PIPE 2.7 overview
A Petri net tool for performance modeling and evaluation

Catalina M. Lladó
Computing and Maths Department
Universitat de les Illes Balears
07071 Palma de Mallorca, Spain
cllado@uib.cat

ABSTRACT

The Petri net modeling formalism allows for the convenient graphical visualization of system models, as well as the modeling and performance analysis of complex stochastic systems. PIPE is an open source, platform independent tool for creating and analysing Petri nets including GSPNs (Generalized Stochastic Petri Nets). It is implemented entirely in Java and provides an easy-to-use graphical user interface that allows creating, saving and loading of Petri as well as its qualitative and quantitative analysis. This paper describes PIPE 2.7, its main features, including its GUI, modeling power, and analysis functionality.

Keywords

Petri Nets, Stochastic Modeling, Performance Analysis, Modeling Tools

1. INTRODUCTION

Any developer of computer or communication systems knows that the most important quality of a system is that it be functionally correct, i.e. that it exhibits certain behavioural, or *qualitative* properties. Once assured that the system behaves correctly, it is also important to ensure that the system meets certain performance-related (or *quantitative*) objectives. Indeed, while system users often take behavioural correctness for granted, it is the latter quantitative properties which often determine the success of one system over another [9].

Petri nets are a popular graphical modeling formalism that can help system designers ensure correctness and performance at design time. While they were originally developed for the study of qualitative properties of systems exhibiting concurrency and synchronization, temporal specifications were introduced about two decades ago in various forms to Petri Nets in order to also allow for quantitative analysis.

The focus of the present paper is on the Petri net tool PIPE (Platform Independent Petri Net Editor) version 2.7. PIPE is an open source, platform-independent tool for creating and analysing Generalized Stochastic Petri Nets (GSPNs). It can be used either in academia since it has a very friendly interface as well as in industry or research since it provides specific modules for model import and export from/to other tools and/or formalism. PIPE [4], is implemented entirely in Java to secure the platform independence and provides an elegant, easy-to-use graphical user interface that allows for the creation, saving, loading and analysis of Petri nets. PIPE was initially built at Imperial College London by means of different student projects. Newer versions with considerable improvements on different aspects of the tool functionality have been implemented at the Universitat de les Illes Balears [11, 10] (PIPE 2.7) and Imperial College London [5].

The rest of the paper is organized as follows: we begin on Section 2 with some background on Petri nets and a quick description of other related Petri net tools. Section 3 outlines the main features of PIPE 2.7 while Section 4 shows a case study that uses some of the new features. Finally, Section 5 reports conclusions.

2. BACKGROUND

This section gives an overview of Petri nets and describes some other highly used Petri net tools.

2.1 Petri nets

Structurally, a Petri net (PN) is a directed bipartite graph comprising *places* and *transitions* [7]. Places, drawn as circles, model conditions or objects. Inside the places are *tokens*, drawn as black dots, which represent the specific value of a condition or object. A particular arrangement of the tokens across all the places is known as a *marking* or *state*. The system begins in some initial configuration known as the *initial marking*. Transitions, drawn as rectangles, are used to describe events that may modify the system state. Directed arcs specify the relation between local states and events in two ways: they indicate the conditions under which the event can occur, as well as the local state transformations induced by the event.

The arcs of the graph are classified (with respect to transitions) as *input* arcs (arrow-headed arcs from places to transitions), *output* arcs (arrow-headed arcs from transitions to places) and *inhibitor* arcs (circle-headed arcs from places to transitions). Multiple (input, output, or inhibitor) arcs between places and transitions are permitted and annotated with a number specifying their multiplicities.

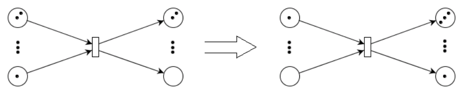


Figure 1: Firing of a transition

A transition is said to be *enabled* in a marking if each input place contains at least as many tokens as the multiplicity of the input arc and if each inhibitor place contains fewer tokens than the multiplicity of the inhibitor arc. Enabled transitions can *fire*, an atomic action which removes one token from all of its input places, and generates one token in each of its *output places*. Figure 1 shows the state of a Petri net before and after the firing of a transition. When arc weights larger than one are used, the number of tokens required in each input place for the transition enabling and the number of tokens generated in each output place by the transitions firing are determined by the weight of the arc connecting the place and the transition.

As an example, Figure 2 illustrates a PN that models two processes accessing a shared resource. The processes can be in one of three possible states: active (doing something that does not involve the shared resource), requesting the shared resource and accessing the shared resource. The shared resource can be idle or busy. The initial marking shows that in the initial state both processes are active and the shared resource is idle.

