

Capturing procedural knowledge

Jan 2003

IST-2001-32429

ICONS

Intelligent Content Management System

www.icons.rodan.pl

Project Partners

Rodan Systems (PL)

The Polish Academy of Sciences (PL)

Centro di Ingegneria Economica e Sociale (IT)

InfoVide (PL)

SchlumbergerSema (BE)

University Paris 9 Dauphine (FR)

University of Ulster (UK)



Capturing procedural knowledge

Project name	Intelligent Content Management System
Acronym	ICONS
Workpackage	WP2
Task	T2.3
Document type	report
Title	Capturing procedural knowledge
Subtitle	
Document acronym	D09
Author(s)	Mariusz Momołko, Gołuch Marek, Gawron Wiesław, Frydrychowski Adam (Rodan), Jacek Płodzień, Ewa Stemposz, Kazimierz Subieta (ICS)
Reviewer(s)	Kazimierz Subieta (ICS), Kieran Greer, Yaxin Bi, Gonde Guo, Hui Wang (Ulster)
Accepting Location	Witold Staniszkis I:\WP2 Multi-paradigm knowledge representation\ICONS WP2 T3 D09 0104.doc
Version	1.04
Date	Jan 2003
Status	final version
Distribution	public

Jan 2003

History of changes

Date	Version	Author	Change description
28.01.03	1.04	Mariusz Momotko	Final
23.01.03	0.3	Mariusz Momotko, Gołuch Marek, Gawron Wiesław, Frydrychowski Adam	The BPMN and XPDL extensions added
15.01.03	0.2	Mariusz Momotko, Jacek Płodzień, Ewa Stemposz, Kazimierz Subieta	The conceptual models added
3.01.03	0.1	Mariusz Momotko	extended abstract creation

Executive summary

This report provides the comprehensive framework for the procedural knowledge representation in the ICONS architecture. We believe that this representation should be specified at two, equally important from the knowledge management standpoint, levels. They are **process definition level**, pertaining to the ICONS knowledge schema, and **process instance level**, pertaining to the ICONS knowledge base. The detailed **conceptual models** of both levels are presented. The models introduce all intrinsic elements and mechanisms that are indispensable in the advanced workflow management engine to be incorporated into the ICONS platform. The models constitute a sound base for the ICONS workflow management engine development.

Additionally, the models are the starting points for other representations. The **deployment model** is for humans and targets aspects related to process development and monitoring. Therefore, the visual notation is proposed to stimulate communication, education and verifiability. The need for processes' visualization on the definition level has been widely accepted and addressed (e.g. recently published Business Process Modeling Notation - BPMN). We postulate, however, to put the equal stress on processes' visualization on the instance level. It allows processes' performers for better understanding the process's history (what was done before, by whom, what were the recommendations, what were the time constraints), presence (what its current state, what are the requirements for the current activity) and future (who will continue the process, what are potential consequences of current decisions). Simply saying, the process instance becomes contextualised and make the performer's knowledge more comprehensive what, in turn, should positively impact productivity.

On the contrary, **the interoperability model** is for distributed machines. It facilitates both process definition and process continuation even outside scope of the originating organization and workflow engine. Using the proposed approach complex interactions of B2B and B2C types can be defined and implemented. As the critical point with respect to process definition and interoperability are standards the compliance with the most promising standards has been assured (i.e. XML Process Definition language [WfMC-TC-1025], Interoperability Abstract specification [WfMC-TC-1012] and Wf-XML binding specification [WfMC-TC-1023]).

Besides the thorough road map of models, languages and notations this report addresses in depth some areas of workflow management that are still waiting for innovative and effective solutions. We especially mean extensions of existing notations with process instance elements, intelligent workflow assignment, advanced routing and time modeling as well as monitoring.

Table of contents

History of changes.....	3
Executive summary	4
Table of contents	5
List of figures	6
List of tables	6
1. Introduction	7
1.1 Objectives.....	7
1.2 Scope	7
1.3 Relations to Other Documents.....	7
1.4 Intended Audience.....	7
1.5 Usage Guidelines.....	7
1.6 Notation Conventions.....	7
2. Road map for procedural knowledge representation.....	8
3. Workflow conceptual model	9
3.1 Introduction	9
3.2 Business process.....	12
3.3 Process definition	12
3.4 Definition Attribute	13
3.5 ECA rule type.....	13
3.6 Activity.....	13
3.7 Route Activity.....	14
3.8 Transition.....	16
3.9 WorkFlow Participant.....	17
3.10 Application	18
3.11 Data Type	19
3.12 Data Container Type.....	19
3.13 Process instance.....	19
3.14 ProcInst state	21
3.15 Instance Attribute	21
3.16 ECA rule.....	21
3.17 Activity instance.....	21
3.18 ActInst state	23
3.19 Transition instance.....	23
3.20 Performer.....	24
3.21 Application	24
3.22 Object	24
3.23 Object	24
3.24 Data Container.....	25
4. Extended BPMN specification	26
4.1 Process definition notation	26
4.2 Process instance notation.....	27
5. Extended XPDL specification	31
5.1 Performer Relationship.....	31
5.2 Pre & Post conditions	31
5.3 Control Flow Conditions	31
6. Summary	32
Bibliography.....	33
External references	33
ICONS references.....	33
Dictionary.....	34

List of figures

Figure 3.1. The workflow conceptual model.....	11
Figure 3.2. A decision example	15
Figure 3.3. A merge example	15
Figure 3.4. An example of AND-SPLIT and AND-JOIN operations.....	15
Figure 3.5. Ordering example.....	16
Figure 3.6. Process instance behavioural model.....	20
Figure 3.7. Activity instance behavioural model.....	22
Figure 4.2. Graphical representations of XOR-SPLIT operation	26
Figure 4.3. The graphical representation of the XOR_JOIN operation.....	26
Figure 4.4. The graphical representation of AND-SPLIT operation	27
Figure 4.5. The graphical representation of the AND-JOIN operation	27
Figure 4.6. The graphical representation of different types of transitions.....	27
Figure 4.7. Representation of different activity instance states	28
Figure 4.8. Graphical representation of different types of delay indicators.....	29
Figure 4.9. Graphical representation of different types of criticality exclamation mark	29
Figure 4.10. Representation of multiple instances and performers.....	29
Figure 4.11. Representation of an activity attribute description	29
Figure 4.12. Representation of an activity status history.....	30

List of tables

Table 2.1. The road map of procedural knowledge representations	8
Table 3.1. Process instance state description.....	20
Table 3.2. Process instance state description.....	23
Table 4.1. Colors used to represent the current state of the activity instance	28
Table 4.2. Colors used to represent delay indicator.....	28

1. Introduction

1.1 Objectives

This report provides a framework for representation of procedural knowledge in the ICONS project. This framework includes detailed description of the conceptual model together with other procedural knowledge models to express interoperability, deployment and internal representation.

1.2 Scope

The scope of this report covers the entire representation of procedural knowledge that is used in the ICONS project. This covers: conceptual, deployment and interoperability models and languages. Additionally stress was put on some specific aspects of workflow management: extensions of existing notations with process instance elements, intelligent workflow assignment, advanced routing and time modeling as well as monitoring.

1.3 Relations to Other Documents

This report extends our initial vision of intelligent workflow management given in [ICONS D06]. Architectural design of the corresponding workflow engine is amply presented in [ICONS D16].

1.4 Intended Audience

The intended audience comprises all members of the ICONS project consortium that want to be familiar with basic concepts on the ICONS procedural knowledge representation. In addition, this report enables the ICONS developers to learn the syntax and semantics languages to represent the procedural knowledge. The deployment model will be also reused as a part of the application design ICONS methodology [ICONS D25].

1.5 Usage Guidelines

The chapters' order drives a potential reader from concepts up to the more advanced features. From some more visual oriented users it may be helpful to start reading from the chapter 2.

1.6 Notation Conventions

In order to assure consistency and readability of the procedural knowledge representation, the UML notation has been used, especially the Class and State chart diagrams [Rumbaugh1999].

2. Road map for procedural knowledge representation

This report provides the comprehensive framework for the procedural knowledge representation in the ICONS architecture. We believe that this representation should be specified at two, equally important from the knowledge management standpoint, levels. They are **process definition level**, pertaining to the ICONS knowledge schema, and **process instance level**, pertaining to the ICONS knowledge base. The detailed **conceptual models** of both levels are presented. The models introduce all intrinsic elements and mechanisms that are indispensable in the advanced workflow management engine to be incorporated into the ICONS platform. The models constitute a sound base for the ICONS workflow management engine development. The conceptual models are described in chapter 3.

