Simulating BPMN Models with Prolog

Darko Andročec
Faculty of Organization and Informatics
University of Zagreb
Pavlinska 2, 42000 Varaždin, Croatia
darko.androcec@foi.hr

Abstract. *A business process simulation is used to assess the cost and time of running a process and to identify potential problems with resources. This article presents how Business Process Management Notation (BPMN) models can be reviewed before simulation execution and consequently simulated using logic programming. It also shows the description of a business process model and its properties (semantic correctness and modeling style) in Prolog. For this purpose, a BPMN model of the e-payment process is shown, described and simulated by means of Prolog and a commercial business process modeling tool. Using this specific model we demonstrate how the same simulation results in terms of the total cost of process execution are obtained either with logic programming or specialized commercial tools like IBM WebSphere Business Modeler. However, using Prolog for BPMN simulation has benefits, as buying expensive commercial software is not required, while a complete control over the simulation execution is maintained.*

Keywords. logic programming, business process, simulation, BPMN, Prolog

1 Introduction

In order to design, redesign, understand and improve business processes various methods and techniques can be used. An effective way to analyze complex business processes is modeling and simulation [1]. Business process simulation is invaluable due to its ability to perform a cost-benefit analysis and the analysis of alternative designs feasibility [2]. The simulation functionality is provided by many business process modeling tools (e.g. IBM WebSphere Business Modeler). These tools, based on notations such as EPC or BPMN, offer user interfaces to specify simulation parameters such as task execution time, cost, and resource availability. They allow users to run simulations and get statistical results (e.g. total cost and time required for process execution). BPMN (Business Process Modeling Notation) [3] is a widespread standard for business process modeling maintained by OMG (Object Management Group). According to BPMN specification ([3]), the primary goal of BPMN is to provide a notation that is readily understandable by all business users (i.e. business analysts, technical developers and business people). In this paper we will show a BPMN model of e-payment and how it can be simulated using the logic programming language Prolog [4]. The business model of e-payment was designed by means of the IBM WebSphere Business Modeler tool. Although we shall not discuss this BPMN model in detail, we use it because it contains the key elements of BPMN (tasks, sequence and message flows, process input and output, roles, repositories, service, business items, stop node) and can be simulated using IBM WebSphere Business Modeler. We will show how the same results are obtained using the logic programming language Prolog. More specifically, for writing the source code presented in this article we used the PIE (Prolog Inference Engine) project from Visual Prolog 7.2 Personal Edition tool.

2 Related Work

There are numerous articles about simulating business process models. Ren et al. [1] discuss the

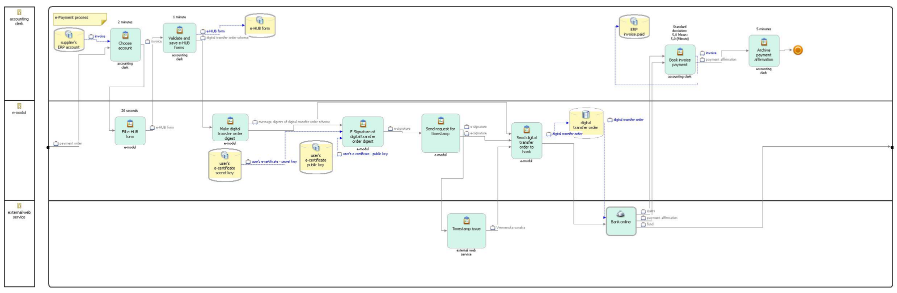
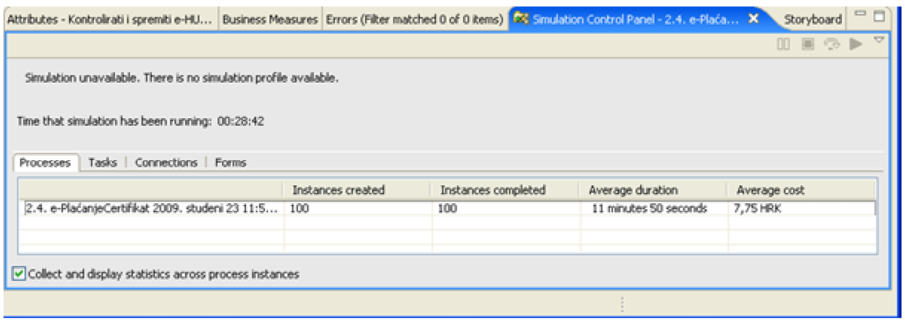


Figure 1: The e-payment BPMN model



Attributes - Kontrolirati i spremiti e-HU... | Business Measures | Errors (Filter matched 0 of 0 items) | Simulation Control Panel - 2.4. e-Plaća... | Storyboard

Simulation unavailable. There is no simulation profile available.

