Dynamic business process in workflow systems

Maciej Kiedrowicz[1*]

[1] Military University of Technology, Faculty of Cybernetics, Institute of Computer and Information Systems, Warsaw, Poland

Abstract. The article presents the concept of business process modeling by using dynamic processes. An extended definition of the business process allows to construct a mechanism for dynamic selection of actions performed in order, which makes it possible to achieve the process goal assumed in the definition. The mechanism for selecting actions subsequently taken in the dynamic process constitutes extension of the functionalities in the workflow process, including the process progress status, WfMS and service environments.

1 Introduction

The process approach applied during the modeling, design, implementation and use of IT systems makes it necessary for the authors to model and design such business processes that shall constitute fundamental and necessary structure elements of the system based on the process approach [1, 2, 4, 6]. It is possible to model basically all of the processes used in the company's operations, however, such a comprehensive approach would cause a significant increase of complexity and hence the costs related to the implementation of such system. If the system based on the process approach is implemented in the incremental form, it means that the processes or groups of processes, which should be modeled or implemented in the first row, need to be distinguished. If the organization already uses the process automation systems, the analysis of definitions and instances of such processes may contribute to the decisions made with respect to their efficiency and quality, using methods and tools for process analyses - process warehouses [7-14]. The main selection criteria include the processes: that are most often implemented within the organization (quantitative factor), that have significant impact on efficiency and operating costs of the company (economic factor) and that have significant impact on the implementation of the company's main targets (strategic factor). On the other hand, the processes rarely implemented or characterized by high variability of their structure require a lot of effort on the part of creators of the business processes, which consequently results in the fact that such processes often have very complex structures. In such processes, the correlation between the cost of their implementation and potential profit during their automated processing may be largely unsatisfactory. It is possible to model and design such processes that would not require from their creators any high structural precision at the beginning [15-18]. The aforesaid processes include adaptation, generic and dynamic processes.

2 Process modeling

One of the most important stages of creating a process is its modeling. It is usually the analyst working with different business employees, responsible for a given business process implemented within the organization and specialized in a given field, who is entrusted with the task of building the aforementioned model. During the modeling process, persons responsible therefor, must identify actions in a given business process, which are both mandatory and optional. Apart from that, additional aspects of each of the aforesaid actions are determined: business owner, input and output data and their necessity, errors that may occur while performing a certain action and methods for dealing with such errors.

As a result of the business process modeling, formalized records are obtained. The format, in which the process model is presented, may differ depending on the adopted method and standard. The form is usually graphical (but may be also different), relatively easy to understand and maintain, with the support of appropriate tools. Currently, the most often applied form of graphic notation is BPMN v 2.0 [3].

The processes may be modeled in different ways, but the best results are produced by one of the verified modeling strategies [25-28]. There are many modeling strategies, but none of them is universal - for one company one strategy may be more efficient than the other, and for another company - rather contrary. We may distinguish four basic strategies [2]:

• top-down - according to this strategy, the modeling of a business process should start from the definition of general elements and then, at later stages, additional details may be added to reach the definition at the lowest level. The main advantage of this strategy is that it focuses around major business needs and then assigns particular detailed actions to be taken with respect thereto. The disadvantage is that the strategy requires

* Corresponding author: maciej.kiedrowicz@wat.edu.pl

extensive knowledge and experience - in case of complex processes, any mistakes made at a higher level may cause problems during the modeling of more detailed business processes.

• bottom-up - the strategy opposite to the one described above. The process modeling starts from the detailed elements of processes and subprocesses, which - when combined into increasingly larger groups - shall eventually create the whole business process. The advantage of such approach is the fact that it results in an accurately designed process, where all detailed actions, products and subprocesses are of great importance. However, one of the drawbacks is that the created business process model may turn out to be too detailed and complex and hence not fully compatible with the general business needs of the organization.

• inside-out (dissemination) - the strategy may constitute a compromise between the two above-mentioned approaches. First of all, the main and most important processes, including their details, are defined, and then some elements, subprocesses and supporting tasks added. The difficulty in determining which tasks (or processes) are the most important may be considered a disadvantage, since any mistakes made in this case can lead to the necessity of re-starting the modeling process.