Additionally, the models are the starting points for other representations. The **deployment model** is for humans and targets aspects related to process development and monitoring. Therefore, the visual notation is proposed to stimulate communication, education and verifiability. The need for processes' visualization on the definition level has been widely accepted and addressed (e.g. recently published a working draft of Business Process Modeling Notation – BPMN [BPMN2002]). We postulate, however, to put the equal stress on processes' visualization on the instance level. It allows processes' performers for better understanding the process's history (what was done before, by whom, what were the recommendations, what were the time constraints), presence (what its current state, what are the requirements for the current activity) and future (who will continue the process, what are potential consequences of current decisions). Simply saying, the process instance becomes contextualised and make the performer's knowledge more comprehensive what, in turn, should positively impact productivity. At the earlier stage, we were working on our own notation to model business processes. However, after publishing the BPMN working draft, we decided to adopt it in the ICONS for our purpose. The BPMN extensions introduced in the ICONS project have been described in chapter 4.

On the contrary, **the interoperability model** is for distributed machines. It facilitates both process definition and process continuation even outside scope of the originating organization and workflow engine. Using the proposed approach complex interactions of B2B and B2C types can be defined and implemented. As the critical point with respect to process definition and interoperability are standards the compliance with the most promising standards has been assured (i.e. XML Process Definition language [WfMC-TC-1025], Interoperability Abstract specification [WfMC-TC-1012] and Wf-XML binding specification [WfMC-TC-1023]). Thanks to flexibility of XPD L we only needed to precise the meaning of several tags in XPD L schema. These propositions have been described in chapter 5.

The proposed procedural knowledge levels and models have been summarized in Table 2.1. The models and notations that are defined/extended in this report are marked in gray color. The extended versions of the languages/notations are marked with the '+' sign as their suffix.

Type of knowledge Level	Conceptual model	Deployment methodology	Interoperability
L1: general process instance	process instance concept	BPMN+ specification	Wf-XML specification
L3: general process definition	process definition concept	BPMN specification	XPD L+ specification

Table 2.1. The road map of procedural knowledge representations

3. Workflow conceptual model

This chapter defines the workflow conceptual model and describes the basic concept related to business processes and workflow management systems (WfM systems). Basic concepts were taken from the WfMC terminology and Glossary standard [WfMC-TC-1011]. The concept of ECA rules was taken from [Ceri1996].

For each class from the conceptual model, its definition and specification of attributes are given. For some classes also their usage is described. In addition, for process instance and activity instance their behavior is described.

3.1 Introduction

A **business process** represents a business behavior. It possesses a set of basic attributes such as name, identifier (ID), description (Desc) and date of creation. In addition, a process definition can possess other attributes that will be defined in the future. They are represented as **definition attributes**. A definition attribute has name, value and type. The type is a simple type such as text, number or date. Since process definition can change in order to satisfy new requirements or to improve its efficiency, it can have many **process definition**. Each process definition has its own attributes: the number of the version, time when it is valid (ValidFrom and ValidTo), and the date of creation. In addition, it is possible to define duration for a version, which means the maximal time between the creation date and the finish date for all instances of this definition. For every version also its owner is defined. The version owner is a user who is responsible for this definition and all its instances.

With a process three different types of data are defined:

- **application data** – data that is application specific and not accessible by the workflow management. In the workflow conceptual model this data is represented as data type (meta-data) or objects (i.e. data type instantiation). Data type and objects are used by the activity performers or applications assigned to activities.
- **relevant data** - data that is used by a Workflow Management System to determine the state transitions of a workflow instance, for example within pre- and post-conditions, transition conditions or workflow participant assignment. In the conceptual model the relevant data are represented by the data type container and data container.
- **control data** –Data that is managed by the WfM system. Such data is internal to the WfM system and is not normally accessible to applications. In the conceptual model control data are represented by the all other model classes.

For a process definition, a **data container type** is specified. This container represents either specific process attributes or references to data type. This data can be used in control flow conditions, parameters for activity applications and workflow participant assignments.

A process definition consists of **activities**, that is logical steps of the process. An activity can be either an **atomic activity**, a compound activity or a route activity. A **compound activity** is represented as another process definition. It is used to express standard sets of activities that are common for many processes in a given organization. A **route activity** represents one of two control flow operations: split or join. There are two types of split operation: **AND-Split** and **XOR-Split**. The former type means that after finishing a given activity, control flow will be passed to all following activities for which transition condition is satisfied. The latter type means that control flow will be passed to only one following activity for which transition condition is satisfied. If transition condition of more than one activities is satisfied, only one of them will be arbitrary selected. Similarly, JOIN operation has also two types **AND-JOIN** and **XOR-JOIN**. The former type means that the next activity will start only if all its previous activities are finished. The latter type means that the next activity will start, if one of the previous activity is finished. A more complex operation can be built as a cascade of the mentioned basic split/join operations.

An activity possesses such basic attributes as name, identifier (ID), description. For every activity also its duration and deadline can be defined. Similarly to the process definition, activity duration means the maximal time between the activity start date and finish date. Activity deadline means the maximal date (with respect to process starting date) that a given activity has to be finished. An activity also possesses cost and priority attributes. In addition, an activity can have pre and post conditions defined. If the former condition is satisfied, the activity can start. If the latter condition is satisfied, the activity can finish. An activity can possess other attributes that will be defined in the future. They are represented as definition attributes.

The order of activities is defined by transitions. A **transition** defines the relation is followed by between two activities. A transition has a condition. If the condition is satisfied, the transition is executed. Conditions are expressed in a procedural language. This language allows such conditions to include functions that operate on relevant, control, as well as audit workflow data. Also attributes from data container type can be used to build a transition condition.

For an activity three additional elements are defined: **workflow participant(s)**, **application** and **data types**, that is who, what and on what data will be performed, respectively. A workflow participant is a user or an automatic agent. There can be more than one workflow participants assigned to given activity initially. A workflow participant assignment (WPA) is expressed in WPAL procedural language. It is possible, to refers to relevant, control and audit workflow data in a WPA. Application refers to a system or software application that will be executed on behalf on a given activity. Specification of application includes its name and set of parameters. It is possible to express as application parameter an attribute belongs to the Data Container type. Also data types are assigned to activities. These types describes data type which are be influenced by a given activity (i.e. objects of such types are created/read/updated/deleted).

For process definition and its activities ECA rule types can be defined. An **ECA rule type** means that if an (E)vent occurs, and (C)ondition is satisfied, an (A)ction will be taken. In the case of WfM system events can be two types: those related to changing the state of process instance or its activities and time events. The former events are generated for every process instance/activity instance. The latter events are generated periodically by the WfM system. ECA rules can be useful to express actions that should be taken when an exception occurs, for example, when an activity instance is delayed.

All mentioned above workflow entities are instantiated during process execution. A process definition that is executed is called a **process instance**. A process instance's behavior is expressed by states and described as a state chart diagram. The history of states in which a given process instance are represented by **ProcInst state** class. Description of an individual state includes its name, DateFrom where the state was entered and DateTo when the state has been left.

A process instance possesses a set of basic attributes: identifier (ID), CreationDate, FinishDate, Priority and Duration. The current process instance state is represented by CurrentState attribute. Similarly to the process definition, process instance can possess other attributes that will be defined in the future. They are represented as **instance attributes**. An instance attribute has name, value and type. The type is a simple type such as text, number or date.

To a process instance a **data container** is defined. This container is an instance of the data container type defined for the related process definition. This container includes either attributes specific for this process instance or references for objects used in activity instances.

A process instance consists of **activity instance**. They are instances of related process activities. An activity instance possesses such basic attributes as ID, CreationDate, FinishDate, Cost and Priority. It also includes information on both time constraints: Duration and Deadline as well as real time values: Working and Waiting Time. The former information refers to the respective values of Duration and Deadline defined for process activity. However the value of deadline is calculated on the base of process instance start date and is represented as a plain date. The latter information refers to time a given instance was waiting in the performer queue to be started, and to time how long this activity instance was executed, respectively.

Similarly to the process instance, an activity instance behavior is expressed by states and represented as a state chart diagram. The history information in which states a given activity was is represented by **ActInst State** class. Description of an individual state includes its name, DateFrom where the state was entered and DateTo when the state has been left. The current state of the activity instance is represented by CurrentState attribute. The real order between activity instances is reported by **transition instances** (i.e. instances of process transitions). Only those transitions that have been satisfied are represented.

With activity instance three additional elements are related to: Performer, Application and Objects that is who, what and on what data this activity instance will be performed, respectively. A **performer** is an instance of workflow participant defined for the related process activity. Always there is only one performer for every activity instance. If more than one workflow participant has been assigned to a given process activity, it will be instantiated by the number of activity instances equals to the number of performers. A performer has identifier (login) and name. The **application** class represents the call of an application and what attributes have been passed. The **object** class represents objects of data types assigned for related process activity that are used by this activity instance.

