TOWARDS A FULL AUTOMATIC APPROACH TO USE INA PETRI NETS ENVIRONMENT FOR BUSINESS PROCESS MODELING

Raida El Mansouri

LIRE Laboratory, Department of Computer Science,
University Mentouri of Constantine, Algeria
raidaelmansouri@yahoo.fr

ABSTRACT

Business process models describe how a business works, or more specifically, how they accomplish missions, activities, or tasks. The automated control and coordination of business processes is made possible by task control constructs that model behaviors like concurrency, asynchronism, and choice. However, there is a real danger of introducing control flow anomalies and behavioral inconsistencies like deadlock, livelock, imperfect termination, and multiple task repetitions [5]. Petri Nets provide a powerful formal modeling method based on solid mathematical fundament while having graphical representation of system models as net diagrams and provide various analysis techniques such as reachability tree, incidence matrix and invariant analysis method, through which properties of the Petri Net model such as liveness, reachability and deadlock can be analyzed.

*This paper proposes an approach to illustrate the use of the Petri Net **INA** (**I**ntegrated **N**et **A**nalyzer) [6] environment for formalizing business process specifications and using analytical techniques to support verification studies. The first step is automated.*

Keywords: *Business process modeling, Petri Nets, INA, Verification, Graph transformation, Meta Modeling.*

1. INTRODUCTION

Business process models describe how a business works, or more specifically, how they accomplish missions, activities, or tasks (henceforth referred to as tasks). A single model shows how a business accomplished a single task. It would take many process models to fully detail the "hows" of most real world enterprises.

A single process can consist of many actors (people, organizations, systems) performing many tasks. In order to accomplish the overall task, the actors must complete specified sub-tasks in a coordinated manner. Sometimes, these sub-tasks can be performed in parallel. Sometimes they are sequential. Some processes require repetition of sub-tasks. Most processes have decision points where process flow can branch depending on either the condition of the system or the particular process execution. In cooperative processes actors must pass information. This information transfer can be the trigger for an actor to begin a sub-task. Other triggers are possible, such as time or interrupts. Some processes are ad-hoc. That is, the sub-tasks do not have well defined triggers. Actors may not need to complete all of a subtask before them or another actor start work on another dependent subtask. Finally, a process can look differently when described from the viewpoint of different actors. A business process modeling methodology needs to be able to represent these different aspects of a process description.

Business process modeling (BPM) provides a conceptual basis for the specification of all business procedures. It helps the coordination and integration of distributed resources, tasks, and individuals, the effective management of all of which is critical to sustaining organizational capabilities. Workflow Management supports both business process specification and automated execution of business procedures, and is a next-generation extension to BPM efforts that emphasizes the increased role that information systems have come to play in today's businesses. Workflow Management involves two phases – (a) the *modeling phase* that abstracts from business procedures and defines computer-implementable *workflow* specifications, and (b) the *execution phase* that executes instances of the workflows to meet business requirements, and both these phases are managed and coordinated by a Workflow Management System (WfMS) [5].

Essentially, a WfMS integrates and automates the execution of steps that comprise a business process, and simultaneously manages resource (information, people, etc.) assignments. This paper focuses on the modeling and analysis issues involved in establishing logical and syntactical correctness of business process specifications before they are implemented. INA is used to illustrate the ideas behind these issues. The work is based on ideas presented in [5], [7], and [8].

The rest of the paper is organized as follows. In section 2 we present some concepts of process modeling that are relevant with our work. In section 3 we recall some notions about Petri nets that are necessary for the specification and analysis of business processes. We will focus especially on Liveness and deadlock-trap properties. In section 4 we will propose our approach and apply it on an example. The last section concludes the paper and gives some perspectives of this work.

2. PROCESS MODELING

Process modeling aims to produce an abstraction of the process that serves as a basis for detailed definition, study, and possible reengineering to eliminate non-value added activities. The process model must allow for a clear and transparent understanding of the activities being undertaken, the dependencies among the activities, and roles (people, machines, information, etc.) necessary for the process. An activity-centered modeling methodology is used for defining process models in that a process is viewed as a sequence of inter-related tasks, the transfer of control between them being determined by logical operations [5].

