

Cover page

The contact author: **Mariusz Momotko**

Address for correspondence:

e-mail Mariusz.Momotko@rodan.pl
conventional **Rodan Systems, oddzial Sopot**
mail **ul. M. Reja 13/15**
 81-874 Sopot
 Poland
tel. **+48 58 550 2024**
fax **+48 58 550 1202**
mobile **+48 504 723 113**

States the reference number: **379**

Topic area: **Industrial Applications & Experience**

Paper title: **Contextualized process execution based on the extended BPMN**

Full list of authors: **Mariusz Momotko, Bartosz Nowicki**

Indicates the topics relevant to the paper:

Business Process Engineering and Execution Support
(primary)
Collaboration/Cooperation
Data and/or Process Models and Design Tools
User Interfaces & Visualization
Workflow Systems

Contextualized process execution based on the extended BPMN*

Mariusz Momotko, Bartosz Nowicki

Rodan Systems
ul. Puławska 465
02-844 Warszawa
Poland

{Mariusz.Momotko, Bartosz.Nowicki}@rodan.pl

Abstract

Humans in order to create, share and improve knowledge about business processes need a common, readable and preferably visual notation. So far, there was a lot of effort put into visualisation of process definition (e.g. recently published Business Process Modelling Notation - BPMN). We postulate to put equal stress on visualisation of process execution allowing process performers to understand the process history, its current state and possible future execution (potential consequences of the current decision). In our opinion, putting activity of interest in its visualised context makes the user knowledge more comprehensive and, consequently, increases productivity.

In this paper, we define the process instance notation as an extension of BPMN. The underlying premise for such approach is the reuse of well defined and commonly accepted concepts (also their graphical form) from process definition level on the process execution level - this again increases chances for the notation to be understandable. The prototype implementation is integrated within the ICONS knowledge management platform.

1. Introduction

* This work was supported by the European Commission project ICONS, project no. IST-2001-32429.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment

Proceedings of the 29th VLDB Conference,
Berlin, Germany, 2003

Flexible and innovative business processes are one of the key elements that enables modern organisations to succeed. In order to define, share and improve knowledge on business processes human need a standard way of describing them. During the last decade there has been much activity laid on defining appropriate visualisation of the business processes. Thousands of business analysts have been studying the way companies work and defining business processes with various modelling techniques. Yet, hundreds of business modelling tools have been developed.

Despite such a huge effort spending on defining business processes, the available modelling techniques differed from each other. There was no a standard, flexible and widely accepted notation that could enable process analysts to speak the same language and to be easily transformed to various range of business process execution languages.

Recently, Business Processes Management Initiative (BPMI) put much effort to develop a standard business process modelling notation that would allow interoperation of business processes at the human level as well as to be easily transform to business process execution languages. So far, there exists a working draft of the Business Process Modelling Notation published by BPMI [BPMN2002].

Modelling of business processes is the first but not the only step to make them useful for organisations. Some people believe that the next step, that is business process execution is even more important for organisations than modelling. Such statement can trigger a lot of discussion, however, it must admit that failure of many business reengineering projects in the last decade were connected with the lack of implementation of very well documented business processes.

In order to avoid such situations many business process management systems (BPM systems) have been developed. In a BPM system a business process is instantiated and executed according to its definition. One

of the main goals of a BPM system is to assure that individual activities of the process will be executed in appropriate order, by right performers and on proper data. Since a performer can be a human, it is useful to visualise information on process execution. Such visualisation allows performers for better understanding the process history (what was done before, by whom, what were the recommendations, what were the time constraints), presence (what its current state is, what are the requirements for the current activity) and future (who will continue the process, what are potential consequences of current decisions). Simply saying, process instance becomes contextualised and makes the performer's knowledge more comprehensive what, in turn, should positively impact productivity. Moreover, in our opinion, process instance visualisation should be specified in a standard notation that is coherent with notation defined for process modelling. It can increase the readability of the notation and make the process of learning simpler. On the other hand, in order not to increase the number of notations, we believe that notation for process instance visualisation should rather extend existing business modelling notation than defining a new one.