For process instance and its activities ECA rules can be defined. An **ECA rule** is an instantiation of the related ECA rule type. This class reports when and what ECA rule has been fired.

3.2 Business process

A set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships.

3.2.1 Usage

- A business process is typically associated with operational objectives and business relationships, for example an Insurance Claims Process, or Engineering Development Process. A process may be wholly contained within a single organizational unit or may span several different organizations, such as in a customer-supplier relationship.
- A business process has defined conditions triggering its initiation in each new instance (e.g. the arrival of a claim) and defined outputs at its completion.
- A business process may involve formal or relatively informal interactions between participants; its duration may also vary widely.
- A business process may consist of automated activities, capable of workflow management, and/or manual activities, which lie outside the scope of workflow management.

3.2.2 Attribute specification

Attribute	Description
Name	The unique (in the organization) name of the business process.
Description	Description of the process (text form).
Creation date	The date and time when the process has been applied in the organization.

3.3 Process definition

The representation of a business process in a form, which supports automated manipulation, such as modeling, or enactment by a workflow management system. The process definition consists of a network of activities and their relationships, criteria to indicate the start and termination of the process, and information about the individual activities, such as participants, associated IT applications and data, etc.

3.3.1 Usage

- The process definition results from work during the process definition mode. and may include both manual and workflow (automated) activities.
- The process definition may contain references to sub-processes, separately defined, which make up part of the overall process definition
- The process definition may make reference to a separate Organization or Resource Model to enable participants to be indirectly defined, for example by reference to attributes such as role or organizational position.
- The WfMC Reference Model includes an interface for the import and export of Process Definitions; this incorporates the Process Definition Meta-Model, which identifies the top level entities within the Process Definition

3.3.2 Attribute specification

Attribute Name	Attribute description
ID	The unique identifier at the level of the WfM system.
Desc	A textual description of the process.
Version	The number specifies the version of this process. Usually it consists of three numbers: major, minor and release numbers.
Creation date	The date when the process has been created (i.e. enters the <i>Open</i> state).
ValidFromDate	The date from this process is valid and available for instantiation.
ValidToDate	The date to this process is valid and available for instantiation.
Owner	The identifier of the system user that is responsible for maintaining this process and all its instances.
Cost	The average cost of for the process, usually it is calculated on the base of its process instances.
State	The current state of this process according to its behavioral model

Priority	A natural number defines the priority of the process.
Documentation	A reference to a file that contains guidelines and suggestions how this process should be performed.

3.4 Definition Attribute

A feature that can be assigned to a process or its activities. A definition attribute has name, value and type. The attribute type may be a simple one and a compound one. The former type is a standard type such as date, integer or text. The latter type may be regarded as a structure that consists of elements with simple or other compound type. It is possible to have a multi-valued attribute, that is an attribute with more than one different value. Definition attributes can be regarded as extended attributes as it is defined in XPDL [WfMC-TC-1025].

3.4.1 Usage

Definition attributes can be used for:

- WfM system specific attributes – to separate them from standard attributes that can be mapped to other systems
- attributes that may be defined in the future and not know at present
- compound attributes, which are hard to process.

3.4.2 Attribute specification

Attribute Name	Attribute description
Name	The name of the attribute
Type	The type of the attribute. The type can be a simple one or a compound one.
Value	The attribute value of type specified in the attribute <i>Type</i> .

3.5 ECA rule type

Definition of a process rule. An ECA rule type means that if an (E)vent occurs, and (C)ondition is satisfied, an (A)ction will be taken. In the case of WfM system events can be two types: those related to changing the state of process instance or its activities and time events. The former events are generated for every process instance/activity instance. The latter events are generated periodically by the WfM system. ECA rules can be useful to express actions that should be taken when an exception occurs, for example, when an activity instance is delayed.

3.5.1 Usage

- to express exceptional situations
- to express complex time constraints

3.5.2 Attribute specification

Attribute Name	Attribute description
Name	The name of the rule
Event	The event of the WfM system.
Condition	An expression based of WFC language
Action	Specification of activity that will be executed if the event occurs and the condition is satisfied.

3.6 Activity

Description of a piece of work that forms one logical step within a process. Activity can be atomic, compound or a route one. An activity is typically the smallest (i.e. atomic) unit of work which is scheduled by a workflow manager during process enactment A compound activity is usually expressed as a sub-process. A route activity is described in detail in the following section.

For every activity its performer is assigned. It can be more than one activity performer. An activity performer is represented as a workflow participant.

An activity can be executed as an application and use data according to the data types defined out of the WfM system.

3.6.1 Usage

- A process definition consists of many process activities, which are logically related in terms of their contribution to the overall realization of the business process.

3.6.2 Attribute specification

Attribute Name	Attribute description
Identifier	The successive number of activity in a given process. This information is assigned automatically by the WfM system.
Name	The activity name.
Description	Description of the activity (text form).
Duration	Time constraint defines the maximal execution time of the activity (i.e. period of time which activity spent in state <code>Open.Running</code>). Duration can be entered in standard time units: minutes(mi), hours (h), and/or days(d). For example 7d3h4mi means 7 days, 3 hours and 4 minutes. This time constraint can either be defined manually by the workflow designer or estimated on the basis of working and waiting times.
Deadline	Time constraint defines the maximal time when, with respecting to the process creation date, the activity has to be finished (i.e. changes its state to <code>Closed.*</code>). Similarly to duration, also deadline is given in standard time units.
WorkingTime	The average time that activity spent in the state <code>Open.*</code> .
WaitingTime	The average time that activity spent in the state <code>Waiting.*</code> .
Priority	Zero or a natural number. 0 means the lowest priority.
Cost	Zero or a float number. 0 means no cost for the activity.
Optional	Indicates whether this activity can be skipped in order to reduce the delay of the process.
Documentation	A reference to a file that contains guidelines and suggestions how this activity should be performed, what are its pre and post-conditions, etc.
PreCond	An activity condition that is evaluated when the activity is created. If this condition is satisfied the activity is assigned to the performer. Otherwise, the activity waits until the condition will be satisfied. Such condition can be used, for example, to express that the activity can not be executed before particular date (time constraint).
PostCond	An activity condition that is evaluated when the activity is finished (i.e. it changes its state to <code>Closed.*</code>). If it is satisfied the activity is finished. Otherwise, the activity remains in its state and appropriate exception is thrown. The performer can try to finish it later. Such condition can be used, for example, to express that the activity can not be finished by a given date or if a given attribute is empty.
WPA	Expression to assign workflow participants to a given activity. It is expresses in WPAL.

Both Pre and Post-conditions are expressed in Business Process Condition Language.

3.7 Route Activity

A control flow element that is used to express alternative and parallel execution of atomic and compound activities. There are four basic types of the route activity: XOR-SPLIT, AND-SPLIT, XOR-JOIN, AND-JOIN.

3.7.1 Alternative execution

In order to express decision in the process the **XOR-SPLIT** operation is used. To every XOR-SPLIT operation one ingoing transition and one or more outgoing transitions are connected. If the ingoing transition has been executed, in the next step, depending on the conditions defined for ongoing transitions, only one of them will be executed (i.e. that one that is satisfied). If more than one or none of them are satisfied the system throws an exception. From programming point of view, the XOR-SPLIT operation can be treated as if-then-else command.

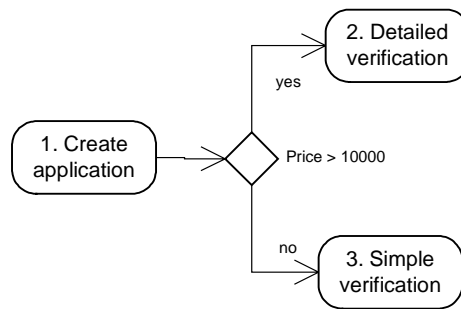


Figure 3.2. A decision example

In the example presented above, depending on the price included in the application, after finishing the A1 activity, either A2 or A3 activity will be performed. It is not possible to perform both of them or none of them. Generally, the transition condition on outgoing transitions should be mutually exclusive.

The **XOR-JOIN** operation is opposite to the XOR-SPLIT operation. It is applied to join two or more mutually exclusive alternative ingoing transitions into one ongoing transition. If more than one or none of the ingoing transition has been executed, the system throws an exception.

