NEW APPROACH IN WORKFLOW PROCESS MODELLING

Lukáš MÁČEL[1]

SUMMARY: This paper is focused on process modelling languages. The two widely adopted languages BPMN and BPEL are discussed, evaluated and compared on the basis of six criteria. The results of this evaluation could be used in order to design different modelling approaches based on a high-level universal programming language. The construction of a process model, including how the basic workflow patterns are defined, is presented using the object-oriented scripting language Python. The resulting model is then evaluated according to the stated criteria.

Keywords: BPEL, BPMN, process modelling, Python, workflow, workflow patterns

Introduction

Management based on workflow processes enables one to organize work better and thus improve the execution of particular activities. Process is defined as a sequence of coordinated activities which are executed in order to achieve a specific goal [6]. In process definition, the order of activities is very important [2, p. 33]. The particular work procedures are often known to their users only. The implementation of processes brings a global view to these procedures and allows a better understanding of the relationship between the activities. It is then possible to expose the inefficient ones and optimize them.

Processes make the coordination of activities easier and can be used for workflow monitoring to detect some deviations which could cause problems later. Process definition comes with important information about data that are necessary for execution, that are about the right timing of execution and about the participants responsible for a particular activity. In addition, the flexibility of work is better if the processes are defined properly. Finally, the processes make it possible to collect information about workflow in order to use it for subsequent analyses and optimization. The process approach is applicable in many areas, not only in companies but also in public administration and in the army.

Nowadays information technologies support many activities and therefore are an essential part of processes. On the one hand, they help reduce the overloading of workers and, on the other hand, they streamline work. Moreover, some activities lend themselves to complete automation. IT supports not only activities but also the possibility of using it for process management and coordination. If we create a computerized representation of a process then we can leave the control of the activities to an information system. A process whose interpretation is partially or fully supported, or even automated by a computer is called *workflow* [4, p. 6]. The system of activity coordination is then called the *workflow system*. Such a system computerises the definition of process and ensures the execution of relevant activities. It assigns tasks to participants of the process and automatically distributes essential data needed for their performance. Simultaneously, it can collect information about the process execution which is useful for further optimization.

[1] Ing. Lukáš Máčel, Faculty of Information Technology, Brno University of Technology, Božetěchova 2, 612 66 Brno, Czech Republic. E-mail: imacel@fit.vutbr.cz

The exact definition of the process model plays a significant role in process automation. The *process model* represents a pattern which describes how a set of similar cases can be solved. The *process instance* is then a particular solution of a case [3]. The model consists of a set of tasks describing activities to perform and also contains the definition of the relationships among these tasks. In model creation the four basic workflow patterns that can be applied include sequence, parallelization, selection and iteration. The *sequence* pattern expresses that two or more tasks must be performed in strict order. The *parallelization* pattern describes tasks that can be performed concurrently. The semantics of the *selection* pattern is defined as logical XOR: either one task is selected or another. Finally, the repeating of two or more tasks describes the *iteration* pattern.

Currently different types of languages can be used for process model specification. The first part of this article classifies the present process languages and critically evaluates their expressive ability. The second part engages in the design of language for process description based on high-level universal programming language. The advantages and disadvantages of this solution are discussed.

1. Process modelling languages

There has been a noticeable standardisation effort in the area of process languages for a long time. Many consortiums and academic groups have been formed in order to create a unified description of process. In [8] Mening, 15 different languages are compared for process modelling. The result of his research shows that there is considerable heterogeneity in the syntax and semantics of process languages. The lack of uniformity complicates the inter-operability of modelling tools and process model interpretation itself. Differences can also be found in the way the process model is represented. [7] Two groups of languages are identified here according to model representation:

1. *Graph-oriented* languages define process through an oriented graph in which oriented edges express time and the logical relationships between the nodes of tasks and events. The types of nodes unfortunately differ from language to language. One important member of this group is BPMN (Business Process Modelling Notation). A detailed description can be found in [5].
2. *Block-oriented* languages specify the process model through control primitives. Usually the model is described as a structured value in XML format. An important member of this group is BPEL (Business Process Execution Language), described in [9, p. 130-134].

In the following text both of the mentioned languages will be discussed and evaluated according to these criteria:

1. **Possibility of direct interpretation of the model through the workflow system**

   Many languages describe the process model only on an abstract level and do not support the direct execution of the workflow system. Transformations must be done to obtain a model which can be directly interpreted. These transformations may be complicated and sometimes there are problems with ambiguity.

