Association for Information Systems
AIS Electronic Library (AISeL)

PACIS 2007 Proceedings

Pacific Asia Conference on Information Systems (PACIS)

2007

Design and Application of Templates in GPenSIM

e Davidrajuh
University of Stavanger, reggie.davidrajuh@uis.no

69. Design and Application of Templates in GPenSIM

Reggie Davidrajuh
Department of Electrical and Computer Engineering
University of Stavanger
reggie.davidrajuh@uis.no

Abstract
This paper talks about design and application of templates in a new simulation tool called General Purpose Petri Net Simulator (GPenSIM). Firstly, GPenSIM is introduced; GPenSIM is designed for simulation of discrete event systems, e.g. information systems. Secondly, the design and application of templates in GPenSIM is stated. Finally, an application example on the use of templates is given.

Keywords: Information system, modeling and simulation, templates, GPenSIM

Introduction

The problem being addressed in this paper is the design and application of templates in the simulation tool called General Purpose Petri Net Simulator (GPenSIM). This paper describes the role of templates is facilitating modeling, simulation, and performance evaluation of information systems. Though Petri net is the basis of GPenSIM, for brevity, basic details about Petri net are not given in this paper. Interested readers are referred to any standard textbooks on Petri net, e.g. Cassandras and LaFortune (1999).

This paper is structured as follows: Section 2 introduces GPenSIM. Section 3 presents the design of templates in GPenSIM. Section 4 presents an application example on simulation of information systems, explaining the role of templates.

GPenSIM

In simulation of information systems, diverse software tools are used to speed up the creation and simulation of computational models. There are mainly two types of tools: visual simulation tools (e.g. Simulink (MathWorks, 2007)), and modeling languages (e.g. SIMSCRIPT (2007)). The tool that is useful for simulation of information systems should possess the following criteria:

1. Modeling discrete event systems: Since information systems are discrete event based, the tool should be capable of modeling discrete connection in information systems.
2. Industry Standard: The tool must run on an industry standard platform, supported by a large number of vendors so that the models created by the tool has a longer lifespan and has the potentials of further extensions in the future.
3. Access to mathematical functions and library: Obviously, a software tool cannot be complete, in the sense it provides all the mathematical functions that are needed for modeling and simulation of any systems. Thus, the tool must be capable of accessing additional libraries and functions to do complex mathematical analysis.
4. Programming language and operating system neutral: A huge diversity of programming languages and operating systems is a barrier to extensive and cross-platform use of the simulation software. Thus, the tool should be based on a language and operating system neutral standard, meaning it should accept inputs and create outputs in a standard interchange format, such as XML, and be able to on run any operating system

GPenSIM satisfies all the 4 criteria stated above (GPenSIM, 2007). GPenSIM runs on MATLAB (MathWorks, 2007) platform. It is designed to accept source files for simulation programmed in both XML language and in GPenSIM language; this flexibility allows the designer to concentrate on the modeling and simulation aspects, rather than on the programming details.

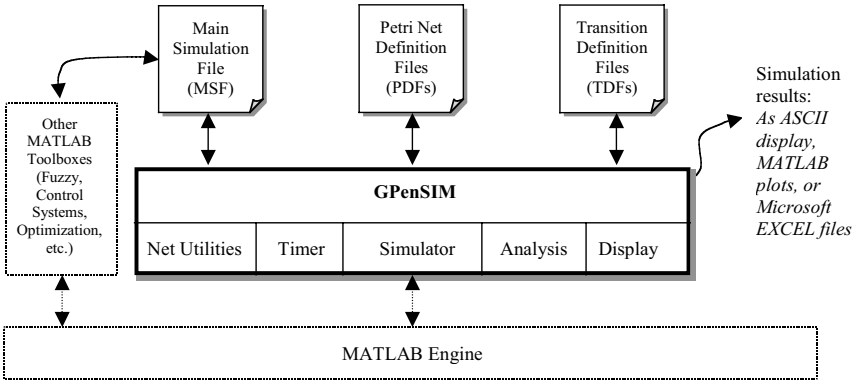

Figure 1: The architecture of GPenSIM

Architecture of GPenSIM

The methodology for simulation with GPenSIM consists of three steps; all the three steps are for creating a Petri net model of the system. The three steps are:

Petri net Definition Files (PDF): There could be many Petri net models, as the system could be divided into many subsystems and each of these subsystems are modeled as separate Petri net models. Each Petri net model is coded in a separate file (PDF).