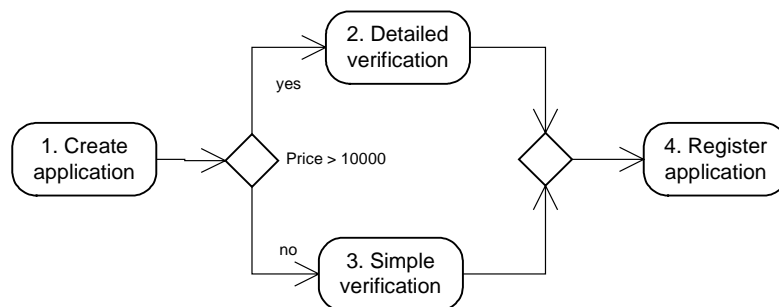


Figure 3.3. A merge example

In the example presented above, the XOR-JOIN operation merges two alternative paths. Only one of these paths is executed according to the price included in the application. After XOR-JOIN operation the A4 activity is executed.

3.7.2 Parallel execution

To fork control flow and enable different activities to be executed in parallel, the **AND-SPLIT** operation is used. To every AND-SPLIT operation there is one ingoing transition and two or more outgoing transitions assigned. If the ingoing transition has been executed, all the outgoing transitions are executed.

The **AND-JOIN** operation is the opposite to AND-SPLIT operation. It is applied to join two or more parallel ingoing transitions into one outgoing transition. To every AND-JOIN operation there are two or more ingoing transition and one outgoing transition assigned. The outgoing transition is executed only if all ingoing transition has already been executed. If not, the system is waiting for the missing transitions.

This type of join operation is represented by merge symbol with more than one ingoing transitions.

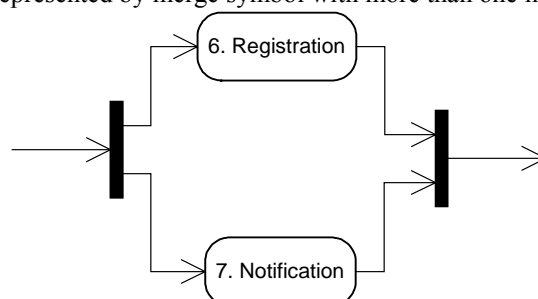


Figure 3.4. An example of AND-SPLIT and AND-JOIN operations.

In the example presented above the activity A6 and A7 can be executed in parallel since there are two outgoing transitions for the AND-SPLIT operation that have been executed. If, for example, A6 has been finished earlier than A7, the AND-JOIN operation will wait for finishing A7 (i.e. all ingoing transition have to be executed).

3.7.3 Usage

- for complex flow operations a cascade of route activities can be used

3.7.4 Attribute specification

A route activity has neither a performer nor an application and its execution has no effect on workflow relevant data or application data. For simulation purposes the following simulation data values should be assumed: Duration 0, Cost 0, WorkingTime 0, WaitingTime 0. For Priority the maximum value should be assumed.

3.8 Transition

A point during the execution of a process instance where one activity completes and the thread of control passes to another, which starts. A transition defines the order between two activities.

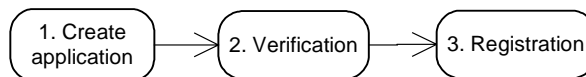


Figure 3.5. Ordering example

The activity A2 can not start before A1 has been finished. Similarly, the activity A3 will start after A2 has been finished. The ordering construct enforces sequential execution of activities.

A **transition** may be unconditional, which means that after completion the ‘from’ activity the transition is executed and then the activity ‘to’ starts. If a transition includes a transition condition, it means that the transition is executed and then the activity ‘to’ starts only if its transition condition is satisfied. A **transition condition** is also expressed in WFC language. This language allows such conditions to include functions that operate on relevant, control, as well as audit workflow data. Also attributes from data container type can be used to build a transition condition.

3.8.1 WFC language (WFCL)

In WFCL a WFC is a *function* that returns a boolean value (i.e. either true or false). A WFC is based on boolean manipulation and can consist of:

- **comparison expression**– a comparison of a data container attribute with a given value. All standard comparison attributes are supported.
- **Boolean operators** (union, intersection, and subtraction) that operate on sets of workflow participants, functions and other WFCs. The operators simplify the notation for complex WFCs,
- **WFC function** – return a boolean value on the basis of control, audit and relevant data. Functions are used to express complex relationships that cannot be represented by other elements of the WFC definition. A function can be implemented as a database procedure, a programming function (e.g. in C) or a class method (e.g. in Java). WFC functions can be reused in definitions of many WFCs,
- **Another WFC** – an already defined WFC. Instead of defining a new WFC from scratch it is possible to use already defined WFCs.

3.8.1.1 WFC grammar

```

<flow_condition> ::= <sum>
<sum> ::= <product> {<op1> <product>}
<product> ::= <expression> {<op2> <expression>}
<expression> ::= <op_NOT> <sum> |
                '(' <sum> ')' |
                <condition> |
                <WFC_function>
<op1> ::= 'OR'
<op2> ::= 'AND'
<op_NOT> ::= 'NOT'
<condition> ::= <attr_name> <op_comp> <value>
<op_comp> ::= '=', '<>', '<=', '>=', '<', '>'
<value> ::= <number> |
           <text> |
           <date>
<attr_name> ::= <text>
    
```

Symbols used:

- ‘OR’ - a boolean OR operator,
- ‘AND’ – a boolean AND operator,
- ‘NOT’ - a boolean negation operator,
- <attr_name> - the name of an attribute defined in the data Container Type,
- <WFC_function> - the name of a other WFC already defined,

Some examples of expressions in WFC:

- The status of the document must be ‘ACCEPTED’:
WfcA =Status = ‘ACCEPTED’
- The status of the document must be ‘ACCEPTED’ and the cost of acceptance is greater than 1000 USD:
WfcB = WfcA AND Acceptation_Cost > 1000

3.8.2 Attribute specification

Attribute Name	Attribute description
Identifier	The successive number of activity in a given process. This information is assigned automatically by the WfM system.
Cond	The transition condition.

3.9 WorkFlow Participant

A resource that performs the work represented by a workflow activity instance. This work is normally manifested as one or more work items assigned to the workflow participant via the worklist.

According to the WfMC’ s definition, a workflow participant is one of the following types: a resource, a resource set, a role, an organizational unit, and the system.

The WfMC’s idea of workflow participant assignment has been extended of flexible language to express complex assignments – Workflow Participant Assignment Language (WPAL) and presented in [Momotko2002]. The foundations of WPAL are described in the next section.

3.9.1 WPAL

In WPAL a WPA is a *function* that specifies (explicitly or implicitly) a set of workflow participants that will perform a given activity. A WPA is based on set manipulation and can consist of:

- **Set of workflow participants** (in particular one participant) – an explicit enumeration of workflow participants - as in the WfMC definition,
- **Set operators** (union, intersection, and subtraction) that operate on sets of workflow participants, functions and other WPAs. The operators simplify the notation for complex WPAs,
- **WPA function** – evaluates a set of workflow participants on the basis of control, audit and relevant data. Functions are used to express complex relationships that cannot be represented by other elements of the WPA definition. Function arguments are WPAs too. A function can be implemented as a database procedure, a programming function (e.g. in C) or a class method (e.g. in Java). WPA functions can be reused in definitions of many WPAs,
- **Another WPA** – an already defined WPA. Instead of defining a new WPA from scratch it is possible to use already defined WPAs. For example, a set of employees that know Java and XML languages can be defined as an intersection of two separate WPAs – one that expresses the sets of employees that know Java and one that expresses employees that know XML. Using other WPAs, it is possible both to simplify the WPA definitions as well as to reduce the cost of the WPA developing and maintenance (e.g. the cost of testing),

The set of workflow participants that are the result of a WPA evaluation is called a *WPA evaluation set*. A WPA is equipped with two additional features: a *modifier* and a *decision*. They are used to delimit the WPA evaluation set that is the result of a given WPA evaluation. A modifier, on the basis of the set of participants evaluated by a WPA, determines the number of participants that will perform a given activity. The modifier can take one from two values:

- **One** – the activity is assigned to the first accepting participant;
- **All** - the activity is assigned to all the participants returned by a WPA.

A decision, on the basis of the set of participants evaluated by WPA, determines whether the set of participants will be assigned automatically or manually. It can take one from two values:

- **Auto** – WFMS automatically evaluates appropriate WPA and assigns the participants to a given activity;
- **Ad-hoc** – As previously, but the final assignment of the returned participants to a given activity is done manually by the performer of the current activity.

To prevent errors in a WPA definition, additional features of the WPA have been introduced: *cardinality* and *default WPA*. They are also present in other WFMSs, e.g. IBM MQSeries Workflow. The cardinality describes how many participants should be included into a WPA evaluation set. A default WPA describes participants that will be assigned to the activity if the real cardinality of a given WPA evaluation set is different from the defined cardinality (e.g. if a WPA evaluation set is empty). Default WPA should be defined explicitly as a set of workflow participants.