2. **Support of basic workflow patterns**

   Modelling language must support basic modelling constructions.

3. **Clarity and comprehensibility of the model**

   These attributes of the model are very important, because if they are not fulfilled, the creation of the model is difficult and it generates many errors.

4. **Compactness of the description and its subsequent maintenance**

The process is typically still in progress. Flexible reaction is very important and therefore it is necessary to have a compact model and simple maintenance.

5. **Ability to use domain objects**

   During the creation of the process model, it is essential to use domain objects in the context of which the process is executed. The description of task inputs and outputs and guard conditions for control constructions are based on the terms of a particular domain. Workflow represents the extension of the domain layer.

6. **Portability of the process model**

   Simple model exchange among different modelling tools is also very important. It allows using and extending already defined models.

2. BPMN

The motivation for the creation of BPMN was the unification of graphic notations for process description. Today it is one of the most accepted standards on the field of process modelling and frequently used by many organizations [10, p. 3]. The objective of a standard is simple and comprehensible notation, which is understandable for all groups of people involved in processes (team leaders, analysts, IT specialists, etc.). Process modelling is not directly interpretable with the help of a workflow system, but it allows transformation to a particular executable process language (e.g. BPEL) [5]. The process model is expressed with the help of a process diagram. This is a directed graph composed of nodes and directed edges which express the relationship of the sequence. Fig. 1 shows a simple model noted in BPMN.

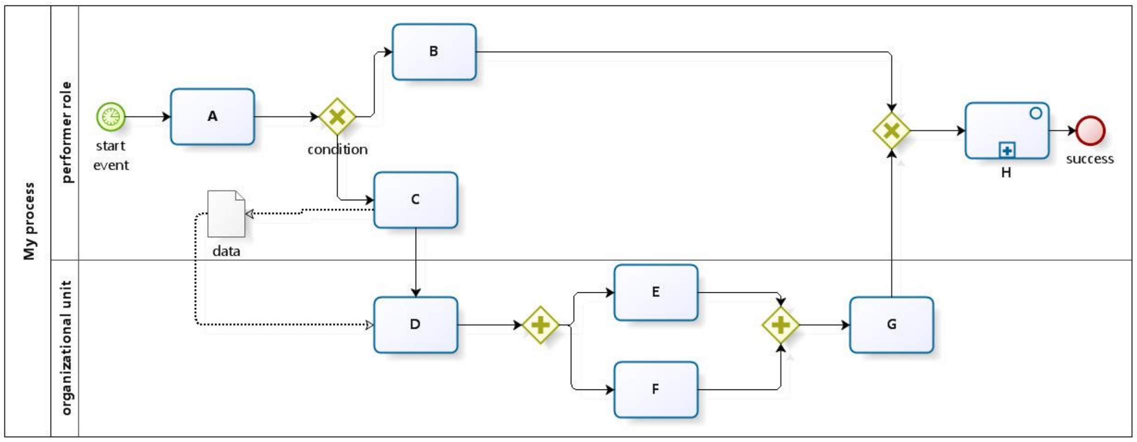

Fig. 1: Process model noted in BPMN

The notation defines three basic types of nodes:

- The *task* is graphically represented as a rectangle and it is labelled with the name of the task. BPMN allows the use of atomic tasks (in fig. *A* to *G*) or composed tasks (*H*). With the composed task, it is possible to further expand and show its model using a sub-process. In this way the hierarchy of the model is defined.
- The *event* describes the process state, which is important from the perspective of its execution. Using this node, it is possible to express the sequence of tasks or their correct timing [1, p. 129]. Each process is started with a particular event (e.g. fulfilling the time condition or receiving a message). Similarly, the end of a process is modeled by an end event. The example uses this to describe the successful completion of the process. Often there are more end events describing error states in order to capture all the possible scenarios of the workflow.

- The *gateway* acts as the control point in the process execution. The node of the gateway always has one incoming edge and multiple outgoing edges. The choice of output depends on the type of gateway and possibly on the associated guard condition. Graphically it is expressed as a diamond shape. In the example, the execution of task *A* is followed by selection gate (XOR), symbolically expressed with *X*. This means that either task *B*, or task *C* will begin on the basis of the given condition. Through the selection it is possible to define the guard condition of the cycle and thus model the iteration. Another example of a gate is the parallel gate (AND-split) expressed with the symbol +. Its semantics express that tasks *E* and *F* will be performed concurrently. The following parallel gate (AND-join) models the place of synchronization. As soon as both tasks are completed, it continues with task *G*.

