Process Modelling using Petri Nets

Katalina Grigorova

Abstract: *This paper discusses the reasons, which impose Petri nets as a conceptual standard for modelling and analysis of workflow. Petri nets notation is used for representation of the main routing constructs as well as for the workflow process description. The correspondence of Petri-net-based model and graph model is demonstrated.*

Key words: Workflow*, Workflow Building Blocks, Process Modelling, Routing Constructs.*

INTRODUCTION

Many of today's Workflow Management Systems (WFMS) are based on Petri nets. It is supposed that they could serve as a conceptual standard for modelling and analysis of workflow. There are several reasons for using Petri-net-based WFMS [4]:

- Formal semantics despite the graphical nature

Using a Petri-net-based WFMS the business logic can be represented by a formal but also graphical language. It is easy to show that the Petri net can be used to model the primitives identified by WFMC [1]. Figure 1 shows how the four workflow primitives identified by WFMC can be mapped onto Petri nets. Tasks are mapped onto transitions and causal relations are modelled by places. These primitives are also present in today's WFMSs.

Many of today's available WFMSs provide ad-hoc constructs to model workflow procedures. Some WFMSs provide exotic constructs whose semantics sometimes is not very clear. To avoid these problems one could use a Petri-net-based WFMS. The exchange of workflow process definitions between two Petri-net-based WFMSs is easy compared to the exchange of workflow process definitions between two WFMSs based on different concepts.

- State-based instead of event-based

In contrast to many other process modelling techniques, the state of case can be modelled explicitly in a Petri nets. Although the distinction between an event-based and a state-based description seems to be very subtle, there are many reasons for using state-based WFMSs instead of event-based. The most important is that event-based WFMSs can only be used satisfactory in situations where the workflow engine is leading, i.e. tasks are triggered by the WFMS instead of environment of the WFMS.

- Expressiveness

Petri nets can explicate both process structure and process dynamism, including concurrency. Therefore, they are capable to serve as a language for process structure mapping, process simulation and workflow management.

- Abundance of analysis techniques

Petri nets are marked by availability of many analysis techniques. This is great asset in favour of a Petri-net-based WFMS. In general these techniques can be used to prove properties (safety properties, deadlock, etc.) and to calculate performance measures (response times, waiting times, occupation rates, etc.). In this way it is possible to evaluate alternative workflows.

PROCESS MODELLING

1. Routing constructs

Process management enables full controls of the work with data during they are routed through a business process. Workflow management is automated co-ordination and control of these work processes. Workflow is a model of the business processes and

contains all information about what has to be done in which order from which person and which data are required to do the work.

The workflow process definition defines the elements that make up a workflow. It could be done either within WFMS or within specific process modelling environment. A variety of different tools may be used to analyse, model, describe and document a process. That is why, the common interchange format is needed to support the transfer of workflow process definitions between separate products. If also defines a formal separation between the development and run-time environments, enabling a process definition, generated by one modelling tool, to be used as input to a number of different workflow run-time products.