In this paper, we suggest how BPMN can be extended of visualisation of process instances. In the first part, based on our experience and suggestions from our clients, we specify requirements for process instance visualisation. Since these requirements also postulate advanced time management, we define process as well as activity instance behavioural model. These models use the time management concept proposed in [Eder1997], [Eder1999] and [Eder2001]. In the second part, the requirements and the behavioural models are then used to define an extension of BPMN of process instance visualisation. Since this extended notation is used in the ICONS project [ICONS D01], in the last part, we describe the primary results of the prototype implementation integrated within the ICONS knowledge management platform. Finally, we provide information on possible future extension of the proposed notation.

2. Requirements for representation of process execution

For the last four years, Rodan Systems have collected carefully requirements for BPM systems, especially Workflow Management systems (WfM systems). The requirements have been influenced by needs of existing as well as potential customers having experience on process modelling and use of WfM systems. The most important requirements were the base for implementing the new functionality of our OfficeObjects® WorkFlow Management System [C+ D54.1], [Momotko2002].

Some of these requirements are connected with visualisation of process execution. The most valuable ones are described in the next sections.

Req. 1 Modelling & execution – a coherent representation

The representation of process execution should be presented in a similar notation as defined for process modelling. It should not introduce new elements to express process elements that already exist in the process modelling notation. The new elements specific for process execution should be compliant with those already defined. Such approach may increase the readability of the representation (i.e. learn once - use anytime).

Req. 2 Different visualisation of elements that have been, can be and will not be executed

From execution point of view three types of process elements can be considered: elements already executed, possible to be executed and not executed. The first type includes elements that have already been executed (e.g. activities) or are currently being executed. The second type includes elements that have not been executed yet but still is a chance to do it. The third type includes all these elements that have not been executed so far and it is sure, that they will not be executed in the future. An example is an alternative path that has not been satisfied. Representation of elements of these three types should be different. The elements of the first type should be the most visible. The elements of the third type should be the least visible or even not shown.

Req. 3 The current state of process instance

Every process instance has its own behaviour. One of the easiest and clear ways of expressing process instance behaviour is to use states. The states reflect to individual steps of process instance execution while transitions between them correspond to process execution events such as 'create', 'suspend', and 'abort'.

It is important to provide a clear and easy-to-understand mechanism to show the current state of a process instance. For humans, such mechanism is simpler and faster than analysing a set of attributes in order to predict what is going on with the process instance.

Req. 4 The current state of activity instance

Similarly to process instances, also activity instances have their own behaviour. Their behaviour is more complex than that for process instance. For example, it requires expressing such situation as waiting in the performer's queue. Activity instance behaviour can also be expressed by states. The states reflect to individual steps of activity instance processing while transitions between them correspond to activity processing events such as 'create', 'open', and 'abort'. This behaviour is further referred to as *operational behaviour*.

Since activities are one of the key elements of the process model, it is very important to provide a clear and easy-to-understand mechanism to show the current state of a given activity instance. Otherwise considering

multiple activity instances as well as multiple performers can cause that the process model will be too complex and not readable enough for humans.

Req. 5 Process/Activity delay indication

Beside operational behaviour, it is also needed to express that a process or activity instance is delayed (i.e. time to finish its execution already expired). Also situation when time left to finish an activity is very restricted should be expressed. This kind of behaviour is referred to as *timing behaviour*. Representing timing behaviour for process and activity instances may warn performers of incoming deadlines and help them in finding already delayed activities.

It should be underlined that two levels of timing behaviour are needed: higher one for process instances, and lower one for activity instances. The representation of timing behaviour should complete information on operational behaviour.