A special type of node is the *data object* (in the example, the *data* node). Through this the language models the output of a particular task or, for example, the content of a received message. This object can then be associated with a different task, thereby modelling the dataflow of the process. BPMN makes it possible to also define information on the participants of the process. The diagram is divided into lanes containing the task nodes, for which the participant given in the description of the particular lane is responsible. In the above example the given *performer role* will solve tasks *A*, *B*, *C*, *H* and *organizational unit* the remaining tasks – *D*, *E*, *F* and *G*.

According to the stated criteria, standard BPMN can be evaluated thus:

1. **Possibility of direct interpretation of the model through the workflow system**

   The model in BPMN cannot be directly interpreted through the workflow system. The standard at least recalls the possibility to transform notation to another executable language.

2. **Support of basic workflow patterns**

   All the basic patterns can be expressed in the diagram.

3. **Clarity and comprehensibility of the model**

   The graphic notation allows for the comprehensive capturing of the process model. This is true primarily for simple models, which are typically used in BPMN examples. With processes using more complex operating logic, which are typical in practical use, the level of clarity decreases. The resulting diagram seems chaotic and difficult to understand.

4. **Compactness of the description and its subsequent maintenance**

   The situation is similar in terms of the compactness of the model. Simple models are not demanding of space and are easily maintained. With increasing complexity, the demands for space also grow. Making changes in a complex diagram often results in the entire diagram having to be redrafted, because it is necessary to organize the nodes differently.

5. **Ability to use domain objects**

   BPMN defines the semantics of data objects, but their relations to domain objects are not formally specified in any other way. The same applies to the definition of guard conditions, which merely have an annotation character.

6. **Portability of the process model**

   The model is very portable. The objective of the standards to unify the graphic definition of the process is successful.

3. BPEL

BPEL is based on XML format. Using the language it is possible to specify the operating logic of the process and define the execution of the tasks through web services. The model contains the definition of a web interface described in WSDL (Web Service Definition Language). The execution is then expressed by an XML file using basic modelling concepts. The interpretation of the model occurs through the orchestration of web services and communication is achieved through the sending of messages. A good example of this can be found in [9, p. 110].

Individual steps of the process are described using *simple activities*. These activities include the receiving and sending of a message (receive, invoke and reply), assigning value to a variable (assign), suspending a process (wait) until a particular moment, or terminating the process (terminate). *Structured activities* can contain other structured or simple activities. They include a sequence mechanism (sequence), concurrent execution (flow), selection (if-else), receiving message (pick) and iteration (while, forEach).

The model in BPEL allows the use of *variables* to keep the context of execution. A message or other important data for process can be assigned to a variable. Variables are declared in the head of the XML document. It is also possible to define *handlers* to handle an event, error or process termination. BPEL uses *partner links* to establish communication with web services.

Evaluation based on the stated criteria is as follows:

1. **Possibility of direct interpretation of the model through the workflow system**
   BPEL is designed to support direct interpretation of the process model.
2. **Support of basic workflow patterns**
   All the basic patterns are supported by using structured activities.
3. **Clarity and comprehensibility of the model**
   Complicated namespaces and numerous XML tags cause the resulting model to be **difficult to read and understand**. Using control constructions and assigning values to variables in XML is also complicated.
4. **Compactness of the description and its subsequent maintenance**
   Description of the process is very long and it is difficult to modify and maintain it.
5. **Ability to use domain objects**
   BPEL supports the defining of relations to the domain, but it is not possible to use domain objects directly. Mapping through an XML scheme is necessary. This solution is less flexible and difficult.
6. **Portability of the process model**
   The model is very portable.

4. Model of processes using universal programming language

From the previous comparison of two currently used languages for describing processes, it is evident that their models are not easily comprehensible and difficult to maintain. The presented languages define the models at too abstract a level. The workflows do not create a super-structure of the domain layer, but are defined independently. This considerably complicates the specification of the process.

Now a different approach for process modelling will be presented. Instead of using abstract language to describe the process, universal programming language will be used to define a process model based on domain objects. Because of this, the impedance will be reduced between the layer of the domain and the operating logic, as is the case today between the data layer (database) and domain layer. This approach brings about several other advantages. It is possible to use the object-oriented paradigm and some of its important concepts such as the inheritance for creating well structured models. If an interpreted language is selected, then the interpreter of this language can be used directly for the execution of the model. A significant advantage is the possibility of using current developmental environments, quality debugging and simulation tools that are at a higher level than tools for specialized process languages. Another advantage is the flexibility and possibility of using already implemented solutions.