2.2 Petri nets tools for performance evaluation

Other Petri net based tools that can be found in literature are:

- GreatSPN 2.0 [2, 1, 8] (GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets): a software package for the modeling, validation, and performance evaluation of distributed systems using GSPNs and their coloured extension: Stochastic Well-formed Nets.

- TimeNet 4.0 [6, 16] (TIME Net Evaluation Tool): a tool for the modeling and analysis of stochastic Petri nets with non-exponentially distributed firing times.

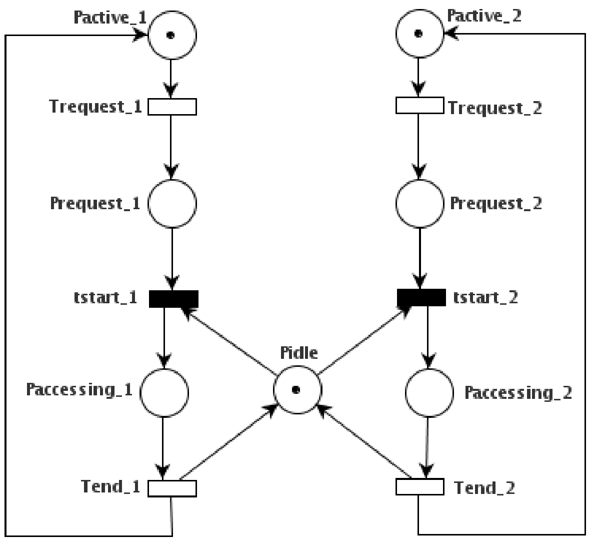


Figure 2: Shared resource GSPN model

- Sharpe [15] (Symbolic Hierarchical Automated Reliability and Performance Evaluator): a tool for specifying and analysing performance, reliability and performability models that allows for different model types. These include queueing networks, Markov and semi-Markov reward models as well stochastic Petri nets.

All these tools have the following characteristics in common: free of charge for academia, somehow maintained, Graphical User Interface (GUI), modeling of GSPNs, parameter specification, inhibitor arcs, structural analysis, GSPN steady-state analysis and simulation: giving all of them at least the subsequent performance results:

- Steady-state probability distribution of tangible states.

- Average number of tokens at each place.

- Throughput of transitions.

The main features provided by PIPE 2.7 with respect to rest of the tools are the import/export transformations and the experimenter, both described in Section 3.

Other important features that are not shared between these tools are summarized in [11].

3. PIPE 2.7

PIPE is an open source, platform independent tool for creating and analysing Petri nets including GSPNs. It is implemented entirely in Java to secure the platform independence and provides an elegant, easy-to-use graphical user interface that allows creating, saving and loading of Petri nets conforming to the PNML interchange format. PIPE also offers a full suite of analysis modules to check behavioural properties, produce performance statistics, and some less common features such as PNs comparison and classification.

PIPE began life in 2002/3 as an MSc Group Project in the Department of Computing at Imperial College London. Successive versions have been implemented and some of them grown in parallel developed by students of both universities, Imperial College London [5] and Universitat de les Illes Balears [11, 10].

In the subsequent subsections the main features of PIPE 2.7 [11, 10, 3] are described.

3.1 The graphical user interface

PIPE was designed with the objective of providing an intuitive, user-friendly tool for editing Petri nets in a easy, fast and efficient way. Anyone familiar with the standard drawing UI can pick up and use PIPE without application specific knowledge. The editor uses standard representation for the different elements that constitute a Petri net.

Following, the remarkable features of the GUI are described:

- It conforms the XML/PNML standard so it could open and work with existing PNML Petri nets. In addition, PNML annotations can also be added, so the user can include text to explain detail of the created models.
- It provides a multiple document interface so that one can work with multiple nets at the same time, each of them located on a tabbed pane.
- Users are able to perform tasks using a menu bar, a toolbar and mouse actions. Shortcuts to all editor options are also available to allow for quicker actions.
- Actions provided: cut/copy/paste and undo/redo.
- Labels are also supported.
- A zoom functionality is provided to grant the user greater flexibility and much easier use when working with large nets.

3.2 Modeling power

The following features can be specified in PIPE 2.7 and are also used in the analysis modules described later:

- Places, immediate and timed transitions, arcs and tokens
- Rate and marking parameters
- Inhibitor arcs
- Capacity restriction of places
- Immediate transition's priorities
- Server Semantics for timed transitions: Single, multiple and infinite.

3.3 Animation mode/token game

PIPE offers an animator so that the user can manually experiment with the token game, firing any of the enabled transitions at each state. The set of enabled transitions is highlighted and the user chooses which one must be fired. Animation history is recorded, i.e. all the fired transitions can be seen on the side of the screen, so from the current state the animation can be stepped forwards or backwards. The automatic execution of a random transition in animation mode is also possible; for this the user specifies the firing delay and the number of firings.