3.9.1.1 WPAL grammar

On the basis of the new definition of WPA a *Workflow Participant Assignment Language* (WPAL) has been created. Its syntax written in BNF is as follows:

```

<wpa_def> ::= <wpa_name> '=' <wpa>
<wpa>      ::= <wpa1> { <op1> <wpa> }
<op1>     ::= '+' | '-'
<wpa1>    ::= <wpa2> { <op2> <wpa2> }
<op2>     ::= '*'
<wpa2>    ::= '(' <wpa> ')' | <set> | <function> | <wpa_name>
<set>     ::= '[' <participant> { ',' <participant> } ']'
<function> ::= <function_name> '(' <arg> { ',' <arg> } ')'
<arg>     ::= <wpa_name> | <function> | <set> | <text> | <role>

```

Symbols used:

- '+' - a union operator,
- '*' - an intersection operator,
- '-' - a subtraction operator,
- <wpa_name> - the name of a WPA,
- <function_name> - the name of a function that is already defined,
- <role> - the name of a defined role,
- <text> - a string representing some description of WPA.

Some examples of a WPA in WPAL:

- A person who is an expert in Java and XML:
WpaA = Expert('JAVA') * Expert('XML')
- A person, who is an expert in Java or Visual Basic, and knows XML (except Doe):
WpaB = (Expert('JAVA') + Expert('VB')) * Knows('XML') - ['Doe']
- Employees that earn less than 20000EUR/year and know Java and XML:
WpaC = Earn('LESS', 'EUR', 'YEAR', 20000) * WpaA
- A person who did the previous activity or the person who started the workflow:
WpaD = Participant(Prev_Activity) + Participant(Start_Activity)

3.9.2 Usage

- A workflow participant can have one or more activities assigned.

3.9.3 Attribute specification

Attribute Name	Attribute description
Name	The name of the resource
ID	The unique identifier f the resource

3.10 Application

A general term for a software program that interacts with a workflow enactment service, handling part of the processing required to support a particular activity (or activities). An application can be called with arguments. It is possible to use as an argument data from the data container type (i.e. workflow relevant data) or from workflow control data. The values of the parameters are instantiated during process execution when information on the activity instance is required.

3.10.1 Usage

- Application is a part of application functions available to the end-users.

3.10.2 Attribute specification

Attribute Name	Attribute description
Name	The name of the application
URL	The reference to the application
Parameters	The list of parameters. If a parameter from workflow control data is used, it is preceded by '\$' prefix.

3.11 Data Type

A type of application data that can be used by an application executed within an activity.

3.11.1 Usage

- A data type is a part of application data.

3.11.2 Attribute specification

A data type must at last provide information on its name and unique identifier.

3.12 Data Container Type

A type of relevant data defined for this process definition. A data container includes definition of all its attributes.

3.12.1 Usage

- A data container type specifies attributes that are relevant data.

3.12.2 Attribute type class

A single definition of type of an attribute that belongs to the data container. An attribute type provides information on its name. There are several basic types available such as text, number, date and reference.

3.13 Process instance

The representation of a single enactment of a process, or activity within a process, including its associated data. Each process instance represents a separate thread of execution which may be controlled independently and has its own internal state and externally visible identity, which may be used as a handle, for example, to record or retrieve audit data relating to the individual enactment.

3.13.1 Usage

- A process instance is created, managed and (eventually) terminated by a WfM system, in accordance with the process definition.
- A process instance can have a number of attributes assigned. Also complex and multi-valued attributes can be assigned.

3.13.2 Behavior

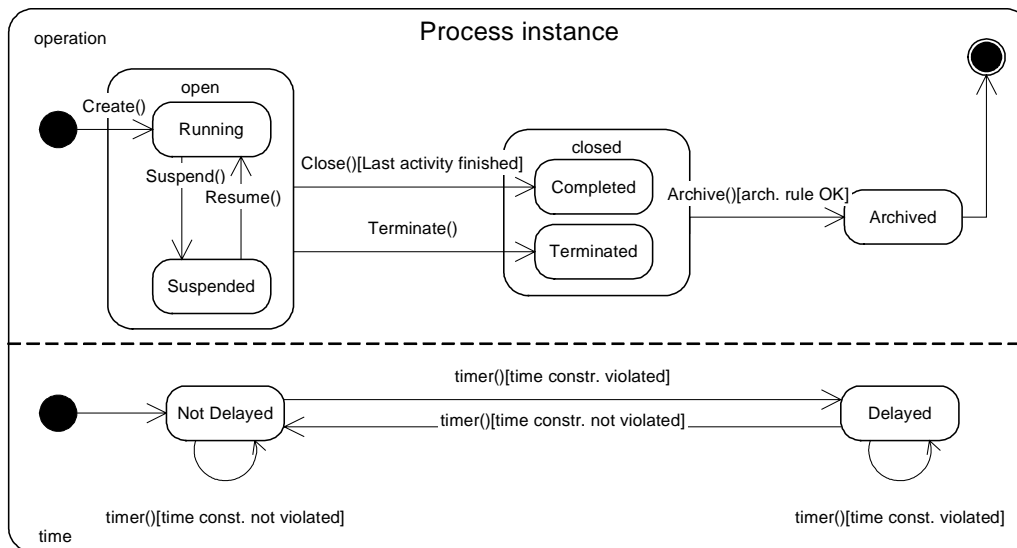


Figure 3.6. Process instance behavioural model

For a process instance two types of behavior can be defined:

- operational – related to the WfM system operations
- timing –related to the possible delay of the process instance.

The meaning of the process instance operational and timing states has been defined in respectively.

State	Description
Operational states	
Open.Running	the process instance has started execution and one or more of its activities may be started.
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 (abnormally) due to error or user request.
Archived	the process instance has been placed in an indefinite archive state.
Timing states	
Not Delayed	The process instance deadline (i.e. given explicitly by the process owner or implicitly by time deadline calculation according to the [Eder2001] algorithm) is still satisfied.
Delayed	The process instance deadline (i.e. given explicitly by the process owner or implicitly by time deadline calculation according to the [Eder2001] algorithm) has already expired.

Table 3.1. Process instance state description

3.13.3 Attribute specification

Attribute Name	Attribute description
ID	A unique identifier assigned by the WfM system.
CreationDate	The date when the process instance was created (i.e. entered the <i>Open</i> state)
FinishingDate	The date when the process instance was finished (i.e. entered the <i>Closed.*</i> state).
CurrentState	The current state according to the process instance behavior.
Cost	Zero or a float number. 0 means no cost for the activity.
Priority	Zero or a natural number. 0 means the lowest priority.
Deadline	Time constraint defines the maximal time when, with respecting to the process creation date, the process has to be finished (i.e. changes its state to <i>Closed.*</i>). A deadline is given in standard time units.
Estimated time constraints (according to [Eder2002])	
Eb	The best earliest possible execution time for the process instance.
Ew	The worst earliest possible execution time for the process instance.

Lb	The best latest possible execution time for the process instance.
Lw	The worst latest possible execution time for the process instance.

3.14 ProcInst state

Information about a state in which a given process instance was or currently is. The state refers to the process instance behavior model.

3.14.1 Attribute specification

Attribute Name	Attribute description
StatusName	A unique identifier assigned by the WfM system.
StartingDate	The date when the process instance entered the state created.
FinishingDate	The date when the process instance left the state.

3.15 Instance Attribute

A feature that can be assigned to a process instance or its activities. A definition attribute has name, value and type. The attribute type may be a simple one and a compound one. The former type is a standard type such as date, integer or text. The latter type may be regarded as a structure that consists of elements with simple or other compound type. It is possible to have a multi-valued attribute, that is an attribute with more than one different value. Definition attributes can be regarded as extended attributes as it is defined in XPDL [WfMC-TC-1025].

3.15.1 Usage

Definition attributes can be used for:

- WfM system specific attributes – to separate them from standard attributes that can be mapped to other systems
- attributes that may be defined in the future and not know at present
- compound attributes, which are hard to process.

3.15.2 Attribute specification

Attribute Name	Attribute description
Name	The name of the attribute
Type	The type of the attribute. The type can be a simple one or a compound one.
Value	The attribute value of type specified in the attribute <i>Type</i> .

3.16 ECA rule

Instantiation of an ECA rule type that has been fired for a given process instance.

3.16.1 Attribute specification

Attribute Name	Attribute description
Name	The name of the ECA rule
CreationDate	Time when this rule has been fired.