Req. 6 Activity criticality indication

It is also important to give information whether a delayed activity instance also delays the whole process. For example, the phase of writing end-user documentation can be delayed but does not delay the whole project because it can be done in parallel to the integration testing phase. On the other hand, since the implementation phase is critical for the whole project, its delay causes a delay of the whole project.

Since, in some way, this indicator is related to the criticality of activities in the context of the whole process, this kind of behaviour is referred to as *criticality behaviour*.

Req. 7 Multiple activity performers

Generally, an activity can be executed by one or more performers. Some BPM systems avoid situation when there is more than one performer, however it seems that such approach may reduce their ability to express different types of business processes. An example of such situation is the process of CV updating that needs to be done by all the employees. It is not possible to assume a priori, during process definition, the fixed number of employees and express activity of CV updating as appropriate number of individual activities. It needs to be expressed as one activity that is performed by all employees.

Information on multiple performers should also be presented in one place together with information on a given activity. It should be possible to switch the context of the activity to read information pertaining to individual performers.

Req. 8 Multiple activity instantiation

An activity that belongs to a process can be instantiated more than once. This occurs if there is a loop that includes the activity or the activity is executed by more than one performer. On the process diagram, all activity instances should be presented together. Different localisation of activity instances increases the diagram complexity and reduces its readability. It should be also possible to switch the context of the activity to read information pertaining to individual activity instances. The presented activity instances are ordered by time of their creation.

Req. 9 Detailed information on activity instance

For every process instance it should be possible to show the values of its attributes as well as history of behaviour. Such information is needed for the 'second view', that is when we investigate the detailed information on activities (e.g. what attributes have been set, how long the activity lasted, what were the parameters of application called within activity).

Req. 10 Loops

In some real business processes there is a need for repeating some activities. In the case when this repetition concerns more than one activity and depends on some conditions, it can be expressed by loops. Since activities and transitions belonged to a loop may be executed more than once, it is a need to express their multiple instantiations.

The mechanism to express multiple instantiation must be simple and easy-to-understand. Sometimes it is not necessary to give a very precise information on multiple instantiation and information on the number of instances in sufficient.

Req. 11 Exceptions as separate layer

Usually, modelling of a business process starts from definition of the case (path) which is the most useful. Exceptional situations are rather modelled later. It seems that the reason comes from the need to present business process in a simple and readable way. Usually, the definition of exceptional situations increases the complexity of the process model and makes it less readable.

Similar situation is for process execution. It is suggested to place execution of the exceptional situations on a separate layer. It should be possible to hide/show this layer from the whole representation.

The process and activity instances behaviour is discussed in more detail in the next section.