Main Simulation File (MSF): Since there could be many Petri net models, the dynamics of each Petri net are gathered together and put in a single file.

Transition Definition files (TDF): Finally, there can be a number of transition definition files, each of them describing the user-defined conditions attached to the transitions.

Figure 1 shows the architecture. There are three types of source files - a single main MSF, one or more PDF, and zero or more TDF - are simulated by GPenSIM engine. GPenSIM engine has a number of modules such as Net utilities module for connecting different subsystems together etc., Timer module (stochastic timer) for time triggering activities, Analysis module for coverability tree analysis, Display module for displaying results of simulation intelligibly, and Simulation module that is responsible for the simulation runs.

Programming the Source Files

There are two approaches to code the source files (figure 2): 1) Using GPenSIM/MATLAB language. 2) Using XML based Petri Net Markup Language (PNML, 2007).

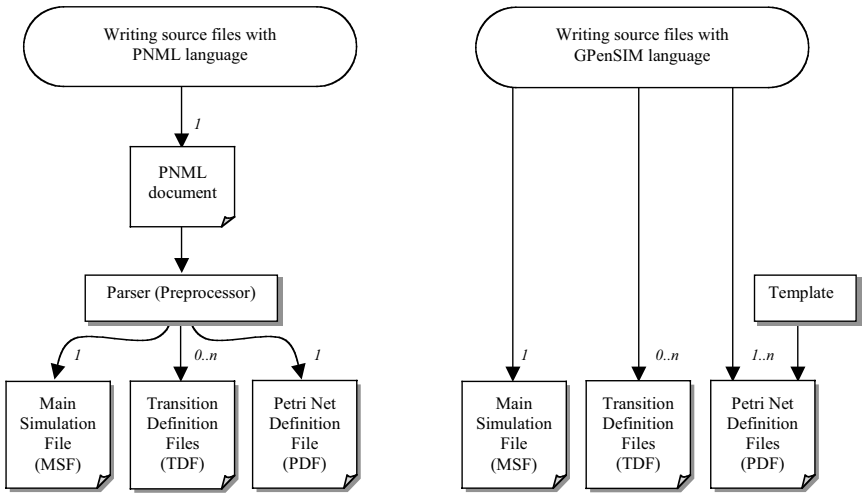

Figure 2: Different approaches for creating the source files for simulation

By the first approach, all the source files (MSF, PDFs, TDFs) are coded in GPenSIM language. By the second approach, the *overall* Petri net model of a system is described in a single source file is called is PNML document, using PNML language; PNML document is then fed to GPenSIM parser, which will convert the PNML document into a set of source files (one MSF, one PDF, and zero or more TDF). The current version of GPenSIM does not allow the use of PNML to model subsystems as separate Petri nets, meaning there will be only one PDF generated by the parser.

Template

As shown in figure 2, a number of Petri net definition files are to be created, each representing a subsystem of the information system. In order to be a useful platform for simulation of information systems, GPenSIM provides a collection of patterns; these patterns are Petri net models that can be used as models of subsystems of an information system, e.g. generator, decision block, etc. Figure 3 shows a group of patterns; GPenSIM manual presents the complete list of patterns.