3.4 Analysis modules

PIPE offers a set of modules to carry out different types of qualitative and quantitative analysis. These set of available modules can easily be increased, provided that the defined interface is followed. The available modules in PIPE are the following:

- Structural analysis. It has different modules to compute:
  - Classification: based on the connectivity between places and transitions, this module classifies a Petri net into one or more of the following types: State Machine, Marked Graph, FC-Nets, EFC-Nets, SPL-Nets, ESPL Nets.
  - Comparison: this module compares two Petri nets based on their PNML files. It confirms whether they are (functionally) the same; otherwise, it works out the differences between them.
  - Incidence and marking: this module shows the incidence matrix and the marking matrix for the given Petri net.
  - Minimal siphons and minimal traps
  - Place and transition invariant analysis
  - Reachability/coverability graph
- GSPN analysis. This module calculates the following performance indexes by exploring the state space of the given Petri net and determining the steady state solution of the model:
  - Average number of tokens
  - Utilization of places
  - Throughput of timed transitions
  - Token probability density
- Simulation: by means of simulation, this module computes the average number of tokens per place along with the 95% confidence interval for each place in the net.
- State space analysis: this module builds a tree of all the reachable markings which is used to determine the qualitative properties of the given Petri net: boundedness, deadlock-free, and safeness. It also provides the shortest path to deadlock in the case that there exists one.

Figure 3 shows the GUI for PIPE 2.7 with the analysis modules on the left side PIPE 2.7.

3.5 Inport/export and transformations

Nets can be printed or exported into two graphical formats: Postscript and PNG. Moreover, as new features for the latest versions of PIPE the following inter-operability functions are also available:

- Nets can also be imported/exported from/to TimeNet [16] format.
- PMIF2 [12] models can also be imported using Model-to-model (M2M) transformations [10].

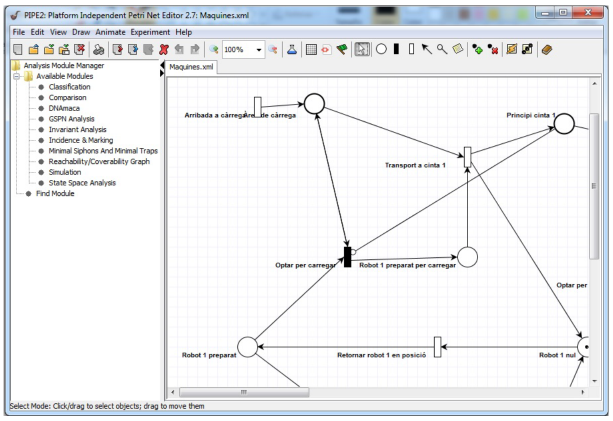


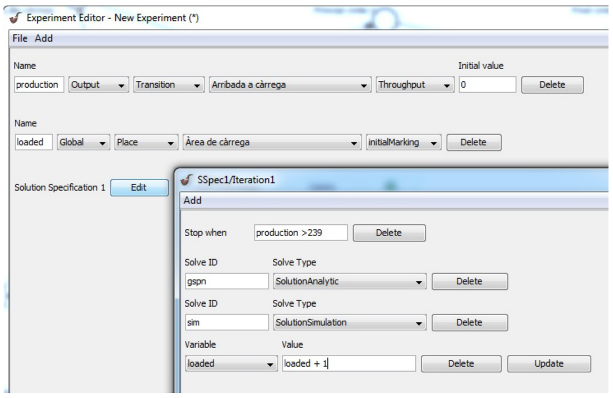
Figure 3: PIPE 2.7 GUI with the analysis modules on the left side

3.6 Experimenter

PIPE 2.7 also has an experimenter that complains with the Experiment Schema Extension specification (Ex-SE) [13] which defines a set of model runs and the output desired from them providing a way of specifying performance studies that is independent of a given tool paradigm. It allows for:

- Experiment editing, validation and execution
- Results on xml format and Excel

Figure 4 shows the experimenter interface for PIPE 2.7.