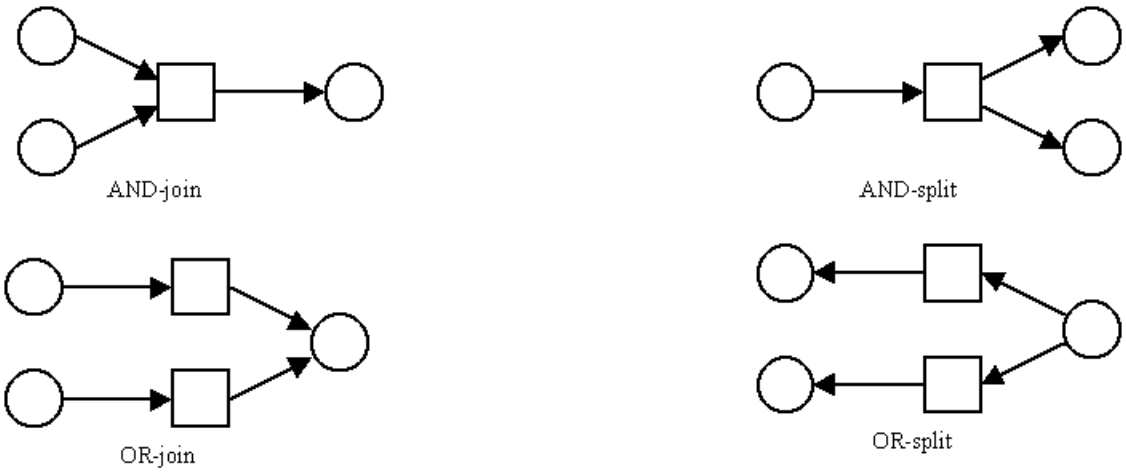

Figure 1. Workflow primitives

In order to standardise workflow process definition WFMC identifies four routing constructs used to spcify the relationships between tasks during the process execution: sequential, conditional, parallel and iterative [1]. Workflow process description is based on these constructs. Figure 1 shows how the workflow primitives are mapped onto Petri nets. Using Petri net based building blocks, the four routing constructs are presented in terms of Petri nets:

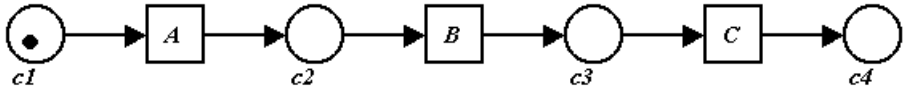

Figure 2. Sequential routing

- sequential routing

It describes sequentially executed tasks where one task is followed by the next task. An example of sequential routing is shown on Figure 2. Task A is executed first, task B starts after completion of task A and before starting of task C.

- parallel routing

Parallel routing describes a situations where two or more tasks are executed at the same time or the order of executions is less strict. For example, two tasks B and C (Figure 3) need to be executed but the order of execution is arbitrary. To model such a parallel routing, two building blocks (Figure 1) are used: AND-split and AND-join. The execution of AND-split, A enables both task B and task C. According to AND-join, D is enabled after execution both B and C, i.e. D is used to synchronise two subflows. As a result, task B and

task C are executed in parallel. This means that B and C are executed in arbitrary order. It is even possible that the execution of B and C overlap in time.

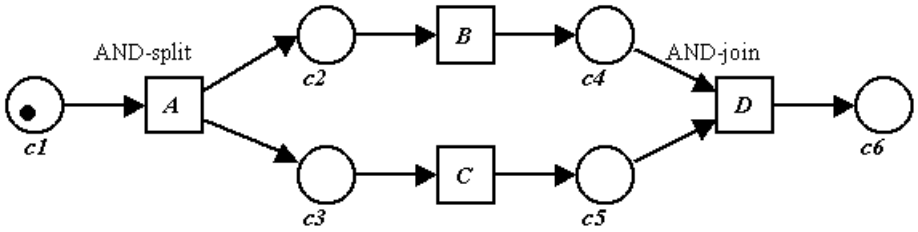

Figure 3. Parallel routing

- conditional routing

This construct is used to model a choice between two or more alternative tasks. After the task A has been completed, the choice is made between the execution of task B or task C. Task D starts when either B or C is completed. To model the choice between two or more alternatives, two building blocks are used: OR-split and OR-join (Figure 4).

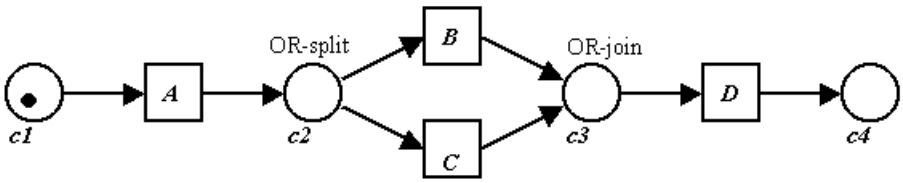

Figure 4. Conditional routing

- iterative routing

Iteration models multiple execution of one or more tasks. Consider for example from Figure 5, task B could be executed repetitively until the condition is met . Task C is a control task, which checks the result of task B. Based on this check, task B may be executed once more.

