An Approach to Web-Oriented Discrete Event Simulation Modeling

Ewa Ochmańska

Warsaw University of Technology, Faculty of Transport
00-662 Warsaw, Poland
och@it.pw.edu.pl

Abstract. The paper describes a methodology for creating simulation models of discrete event systems and for executing them on the Web platform. Models defined as extended Petri nets are built following schemas describing their available elements and admissible structures. Simulation portal provides access to Java class libraries of model elements, to XML documents of process schemas and defined models, as well as to several functions concerning model definition, execution of simulation jobs, analysis and visualization of results.

1 Introduction

Some of recent efforts concentrated on sharing computational resources concern Grid environments, providing middleware platform to organize transparent controlled use of advanced computing resources and cooperative Web-based technologies [1, 2].

This paper presents an approach to construct a frame for cooperative Web-based DES environment, founded on a particular method for defining, building and executing simulation models based on extended Petri nets. The method, implemented in simulation modeling of transport processes [3,4], comprises data-driven construction of object-oriented models of semantic classes of net elements, following predefined schemas. Such modeling approach can be implemented in the collaborative simulation portal based on Java / XML and Grid technologies, giving common access to the program and data resources and permitting to exploit and develop them collectively.

2 Modeling Principles and Implementation of Simulation Models

Model of a process is represented by Petri net: a bi-graph with two disjoint subsets of nodes, transitions and places; places are passive containers for tokens; dynamic transitions change net states by consuming tokens from input places and producing them in output places, as defined in [5], according to so-called enabling rule.

Simulation of a process represented by Petri net consists in changing its states by dynamic behavior of transitions. Various extensions, in particular concerning timing and control rules [6], were proposed to increase semantic expressiveness of the Petri net formalism. Some of them have been adopted in the presented modeling approach:

- Data structures assigned to tokens describe processing or processed entities.
- Timestamps assigned to tokens record a time of entity appearance or creation.

- Contents of all tokens, including timestamps, describe current state of a process.
- Predicates, defined on values of input tokens, extend enabling rules of transitions.
- Transitions perform actions transforming values of tokens consumed on input to values of output tokens, timestamps comprised, resulting in new process states.
- The model has the dual structure shown on Fig. 1, which comprises a bi-graph of Petri net along with a list of transitions forming a queue of planned events.

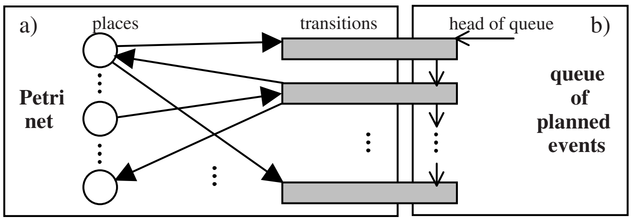

Fig. 1. Dual structure of a model: a) bi-graph of places and transitions, b) list of transitions

Process schemas define families of models for various categories of processes, with particular structure and semantics, by specifying following construction rules:

- a set of semantic place classes of with proper classes of tokens (data structures)
- a set of semantic transition classes with proper of input and output place classes
- partial ordering of transition subclasses according to the processing flow.

Java classes of model components are implemented in category-specific libraries as sub-classes of four base classes: token, place, transition and process equipped with following attributes and methods, resumed in Table 1. Program items marked by *(o)* in the table are overridden (redefined) in semantic subclasses of a process category.

Table 1. Attributes and methods of base object classes of an extended Petri net model

Java class	Attributes	Methods
token	time TimeStamp object DataStructure *(o)* token NextToken	void RemoveFromPlace () void InsertTo (Place P)
place	token TokenList *//semantic place subclasses differ just by the classes of contained tokens* transition[] OnInput, OnOutput	
transition	token[][] CandidateTuples place[] Input, Output transition NextTransition	token[][] EnablingPredicate () *(o)* void Action () *(o)* time TimeFunction ()
process	transition FirstTransition time CurrentTime, TimeLimit	transition[][] OperativePredicate () transition[] DecisivePredicate (transition[][] Sets) *(o)* void ExecuteSimulation (), void SimulationStep ()

Simulation program is in fact an instance of a category-specific subclass of process, say myProcess. It calls ExecuteSimulation method to execute a loop of simulation steps:

```
while (myProcess.FirstTransition.TimeFunction < TimeLimit) {
  myProcess.CurrentTime = myProcess.FirstTransition.TimeFunction;
  SimulationStep ();
}
```

During a simulation step, EnablingPredicate computes CandidateTuples of tokens for each transition enabled at CurrentTime. OperativePredicate returns a decision space for current state of simulation i.e. alternate non-conflict subsets of enabled transitions with proper CandidateTuples. Category-specific DecisivePredicate chooses one of these subsets. Action transforms the chosen candidate tuple of tokens from Input places to a new tuple of tokens in Output places for each of the chosen ActivatedTransitions:

```
void SimulationStep () {
  transition T = process.FirstTransition;
  while (T.TimeFunction == CurrentTime) {
    T.CandidateTuples = T.EnablingPredicate;
    T = T.NextTransition; }
  int j;
  transition[ ] ActivatedTransitions = new(DecisivePredicate (OperativePredicate ()));
  for (j=0; j < ActivatedTransitions.Length; j++) {
    ActivatedTransitions[j].Action (); }
}
```

Executing multi-thread model of concurrent processes demands synchronization of their local times, involving current communication between processes executed as separate program threads. Running all threads on single machine, it can be implemented by a meta-process class using some of the common synchronization strategies [7] to control cooperation of processes. Distributed meta-process simulation requires autonomous mechanism for suspending/unrolling built into process instances.