Experiment Editor - New Experiment (*)
File Add

Name					Initial value	
production	Output	Transition	Arribada a càrrega	Throughput	0	Delete

Name					
loaded	Global	Place	Àrea de càrrega	initialMarking	Delete

Solution Specification 1 Edit

SSpec1/Iteration1
Add

Stop when production >239 Delete

Solve ID	Solve Type	
gspn	SolutionAnalytic	Delete

Solve ID	Solve Type	
sim	SolutionSimulation	Delete

Variable	Value		
loaded	loaded + 1	Delete	Update

Figure 4: PIPE 2.7 Experimenter interface

4. CASE STUDY

To illustrate the distinguished capabilities of PIPE 2.7, we present a case study that uses the import transformation from a QN model specified using PMIF [12]. We use the Oracle example described in [14], with 3 servers (CPU, UserThink and Delay). Its PMIF model is imported into PIPE2 and the PN seen for this example in as shown in Figure 4, only with some transitions and places moved a little bit so the drawing is clearer (the transformation output leaves some arcs crossed). Performance indexes obtained for this example are exactly as shown in [14], so it demonstrates the correct transformation of our tool.

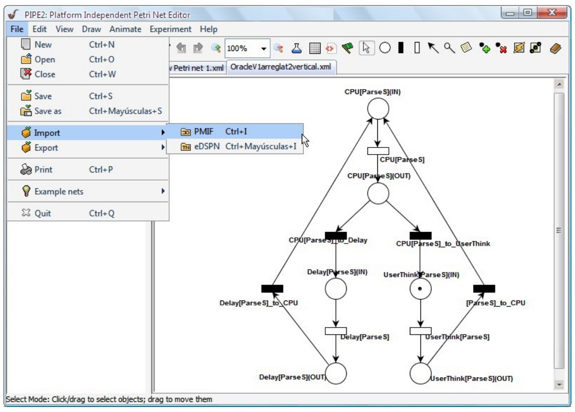


Figure 5: Oracle case study

5. CONCLUSIONS

Petri nets are a popular graphical modeling formalism that can help system designers ensure correctness and performance at design time. Petri nets theory has been widely used to implement a variety of modeling and evaluation tools. In this paper we have focus on PIPE open source Petri net editor, version 2.7. We have also described PIPE 2.7 main features, including its GUI, modeling power, and analysis functionality.

We are now working on making the tool available through a Github license, though currently PIPE 2.7 can be found on [3]. Some more detailed future work, related for example to the PMIF import could be improving the automatic drawing of the nets (as mentioned in the previous section, it is not yet right) as well as the import of more complex QN models, for example, allowing for different scheduling policies.

6. REFERENCES

[1] The GreatSPN Framework version 3.0. `https://github.com/greatspn/`.

[2] GreatSPN GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets. `http://www.di.unito.it/~greatspn/`.

[3] Pipe 2.7 petri net modelling tool: Pipe 2.7. `http://mifs.uib.cat/tools/`.

[4] Platform Independent Petri net Editor 2. `http://pipe2.sourceforge.net/`.

[5] Platform Independent Petri net Editor 5. `https://sarahtattersall.github.io/PIPE/`.

[6] TimeNET TIMEd Net Evaluation Tool. `https://timenet.tu-ilmenau.de/#/`.

[7] M Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling With Generalized Stochastic Petri Nets*. Wiley, 1995.

[8] Souheib Baarir, Marco Beccuti, Davide Cerotti, Massimiliano De Pierro, Susanna Donatelli, and

Giuliana Franceschinis. The greatspn tool: recent enhancements. *SIGMETRICS Perform. Evaluation Rev.*, 36(4):4–9, 2009.

[9] F. Bause and P.S. Kritzinger. *Stochastic Petri Nets*. Vieweg, 2nd edition, 2002.

[10] P. Bonet and C.M. Lladó. Importing pmif models into pipe2 using m2m transformation. In *Proc. of the 1st Joint WOSP/SIPEW International Conference on Performance Engineering (ICPE)*, 2012.

[11] P. Bonet, C.M. Lladó, R. Puijaner, and W.J. Knottenbelt. Pipe v2.5: A petri net tool for performance modelling. In *Proc. 23rd Latin American Conference on Informatics (CLEI 2007)*, October 2007.

[12] C. U. Smith, C. M. Lladó, and R. Puigjaner. Performance Model Interchange Format (PMIF 2): A comprehensive approach to queueing network model interoperability. *Performance Evaluation*, 67(7):548 – 568, 2010.

[13] C. U. Smith, C. M. Lladó, and R. Puigjaner. Model interchange format specifications for experiments, output and results. *The Computer Journal*, 2011.

[14] C.U. Smith and C. Milsap. Software performance engineering for oracle applications: Measurements and models. In *Proc. Computer Measurement Group, Las Vegas, NV*, December 2008.

[15] K.S. Trivedi and C. Hirel. SHARPE: Symbolic Hierarchical Automated Reliability and Performance Evaluator. `https://sharpe.pratt.duke.edu/`.

[16] A. Zimmermann, M. Knoke, A. Huck, and G. Hommel. Towards Version 4.0 of TimeNET. In *13th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems, MMB 2006*, pages 477–480, March 2006.