

# Visualisation of (Distributed) 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 on 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). In this paper we postulate to put equal stress on visualisation of process execution (also distributed or cross-organisational) allowing process performers to understand the process history, its current state and possible future execution. In our opinion, putting activity of interest in its visualised context makes the user knowledge more comprehensive and, consequently, increases productivity. We also 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 from the process definition level on the process execution level. The prototype implementation is integrated within the ICONS knowledge management platform.

## 1. Introduction

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

Despite such a huge effort spending on defining business processes there was no standard, flexible and widely accepted modelling 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 such a standard business process modelling notation. So far, there exists a working draft of the Business Process Modelling Notation published by BPMI [2].

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). Also for distributed or cross-organisational process that are executed in the Web environment (e.g. using Wf-XML operations over HTTP [14]), such visualisation may help performers to monitor/control the progress of work in the other organisations/workflow engines and to react early when an exceptional situation occurs. Simply saying, process instance becomes contextualised and makes the

---

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

performer's knowledge more comprehensive what, in turn, should positively impact productivity.

Moreover, in our opinion, process instance visualisation should be specified as an extension of a standard notation that is used for process modelling. Such approach may increase the readability of the notation and makes the process of learning simpler.

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 extend the process and activity instance behavioural models defined in [1, 13]. These models use the time management concept proposed in [4, 5] and recently extended in [9]. In the second part, the requirements and the behavioural models are then used to define a BPMN extension of process instance visualisation. In the last part we describe the primary results of the prototype implementation of this notation within the ICONS knowledge management platform and OfficeObjects® WorkFlow system. Finally, we provide information on possible future extension of the proposed notation.

## **2. Requirements for visualisation of distributed 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 system [6, 10]. Some of these requirements are connected with visualisation of both inter and cross-organisational process execution. The most valuable ones are described in the next sections.

### **Req. 1 Modelling & execution – a coherent representation**

The representation of process execution for both types of processes 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 Process instance information – visibility levels**

Especially for business processes that cross different organisations or distributed departments/branch offices

within an organisation, it is needed not to expose outside the whole detailed information on process execution. Instead, it should be possible to check the basic process execution parameters such as progress of work, the current operational and delay status (also reported in [3]), and other process specific information.

This process specific information may be valuable in the case, when information on the current operational process state is too 'workflow oriented' and should be translated or mapped into a simpler and more human-readable form. For instance, a customer would like to know whether its order is still being verified, or maybe it has already been accepted and now is executed. It is likely, that this information is not available directly from history of process execution (e.g. three successive activities represent verification). The mentioned process parameters correspond to the workflow Quality of Service dimensions such as time, cost defined in [11].

### **Req. 3 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. 4 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 simple graphical 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. 5 The current state of activity instance**

Activity behaviour is more complex than behaviour defined 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 refers to individual steps of activity instance processing while transitions between them

correspond to activity processing events such as 'create', 'open', 'close', and 'abort'. This behaviour is further referred to as *operational behaviour*.

#### **Req. 6 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. 7 Activity criticality indication**

It is also important to give information whether a delayed activity instance also delays the whole process. For instance, the phase of writing end-user documentation can be delayed but does not delay the whole project because it may be done in parallel to the acceptance 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. In the time management technique proposed in [4], such situation may occur when an explicit deadline for a given activity has been set.

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. 8 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. 9 Multiple activity instantiation**

An activity that belongs to a process can be instantiated more than once. It 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 reduces its readability (i.e. number of elements placed in the diagram increases and it is not possible to see that all these activity instances instantiated one activity). It should be also possible to switch the context of the activity to read information pertaining to individual activity instances. The presented activity instances should be ordered by time of their creation.

#### **Req. 10 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. 11 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 readable. Sometimes it is not necessary to give a very precise information on multiple instantiation and information on the number of instances is sufficient.

#### **Req. 12 Exceptions as separate layer**

Usually, modelling of a business process starts from definition of the cases (paths) which are the most useful (i.e. the most often used). 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.

### **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 [1, 12].

Unfortunately, none of them includes information on timing and criticality behaviour. In the next sub-sections we extend the existing behavioural models of timing and criticality behaviour.

