Workflow Object Driven Model

Włodzimierz Dąbrowski[1,2], **Rafał Hryniów**[2]

Abstract: Within the last decade the workflow management makes an incredible career. Technology connected with the workflow management is becoming the key element of information systems. Despite of multiple advantages coming out of incorporation of systems to workflow management there are significant limitations visible. One of them is assumption, that business processes are not changing during their execution. It's not true for the most of the real processes, requiring adaptation to the dynamic changes both within the workflow and in the process environment. Classical solutions of the workflow processes control are based on a sequence transformations controlled by Petri's nets methods. There are many disadvantages of such approach which lead to many problems. Within the article there are shown new approach methods for process modeling problems, based on the object model, taking into consideration the event steering of workflow processes.

Keywords: workflow, business processes, business modelling

1. Introduction

In 1996 Workflow Management Coalition published a glossary of terms related with Workflows. The Workflow definition developed by WfMC looks as follows:

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 (WfMC-TC-1011, 1999).

During the last decade workflow management has had a successful career. Workflow management technology has become a critical component of IT applications, as have databases, industrial information technology systems or transaction processing (Aalst W van der, 2002), (Momotko M. and Subieta K., 2002), (Momotko M. and Subieta K., 2004). Despite many advantages resulted from application of workflow management systems, significant limitations were also observed. One of the major restrictions was the assumption that business processes do not change very often during their execution. Such an assumption does not hold for most of real, less rigid business processes which need to adapt to dynamic changes in the workflow environment (i.e. data, resources, applications) as well as the workflow

[1] Warsaw University of Technology
Koszykowa 75, 00-662 Warsaw, POLAND
e-mail: w.dabrowski@ee.pw.edu.pl
[2] Polish Japanese Institute of Information Technology
ul. Koszykowa 86, 02-008 Warsaw, POLAND
e-mail: rhryniow@pjwstk.edu.pl

itself.

Classical solutions of the control of workflow processes are based on sequence transformations which could be described using Petri nets (Aalst W van der, 2002). Defects of such approach are numerous and make for following problems:

- the dynamic change of the process during its activity
- the parallel execution of processes with synchronization (present synchronization resources are too primitive and insufficient)
- the parallel execution of processes on many servers
- breaking of executable process or his parts as a result of some events that could be recognized automatically
- the response on exceptions or events that happen in the system environment
- the resource management (like people, budget, computers, infrastructure, etc.)
- the generating of new processes that could be automatically running

Some languages like YAWL (YAWL, 2007) overcome most of this problems by introducing new workflow patterns that can be used to solve specific problems. We believe that this problem may not be solved by this kind of approach, but need a new paradigm.

In this paper we would like to introduce an alternative paradigm based on object-oriented model where processes are objects with defined attributes and behavior. This solution can be used to model all workflow patterns (basic and advanced) in an uniform manner (Aalst W van der, 2000), (Aalst W van der and Kees van Hee, 2002). Another advantage of using a object-based model is ability to make dynamic changes to executed processes e.g. insert new process or remove an existing one as well as modify a behavior of active processes.

The proposed paradigm makes workflows close to PERT nets that are vastly used in project planning to analyze critical paths. PERT nets assume inherent parallelism of processes, constrained however by their in-out dependencies and availability of resources necessary for execution.

We would like to emphasize that we do not try to create an extension to object-oriented model or map current solution to object model (Mann D, 2007), (Schmidt M), but to show a different approach to workflows that base itself on object model.

2. Workflow Object Model

Object-oriented approach for Workflow process model treats the work process instances as object structures with defined attributes that can be queried and used

in evaluation during run-time. The procedural (behavior) part provides the modeling of work process dynamics called out by proper external events. This approach makes a workflow process a net in which we would define constraints on starting a new process.

The object model gets away from pre-defined flow of work process control and replaces them with approach based on event model. To provide univocal identification of process task sequence, the object model for workflow is based on following assumptions:

- each process has initial condition that starts the process (precondition)
- each process has conditions for completing the process (postcondition),
- each process has a state,
- each process is parallel by definition,
- processes do not create an explicit hierarchy. The hierarchy is defined in implicit manner by defined conditions,
- there can be as many instances of a given process as it is needed,
- each process has a definition that is used for creating an instances of a process,
- precondition, post-condition and procedural part of a process are first class programming objects, so that it is possible to modify them in the process instances e.g. replace postcondition with its new version for a particular process instance.

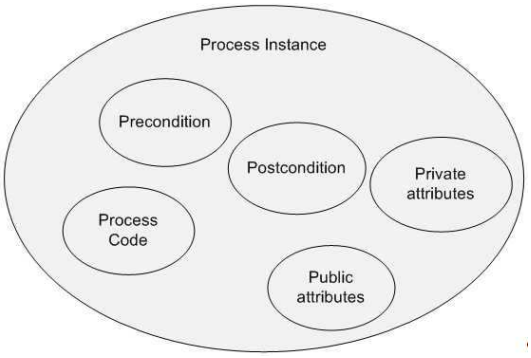

Figure 1. Model of Workflow Object as a process instances

Preconditions and postconditions are determined by a query language which may work on the entire workflow environment and the state of all the population of workflow processes.