The complete specification of business processes includes (a) the control flow, i.e., the partial and total ordering specifying the sequence of the various tasks, (b) the data flow, i.e., the information requirements, and the resource (people, machines, etc.) allocations for the execution of the various tasks. This is required for identifying the input and output requirements for each task, and also to *put together* a skeletal outlay of the process that is both conceptually and descriptively complete. There has been significant research in developing *process meta-models*, namely, a representational language in which to express workflow models amenable to automation. Stated simply, the ability to represent behaviors like concurrency and choice increases the chances of defining logically incorrect models with control flow errors, the execution of which could result in deadlock, livelock, etc. The focus of this paper is to highlight the use of Petri nets as a technique for formalizing business process models to analyze verification issues, and to support performance evaluation studies. **INA** is used to illustrate these issues.

The ease and flexibility of graphical modeling languages brings with it a possibility for introducing control flow anomalies in process specifications. The major control flow verification issues, i.e., checking for deadlock, livelock, multiple repetition, etc. are described as follows:

2.1. SOME CONTROL FLOW ANOMALIES

2.1.1 DEADLOCK SITUATION

When control flow from one of several required merging paths is missing [5].

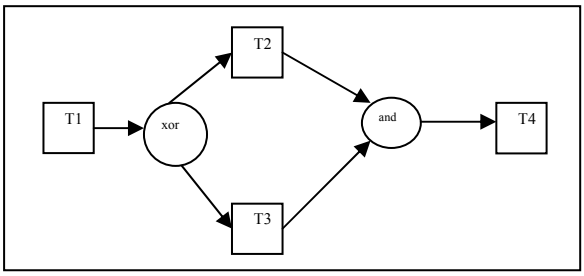

Figure 1 : Deadlock Situation

2.1.2 MULTIPLE REPETITION SITUATIONS

When control flow arrives from multiple sources, but only one is necessary.

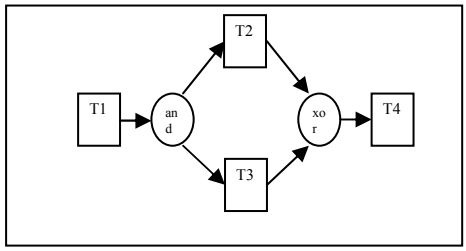

Figure 2 : Multiple répétition

2.1.3 LIVELOCK SITUATION

When control flow fails to exit out of a set of previously executed tasks.

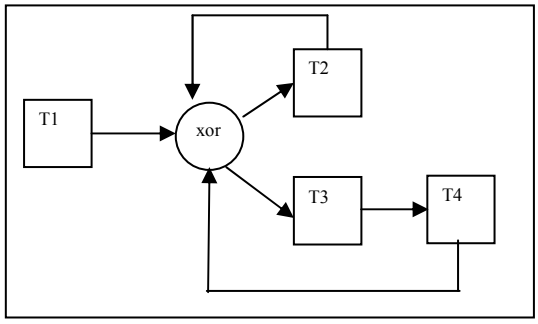

Figure 3 : Livelock Situation

2.2. CONTROL FLOW CORRECTNESS

Create a *control-flow model* specifying just the tasks, and the ordering required within, without the overhead of resource, data requirements - Petri nets have emerged as a very popular technique for such abstractions [5].

These models have been used to answer the following questions: (a) the *initiation problem* is to determine if there is a sequence of task executions that will lead to the execution of a particular task – this has been shown to be NP-complete, and (b) the *termination problem* is to determine if the control-flow specification will lead to a terminal state – this has been shown to require exponential storage requirements.

3. PETRI NETS [MUR 89]

3.1. PETRI NETS: TERMINOLOGY AND NOTATION

This section introduces the basic Petri net terminology and notations. The classical Petri net is a directed bipartite graph with two node types called *places* and *transitions*. The nodes are connected via directed *arcs*. Connections between two nodes of the same type are not allowed. Places are represented by circles and transitions
by rectangles.

Definition 3.1.1

A Petri net is a triple (P; T; F)*:*

- P is a finite set of **places**,
- T is a finite set of **transitions** *(*$P \cap T = \varnothing$ *;),*
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of **arcs** (flow relation)

At any time a place contains zero or more *tokens*, drawn as black dots. The *state*, often referred to as marking, is the distribution of tokens over places. The number of tokens may change during the execution of the net. Transitions are the active components in a Petri net: they change the state of the net according to the following *firing rule*:

(1) A transition t is said to be *enabled* iff each input place p of t contains at least one token.
(2) An enabled transition may *fire*. If transition t fires, then t *consume* one token from each input place p of t and *produces* one token for each output place p of t.

Definition 3.1.2
A Petri net (PN;M) is **live** iff, for every reachable state M' and every transition t there is a state M" reachable from M' which enables t. A Petri net is structurally live if there exists an initial state such that the net is live.