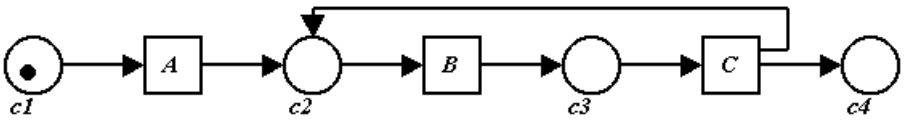

Figure 5. Iterative routing

2. Workflow net (WF-net)

In the process definition it is specified which tasks need to be executed and in what order. Modelling a workflow process in terms of a Petri net is rather straightforward: tasks are modelled by transitions, conditions are modelled by places, and cases are modelled by tokens.

A Petri net which models workflow process definition is called Workflow net (WF-net). A WF-net satisfies two requirements [4]:

- it has one input place and one output place. A token in input place corresponds to case which needs to be handled, a token in output place corresponds to a case which has to be handled;

- every transition (place) should be located on a path from input place to output place.

3. Petri net based process model

[2] describes a method for process manipulation. The processes are represented as graphs whose nodes correspond to the activities (tasks), composing the process and the arcs correspond to triggering and terminating events for each activity.

The conditions for starting the process are defined by triggering events, whereas the terminating events describe the results of the process execution. The events of the process should meet the following requirements:

- the triggering event of the process is a triggering event and of one sub-process only;
- the terminating event of the process is a terminating event at least of one sub-process;
- the terminating events of the sub-processes (except of the terminating event of the whole process) are triggering events of other sub-process.

When the above circumstances take place, the sub-processes of the given process form a directed graph as shown on Figure 6. The nodes correspond to the sub-processes. The node X is connected by an edge with node Y if and only if the terminating event of X is a triggering event (or a part of it) for Y. There is only one node without predecessors (its triggering event is a triggering event of the whole process too) and one or more sub-processes that do not have successors, as their terminating events are terminating events of the whole process.

Each node has exactly one triggering and one terminating event. The start of an activity is possible if the triggering event is available. The terminating event is produced as a result of an activity execution.