State of an object can be used as a basic mechanism for control transition from one process to the other. Assume we have processes A and B within the same environment with clause, that B may be started after A is finished, the process A after it has finished sets its own status for "completed", and the process B has in the initial clause a predicate: $(A.process_{status})$ = "*completed*". Of course, our model allows a lot of different modes for control transitions and process activity parallelization.

Within the object model the work process is treated as an object with defined following elements.

2.1. Unique internal ID

Unique internal ID identifies the given object the univocal way. It's unreadable and unprintable, cannot be used directly within the source code of query or program. In other context is called reference. Reference is necessary in the stack approach, e.g. for imperative construction realization. It exists also as value of the pointer object, realizing the connections among objects. May be implicit used parameter for procedures, functions or methods.

2.2. External name

External (business) name of the object developed during design of analytic application model, given by admin or other authorized user. It's incorporated for object identification within the source codes for queries/programs, but is not unique. There may be many processes possessing the same name.

2.3. The flag, setting it is a process instance not a simple object

The flag "I'm a process instance" serves for distinguishing processes from other object in the same environment. May be implemented as display attribute with reserved name, which permits the interpreter to recognize processes. In a object-based system with implemented classes an inheritance tree may be used as a flag e.g. inheriting from special class or interface sets the flag.

2.4. Public attributes

Public attributes are attributes accessible outside of the object. They serve for process identification and eventually to change its status by any external program, i.e.: other processes or administration modules. Some of the attributes should be distinguished, with reserved names and eventually externally unalterable, especially:

- process status with values: "inactive", "running", "completed", "suspended", "interrupted", "transferred for realization to other machine" etc.;
- process creation time;

- process start time;
- process end time;
- process priority;

Apart from those attributes there may be more, which were foreseen during creation of definition for the given process instance (the definitions would be described below). Those attributes can store any information about substantial or abstractive resources assigned to given process, for the started and finalized processes the substantial resources should be stored etc. The attributes may have unrestricted number and complexity: may be complex, optional, repeatable etc.

2.5. Private attributes

Private attributes are externally invisible. They may be used by any internal query/method e.g. for calculation of postcondition.

2.6. Single distinguished method containing process code (procedural part)

The method containing process code (procedural part) determines the behavior of an object. Basically , the code may contain any imperative instructions and we suppose that it is single-threaded, has no splitting and synchronizing (joining) elements. This code may also create new processes and modify environment.

Multi-threading may be realized with many parallel processes. Few instructions may be important for work processes, particularly:

- sending a mail with dynamically created content to defined addresses;
- generation of HTML, XML, TXT, PDF or other document;
- sending any document to defined place;
- starting external application on certain computer with defined parameters and file, particularly starting Web Services;
- time control in form of instruction synchronizing the clocks of all computers being under the management of our workflow application, setting timers and reaction for timers etc.;
- cooperation with defined application, installed from the client side, i.e. application graphically setting the task queue to execute by certain user (setting the task to the queue, claiming the queue etc.);
- transferring the defined process for realization to other defined machine;
- synchronization with other processes - “wait for” instructions.

2.7. Precondition

The initial clause is evaluated by interpreter within the cycle designed by designer. The cycle for checking the clause for very fast processes, requiring nearly real-time reaction, may occur every second, for less time dependant system every minute, etc. It depends on time scale the business uses. The precondition is a typical query valued true/false, which may use any other element of the whole workflow application environment, particularly the status of other processes, local and global attributes, clock, timers. Until the query evaluation returns false, nothing happens. In the moment the query returns true, the process is started. This means two things:

- if the defined process has a procedural part, the interpreter starts its execution.
- the query for the started process is no longer checked. The process may be interrupted (with proper authorized instruction changing its status). Interrupting the process doesn't mean its removal, removal is a separate operation controlled e.g. by administrator . The process may also be suspended and continued.

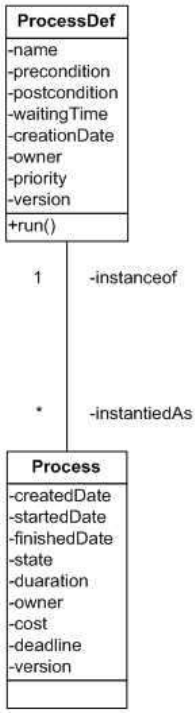

Figure 2. Model of Workflow Object as a process instances

2.8. Postcondition

The final clause determines the termination of a process. It's an optional element. If it does not exists we assume that process is finished when the process active part stops executing.