XML technologies provide means to define simulation models as XML documents. Actual simulation program is synthesized by parsing such document and building specified net structure of proper subclasses of components. Formal and semantic correctness of model definition is controlled by an XML Schema for process category describing model structure and data types of semantic token subclasses. XSLT/XPath techniques permit to automatically generate context-dependent user interface for defining simulation models and tasks from XML process Schemas.

3 Web Based Simulation Environment

Functionality of simulation environment for processing input data, including user activity, to produce results specific for different phases of simulation experiment, is resumed in Tab. 2 in the context of previously described concepts.

Web-based simulation environment can be constructed as a virtual grid application in OGSA architecture [8], accessed via specialized Web simulation portal. Client-side activities are localized in the frame of a Web browser providing user with dynamic, context-dependent GUI for interacting with particular functionalities of simulation environment. Several tools suitable for such purpose are available. XML/XSL standards can be used in connection with Java based scripting technologies such as JSP, in order to transform XML process schemas in adaptable user interfaces for

Table 2. Input and output data in the phases of simulation experiment

Phase	User activity	Input	Output
Defining simulation model	Choosing process categories. Building model configuration	XML Shemas for process categories	XML definition of simulation model
Specifying simulation task	Defining model parameters. Specifying initial states	XML model definition & process XML Shemas	XML specification of simulation task
Synthetizing simulation model	Demanding execution of specified simulation job	XML model definition	Java program with model instantiation
Running simulation job	“	XML job specification parameters, initial states	Row result recorded by simulation passes
Elaboration of simulation results	Queries on simulation output; choosing presentation forms	Recorded simulation results	Analytical/synthetic views of results, visualization

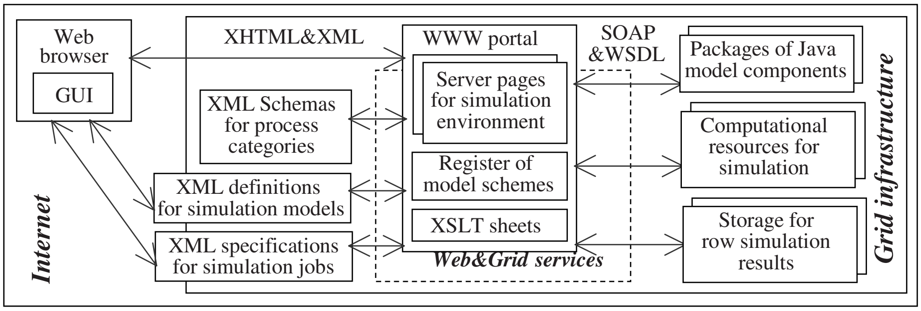
Web browser
GUI
XHTML&XML
WWW portal
SOAP &WSDL
Packages of Java model components
Server pages for simulation environment
XML Schemas for process categories
Computational resources for simulation
Register of model schemes
XML definitions for simulation models
XSLT sheets
Storage for row simulation results
XML specifications for simulation jobs
Web&Grid services
Internet
Grid infrastructure

Fig. 2. The functional structure of a Web-based simulation environment

defining valid models and specifying well formulated tasks to perform simulation research. A middleware layer, with GT3 implementing OGSI on top of the Web service SOAP and WSDL protocols, can organize transparent and secure access to virtual simulation machine composed of distributed software, computation and storage resources. A functional structure of such an environment is outlined on Fig. 2.

Model definitions can be stored locally by users or archived by simulation portal to be shared among cooperating groups of users. Besides, all kinds of resources may be situated anywhere in the Web. XML process schemas defined with namespaces are related to providers of component implementations for various process categories.

References

1. `http://www.computingportals.org`
2. Nemeth Z., Sunderam V.: A Comparison of Conventional Distributed Computing Environments and Computational Grids. In: Computational Science - ICCS 2002. Part II, Vol. 2330 of LNCS, Springer-Verlag (2002)
3. Ochmańska E.: System Simulating Technological Processes. ESM’97, Proceedings of the 11th European Simulation Multiconference, Istambul (1997)
4. Ochmańska E., Wawrzynski W.: Simulation Model of Control System at Railway Station. Archives of Transport. Polish Academy of Science, Committee of Transport. Warsaw 2002-

5. Desel J., Reisig W., Place/Transition Petri Nets. In: Lectures on Petri Nets I; Basic Models, Vol. 1491 of LNCS, Springer-Verlag (1998)
6. Ghezzi C., Mandrioli D., Morasca S., Pezzè M.: A General Way to Put Time in Petri Nets. Proceedings of the 5th International Workshop on Software Specification and Design, IEEE-CS Press, Pittsburg (1989)
7. Yi-Bing Lin, Fishwick P.A.: Asynchronous Parallel Discrete Event Simulation. `http://www.cis.ufl.edu/~fishwick/tr/tr95-005.html`
8. Foster I. at al.: The Physiology of the Grid. `http://www.globus.org/research/papers.html`