• mixed - this approach is a hybrid of all of the above-mentioned strategies. Intelligent use of the advantages related to other strategies helps to achieve better results and more efficient modeling, while minimizing the impact of drawbacks resulting from the application of the particular strategies separately.

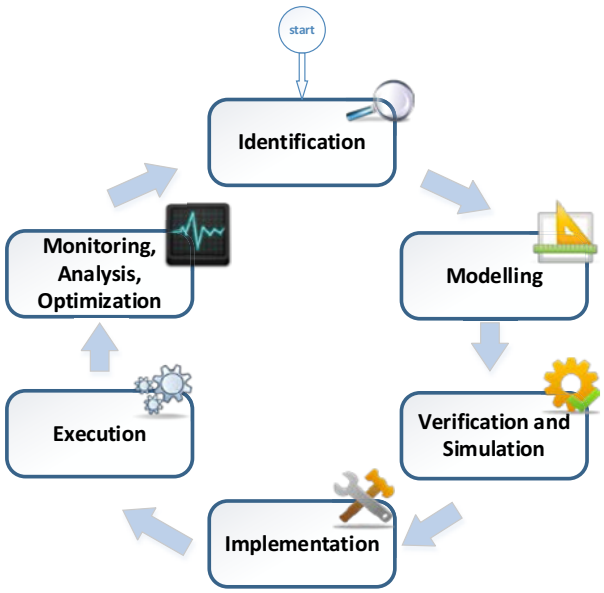

Fig. 1. Life cycle of business process.

The process approach is characterized by constant and cyclical sequence of steps, which are aimed at modeling, implementing and improving processes. The sequence of such steps executed cyclically is defined as the process life cycle (Fig. 1).

3 Ad-hoc, adaptive and generic processes

The process models applied in the mainstream process modeling must be complete in terms of achievement of the assumed goals. It means that the authors must develop business processes which include many situations and cases [19-22]. Based on such approach, the processes are usually very complex and susceptible to unexpected changes, which may occur in their environment, e.g. amended rules and regulations. Therefore, an alternative to the said approach in the process modeling might be ad-hoc, adaptive or generic processes.

The ad-hoc processes were intended to introduce an element of ambiguity and dynamics obtained during the process implementation, with initially non-defined form of such process, in whole or in part. They may be applied at a given place (as part of a certain activity) under the undefined subprocess, whose actual implementation means the choice of a particular set of processes possible to use at a given time. Currently, the most often applied method of defining subsequent processes under the ad-hoc process is manual selection of the processes by system users. Fig. 2 shows basic definition of the process using the ad-hoc process.

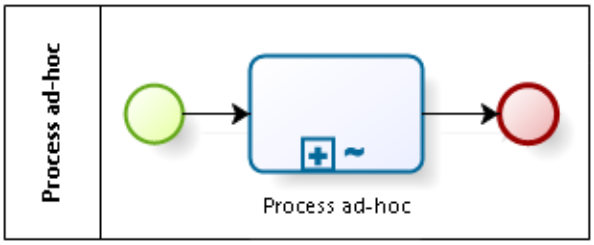

Fig. 2. Example of simple business process with ad-hoc process

Adaptive processes are the processes which may be adapted to new conditions encountered while implementing the process, in a simple manner, without any significant labor input, time consumption or financial contributions. The adaptation may be controlled by a person or automatically supported (in whole or in part). The current workflow systems support the said processes in a very limited manner, by using ad-hoc actions, which - however - need to be forecasted by the process designer prior to the process creation. Full adaptation to new, unpredicted conditions may be achieved with the help of the mechanism of dynamic processes.

Generic processes are universal and may be used to implement any general process, in case of which further clarification takes place at the time of implementing a given instance of the process. Current standards on the process modeling do not provide for a possibility of creating the aforementioned generic processes.