3.17 Activity instance

The representation of an activity within a (single) enactment of a process, i.e. within a process instance.

3.17.1 Usage

- An activity instance is created and managed by a workflow management system when required within the enactment of process, in accordance with the process definition.
- Each activity instance represents a single invocation of an activity, relates to exactly one process instance and uses the process instance data associated with the process instance. Several activity instances may be associated with one process instance, where parallel activities exist within the process, but one activity instance cannot be associated with more than one process instance.

3.17.2 Behavior

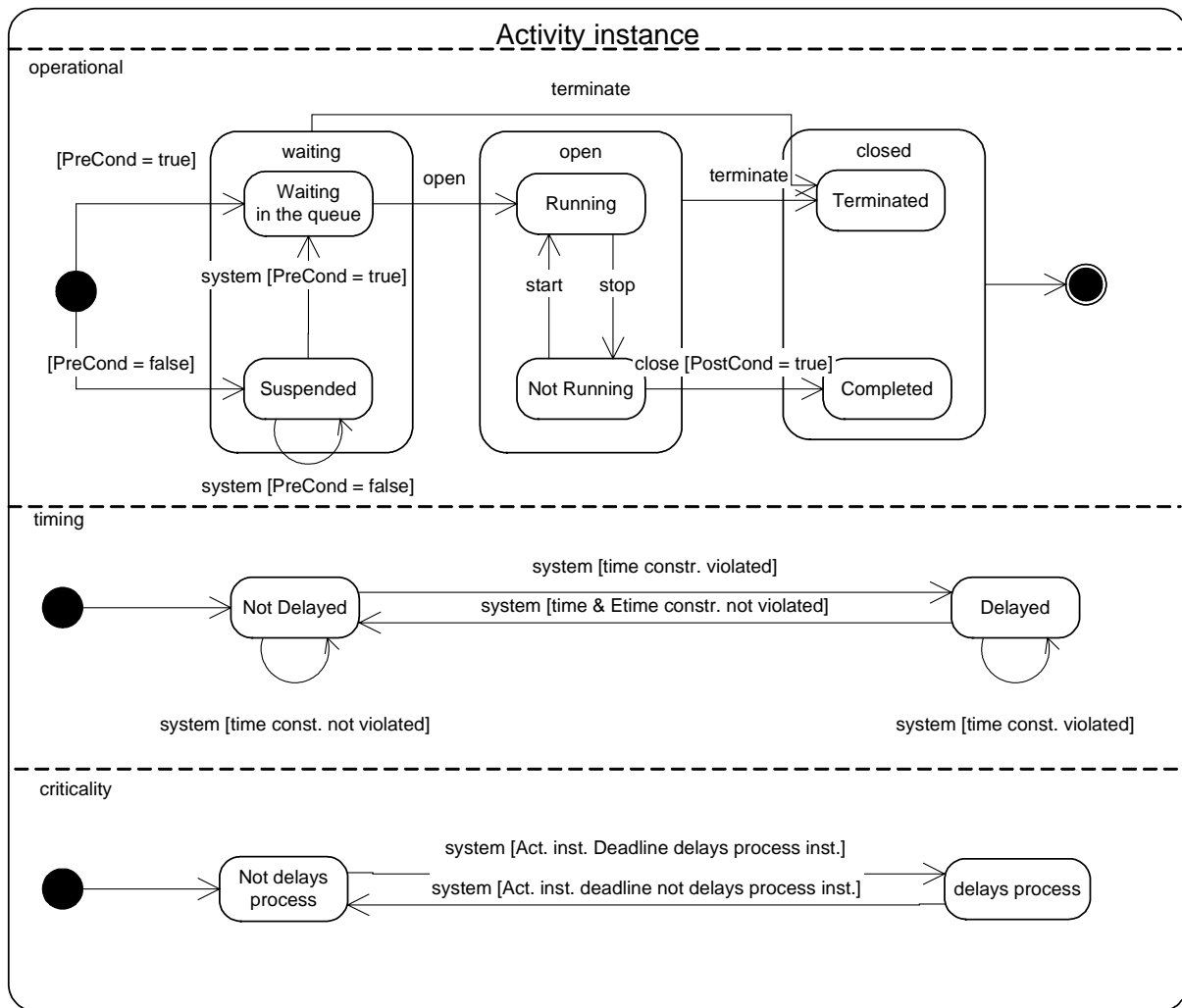


Figure 3.7. Activity instance behavioural model

For an activity instance three types of behavior can be defined:

- operational – related to the WfM system operations
- timing –related to the possible delay of the activity instance.
- criticality – indicates if a given activity instance delays the process instance. On other word, it indicates if the activity instance belongs to the critical path of the process instance.

The meaning of the process instance operational and timing states has been defined in respectively.

State	Description
Operational states	
Waiting.Waiting in the que	The pre-condition for the activity instance has been satisfied, assigned to a performer and started successfully. However, so far, the workitem related with this activity has not been executed by the performer.
Waiting.Suspended	The pre-condition for the activity instance has not been satisfied and the activity instance is waiting for its satisfaction.
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.
Timing states	
Not Delayed	The activity instance deadline and duration (i.e. given explicitly by the process owner or implicitly by time deadline calculation according to the [Eder2001] algorithm) are

	still satisfied.
Delayed	The activity instance deadline or duration (i.e. given explicitly by the process owner or implicitly by time deadline calculation according to the [Eder2001] algorithm) has already expired.
Criticality states	
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, in fact, delays it.

Table 3.2. Process instance state description

3.17.3 Attribute specification

Attribute Name	Attribute description
ID	A unique identifier
CreationDate	The Date when the process instance enters the state <code>Waiting.*</code> .
StartingDate	The Date when the process instance enters the state <code>Open.*</code> .
FinishingDate	The Date when the process instance enters the state <code>Closed.*</code> .
CurrentState	The current state of the process instance according to its behavioral model.
Cost	Zero or a float number. 0 means no cost for the activity.
Priority	Zero or a natural number. 0 means the lowest priority.
Duration	Time constraint defines the maximal execution time of the activity (i.e. period of time which activity spent in state <code>Open.Running</code>). Duration can be entered in standard time units: minutes(mi), hours (h), and/or days(d). For example 7d3h4mi means 7 days, 3 hours and 4 minutes. This time constraint can either be defined manually by the workflow designer or estimated on the basis of working and waiting times.
Deadline	Time constraint defines the maximal time when, with respecting to the process creation date, the process has to be finished (i.e. changes its state to <code>Closed.*</code>). A deadline is given in standard time units.
Calculated time values	
WaitingTime	The time that activity spent in the state <code>Waiting.*</code> .
WorkingTime	The time that activity spent in the state <code>Open.*</code> .
Estimated time constraints (according to [Eder2002])	
Eb	The best earliest possible execution time for the process instance.
Ew	The worst earliest possible execution time for the process instance.
Lb	The best latest possible execution time for the process instance.
Lw	The worst latest possible execution time for the process instance.

3.18 ActInst state

Information about a state in which a given activity instance was or currently is. The state refers to the activity instance behavior model.

3.18.1 Attribute specification

Attribute Name	Attribute description
StatusName	A unique identifier assigned by the WfM system.
StartingDate	The date when the process instance entered the state created.
FinishingDate	The date when the process instance left the state.

3.19 Transition instance

The representation of a transition within a (single) enactment of a process, i.e. within a process instance.

3.19.1 Usage

- A transition instance is created and by a workflow management system when required within the enactment of process, in accordance with the process definition.
- Each transition instance represents a single connection between two activity instances.

3.20 Performer

A workflow participant that has performed or is performing a given activity instance.

3.20.1 Usage

- For a given activity one and only one performer is assigned.

3.20.2 Attribute specification

Attribute Name	Attribute description
ID	A unique identifier
Name	The name (i.e. login) of the performer.

3.21 Application

The representation of a data type within a (single) enactment of an activity, i.e. within an activity instance.

3.21.1 Usage

- Application is a part of application functions available to the end-users.

3.21.2 Attribute specification

Attribute Name	Attribute description
Parameters	The list of real parameters passed to the application.

3.22 Object

Instantiation of a data type related to a given activity. This object is used by the activity instance performer or application to execute the activity instance.

3.22.1 Usage

- An object is a part of application data.

3.22.2 Attribute specification

Attribute Name	Attribute description
ID	A unique identifier.
Parameters	The list of real parameters passed to the application.

3.23 Object

Instantiation of a data type related to a given activity. This object is used by the activity instance performer or application to execute the activity instance.

3.23.1 Usage

- An object is a part of application data.

3.23.2 Attribute specification

Attribute Name	Attribute description
ID	A unique identifier.