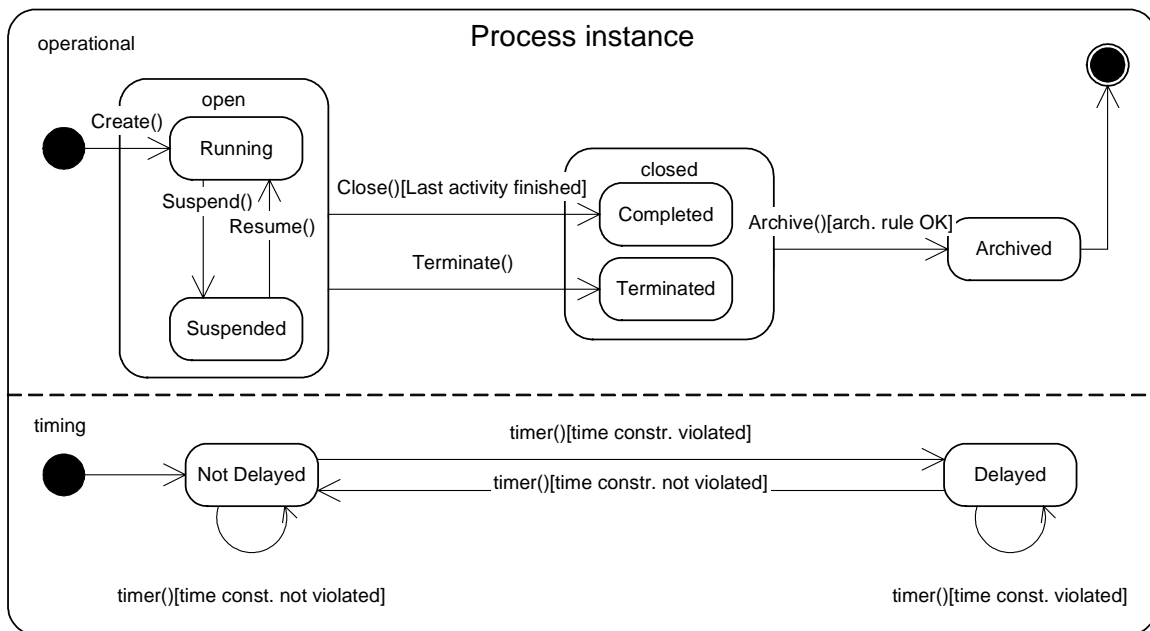


Figure 1: Process instance behavioural model

3. Process Instance/Activity behaviour

As was stated earlier process and activity behaviour is one of the key elements of process execution representation. Behaviour for these process entities is described in most standards for business processes such as [BPML2002] and [WfMC-TC-1011]. Unfortunately, none of them includes information on timing and criticality behaviour. To order the mentioned behaviour we decided to provide an extended behavioural model for process instance and activity instance. The timing and criticality behaviour has been defined on the base of work on time management described in [Eder1997], [Eder2002] and recently extended in [Momotko2003].

3.1 Process instance behavioural model

For a process instance two types of behaviour can be defined: operational behaviour related to business process execution operations, and timing behaviour related to possible process delays. These kinds of behaviour do not depend (directly) on each other. The operational behaviour presented in this paper is an example of such behaviour defined in OfficeObjects® Workflow.

For **operational behaviour** when a process instance is created, it is automatically set in the Open.Running state. This instance can be suspend – in such case it changes its state to Open.Suspend. From the latter state, the process instance can be resumed and change its state again to the Open.Running state. If all the activity instances of the process have been finished (i.e. changed their state to Closed.*¹ - see next section) the process changes its state

from Open.* to Closed.Completed. Anytime, when the process is in the Open.* state it can be terminated. In such case it changes its state to Closed.Terminated. If the rule for archiving processes is satisfied, the process changes its state from Closed.* state to Archived.

For **timing behaviour** when a process instance is created, it is automatically set to the NotDelayed state. At this time all estimated time constraints (i.e. durations and deadlines) are re-calculated. An example of such estimation algorithm is given in [Eder2002]. If during process execution one of the estimated deadlines is violated, the whole process can be delayed. In this case the process changes its state to PossibleToDelay. If in this state the process owner adjust the activity deadlines and after calculation none of their deadlines is violated, the process instance changes its state to NotDelayed. The meaning of the states is summarised below:

- Open.Running - the process instance has started execution one or more its activities.
- Open.Suspended - the process instance is quiescent; no further activities are started until it is resumed
- Closed.Completed - all activity instances belong to the process instance has achieved the Closed.* state.
- Closed.Terminated - the execution of the process has been stopped due to error or user request.
- Archived - the process instance has been placed in an indefinite archive state.
- Not Delayed - The process instance deadline (i.e. given explicitly by the process owner or implicitly by time deadline calculation) is still satisfied.
- Delayed - The process instance deadline (i.e. given explicitly by the process owner or implicitly by time deadline calculation) has already expired.

¹ The arrow means any valid state of this super-state.