The objective is not to create a new language for modelling processes, but to use some current high-level universal programming language and even just to complete some constructions, so that the process model could be defined simply. Some of the important attributes of the considered language should include:

- *Interpretability* – the interpreter of the language can be used to execute the model
- *Object orientation* – a quality structured model that is easy to maintain
- *Compact and comprehensible syntax* – good readability of the model

The scripting language *Python* has been selected for the process modelling. However, there is nothing preventing the use of another language from fulfilling the given characteristics. As with BPMN and BPEL, a method of defining construction elements of the model will now be shown as well as the expression of general workflow patterns.

The process describes a certain working algorithm and therefore it is expressed with a function. Individual tasks are represented as methods of objects modelling participants in workflow. This approach allows the use of domain objects, complete behaviour, which is important from the perspective of workflow, and using the resulting object directly in the process definition. This results in the valuable interconnection between the domain layer and the workflows.

The BPMN example in Fig. 1 is defining tasks *A* to *H* and the two actors *performer role* and *organizational unit.* Their definition in Python is shown in the figure below. The implementation of individual tasks is not resolved here.

```
class PerformerRole(Object):
 def A(): pass
 def B(): pass
 def C(): pass
 def H(): pass
```

```
class OrgUnit(Object):
     def D(): pass
     def E(): pass
     def F(): pass
     def G(): pass
```

Fig. 2: Definition of the classes of actors

With the help of a general programming language, one can also simply express the pattern of sequence, selection and iteration, because the language contains the appropriate constructions. This definition is shown in Fig. 3.

```
#Sequence
o.A()
o.B()
```

```
#Selection
If obj.getValue() > 10:
  o.A()
else:
  o.B()
```

```
#Iteration
while repeat:
   o.A()
   o.B()
```

Fig. 3: Expressing the sequence, selection and iteration

The sequence of tasks *A* and *B* can be modeled simply by calling *method A* and subsequently calling *method B*. For selecting, the *if-else* construction is used. It is, however, important that the condition be expressed very precisely and the random object of the domain be used within it. Finally, iteration is made with the construction *while* and again a guard condition. As long as the expression of condition *repeat* is valid, it results in the calling of *method A* and *B*. The last pattern for parallelization is more complex, because Python does not have native inherent support for concurrent execution. It is necessary to implement extra functionality for the initiation and synchronization of concurrent processes. One of the possible solutions is in Fig. 4.

```
#Concurrent processess
#p1 and p2
     def p1(p, o):
      o.B(p)
     def p2(p, o):
      o.C(p)
```

```
def MyProcess(p, o):
      o.A(p)
      p.newProcess(p1, o)
      p.newProcess(p2, o)
      p.join(p1, p2)
      o.D(p)
```

Fig. 4: Solving parallelization

The figure shows that the function describing the whole process algorithm as well as each task method has to obtain as a primary parameter a reference for the instance of process *p*. Although this entry is more complicated, it is essential, because each task must know the process within which it is being executed, similar to how each method knows its context object. The process instance also contains methods for creating parallel processes. In Fig. 4, it shows that firstly it is necessary to define the functions of the processes of both concurrent flows *p1* and *p2*, which execute tasks *B* and *C*. To begin, the main process itself performs task *A* and then creates the instance of the defined processes with the help of the method *newProcess()*. Synchronization ensures the calling of the method *join()* on the main process instance *p*. In the end, task *D*, the final task, is executed.

Now it is possible to define the model process from Fig. 1 with the help of the established constructions. The resulting description is shown in the figure below. The existence of classes defined in Fig. 2 is presumed.

```
def MyProcess(p, performer, orgUnit, condition):
   performer.A(p)
   if (condition):
      performer.B(p)
   else:
      data = performer.C(p)
      orgUnit.D(p, data)
      p1 = p.newProcess(lambda p, orgUnit: orgUnit.F(p))
      p2 = p.newProcess(lambda p, orgUnit: orgUnit.G(p))
      p.join(p1, p2)
      orgUnit.G()
   performer.H()
```

Fig. 5: Model of processes in Python

The function *MyProcess* is parameterized by the objects of the process participants and the data *condition*. It is worth noting the simple and straight-forward method of transferring *data* from task *C* to task *D*. The lambda function is used for a more compact description of the parallel execution of tasks *F* and *G*. This is a short form for function definition with the arguments *p* and *orgUnit*; after ":" the function body follows.