3.24 Data Container

Instantiation of a data container type related to a given process instance. This data container is used to check flow conditions during process instance execution.

3.24.1 Usage

- Attributes from Data Container are relevant data.

3.24.2 Attribute specification

Attribute Name	Attribute description
ID	A unique identifier.

3.24.3 Attribute class

A single instantiation of type of an attribute that belongs to the data container. An attribute type provides information on its name. There are several basic types available such as text, number, date and reference.

4. Extended BPMN specification

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 visualization of process definition (e.g. recently published Business Process Modeling Notation – BPMN). We postulate to put equal stress on visualization of process instances allowing process performers to understand the process history and its current state (context) as well as possible future execution (potential consequences of the current decision). In our opinion, putting activity of interest in its visualized context makes the user knowledge more comprehensive and, consequently, increases productivity.

In this report, we define the process instance notation as an extension of BPMN [BPMN2002]. 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 instance level - this again increases chances for the notation to be understandable.

4.1 Process definition notation

One of the open issues declared in BPMN is enhanced graphical mechanism for forking, joining, and merging [BPMN2002]. in the ICONS project, to give a comprehensive notation for process modelers, we decided to propose a notation of routing elements on the basis of notation suggested in [Eishuis2001].

4.1.1 Routing elements notation

In order to express decision in the process the **XOR-SPLIT** operation is used. To every XOR-SPLIT operation one ingoing transition and one or more outgoing transitions are connected. If the ingoing transition has been executed, in the next step, depending on the conditions defined for ongoing transitions, only one of them will be executed (i.e. that one that is satisfied). If more than one or none of them are satisfied the system throws an exception. From programming point of view, the XOR-SPLIT operation can be treated as if-then-else command. This type of split operation is represented by the diamond symbol with more than one outgoing transitions.

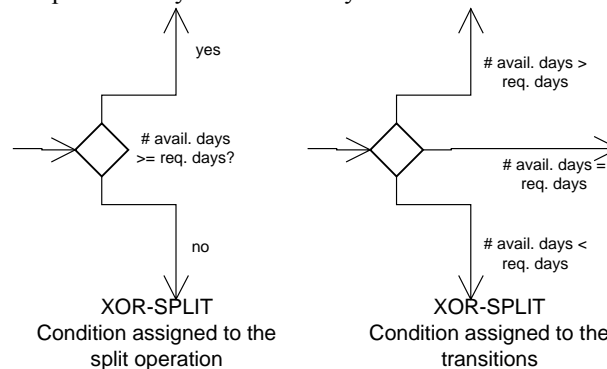


Figure 4.1. Graphical representations of XOR-SPLIT operation

The decision can be specified either as a text box in the diamond or on individual ongoing transitions. In the former case the decision can be treated as the left side of the transition condition common for all outgoing transitions, while the values assigned to the individual transitions are the right value of their transition conditions. In the latter case, the transition condition is assigned to the individual transitions explicitly.

The **XOR-JOIN** operation is opposite to the XOR-SPLIT operation. It is applied to join two or more mutually exclusive alternative ingoing transitions into one ongoing transition. If more than one or none of the ingoing transition has been executed, the system throws an exception.

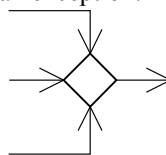


Figure 4.2. The graphical representation of the XOR_JOIN operation

This type of join operation is represented by the diamond symbol with more than one ingoing transitions.

To fork control flow and enable different activities to be executed in parallel, the **AND-SPLIT** operation is used. To every AND-SPLIT operation there is one ingoing transition and two or more outgoing transitions assigned. If the ingoing transition has been executed, all the outgoing transitions are executed.

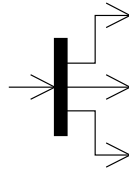


Figure 4.3. The graphical representation of AND-SPLIT operation

This type of split operation is represented by fork symbol with more than one outgoing transitions. The **AND-JOIN** operation is the opposite to AND-SPLIT operation. It is applied to join two or more parallel ingoing transitions into one outgoing transition. To every AND-JOIN operation there are two or more ingoing transition and one outgoing transition assigned. The outgoing transition is executed only if all ingoing transition has already been executed. If not, the system is waiting for the missing transitions.

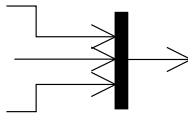


Figure 4.4. The graphical representation of the AND-JOIN operation

4.2 Process instance notation

This section specifies issues related to process instance visualization and for each issue proposes a graphical notation.

4.2.1 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 gray color 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 behavior. 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 will 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 creates loops and have been instantiated more than once are labeled with number of instances. This number is displayed in brackets.

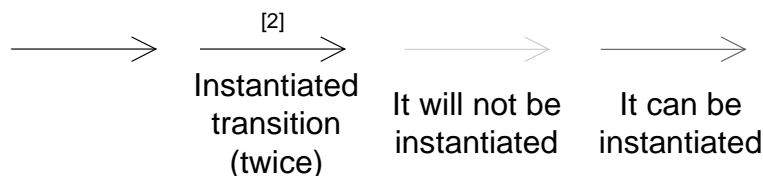


Figure 4.5. The graphical representation of different types of transitions

4.2.2 Activity instance notation

On of the most important element to represent on the process model are activity instances. Since they have different attributes and behavior that process activity, the activity notation needs to be extended significantly.

4.2.2.1 Current state

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

Operational behavior

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

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

Table 4.1. Colors used to represent the current state of the activity instance

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

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

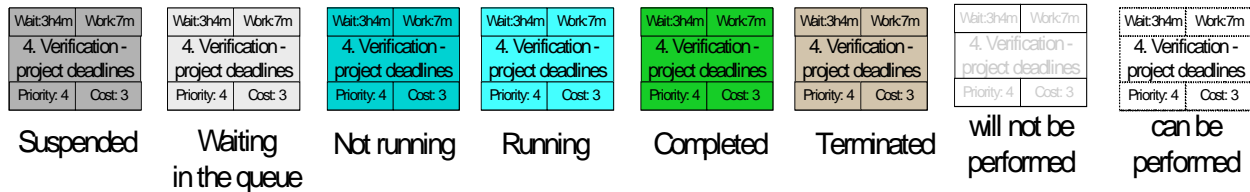


Figure 4.6. Representation of different activity instance states

Timing behavior

The current timing state of activity instance is represented by a colored 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 color as the activity is displayed with the black exclamation mark inside.

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

Table 4.2. Colors used to represent delay indicator

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

- Time.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 colors to show lapse of time (i.e. i.e. starting from yellow and continuing to red).
- Time.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 color depending on the number notifications that have been sent so far (i.e. starting from red and continuing to black).

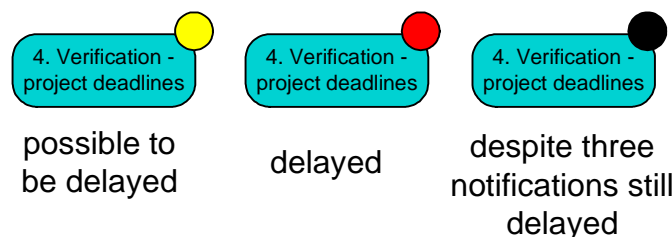


Figure 4.7. Graphical representation of different types of delay indicators

Criticality behavior

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.

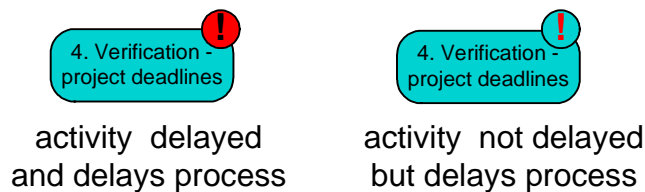


Figure 4.8. Graphical representation of different types of criticality exclamation mark

4.2.2.2 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 careful 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 left-bottom 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 color as the rest of the activity box.

Since the instantiation of a process role can consist of more than one workflow participant is has to be displayed in the activity. In this language it is represented by the upper-left tabs. If you click on a tab, information related to activity performed by a given participant is displayed. In addition, information on the left side of the swimline is updated. The current tab is marked in the same color as the rest of the activity box.

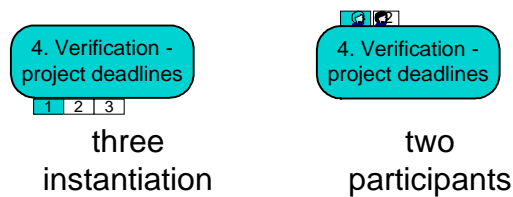


Figure 4.9. Representation of multiple instances and performers

4.2.2.3 Detailed information

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.