Definition 3.1.3
A Petri net (PN;M) is **bounded** iff for each place p there is a natural number n such that for every reachable state the number of tokens in p is less than n. The net is **safe** iff for each place the maximum number of tokens does not exceed 1. A Petri net is **structurally bounded** if the net is bounded for any initially state.

Definition 3.1.4
A Petri net PN is **well-formed** iff there is a state M such that (PN;M) is live and bounded. Paths connect nodes by a sequence of arcs.

Definition 3.1.5
A Petri net is a **free-choice** Petri net [1] iff, for every two transitions t1 and t2, $\cdot t1 \cap \cdot t2 \neq \varnothing$ implies $\cdot t1 = \cdot t2$.

Deadlock-trap-property [6]

A net satisfies the *deadlock-trap-property*, if the maximal trap in each minimal deadlock is sufficiently marked [6].
A *trap* is a set of places that, if it contains tokens, cannot become clean, because every transition which subtracts tokens from one place in this set has a post-place in this set, and thus returns tokens to the set. Hence, the empty set is a trap.

A trap is *maximal*, if it is not a proper subset of a trap. A deadlock is a non-empty set of places that cannot be marked again once it is clean, because every transition which would fire tokens onto a place in this set has a pre-place in this set (and so cannot fire).

A deadlock is **minima**l, if it does not properly contain a deadlock. A set of places is sufficiently marked, if it contains a place which contains sufficiently many tokens to enable all its post-transitions.

3.2. PETRI-NET FORMALIZATIONS OF BUSINESS PROCESS MODELS

Any process can be understood to be a collection of events, the conditions that enable these events to occur, and the conditions that are satisfied following the completion of these events. A Petri net ideally describes this intuition, and explicitly separates the conditions, and the events involved in a process, and models state changes involved therein, through a simulated movement of tokens. To map the business processes to Petri nets, we have used the ideas propose in [5]. For example the Petri net model in Figure 4 is the mapping of the process model in Figure 1.

Petri-nets offer the advantage of graphical appeal coupled with a rigorous formalism that has found tremendous use in behavior systems and processes that exhibit asynchronism, concurrency, and determinism. Petri nets are especially attractive for formalizing and analyzing business processes for the following reasons: (i) clear and unambiguous description of process logic, (ii) intuitive ease and feel of a self-documenting graphical formalism that retains complete conceptual clarity, and (iii) extensive analysis capabilities that vastly extend the power and usefulness of structured process description languages like IDEF3. The control flow issues highlighted previously are readily expressed in Petri-net theoretic terms, e.g., reachability, deadlock, liveness, etc. [5]. Moreover, Petri nets allow for a study of both (a) *structural properties* pertaining to the static aspects of the process's definition, and (b) *Behavioral properties* pertaining to the dynamic aspects of the process observed during its execution.

4. THE APPROACH OF USING INA ENVIRONMENT

In order to use the INA environment for formalizing business process specifications and the use of analytical techniques to support verification studies, we propose the following steps.

1) First of all each business process model is mapped to an equivalent Petri net representation. The mapping is based on the ideas proposed in [5]. The mapping process is performed automatically using our tool [2] based on graph transformation and Meta modeling [3].
2) Then each graphical representation of the obtained net is mapped to a textual representation [6]. This step is being automated as an extension of our developed tool [2]. This automation is simple thanks to graph transformation and Meta modeling [3].
3) Then we have used the INA environment for analyzing the represented net.
4) Feed backs are given to the user to correct his business process model.

4.1. APPLICATION OF THE APPROACH

We have used this approach for the three above situations as follows.

4.1.1. DEADLOCK SITUATION

a) We have used our tool for mapping the business model of the figure 1 obtained the following equivalent Petri nets representation.

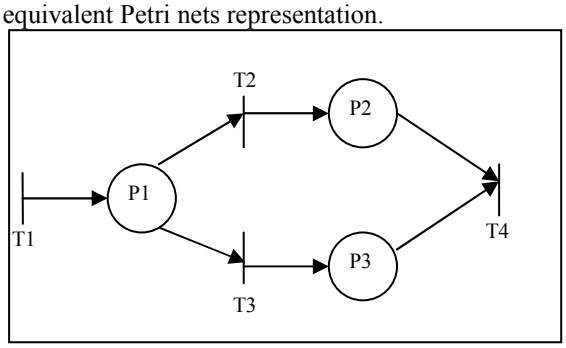

Figure 4: Petri net model representing the process of Figure 1

b) Then we have mapped this representation to the following textual form in the file **dead2.pnt**

```
P   M   PRE,POST   NETZ 1:3_prog_2_term
 0 1     , 1
 1 0     1, 2 3
 2 0     2, 4
 3 0     3, 4
 4 0     4
@
place nr.          Name capacity time
      0: p0        oo    0
      1: P1        oo    0
      2: P2        oo    0
      3: P3        oo    0
      4: p4        oo    0


@
trans nr.          Name priority time
      1: T1          0    0
      2: T2          0    0
      3: T3          0    0
@
```