2.1 Dynamic business processes

The aforementioned types of processes: ad-hoc, adaptive or generic are aimed at developing a general and undetermined definition of the process, thanks to which such process would be adaptable to variable conditions [23, 24]. One of the possibilities is to dynamically adapt the process to the assumed target in the variable conditions. Such business processes may include dynamic processes. The processes may constitute extension of the idea of ad-hoc processes, where after determining the goal and indirect goals of the process, the sequence of the actions would be automatically or semi-automatically defined (Fig.4). Selection of appropriate processes and actions to be taken to achieve the process goal or goals requires definition of additional properties for each process or action, which may be used as part of dynamic selection of next steps of the process to accomplish a given objective. When selecting the processes or actions in the subsequent steps, it is necessary to consider the changing process environment, due to which the implementation of the processes with the same goal does not have to have the same form. Such selection requires fulfillment of initial conditions for the performance of a given action. The factors determining such initial conditions of the process use may include, for example, the general process status (which may be permanent or affected by individual actions), previously performed actions or data obtained from the previously performed actions. The conditions may be obligatory or needed to support the selection of a given action. In combination with the definition of weights, we shall obtain the mechanism allowing to determine the limiting conditions or conditions permitting to benefit from the possibilities of a given process, based on current status of the process instance, process definition, condition of the workflow environment and its surrounding (*pre*section – Fig.4). When checking the initial conditions (both obligatory and supporting), we may obtain a hint whether it is possible to use the said actions and processes in the process instance, at a given time of implementing such process instance. By using the aforementioned mechanisms and obtained results, we are provided with a special tool supporting the decision-making process regarding the choice of appropriate actions and processes in the next steps of a given process instance.

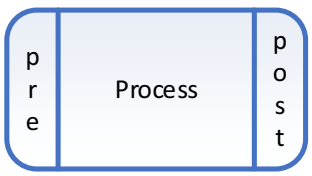

Fig. 4. Schema of the dynamic process with pre and post sections [7]

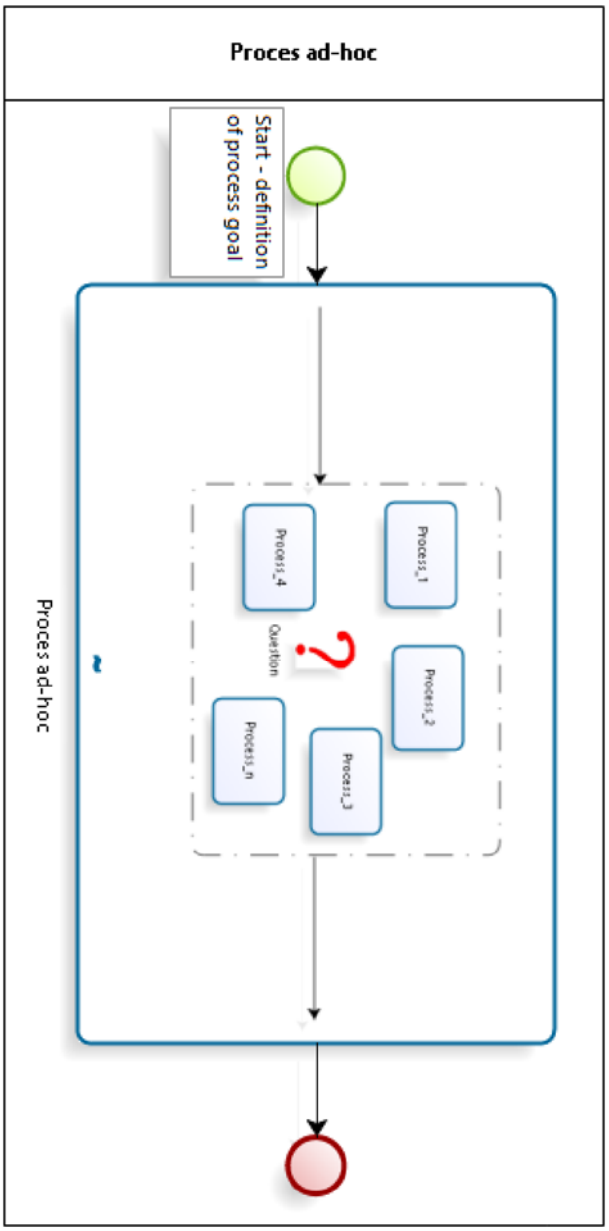

Fig. 5. Ad-hoc process as the process of dynamic selection of actions

Apart from the conditions for using a given process and their weights, the *pre* section may allow to define a set of guidelines and instructions influencing the status of a given process instance, workflow environment and its surrounding. Such a set of guidelines and instructions may be included in the *post*-section, which is activated and implemented after the workflow management system performs a given action. Such instructions may have impact on the process status and particular attributes describing the process, which shall result in greater possibilities of managing dynamic process development.