Time that simulation has been running: 00:28:42

Processes | Tasks | Connections | Forms

	Instances created	Instances completed	Average duration	Average cost
2.4. e-PlaćanjeCertifikat 2009. studeni 23 11:5...	100	100	11 minutes 50 seconds	7,75 HRK

☑ Collect and display statistics across process instances

Figure 2: Results of the simulation

conflict between usability and flexibility of software tools in the business process modeling and simulation market. Melao and Pidd [5] show how component technology can be used to develop a simulation library for business process modeling. Experiences and technical challenges related to building a Java based simulation engine for BPMN models are described in [6]. Whereas Jansen-Vullers and Netjes [7] evaluate a number of business process simulation tools, Bosilj-Vuksic et al. [8] propose criteria for evaluation of such tools.

A great number of papers deal with discovering model properties of UML class diagrams, structure charts and flow diagrams [for examples, see 9-12]. However, the literature on applying the ideas to concrete business process models is fairly scarce. Gruhn and Laue [13,14] have shown that logical programming can be exploited for verifying and finding the properties of EPC (event driven process chain) business process models, including semantic (structural) correctness, modeling style, modeling errors and consistency between models.

Plexousakis [2] describes an approach to analyzing business processes using the high-level logic programing language GOLOG. This language is an extended version of the situation calculus and incorporates a formal theory of action. In this article business processes are shown as actions that affect the state. To our knowledge, using the logic programming language Prolog for BPMN model simulation has so far not been described in literature.

3 Defining BPMN Models in Prolog

A logic programming language deals with logical facts and logical rules. We can describe every BPMN element by means of logical predicates, e.g.:

```
role(r1).
roleName(r1, "accounting clerk").
roleCostPerTimeUnit(r1, "1h").
roleCostValue(r1,30.00).
roleCostValueCurrency(r1,"HRK").
```

Here we define the role r1 with the name accounting clerk and the cost of 30 HRK (kuna - Croatian currency) per hour.

```
task(t3).
taskName(t3, "Fill e-HUB form").
taskDescription(t3, "Fill e-HUB form").
taskRoleRequirement(t3, r2).
taskRoleSecondsRequired(t3, 20).
taskExecutionCost(t3,0).
```

Above we describe the task t3, its name and description using Prolog. This task requires the role r2 for 20 seconds, and has not got a fixed execution cost. All the other elements of the BPMN model can be explained in a similar manner. We define connections between BPMN elements as:

```
connection(con1).
connectionLink(con1,bp1,t5).
connectionAssociatedData(con1,bi1).
```

The connection con1 (referred to as sequence flow in BPMN specification) is a link between the process begin node bp1, task t5 and transfers business item bi1.

4 Checking BPMN Model Properties

Having defined BPMN elements, we need to list some basic statements to describe the BPMN model of e-payment (definition of negation, existence of connection between two specific BPMN elements, element types, definition of successor). Next we will show some useful Prolog queries for finding properties of the business process model. For that purpose we will use the rules defined in the BPMN specification [3]. Every BPMN model must satisfy the following rules:

1.A connection between process input and output is not allowed.

```
rule1:-beginProcess(X),endProcess(Y),
not(connectionLink(_,X,Y)),
not(connectionLink(_,Y,X)).
```

2. A connection between two stop events is not allowed.

```
rule2:-stopNode(X),stopNode(Y)
,not(connectionLink(_,X,Y)),
not(connectionLink(_,Y,X)).
```

3. A connection between two start events is not allowed.

```
rule3:-startNode(X),startNode(Y),
not(connectionLink(_,X,Y)),
not(connectionLink(_,Y,X)).
```