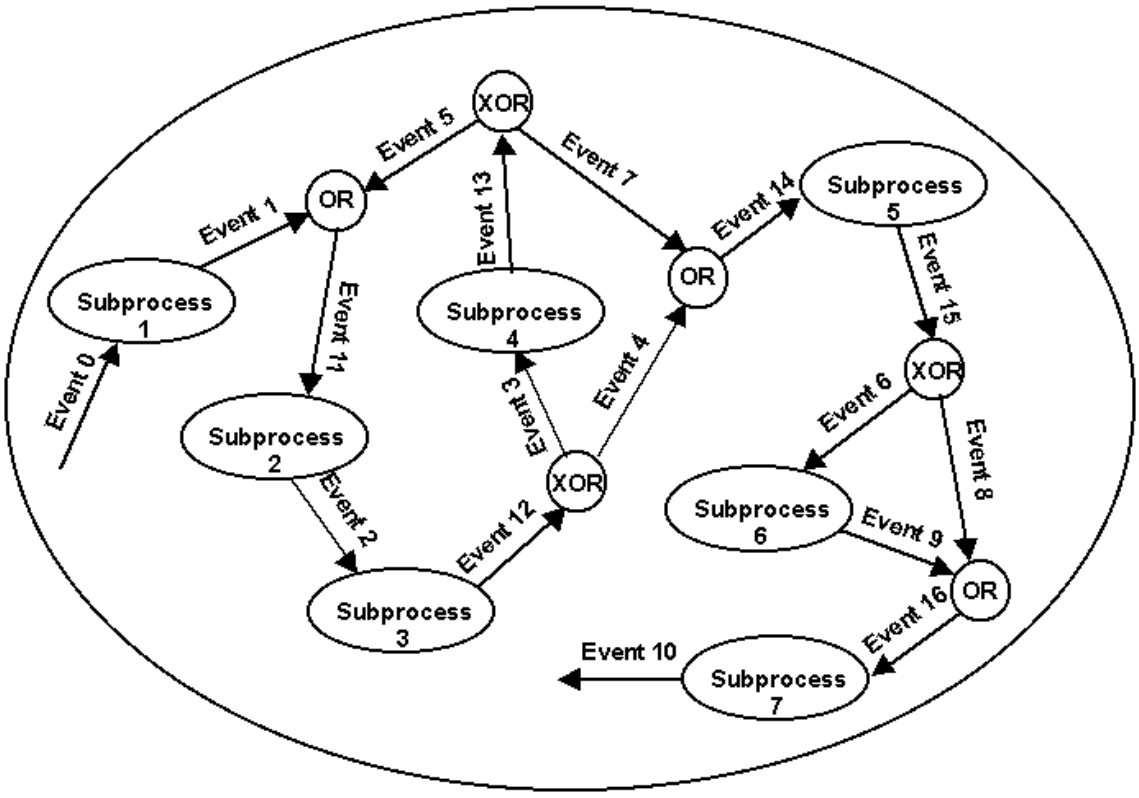

Figure 6. Process graph

Some of the activities require the availability of more than one event or could be started after availability of different events. Similarly, as a result of an activity execution, more than one event could be created. These situations are described by composite events, formed as logical functions of simple events.

There are six types of composite events:

- triggering, created as AND function between two or more other events;
- terminating, created as AND function between two or more other events;
- triggering, created as OR function between two or more other events;
- terminating, created as OR function between two or more other events;
- triggering, created as XOR function between two or more other events;
- terminating, created as XOR function between two or more other events.

Transformation of the graph model into Petri net based model requires identification of an appropriate mapping of graph components onto Petri net. One can see that the graph nodes properly map the transitions and simple events map places. The composite events need to be substituted by building blocks.

Note that the composite events formed as AND and OR functions are straightforward transformed into primitives presented on Figure 1. Regarding the composite events formed as XOR function, the corresponding primitives do not exist. That is why two new building blocks are introduced: XOR-split and XOR-join. In order to distinguish AND-split from XOR-split and AND-join from XOR-join the involved transitions are marked as shown on Figure 7. The primitives OR-split and OR-join remain the same.

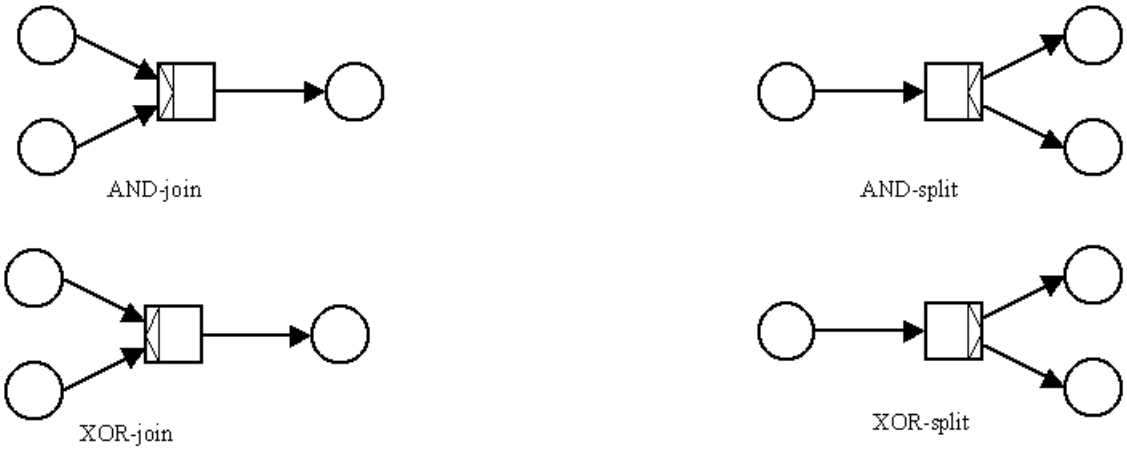

Figure 7. Improved workflow primitives

The method described in [2] manipulates process data organised within traditional DBMS environment. It aims unification of model components. These are the activities forming the process and called subprocesses. Graph nodes represent them. Events are represented as subprocess characteristics. In this way, the graph model consists of one type of components only. This allows application of classical graph processing algorithms. They are used for new process generation or existing process modification. Additional possibility is ordering of subprocesses and analysis of graph connectivity.