In conclusion, we will evaluate the model in Python based on the same criteria as in BPMN and BPEL:

1. **Possibility of direct interpretation of the model through the workflow system**
   The model is directly executed through an interpreter of Python. The workflow system implements only the operation itself based on the given model.
2. **Support of basic workflow patterns**
   All the basic patterns are supported. The most complicated is the definition of parallel execution.
3. **Clarity and comprehensibility of the model**
   The model is well structured and comprehensible. However, the entry is only readable by an IT specialist and not by others (e.g. team leaders). This limitation is necessary if general programming language is used.
4. **Compactness of the description and its subsequent maintenance**
   The model is as compact as possible. High quality developmental environment and debugging tools can be used for its maintenance.
5. **Ability to use domain objects**
   The model is defined right above the domain objects, without any excessive mapping.
6. **Portability of the process model**
   The model is easily portable, because it uses Python code, which is supported by various platforms.

Conclusion

Currently there is a wide collection of languages which differ in their syntax and semantics. Two important groups of languages can be identified – graph-oriented and block-oriented. On the basis of six criteria, two often used languages are evaluated. From the group of graph-oriented languages, the BPMN standard is used; of the block-oriented ones, the BPEL language is used.

Furthermore, the article presents a different method of defining processes with the help of a high-level universal programming language. The aim of the presented definition of process is the emphasis on the compactness and readability of the model and especially the ability to make use of domain objects that are important for executing tasks in the modeled process. The description of the process is based on Python scripting language. What is demonstrated is the ability of the language to describe the basic building blocks of the model and four general workflow patterns, which include sequence, selection, parallelization and iteration. There is also a demonstration of the use of language on a practical example and an evaluation based on the same criteria.

So far, an informal description of the language has been done using several sets of examples. The next phase will involve the formal definition of the modelling language, especially showing proof of the transformation of the current process models described in BPMN or BPEL in regards to the presented model. Subsequently, it will be necessary to make a draft of the core of the workflow system that will simulate the execution of tasks based on given definitions. It will be necessary to design a method through which it will be possible to suspend the interpretation of the process algorithm, because executing tasks could take a longer time, whereas the function which represents this task is executed almost immediately.

The solution must ensure the saving of the entire context of the execution and then renew it with respect to the assurance of data consistency.

References

[1] Řepa, V.: *Podnikové procesy: Procesní řízení a modelování*. Grada Publishing, a.s., Praha, 2006. 268 s. ISBN 80-247-1281-4.

[2] Aalst, W., Hee, K.: *Workflow Management: Models, Methods and Systems*. The MIT Press, London, 2004. 384 s. ISBN 978-0262720465.

[3] Weske, M.: *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007. 368 s. ISBN 978-3540735212.

[4] Workflow Management Coalition: *The Workflow Reference Model*, 1995, dostupný na URL: <http://www.wfmc.org/standards/docs/tc003v11.pdf>.

[5] OMG: *Business Process Model and Notation (BPMN): ver. 1.2.* 2009, dostupný na URL: <http://www.omg.org/spec/BPMN/1.2/>.

[6] Juric, M. B., Pant, K.: *Business Process Driven SOA using BPMN and BPEL*. Packt Publishing, Birmingham, 2008. 311 s. ISBN 978-1-84719-146-5.

[7] Mendling, J., Lassen K., Zdun, U.: *Transformation Strategies between Block-Oriented and Graph-Oriented Process Modelling Languages*. In: F. Lehner, H. Nösekabel, P. Kleinschmidt, eds.: Multikonferenz Wirtschaftsinformatik 2006 (MKWI 2006), Band 2, XML4BPM Track, GITO-Verlag Berlin, 2006, ISBN 3-936771-62-6, pages 297-312.

[8] Mendling J., Nüttgens M. a Neumann G.: *A Comparison of XML Interchange Formats for Business Process Modelling*. In F. Feltz, A. Oberweis and B. Otjacques, editors, Proceedings of EMISA 2004, LNI 56, pages 129–140, 2004

[9] Havey M.: *Essential business process modeling*. O'Reilly, 2005. 332 s. ISBN0-596-00843-0.

[10] Silver, B.: *BPMN Method & Style*. Cody-Cassidy Press, 2009. 213 s. ISBN 978-0-9823681-0-7.