4. There must not exist a connection between a start and an end event.

```
rule4:-startNode(X),endNode(Y),
not(connectionLink(_,X,Y)),
not(connectionLink(_,Y,X)).
```

5. A start event must be a source of a sequence flow.

```
rule5(X):-startNode(X),
connectionLink(_,X,_).
```

6. An end event must be a target of a sequence flow.

```
rule6(X):-endNode(X),
connectionLink(_,_,X).
```

7. An end event must not be a source of a sequence flow, there must not be an outgoing sequence flow.

```
rule7(X):-endNode(X),
not(connectionLink(_,X,_)).
```

5 Reviewing Model before Simulation

The IBM WebSphere Business Modeler tool can simulate business process models. In its help files there are rules which must be satisfied before simulation can be executed. According to [15], reviewing the structure of the model is recommendable for ensuring that its simulations and analyses produce information as expected. The first rule is that all possible process paths eventually need to reach a terminate node. To check this, we must use some auxiliary functions [16]:

```
member(X,[X|R]).
member(X,[Y|R]) :- member(X,R).
travel(A,B,P,[B|P]) :-
connectionLink(_,A,B).
travel(A,B,Visited,Path) :-
connectionLink(_,A,C),
C\==B,not(member(C,Visited)),
travel(C,B,[C|Visited], Path).
```

The first rule of the simulation review in the aforementioned tool can be formulated as:

```
endEventNode(X):-
endProcess(X);stopNode(X).
endEvent(X):-endEventNode(X),
noOutgoingConnection(X).
getAllPath(Path):-startNode(X),
endEvent(Y),travel(X,Y,[X],Path).
check1:-getAllPath([First|Path]),
endEvent(First).
```

The second rule describes the requirements for loops if simulations are to be executed correctly (every loop must contain a terminate node or expression defined for exiting the loop, loop conditions must make reference only to input values). As our sample model does not have loops, we did not implement this rule. There are also additional restrictions which apply if static process case analyses are conducted on a model. In that case, the model must not contain any of the following elements: repositories, notification broadcasters, notification receivers, observers, timers and maps:

```
check2IsNotTrue:-repository(X);
notificationBroadcaster(X);
notificationReceiver(X);
observer(X);timer(X);map(X).
```

Our e-payment model has repositories and does therefore not need to comply with this rule.

6 BPMN Simulation with Prolog

In this section we will show that the same simulation results in terms of the total cost of process execution can be obtained by either using logic programming or specialized commercial tools like IBM WebSphere Business Modeler. Logical rules which define costs per tasks are:

```
taskCostRoleExists(T):-
taskRoleRequirement(T, R),
taskRoleSecondsRequired(T, _).
divide(X,Y,Z):-Z is X/Y.
taskCostRolePerHour(T,Z):-
taskRoleRequirement(T, R),
roleCostValue(R,V),
taskRoleSecondsRequired(T, X),
 Z is ((X/3600)*V).
```

Above we compute the cost per hour of a specific task depending on the required work time of a role and its cost. We get all the costs of a relevant task by summing all task role costs and fixed execution costs (for example, the fixed cost of a timestamp issue per e-payment transaction), if there are any:

```
taskAllCost(T,Z):-
(taskCostRolePerHour(T,X),
taskExecutionCost(T,Y), Z is X+Y);
(not(task(T)), Z is 0).
```

Then we make logical rules for the point-to-point cost calculation of the BPMN model:

```
travelComputeCost(A,B,P,[B|P]) :-
connectionLink(_,A,B).
travelComputeCost(A,B,Visited,Path)
:-connectionLink(_,A,C), C\==B,
not(member(C,Visited)),
travelComputeCost(C,B,
[C|Visited],Path).

listCostAll([], Cost, PastCost):-
Cost is PastCost.
listCostAll([Head|Tail],
Cost,PastCost)
:-taskAllCost(Head, R),
NewCost is PastCost+R,
T is Tail,
listCostAll(T,Cost,NewCost).
```

Afterwards we can make queries in the Dialog window of PIE, e.g.:

```
listCostAll(["stop1","t1","t2","srv1",
"t7","t10","t9","t8","t6","t4","t3",
"t5","bp1"], Cost, 0).
COST = 7.75.
```

The result is the same as presented in Figure 2. (IBM Websphere Business Modeler simulation results of the e-payment model).