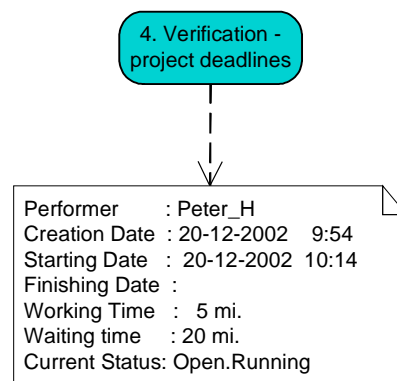


Figure 4.10. Representation of an activity attribute description

4.2.2.4 Status history

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

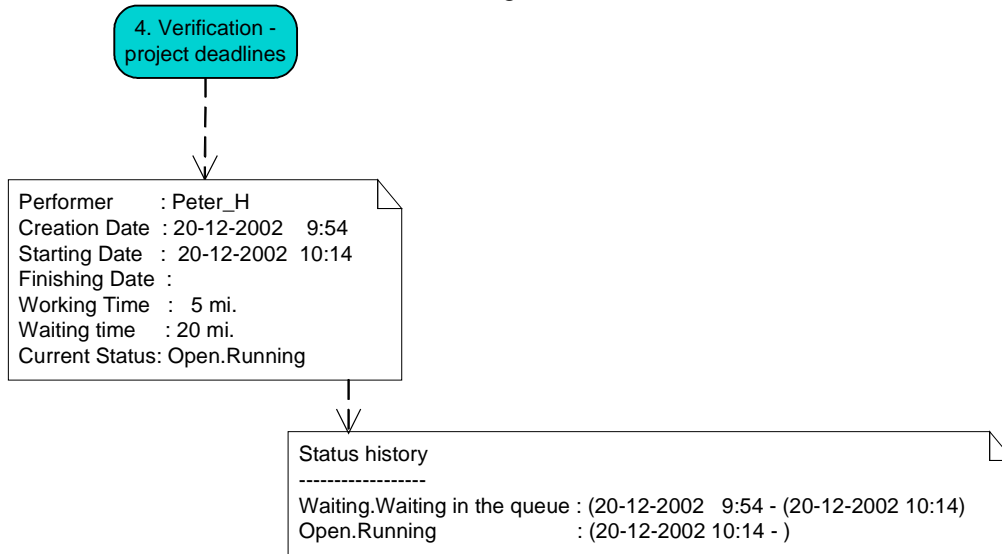


Figure 4.11. Representation of an activity status history

5. Extended XPDL specification

One of the most important elements of XML Process Definition Language (XPDL) is a generic construct that supports vendor specific attributes (i.e. extended attributes) for use within the common representation. This representation enables a number of different tools and approaches to be expressed in XPDL.

Thanks to this high flexibility of XPDL, in the ICONS project we only had to add two extended attributes for activity, namely Pre and Post conditions as well as to precise the meaning of two workflow entities: Performer and Transition Condition. These issues are described in the following sections.

5.1 Performer Relationship

According to the Workflow Participant Assignment language described in section 3.9.1 we precise the meaning of the XPDL's Performer. In our approach the body of the tag <Performer> is fulfilled with the WPA expression. For example, the following performer assignment

```
<Performer>ONE#AUTO#Group_Member('A')</Performer>
```

means a workflow participant that is a member of the 'A' group.

5.2 Pre & Post conditions

In order to express Pre & Post-conditions we add two additional extended attributes to the activity specification. These conditions are expressed in WFCL described in section 3.8.1.

An example for definition of these two attributes for an activity is the following:

```
<ExtendedAttribute Name="PrePostConditions">  
  <PrePostConditions>  
    <Attribute Name = "PreCondition=" Value="x > 500" />  
    <Attribute Name = "PostCondition=" Value="Status = 'OK'" />  
  </PrePostConditions >  
</ExtendedAttribute>
```

5.3 Control Flow Conditions

According to the Workflow Condition Flow language described in section 3.8.1 we precise the meaning of the XPDL's Transition Condition. In our approach the body of the sub-tag <Condition type>, the tag <Condition> is fulfilled with the WFC expression. For example, the following transition condition assignment:

```
<Condition Type="CONDITION">STATUS= 'ZAAKCEPTOWANA WST.'</Condition>
```

means that is true if the attribute status has value 'ZAAKCEPTOWANA WST.'.

6. Summary

This report describes the framework of procedural knowledge representation. In order to be flexible and coherent with other work in the workflow domain, we decided to use well-known and well-defined standards both from the Workflow Management Coalition and Business Process Management Initiative. Since both BPMN and XPDL did not fully satisfy our requirements for the contemporary workflow management, we decided to extend these standards. Especially for BPMN, we suggested quite a significant extension for supporting process instance representation according to the process definition.

Our propositions, however, should be treated as preliminary and will be revised in the next stage of the ICONS project.

Bibliography

External references

- BPMN2002 Business Process Management Initiative; Business Process Modeling Notation; working draft, version 0.9, Nov 2002.
- Casati1996 F. Casati, S. Ceri, B. Pernici, G. Pozzi; *Deriving Active Rules for Workflow En-actment*; Int. Conf. on Database and Expert System Applications; Zürich, Switzerland, 1996.
- Eder1997 Eder, J.; Pozewaunig, H., Liebhart, W., ePERT: Extending PERT for Workflow Management Systems, Proceedings of the 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, Proceedings of the 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., Book Division, 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
- Momotko2002 Momotko, M., Subieta, K., Dynamic change of Workflow Participant Assignment. Paper accepted to Advances in Database Information Systems, ADBIS'2002, Bratislava, 2002.
- Rumbaugh1999 Rumbaugh J., Jacobson I., Booch G., The Unified Modeling Language Reference Manual, Addison Wesley, 1999
- WfMC1994 Workflow Management Coalition, Information Pack, Grenoble, France, July 1994
- WfMC1996 Workflow Management Coalition, Workflow standard, Interoperability abstract specification, WfMC-TC-1012 version 1.0, Oct 1996.
- WfMC1999 Workflow Management Coalition, Workflow standard, 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.
- WfMC-TC-1023 Workflow Management Coalition, Workflow standard, Wf-XML Binding, WfMC-TC-1023 version 1.1, Nov 2001.
- WfMC2001 Workflow Management Coalition, Workflow standard, [Wf-XML Binding](#), [WfMC-TC 1023, Final draft, Nov 2001 Version 1.1.](#)

ICONS references

- [ICONS D06] The IST-2001-32429 ICONS Consortium, Research Base for the ICONS Project, www.icons.rodan.pl, June 2002
- [ICONS D16] The IST-2001-32429 ICONS Consortium, Specification of the ICONS Architecture, www.icons.rodan.pl, December 2002
- [ICONS D25] The IST-2001-32429 ICONS Consortium, The knowledge-based content management application design methodology, www.icons.rodan.pl, to be developed

Dictionary

Notion (short name)	Meaning
Business process	A set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships.
BPMN	Business process Modeling Notation
XPDL	XML Process Definition Language
Wf-XML	Workflow XML Interoperability Binding
Workflow management system (WfM system)	A system that defines, creates and manages the execution of workflow processes through the use of software, running on one or more workflow managers, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications.
Workflow process (workflow)	The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.
Workflow process definition (workflow definition)	The representation of a business process in a form which supports automated manipulation, such as modeling, or enactment by a workflow management system.
Workflow process activity (activity)	A description of a piece of work that forms one logical step within a workflow process.
Workflow process instance (process instance)	The representation of a single enactment of a workflow process for a particular case.
Workflow Activity Instance (activity instance)	The representation of a single enactment of a workflow activity within a workflow process instance.
Workflow participant	A resource which performs the work represented by a workflow activity instance.
Workflow participant assignment (WPA)	Assignment of one or more workflow participants to a given workflow activity instance.
Worklist	A list of work items associated with a given workflow participant
Workitem	The representation of the work to be processed (by a workflow participant) in the context of an activity within a process instance.
Workflow Application	A software program that interacts with a workflow enactment service, handling part of the processing required to support a particular activity (or activities).
Application data	Data that is application specific and 'read-only' for workflow management system.
Workflow relevant data (relevant data)	Data that is used by a Workflow Management System to evaluate the control flow conditions (e.g. pre- and post-conditions, transition conditions) and workflow participant assignments. This data includes both application data as well as workflow control and process data.
Workflow control data	Data that is managed by the Workflow Management System and/or a Workflow Manager.
Workflow process data	A historical record of the progress of a process instance from start to completion or termination.
Workflow interoperability	The ability for two or more Workflow Managers to communicate and work together to co-ordinate work.
Workflow manager	A software service or "engine" that provides the run time execution environment for a process instance.