is positive, Sunny Side makes a payment to the insured party; if the result is negative, Sunny Side sends a rejection letter.

For complex claims, Sunny Side carries out three steps independently: it checks the insurance policy of the insured party for validity, retrieves the statement of a local authority, and asks for two witness statements. Again, when every outcome is available, Sunny Side checks the statements of the local authority in the next step and witnesses against the policy. If the result is positive, Sunny Side makes a payment to the insured party; if the result is negative, Sunny Side sends a rejection letter.

2. Repairing Incorrect Process Model

Example 2: Traffic Fine Management

Consider an Italian municipality's (simplified) process of handling road-traffic fines.

When a fine is issued, it is created in the system and sent to the offender. Since penalties are sent through certified emails with the confirmation receipt, the receipt is eventually received back by the municipality, which inserts the notification of the fine. If the offender has any complaints, (s)he can appeal to the judge: if the appeal is accepted, the fine is closed; otherwise, the process continues with not appealing. Note that closing the penalty is an activity of the process because it requires work such as updating the management system by the municipality. Also, note that one single appeal is possible, and the process model must enforce this.

At any moment, the offender can pay the fine, which also brings the fine to a close. If the fine is never paid, this is eventually sent to an agency for credit collection, after which the penalty is also closed.

Let us suppose a process analyst to have drawn the Petri-net process model in Figure 3 without much care for soundness verification. See file ***`FineInItaly.pnml`*** for the current model. What should be done to make the model sound? Create the sound model using WoPeD.

The model given in *Figure 3* is not soundness because:

- *Create Fine* is enabled by the token in p1 and produces one token in p3.
- *Send to Defender* consumes token p3 and produces one token in p4.
- *Insert Fine Notification* consumes one token in p4 and produces two tokens in p5 and p7.
- If either *Payment* or *Send to Credit Collection* transitions is enabled, then the token in p7 remains.

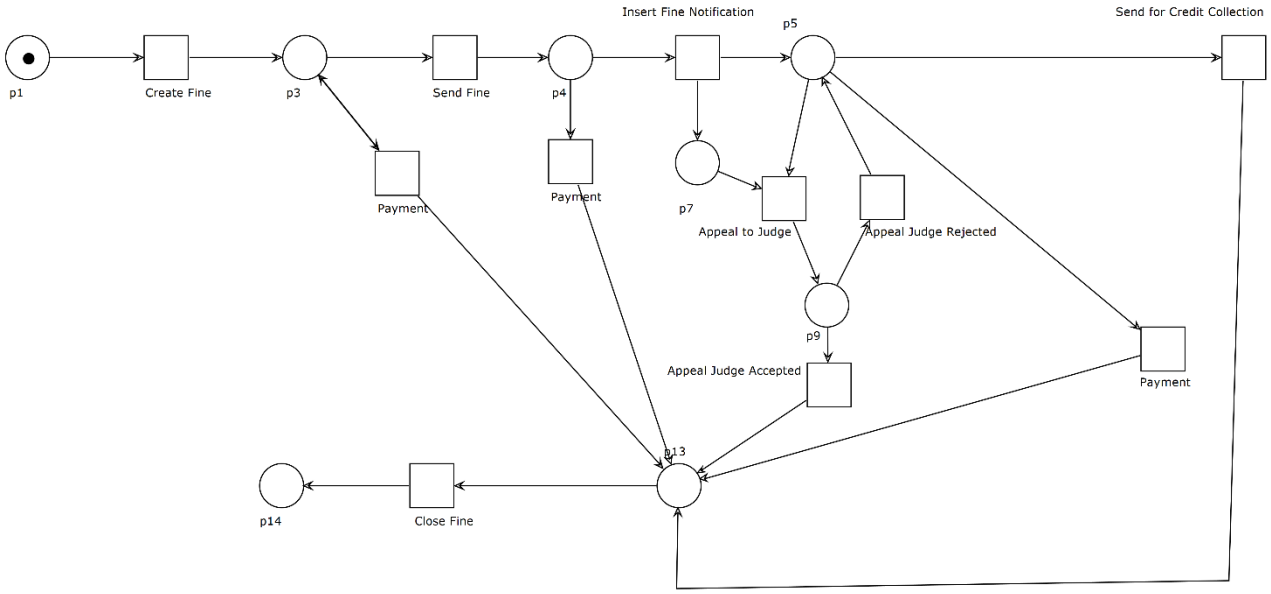

Figure 3. Petri net for the model of traffic fines management