2.2 Structure of *pre* and *post* sections of dynamic process definition

The definition of the standard for recording input conditions of actions (the concept described in the previous chapter) and operations performed after

completing a given action, which may be used to modify the process status, constitutes an important element of the designed extension of XPDL language [5].

The *pre* conditions may have different forms, refer to a simple process parameter, several interdependent parameters, rules, functions, general process status or its current progress. To stimulate the process progress, it is required to store the history of the previous process progress in terms of the order and number of actions performed as well as the process status at a given point in time. The definition of simple conditions regarding single parameters is rather easy and the functionality of their verification in the process runtime environment may be implemented without any problems, regardless of the form of recording. The problem occurs in case of more complex conditions of the structure of rules that may be defined in the *pre*-section. One of the solutions analyzed from the point of view of the project process itself would be to use a language, in which it is possible to record the assumed conditions and rules, e.g. by applying the first order account properties predicate. The application of such mechanism would eventually allow to implement a universal solution, similar to, for example, Prolog language. However, such solution implies the occurrence of a complex program for interpreting conditions recorded in the aforesaid language - this is the task much more complex that the scope of this study. Therefore, even though the creation of such a new language seems a bit exaggerated now, in the future it might turn out that it is quite reasonable and even necessary as part of development and popularization of the dynamic process concept. A compromise solution is to define a set of operators and functions allowing to handle the *pre* conditions and *post* operations, which need to be made available by the runtime environment of dynamic processes. The main disadvantage of the said solution is the lack of such developed universality that would be allowed by the above-mentioned solution. However, a set of the aforesaid functions may be so developed (e.g. thanks to additional libraries) that it offers sufficient universality level and, in some aspects, provides much greater possibilities, as the implementation of such functions may be definitely more extended that the implementation offered by the language of predicates (apparently, we may use another language, e.g. any programming language, which would be interpreted or complied during the implementation, but it would create rather severe problem regarding the issue of security, thus, it is not a good solution). Besides, this solution is much safer, as the implementation of the functions constitutes a part of the runtime environment, not the process itself. Apart from the security issue, another advantage of this solution is better resistance to general errors and basically complete elimination of syntactic errors. Furthermore, the processes using such solution shall be more transparent and easier to interpret.

The basis for the draft definition of the dynamic process is the XPDL standard [5], thus, in case of the basic definition of a part of the process description, the XPDL standard structure is used. Some elements of the process definition, describing the the aspects of dynamics, constitute extension of the definition of the XPDL process standard by adding supplementary structures required to save the dynamic process definition. Modification of the XPDL language refers to a number of aspects:

- recording of actions that might be used in the process,
- recording of security level of actions,
- extension of the *Activity* and *WorkflowProcess* structure by elements required due to the dynamic nature of the process.

The first element shall be implemented by adding the*Type* attribute to the *Activity* node. The attribute may have one of the two following values: *ProcessElement* and *SpareElement*. The activities of the *ProcessElement* type belong to the process flow. The process may be composed of a number of such activities. The activities of the *SpareElement* type belong to a set of activities that might be used in a given process. In case of applying such activity, it shall be copied to the process definition, whereas the *Type* attribute of the copied activity set to the *ProcessElement* value. The runtime environment of dynamic processes may provide access to a set of additional and public actions, which could be implemented in any process, by using e.g. a XPDL document containing the definition of only the *SpareElement* action type. However, every environment may implement such option in any way or not make it available at all.

The action security level shall be recorded through the *SecurityLevel* action attribute. The attribute may adopt one of the two following values:

- *Secure* - secure level
- *Unlimited* - level allowing unlimited modifications

The third aspect requires considerably greater extension of the definition structure determined in the XPDL standard. The first components to be added is the information on the process progress, i.e. actions performed. Inside of the activity node, the node called *Executions* was added, including the information on further executions of a given action in the *Execution* nodes (there might be more than one execution of a given action, e.g. due to the process loop). Every single node of the action execution shall include execution identifier, date and time of commencement of a given action as well as its end time. It may also contain some information on the input and output data of the activities in the form of copied nodes - *InputSet* and *OutputSet*, including defined values of the parameters or artifacts.