As with precondition, the clause is periodically tested and the testing cycle is set by process designer and depends on time scale of the business. As previously, the final clause is a query returning true/false value and may refer to any global and local information of the environment for the defined workflow application. If the final clause for the given process is true the process is set to "completed" and nothing more happens except for archive events (reporting, data mining etc.).

The final clause fulfils simultaneously the part of parallel process synchronizer. Within this clause one may e.g. test, if all the other depended processes were set to "finished" status.

2.9. Connections with the others accessible in the workflow process environment

The interconnections of the process with other objects serve to connect the process with information stored within any other object. The examples of such connections are below:

- connection of the process with abstract resource (required or planned), i.e. executor's part, abstract computer resource (printer), etc.
- connection of the process with substantial resource (currently in use or used), i.e. defined executor performing the given task, defined executor who already performed the given task, substantial computer resource (printer) provided for the process or used by it etc.
- connection of the process with the object storing its definition;
- connection of the process with metadata (ontology), which permit finding the process and further operations basing on its certain business attribute.
- connecting the process with information setting the security rules: access rules, privacy, permissions etc.
- connection of process with the documents it uses, i.e. invoice;
- connection of process with any business data of the workflow application environment, i.e. warehouse contains or marketing data
- connection of process with applications used by the process;

The connections may be local (private for certain process) or global (visible for external requests). They may also become a part of complex attributes. For

example, connection to certain executor may exist within the complex attribute, containing also the executors part, job start, job finish, the time quantity used, hour payment rate, etc.

2.10. Connections with objects storing the constants for objects, what means classes

The connection of the object with classes may be implemented to incorporate the constants of the objects included in those classes (as normally within objects). Let's suppose, that class is a certain database object, sitting on the server side (it's constant and of first programming category). The main constants stored in the class may be:

- the typology information, needed for typological control of the objects and static typological control of queries/programs operating on the object;
- the methods codes stored within the class (their binders are stacked on the environment pile in the moment of opening the internal part of the object;
- static (Class) attributes;
- the codes for integrity clauses, triggers or overload perspectives;
- information regarding the security rules;
- information regarding the object visualization (e.g. graphics for visualization icon or visualization method);

Classes may become connected into hierarchy of inheritance, as standard within the objects.

3. Solved Problems

We would like to present some of the problems we have mentioned with the description of how our approach solves it:

- the dynamic change of the process during its activity - precondition, postcondition and active part are first class programming objects that can be modified, substituted during process execution
- the parallel execution of processes with synchronization - in our approach everything is by definition parallel. Synchronization methods are "waitFor" command, preconditions and postconditions,
- breaking of executable process or his parts as a result of some events that could be recognized automatically - the same explanation as for the first one.

4. Conclusions

We have proposed a new approach to workflows that is based strictly on object model and simulates a PERT net behavior. We believe that modeling processes as objects with formal attributes and behavior may solve all problems introduced by Petri nets and its successors. The implicit parallelism allows for much easier creation of distributed workflows. Furthermore by making preconditions, postconditions and active parts are a first class programming objects we can easily create and adaptive workflow without any alteration of our assumptions as e.g. modification of postcondition during process runtime can be used for making an adaptive-or or adaptive-and pattern.
Currently we are working on prototype based on object-oriented database ODRA which is constructed on a SBA paradigm (Subieta K, 2004), (Subieta K, 2006). We use SBQL as query language used for evaluation of preconditions and postconditions.

References

Aalst van der and Kees van Hee (2002) *Workflow Management: Models, Methods, and Systems.* Massachusetts London, England

Aalst van der and Kees van Hee (2000) *Workflow Management.* MIT Press Cambridge.

Kenan, I (2004) *Adaptive Workflow Patterns.* The University of Manchester.

Mann, D (2007) *Workflow in the 2007.* Microsoft Office System, Apress.

Momotko, M and Subieta, K. (2002) Dynamic change of Workflow Participant Assignment. *6th East-European Conference on Advances in Database Information Systems, ADBIS'2002, Bratislava, Slovakia.*

Momotko, M and Subieta, K. (2004) Business process Query Language - a Way to Make Workflow Processes More Flexible *8th East-European Conference on Advances in Database Information Systems, ADBIS'2004, Budapest, Hungary*

Subieta, K (2004) *Theory and Construction of Object-Oriented Query Languages.* Editors of the Polish-Japanese Institute of Information Technology.

Subieta, K (2006) *Stack-Based Approach (SBA) and Stack-Based Query Language (SBQL)*
`http://www.ipipan.waw.pl/~subieta` ? Description of SBA and SBQL

Schmidt, M Building Workflow Business Objects,
`http://jeffsutherland.com/oopsla98/mts.html`

WfMC-TC-1011 (1999) *Workflow Management Coalition: Terminology & Glossary, WfMC-TC-1011, version 3.0*

YAWL (2007)
http://yawlfoundation.org/