Main disadvantage of the graph model is the lack of proper simulation procedures. The executability of Petri nets is based on their marking concept. The marking concept shows very clearly the relationships between process structure and process instances. Moreover, it can be the basis of experimental simulation and of formal process analysis. The performance of techniques for Petri net analysis is also applicable. In order to do that, it is necessary to transform graph model into WF-net which could be done analogously.

The method is implemented as an element of programming module developed within MS Access[3].

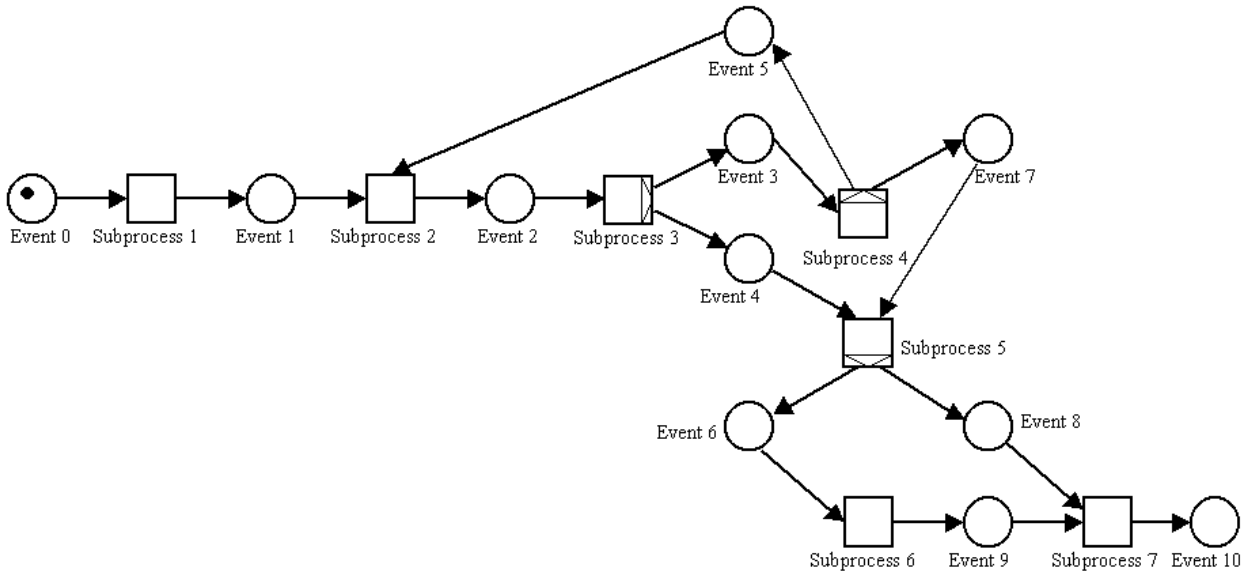

Figure 8. A Petri net based process model

CONCLUSIONS AND FUTURE WORK

Using Petri nets based process modelling allows application of different analysis techniques. These techniques can be used to examine the behaviour of the process and to calculate its performance measures. Transforming Petri net model into graph model gives possibility of using classical graph processing algorithms.

A process model that accommodates different modelling perspectives is very flexible and makes developed models more understandable and usable.

The future work will be focused on enhancement the model presented in [3] with data necessary for process evaluation and choice of alternative workflows.

REFERENCES

[1] WFMC. Workflow Management Coalition Terminology and Glossary (WFMC-TC-1011), Workflow Management Coalition, Brussels, 1996

[2] Grigorova K., Process manipulation within DBMS environment, CompSysTech'2002 - Bulgarian Computer Science Conference - 20-21.06.2002, Sofia, Bulgaria, III.28-1-6

[3] Grigorova K., Workflow Representation within DBMS environment, CompSysTech'2001 – Bulgarian Computer Science Conference – 21-22.06.2001, Sofia, Bulgaria, САИ, II.7-1-II.7-5

[4] Van der Aalst W.M.P., The Application of Petri Nets to Workflow Management, http://wwwis.win.tue.nl/~wsiwa/jcsc

ABOUT THE AUTHOR

Katalina Petrova Grigorova, Department of Informatics and Information Technologies, University of Rousse, Phone: +359 82 888 470, E-mail: katya@ami.ru.acad.bg