- Waiting.Suspended - the pre-condition for the activity instance has not been satisfied so far.
- Open.Running - the workitem has been taken by the performer and is currently executing.
- Open. Not Running - the workitem has been taken by the performer but is not currently executing.
- Closed.Completed - the activity has been finished.
- Closed.Terminated - the activity has been stopped (abnormally) due to error or user request.
- Archived - the activity instance has been placed in an indefinite archive state.
- Not Delayed - the activity instance deadline and duration (i.e. given explicitly by the process owner or implicitly by time deadline calculation) are still satisfied.
- Delayed - the activity instance deadline or duration (i.e. given explicitly by the process owner or implicitly by time deadline calculation) has already expired.
- Not delays process - execution of the activity instance does not influent deadline for the process.
- Delays process - execution of the activity instance influents deadline for the process and delays it.

4. An extension of BPMN

On the basis of the specified requirements and the behavioural models we describe how these requirements can be satisfied and propose their graphical representation.

4.1 Proces instance current state

The current process instance state is represented via colours of the process label box. This box is usually placed on the top of the model.

State	Colour
Open.Running	Blue
Open.Suspended	Grey
Closed.Completed	Green
Closed.Terminated	Brown
Archived	Dark brown

Table 1: Colours used to represent the current state of process instance

4.2 Activity instance current state

To represent the current state of the activity instance colours, a circle indicator and the exclamation mark are used.

The current **operational state** of an activity instance is expressed as different colour of the activity box. Such approach enables performers to check quickly what is the current status.

State	Colour
Waiting.Suspended	Dark grey
Waiting.Waiting in the queue	Grey
Open.Running	Blue
Open.Not running	Navy
Closed.Completed	Green
Closed.Terminated	Brown
Archived	Dark brown

Table 2: Colours used to represent the current state of activity instance

In addition, two on-the-fly' calculated states are represented via colours:

- dotted black borders – the activity can be performed in one of the next steps of the process instance execution.
- light grey borders – the activity will not be executed in this process.

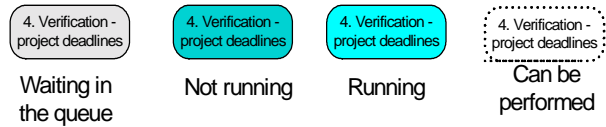


Figure 3: Examples of representation for activity instance states

The current **timing state** of activity instance is represented by a coloured circle indicator. It is also called a delay indicator since it represents information whether the activity is delayed. It is placed in the upper-right corner of the activity instance box. Basically, if the activity is not delayed the indicator is not shown. There is one exception, if the activity delays process but is not delayed itself, an indicator with the same colour as the activity is displayed with the black exclamation mark inside.

State	Colour
Time.NotDelayed	Not represented
Time.Possible to be delayed	Yellow
Time.Delayed	Red
Time.Delayed and 3 notification sent	Black

Table 3: Colours used to represent delay indicator

In addition two on the fly calculated states are represented via the delay indicator:

- Possible to be delayed – yellow – time to perform a given activity instance is very short (what 'very short' means depends on concrete application, for example it can be set on 4 hours). It is also possible to graduate colours to show lapse of time (i.e. starting from yellow and continuing to red).
- Delayed and n notifications sent – black – after sending n notification about the activity delay, it is still delayed. It is possible to graduate the indicator colour

depending on the number notifications that have been sent so far (i.e. starting from red and continuing to black).

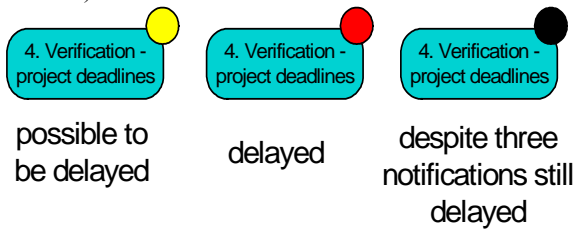


Figure 4: Representation of delay indicators

The current **criticality state** of an activity instance is expressed as the exclamation mark placed inside the delay indicator. If the activity does not delays the process instance, the mark is not shown.