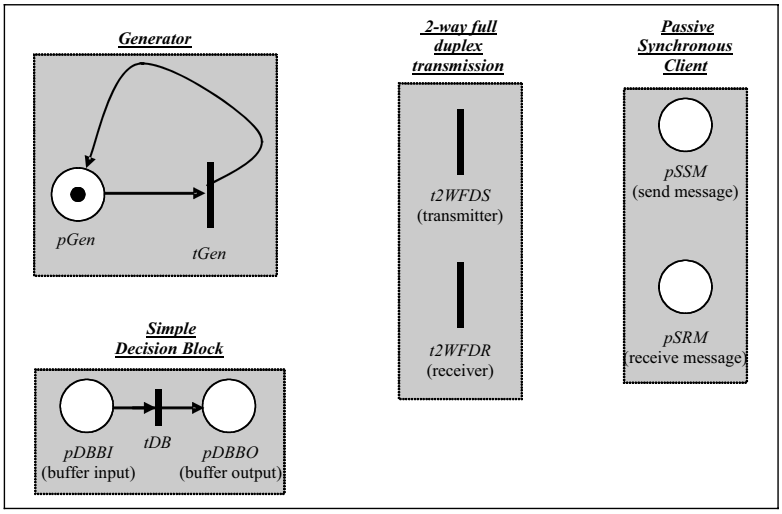

Figure 3: A group of patterns

Using Template

During model building, a designer divides an information system into identifiable subsystems. Then rather than creating (reinventing) individual source files (PDFs) for these subsystems, he checks the template for the correct patterns and simply uses these patterns as the models of the subsystems. By using the modular model building approach, model builders can quickly build models of large systems by simply selecting the suitable patterns for subsystems and then connecting the patterns together forming the complete systems.

Template Design

The functionality of the template is to process an input of text strings (known as 'concept') and to generate an output ('pattern') that represents some information; see figure 4. Hence, the problem of template design is an instance of the problem of knowledge representation (Hobbs and Israel, 1994). Thus, the following issues that are related to knowledge representation are to be considered for template design:

1. What are the elements in the knowledge representation (pattern)?
2. How are the elements related to each other?
3. What are the properties of the elements?
4. How the properties change?
5. What is the granularity of the knowledge representation instances?
6. What is the level of abstraction of the knowledge representation instances?

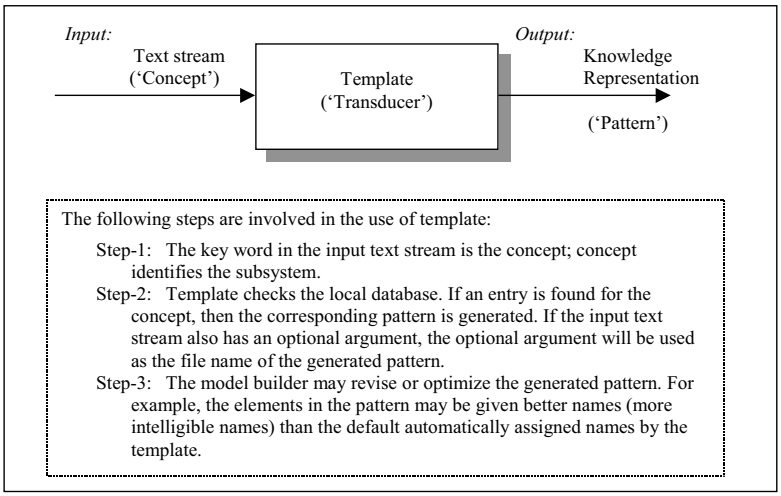

Figure 4: Template in action

Fortunately, it is easy to choose the parameters for the first four questions, because of the use of Petri net as the representational model for the domain of modeling applications:

1. The basic elements in the pattern are places and transitions
2. The basic elements are connected by bipartite arcs.
3. Some of the basic properties of the elements are: 'tokens' for places and firing times for transitions.
4. These properties do not change in the pattern that represents the static Petri net graph. Thus, the initial values of the properties should be placed somewhere else. During simulations, the properties may change.

The issue about the granularity is a matter of time available for developing the collection patterns. Initially, only a small set of patterns representing some obvious subsystems of information systems such as some types of servers (Web server, database server), some types of clients (passive client, synchronous client, asynchronous client), connections (e.g. the Internet transmission), repeaters, decision blocks, etc. are planned. Final issue is the level of abstraction or the level of complexity of the generated patterns. Currently, the generated patterns have higher level of abstraction meaning as they present less internal details of the subsystems; the generated patterns have only a few elements (places and transitions), and the relations between the elements are represented by minimal sets of arcs.

4. Application Example

Figure 5 shows a simple application example. Due to brevity, the example is simplified. In this example, a Web (synchronous) client requests a service for a Web server. The Web server, which is implemented as a Web service, further requests data from a database.