7 Conclusion

Valid design of business processes is crucial to the success of any business. Whereas business process modeling enables a comprehensive analysis of a business process, simulation is an effective way to perform both cost-benefit analysis and the analysis of alternative designs feasibility. Although we can use commercial tools for business process modeling and simulation, they do not allow access to used methods and techniques which produce results from simulation input parameters. The flexibility to change this process is not provided either.

This paper deals with simulating a BPMN model with the logic programming language Prolog. We first show how to define elements of BPMN diagram (using example of e-payment BPMN model) and then how to check its properties and review the model before simulation. We believe that the potential of the idea of using Prolog to simulate business process models lies in the flexibility and better control over the simulation execution that it provides.

References

[1] Ren C., Wang W., Dong J., Ding H., Shao B., Wang Q.: Towards a flexible business process modeling and simulation environment, Proceedings of the 2008 Winter Simulation Conference, Miami, USA, 2008, pp. 1694-1701.

[2] Plexousakis D.: Simulation and analysis of business processes using GOLOG, Proceedings of conference on Organizational computing systems, Milpitas, USA, 1995, pp. 311-322.

[3] OMG: Business Process Model and Notation (BPMN) Version 1.2, available at `http://www.omg.org/spec/BPMN/1.2/`, Accessed: 8th December 2009.

[4] Nilsson U., Maluszynski J.: Logic, Programming and Prolog, available at `http://www.saber.ula.ve/bitstream/123456789/16227/2/libro-texto.pdf`, Accessed: 30th March 2010.

[5] Melao N., Pidd M.: Using component technology to develop a simulation library for business process modeling, European Journal of Operational Research, Amstredam, Netherland, 2006, pp. 163-178.

[6] Lanner Group: BPMN and Simulation, available at `http://www.dynamic.co.kr/Witness_Training_Center/Articles/Bpmn%20-%20simulation.pdf`, Accessed: 2nd April 2010.

[7] Jansen-Vullers M.H., Netjes M. : Business Process Simulation - A Tool Survey, available at `http://www.daimi.au.dk/CPnets/workshop06/cpn/papers/Paper05.pdf`, Accessed: 2nd April 2010.

[8] Bosilj-Vuksic V., Ceric V., Hlupic V.: Criteria for the Evaluation of Business Process Simulation Tools, Interdisciplinary Journal of Information, Knowledge, and Management, Santa Rosa, USA, 2007, pp. 73-87.

[9] Mammar, A.: A formal approach and its tool support for the specification and the verification of structural properties on UML activity diagrams, Software Engineering Research and Practice, Las Vegas, USA, 2006, pp. 988-994.

[10] Kielland, T., Borretzen, J.A.: UML consistency checking, available at `http://www.idi.ntnu.no/grupper/su/sif8094-reports/2001/p8.pdf`, Accessed: 22th April 2010.

[11] Gustafsson, J., Paakki, J., Nenonen, L., Verkamo, A.I.: Architecture-centric software evolution by software metrics and design patterns, Proceedings of the Sixth European Conference on Software Maintenance and Reengineering, Washington, USA, 2002, pp. 108.

[12] Tse, T.H., Chen, T.Y., Chan, F.T., Chen, H.Y., Xie, H.L.: The application of Prolog to structured design, Software: Practice and Experience, New York, USA, 1994, pp. 659-676.

[13] Gruhn, V., Laue, R.: Validierung syntaktischer und anderer EPK-Eigenschaften mit PROLOG, available at `http://ebus.informatik.uni-leipzig.de/~laue/papers/epk-Prolog.pdf`, Accessed: 8th December 2009.

[14] Gruhn, V., Laue, R.: Checking Properties of Business Process Models with Logic Programming, available at `http://ebus.informatik.uni-leipzig.de/~laue/papers/bpm-analyse-Prolog.pdf`, Accessed: 8th December 2009.

[15] IBM Help: Reviewing your model before simulation, available at `http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.`

ibm.btools.modeler.advanced.help.doc/doc/tasks/simulating/reviewstopend.html, Accessed: 8th December 2009.

[16] Fisher, J.R.: Prolog Tutorial, available at http://www.csupomona.edu/~jrfisher/www/Prolog_tutorial/, Accessed: 9th December 2009.