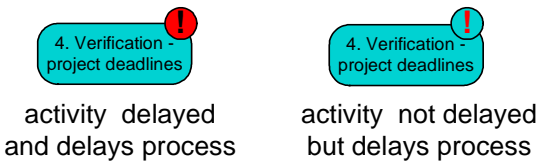


Figure 5: Representation of criticality exclamation mark

4.3 Multiple instantiation and multiple performers

The next new element for activity instance comparing to the related activity in the process definition is information on multiply activity instances. Since a process definition can include loops, some activities can be instantiated many times. A simple example is verification and then modification of a vocation application that occurs three times (our boss is very cautious in giving vacations for his employees and prefers to give less number of days that it is expected). Information that activity has been executed more than once is expressed by tabs on the bottom-left corner of the activity. Each tab has its number (i.e. the number of instantiation of this activity). If you click on a given tab, information on a given instantiation of this activity is displayed. The current tab is marked in the same colour as the rest of the activity box.

Instantiation of a process role can consist of more than one performer. It is represented by the upper-left tabs. If you click on a tab, information related to a given participant is displayed. The current tab is marked in the same colour as the rest of the activity box.

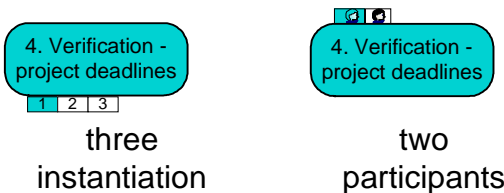


Figure 6: Representation of multiple instances and performers

4.4 Detailed information on activity

For every activity instance it is possible to display information on the activity detail. This information is presented as a text annotation. Individual attributes of activity instance are listed one after another. For every attribute its name and value is displayed.

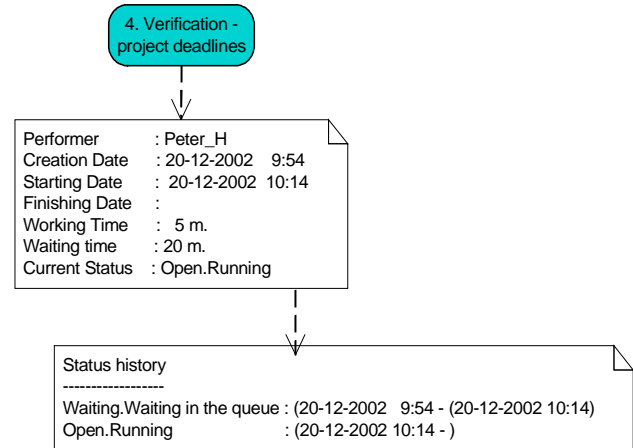


Figure 7: Representation of an activity information

Together with detailed information on activity instance also information on its status history is given. It is included in the text annotation field. The historical states are listed in chronological order. For every state information when it was entered and left is given.

4.5 Extended control flow notation

In general the representation of control flow is very similar to the process definition. The differences are the following:

- transitions and other control flow elements are drawn in light grey colour if it is sure that they will not be instantiated in this process instance (e.g. the application of John_B did not need to be modified, thus the Modification activity will not be performed in this process instance). This information is not expressed directly via activity instance behaviour. It is determined on-the-fly during generation of the activity instance representation.
- transitions and other control flow elements are drawn in dotted lines if it is possible that they can be instantiated in this process instance. Similarly, to the previous type of transitions, transitions of this type are also determined on the fly.
- transitions that create loops and have been instantiated more than once are labelled with number of instances. This number is displayed in brackets.

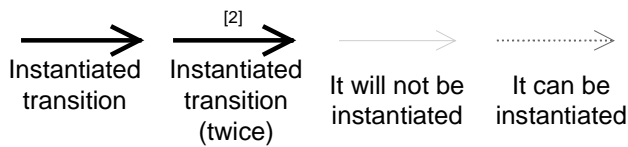


Figure 8: Representation of transitions

5. Practical results

A prototype implementation of the extended notation described in this paper is integrated within the ICONS knowledge management platform. We use it to provide information about process execution in OfficeObjects® Intelligent WorkFlow Manager [ICONS D16]. So far we observe some pros & cons of this notation.