Note: We have added two places (Start: here place 0 and End: here place 4).

c) Then we have applied the INA environment (INA.exe) with the option **A** (Analyse) to the file dead2.pnt and we have obtaine the following results.

```
        >>>>>>>>>>>>> Welcome to the Integrated Net
                                  Analyzer! <<<<<<<<<<<<<
Version 2.2          Jul 31 2003          Peter Starke,
Berlin

Current net options are:
   token  type: black        (for Place/Transition nets)
   time option: no times
   firing rule: normal
   priorities : not to be used
   strategy   : single transitions
   line length: 80

Do You want to
   edit ? ….....E
   fire ? ….....F
   analyse ? ….....A
   reduce ? ….....R
   read the session report ? ….....S
   delete the session report ? ….....D
   change options ? ….....O
   quit ? ….....Q
 choice > A
Netfiles:
altbit  ampel  bpm05  dead1  dead2  dinner
ININET  live01
reduce_a reduce_b reduce_c reduce_f reduce_m
reduce_u reduce_v reduce_w
red_simp stateeq  terminal
Petri net input file > dead2.pnt

Information on elementary structural properties:
Current name options are:
    transition names not to be written
    place names not to be written
….....Reset options? Y/N N
….....Print the static conflicts? Y/N N
The net is not statically conflict-free.
The net is pure.
The net is ordinary.
The net is homogenous.
The net is not conservative.
The net is subconservative.
The net is structurally bounded.
The net is bounded.
There are no proper semipositive T-surinvariants.
The net is not live.
The net is not live and safe.
The net is not a state machine.
The net is free choice.
The net is extended free choice.
The net is extended simple.
The net has places without pre-transition.
The net is not state machine decomposable (SMD).
The net is not state machine allocatable (SMA).
The net is not strongly connected.
The net is not covered by semipositive T-invariants.
The deadlock-trap-property is not valid.
The net has places without post-transition.
The net is marked.
The net is marked with exactly one token.
```

Interpretation of the result

The net is not live and the *deadlock-trap-property* is not valid. So there is a deadlock situation.

We have also used our approach to verify the situations of multiple repetition (Figure 2) and livelock (Figure 3) and we have obtained the expected results.

5. CONCLUSION AND FURTHER WORK

In this paper we have proposed an approach to use the INA Petri nets environment for formalizing business process specifications and using analytical techniques to support verification studies. Three properties have been verified: Deadlock, livelock, and multiple repetition using INA environment. We have automated the first step of our proposed approach and we plan to automate the steps 2 and 3. To this end, we will use the tool ATOM3 [3] for mapping graphical representation of business processes to Petri nets models.

REFERENCES

[1] J. Desel and J. Esparza, Free Choice Petri Nets, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.

[2] R. Elmansouri, A Full Automatic Approach based on Meta-Modelling and Graph Grammars to Generate Petri Nets models for Business Processes, internal report No 1, Department of Computer Science, University Mentouri Constantine, 2007. Submitted to a journal.

[3] J. de Lara, H. Vangheluwe, ATOM3 : A Tool for multi-formalism and meta-modeling, LNCS No 2306, 2002.

[4] T. Murata, "Petri Nets: Properties, Analysis, and Applications". *Proceedings of the IEEE*, 77(4):541–580, 1989.

[5] E. Sivaraman and M.Kamath
"On The Use of Petri Nets for Business Process Modeling", Proceeding of the 11th Annual Industrial Engineering Research Conference, Orlando, FL., May 2002.

[6] P.H. Starke and S. Roch, "INA: Integrated Net Analyzer", 2003.

[7] Wil M.P. van der Aalst**,** Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques, In Aalst, W.M.P., Desel, J., and Oberweis, A., editors, *Business Process Management – Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 161–183. Springer-Verlag, 2000.

[8] Wil M.P. van der Aalst, Arthur H.M. ter Hofstede, and Mathias Weske Business Process Management: A Survey, Lecture Notes in Computer Science 2678 Springer 2003, ISBN 3-540-40318-3 2003.