### 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.

Within the **operational behaviour** we defined the following states:

- Open.Running - process instance has started execution one or more its activities.
- Open.Suspended - process instance is quiescent; no further activities are started until it is resumed
- Closed.Completed - all activity instances belong to the process instance has moved to the Closed.\*<sup>1</sup> state.
- Closed.Terminated – process execution has been stopped due to error or user request.
- Archived - the process instance has been placed in an indefinite archive state.

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 [4]. If during process execution one of the estimated deadlines-worst case is violated, the whole process changes its state to Delayed. Otherwise, if one of the estimated deadlines-best case is violated and all estimated deadlines-worst case are not violated, the process changes its state to PossibleToDelay. If in the mentioned states 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.

- Not Delayed - process instance deadline (i.e. given explicitly by the process owner or implicitly by time deadline calculation) is still satisfied.
- Delayed – even skipping optional activities and selection of shortest alternatives do not assure that the process instance will be executed on time. In this case further execution of process activities must be shorter than expected.
- PossibleToDelay – in order to finish process instance on time either shorter alternatives must be chosen or optional activities have to be skipped.

### 3.2 Activity instance behavioural model

For an activity instance three types of behaviour can be defined: operational behaviour related to activity processing operations, timing behaviour related to possible activity delays and criticality behaviour indicates whether a given activity instance delays the whole process. These kinds of behaviour do not depend (directly) on each other.

Within **operational behaviour** we defined the following states:

- NotOpen.Waiting in the queue - the pre-condition for a given activity instance has been satisfied, and the instance was assigned to the performer and started successfully. However, so far, the performer has not opened the work item related to this activity.
- NotOpen.Suspended - pre-condition for the activity instance has not been satisfied.
- Open.Running - work item has been taken by the performer and is currently executing.
- Open. Not Running - workitem has been taken by the performer but is not currently executing.
- Closed.Completed - the activity has been finished.
- Closed.Terminated - activity has been stopped (abnormally) due to error or performer/process owner request.
- Archived - the activity instance has been placed in an indefinite archive state.

For **timing behaviour** when the activity instance is created, it is automatically set to the NotDelayed state. If for this activity an explicit duration or deadline has been set, it is automatically assigned to it. If during process execution the explicitly set duration or deadline has been violated, the activity changes its state to Delayed. If in this state the process owner adjust the activity duration or deadline and none of them is not violated, the activity instance changes its state to NotDelayed. The meaning of the states is summarised below.

- Not Delayed - activity instance deadline and duration (i.e. given explicitly by the process owner or implicitly by time deadline calculation) are still satisfied.
- Delayed - activity instance deadline or duration (i.e. given explicitly by the process owner or implicitly by time deadline calculation) has already expired.

For **criticality behaviour**, when the activity is created, it is automatically set to NotDelaysProcess state. Also estimation for duration and deadline is calculated. If during activity execution one of these implicit time constraints is violated, the activity delays process and changes its state to DelaysProcess. If in this state the process owner adjust the activity duration or deadline and after re-calculation of the estimated time constraints none of them is violated, the activity instance changes its state to NotDelaysProcess. The meaning of the states is summarised below.

---

<sup>1</sup> Closed.\* means every sub-state within the Closed state.

- Not delays process - the activity instance does not delay the process.
- Delays process - the activity instance delays the process.

If an explicit deadline is set for an activity, and the estimated deadline is greater than that given explicitly, the activity can be delayed and not delays the whole process.

#### 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 Process instance detailed information

In BPMN, both inter-organisational and cross-organisational (or distributed) processes are represented by a process label box and a set of Pools (and Lines). The process label box displays the basic information on the process instance such as date of creation, the process owner, and reference to the process definition. Usually, this box is placed on the top of the process instance representation. A Pool is a “swimlane” and a graphical container for partitioning set of activities from other Pools. A Line may be treated as a “sub-swimlane” within a Pool.

In an inter-organisational process, usually “white box” Pools are used. These Pools expose detailed information on its activities and sequence flow between them. In a cross-organisational process, two types of Pools are used “white box” and “black box”. If a Pool is internal for a given organisation it is represented as a “white box” one a Pool is external for the organisation, it is usually represented as a “black box” Pool with restricted details exposed. The latter Pool can be treated as a sub-process.