The main advantage of this notation is its compliance with the defined notation for process modelling. Being an extension of a standard business modelling notation, that is BPMN, gives an opportunity to be independent of a particular BPM system and increases the possibility to be widely used.

From end-users point of view, the process execution model is simple to understand. Formerly, the main problem they complained about, was too complex and not easy-to-understand information on process execution (i.e. in our system we used a textual notation to show history of process instance execution). Now they are supported with human-readable information about process history, its current status and its possible continuation. In addition information on time behaviour enables performers to understand what activities do delay the process and what will happen if a given time constraint is changed.

And finally, our first impression is the following: introducing a simple but quite powerful subset of new elements to BPMN increase its functionality and does not increase its complexity.

On the contrary we understand that approach that use colours to represent states may be hardly accepted by colour-blind people. Also the representation of the loops is not straightforward. For example, it is hard to see the relations between different transitions belong to the same loop.

6. Summary

So far the prototype implementation of the extended notation confirms its practical usefulness. It really may help process performers understanding, sharing and improving knowledge on their business processes. In addition, compliance with the BPM notation, may increase the opportunity to share the process execution knowledge between the users of different BPM systems and thus between different organisations.

In the next stages we would like to provide better mechanism for loop representation. Also some elements for process modelling proposed in BPMN need to be

extended. Some of them are reported in the BPMN specification as open issues. However, a few of them such as time modelling (i.e. representation of activity duration and deadline) have not been mentioned in the specification.

References

- [BPML2002] Business Process Management Initiative; Business Process Modelling Language, Nov 2002.
- [BPMN2002] Business Process Management Initiative; Business Process Modelling Notation; working draft, version 0.9, Nov 2002.
- [C+ D54.1] The IST-1999-20162 Component+ Consorcium, Pilot projects - Architectural description, Oct 2002.
- [Eder1997] Eder, J.; Pozewaunig, H., Liebhart, W., ePERT: Extending PERT for Workflow Management Systems, 1st East-European Conference on Advances in Databases and Information Systems (ADBIS'97), 1997.
- [Eder1999] Eder, J.; Panagos, E., Pozewaunig, H., Rabinovich, M., Time Management in Workflow Systems, 3rd International Conference on Business Information System (BIS'99), p. 265-280, 1999.
- [Eder2001] Eder, J., Paganos, E., Managing Time in Workflow Systems, in Workflow Handbook 2001, Layna Fischer (Ed.), Future Strategies Inc., 2001, USA.
- [Eshuis2001] R. Eshuis and R. Wieringa. A real-time execution semantics for UML activity diagrams. In H. Hussmann, editor, Proc. Fundamental Aspects of Software Engineering (FASE), LNCS 2029, pages 76-90, Springer, 2001
- [ICONS D01] IST-2001-32429 ICONS Consortium, Research Base for the ICONS Project, April 2002, www.icons.rodan.pl.
- [ICONS D16] IST-2001-32429 ICONS Consortium, Specification of the ICONS Architecture, Jan 2003.
- [Momotko2003] Implementation of advanced time management in Workflow Systems, paper submitted to 7th East-European Conference on Advances in Databases and Information Systems, 2003, Dresden, Germany.
- [Momotko2002] Dynamic change of Workflow Participant Assignment, 6th East-European Conference on Advances in Databases and Information Systems, Vol. 2, Sep 8-11, 2002, Bratislava, Slovakia.
- [WfMC-TC-1011] Workflow Management Coalition, Workflow terminology & glossary, WfMC-TC-1011 issue 3.0, Feb 1999.
- [WfMC-TC-1025] Workflow Management Coalition, Workflow standard, Workflow process definition language – XML process definition language, WfMC-TC-1025 draft 0.03a, May 2001.