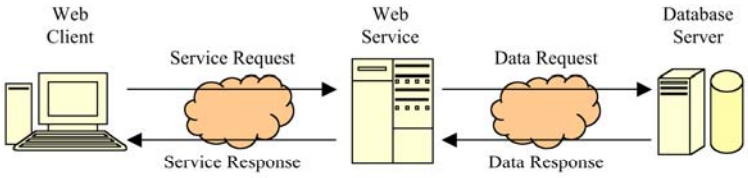

Figure 5: A simple information systems

The Petri net model of the example is shown in figure 6. The Petri net model has 5 subsystems; each subsystem can be modeled as a separate Petri net. For brevity, the main simulation file (MSF) given below, shows only the code relevant for generation of patterns for the subsystems; complete code is given (TemplateExample, 2007):

```
% generate pattern for client in PDF  'Client_def.m'
template('PassiveClient', 'Client');
% generate pattern for Internet transmission between client/WebServer in PDF  'Internet_1_def.m'
template('FullDuplexTransmission','Internet_1');
% generate pattern for Web server in PDF  'WS_def.m'
template('WebService','WS');
% generate pattern for Internet transmission between WebServer/Dbase in PDF  'Internet_2_def.m'
template('FullDuplexTransmission','Internet_2');
% generate pattern for Database server in PDF  'DBServer_def.m'
template('DBServer','DBServer');
% now build the complete model by connecting the subsystems
% the interconnection between the subsystems is defined in PDF 'Connect_def.m'
PN = build('Client_def', 'Internet_1_def', 'WS_def', 'Internet_2_def', 'DBServer_def', 'Connect_def');
% the rest of the MSF is about assigning the initial dynamics, simulation of the systems, and printing the results
```

Conclusion

This paper presents a new tool for simulation of information systems called General-purpose Petri net Simulator (GPenSIM). The architecture of GPenSIM reveals its most favorable characteristics for simulation of information systems:

1. Extensible: In additions to the GPenSIM functions, model builders can make use of hundreds of diverse functions (Statistics, Control systems, Fuzzy logic, database, etc.) available in numerous tool boxes in the MATLAB environment. For example, by combining functions from GPenSIM and Fuzzy logic toolbox, model builders can experiment with Fuzzy Petri net models.
2. Modular model building: This work emphasizes the use of patterns in GPenSIM as patterns enable painless and faster model building.
3. Language and operating systems neutral: To use GPenSIM, model builders need not learn GPenSIM or MATLAB language, if they are familiar with XML. Use of GPenSIM is platform-independent, as MATLAB is available for all the major operating systems.

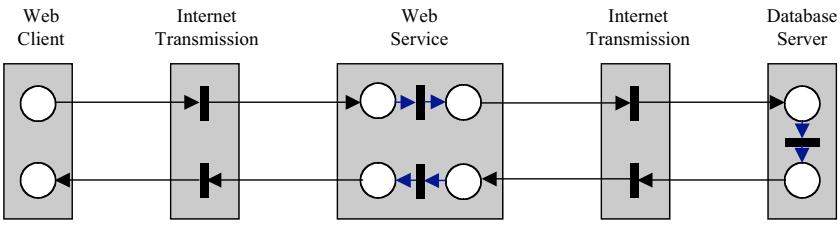

Figure 6: The Petri net model for simulation

Further work: we point out just two topics. 1) The available prototypical system only has a minimal set of patterns as knowledge representations of obvious subsystems of information systems such as Web server, Web client, etc. The apparent further work is to categorize the information systems and subsystems and to create patterns for these. Further, the level of abstraction of the generated patterns may also be lowered as to represent more details of the subsystems; currently, the patterns are at the highest level – they only have a few elements (places and transitions) and relations. 2) GPenSIM accepts model descriptions (inputs) in XML (PNML) language. Obvious further work is to output simulations results also as XML documents so that they can be further processed by any other software.

References

Cassandras, G. and LaFortune, S. Introduction to Discrete Event Systems. Kluwer Academic Publications, Hague, 1999

Hobbs, J. and Israel, D. "Principles of Template Design", in ARPA Workshop on Human Language Technology, M. Kaufmann, March 1994, pp 172-176

GPenSIM (2007) Available: http://ilab16.ilab.ux.his.no/GPenSIM/

MathWorks (2007) Available: http://www.mathworks.com

PNML (2007) Available: http://www2.informatik.hu-berlin.de/top/pnml/

SIMSCRIPT (2007) Available: http://www.simscript.com/

TemplateExample (2007) Available: http://ilab16.ilab.ux.his.no/GPenSIM/TemplateExample