In a “black box” Pool, information about process QoS factors and process specific data is displayed. The main process QoS factors are: the current progress (percentage), current delay (datetime) and cost (float). Both types of information are displayed as a text annotation within the Pool. The current operational state of the sub-process is represented by different colours of the annotation text. Such approach enables performers to check quickly what is the current status of the Pool. The current timing state is represented by a coloured circle indicator. It is also called a delay indicator since it represents information whether the sub-process is delayed. It is placed in the upper-right corner of the annotation text. Basically, if the sub-process is not delayed the indicator is not shown.

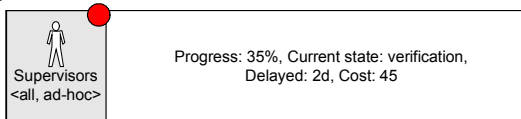


Figure 1: An example “black-box” Pool

##### 4.2 Activity instance current state

To represent the current state of the activity instance within a “white box” Pool, 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. Some examples of using colours to represent different states are given below. Such approach enables performers to check quickly what is the current status.

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

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

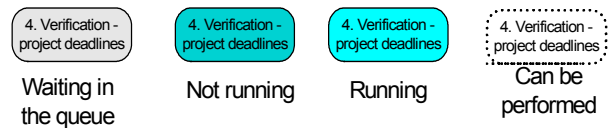


Figure 2: Representation of activity operational behaviour

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. For representing a Delayed activity the red-coloured indicator is used. 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 defined as 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).

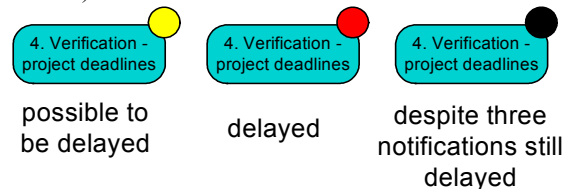


Figure 3: Representation of timing behaviour

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

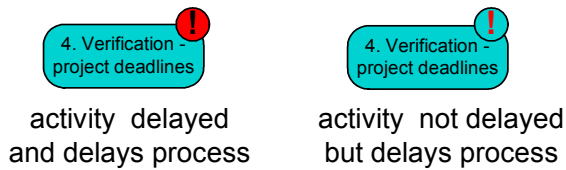


Figure 4: Representation of criticality behaviour

### 4.3 Multiple instantiation and multiple performers

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). If such situation occurs, activity is instantiated more than once (i.e. multiple activity instances). 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.

Also 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.

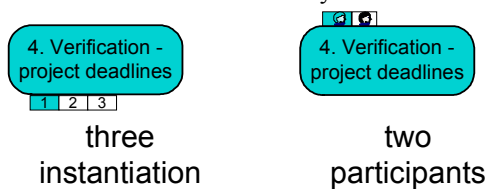


Figure 5: Representation of multiple instances and performers

### 4.4 Detailed information on activity

For every activity instance it is possible to display information on its details. 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.

Together with detailed information on activity instance also information on its status history is given. It is also 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.

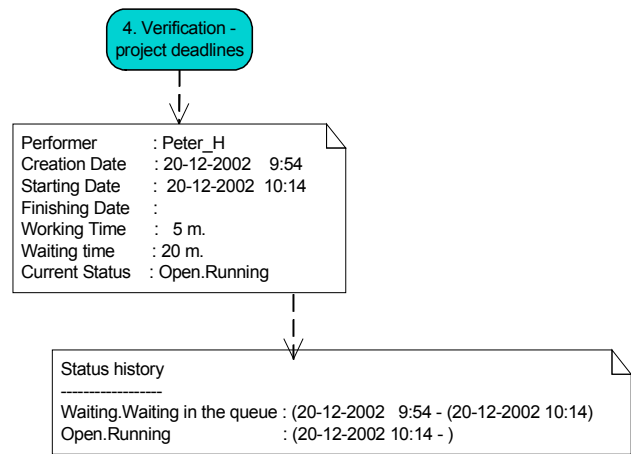


Figure 6: Representation of an activity information

### 4.5 Extended control/message flow notation

In general, the representation of control/message flow is very similar to the process definition. The differences are as follows:

- a control/message flow that has been executed is drawn with bold solid/dashed line.
- a control/message flow that has not been executed but still may be executed is drawn with a normal solid/dashed line.
- a control/message flow that will not be executed for sure is drawn with a light grey normal solid/dashed line.
- a control/message flow that has been executed more than once is labelled with the number of executions. This number is displayed in brackets.

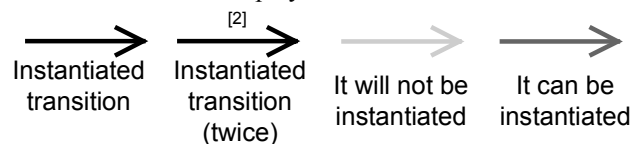
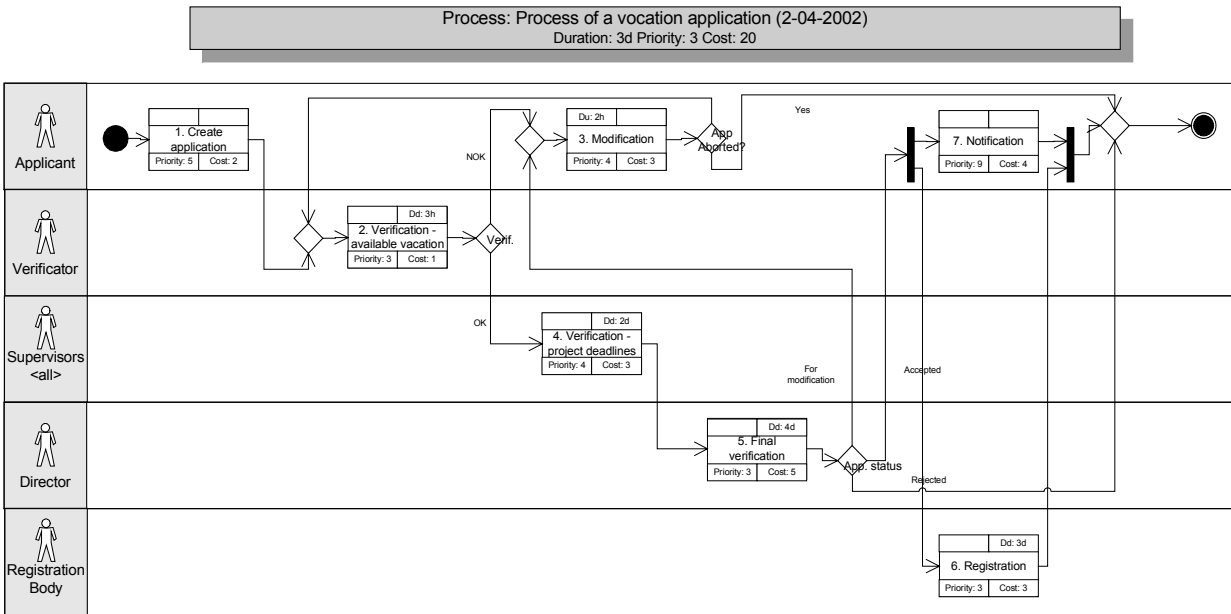


Figure 7: Representation of control flow

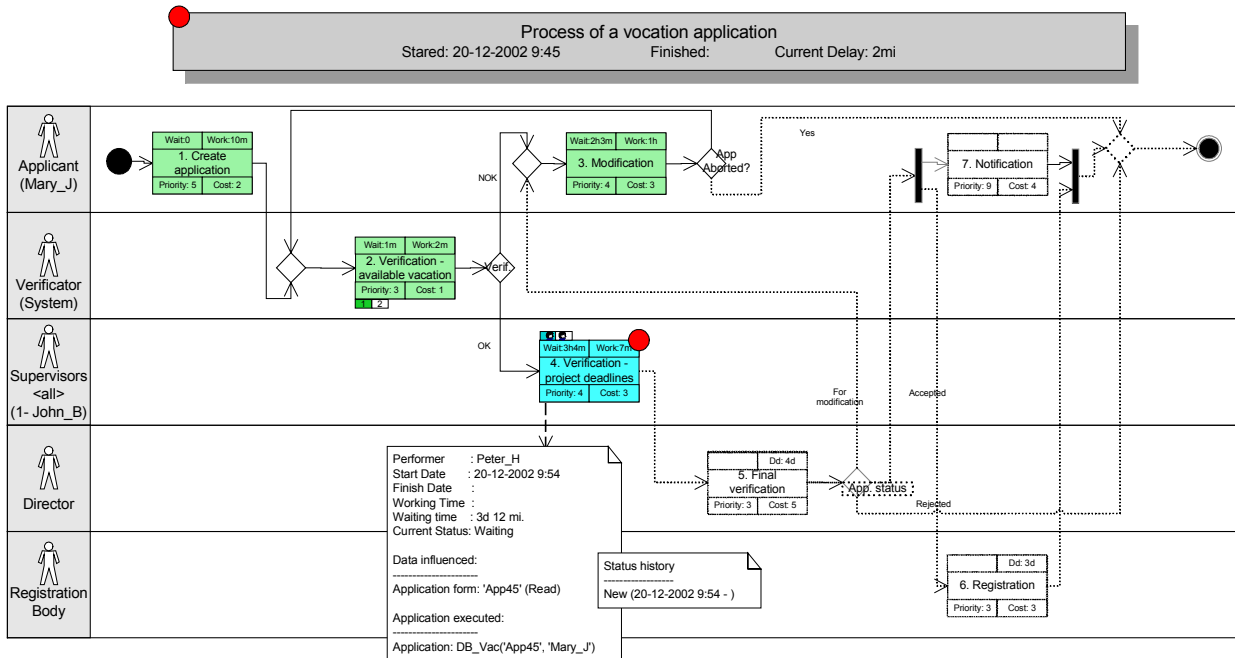
## 5. Practical results

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 [7, 8]. 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.



**Figure 8: An example of process of a vocation application**



**Figure 9: An example of process instance execution**

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 increases 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

- [1] Business Process Management Initiative, *Business Process Modelling Language*, Nov 2002.
- [2] Business Process Management Initiative, *Business Process Modelling Notation*, working draft, version 0.9, Nov 2002.
- [3] CrossFlow Consortium/GMD, *Specification of QoS Model, Architecture, and Interfaces*, D9.a report, Oct 1999.
- [4] Eder, J., E. Panagos, H. Pozewaunig, M. Rabinovich, *Time Management in Workflow Systems*, 3<sup>rd</sup> International Conference on Business Information System (BIS'99), 1999, pp 265-280.
- [5] Eder, J., E. Paganos, *Managing Time in Workflow Systems*, in *Workflow Handbook 2001*, Layna Fischer (ed.), Future Strategies Inc., USA, 2001.
- [6] The IST-1999-20162 Component+ Consortium, *Pilot projects - Architectural description*, D54.1 report, Oct 2002.
- [7] IST-2001-32429 ICONS Consortium, *Research Base for the ICONS Project*, D01 report, April 2002, [www.icons.rodan.pl](http://www.icons.rodan.pl).
- [8] IST-2001-32429 ICONS Consortium, *Specification of the ICONS Architecture*, D16 report, Feb 2003.
- [9] Momotko M., L. Zalewska, *Implementation of Advanced Time Management in Workflow Systems*, submitted to 7<sup>th</sup> East-European Conference on Advances in Databases and Information Systems, Dresden, Germany, 2003.
- [10] Momotko M., *Dynamic change of Workflow Participant Assignment*, 6<sup>th</sup> East-European Conference on Advances in Databases and Information Systems, Vol. 2, Bratislava, Slovakia, Sep 2002.
- [11] Sheth, A.; J. Cardoso, J. Miller, J. Arnold, *Modelling Quality of Service for Workflows and Web Service Processes*; submitted to VLDB Journal, 2002.
- [12] Workflow Management Coalition, *Workflow Terminology & Glossary*, WfMC-TC-1011 issues 3.0, Feb 1999.
- [13] Workflow Management Coalition, *Workflow Process Definition Interface – XML Process Definition Language*, WfMC-TC-1025, version 1.0, June 2002.

- [14] Workflow Management Coalition, *Interoperability – Wf-XML Binding*, WfMC-TC-1023, version 1.1, Nov 2001.