Another addition to the standard XPDL shall be the *pre* section including a description of the conditions verified during the dynamic modification of the process definition. Inside the *Activity* tag, the *PreSection* tag shall be added, which contains the *Conditions* tag, which shall include individual conditions within the framework of the *Condition* tags. Each *Condition* tag has the *Type* attribute that adopts one of the three following values:

- *Required* - the condition must be fulfilled in a security mode
- *Supporting* - the condition is a supporting condition, so its fulfillment is not required in any of the security modes

- *Sufficient* - the fulfillment of such condition is sufficient to benefit from a given activity in a security mode, regardless of the results of other *Required* type conditions.

The *Condition* tag may include a single simple condition - *SimpleCondition* or complex condition - *And* or *Or*. The *And* and *Or* tags may contain other *And* or *Or* tags as well as *SimpleCondition*. To meet the main condition, all other conditions contained in the *And* tag or any condition in the *Or* tag must be met in accordance with the principles of the Boolean algebra. The *And* and *Or* tags group other conditions, and their name defines behavior, this, they have no additional attributes. The *SimpleCondition* tag is defined by the operator in the *Type* attribute. The operator types of such operators are the following: *Eq* (equal to), *Ne* (not equal to), *Lt* (less than), *Le* (less than or equal to), *Gt* (greater than), *Ge* (greater than or equal to), *Contains*. The *SimpleCondition* tag includes two tags defining *LValue* and *RValue* of the operation. Each of them has two attributes: *DataType* - a type of the parameter data and *Type* - a type of the parameter. The *Type* attribute may adopt one of the three values:

- *Value* - a value of the feature,
- *Parameter* - a parameter used in the process as an internal *Parameter* element, with the *Type* (defining the parameter type) and *Name* (defining the name of the parameter in the process) attributes.
- *Function* - the function is used if the value is defined by the function stipulated in the *Name* attribute of the internal *Function* tag. If the function adopts certain input parameters, the *Function* tag must include a set of such parameters. The parameters are grouped through the *FParameters* tag. The single parameter is defined by the*FParameter* tag, which is a copy of the following *tags: LValue*/*RValue*, including the additional *Name* attribute, which defines name of the parameter. It should be emphasized that even though, in this case, the structure allows for practically unlimited incorporation of the results of the function operation, as an input parameter of another function, the runtime environment of the dynamic processes shall always somehow limit this possibility.

The *WorkflowProcess* tag was supplemented with the *Goal* attribute, whereas the *PreSection* tag - with a set of the *GoalWeights*/*GoalWeight* tags, including *Name* and *Weight* attributes, defining weights for particular targets.

The last element, which should be added, is the definition of the post-section, where the operations performed after the completion of a given action are defined. The post-section is defined via the *PostSection* tag, which may cover any number of operations. Every operation is represented by the *Operation* tag, with one *Type* attribute, defining the type of such operation (operator). The *Is* operator functions similarly to other operators known from the *pre* section; the sole difference is the that only the process parameter may constitute L-value. The *Procedure* operator allows to perform any function, which does not return any result - the structure of the *Function* tag is identical to the one described above. All post-section operations should be performed in order according to the record in the XPDL structure.

Below are the selected fragments of the process definition, in the form of the modified XSD structure for XPDL format, extended by the previously described tags, used to construct dynamic business processes.

```
<xsd:attribute name="Type">
 <xsd:simpleType>
  <xsd:restriction base="xsd:NMTOKEN">
   <xsd:enumeration value="ProcessElement"/>
   <xsd:enumeration value="SpareElement"/>
  </xsd:restriction>
 </xsd:simpleType>
</xsd:attribute>

<xsd:attribute name="SecurityLevel">
 <xsd:simpleType>
  <xsd:restriction base="xsd:NMTOKEN">
   <xsd:enumeration value="Secure"/>
   <xsd:enumeration value="Unlimited"/>
  </xsd:restriction>
 </xsd:simpleType>
</xsd:attribute>

<!--WorkflowProcess-->
<xsd:attribute name="Goal" type="xsd:string"/>


<xsd:element name="PreSection" maxOccurs="1">
 <xsd:complexType>
  <xsd:element name="Conditions" maxOccurs="1">
   <xsd:complexType>
    <xsd:element name="Condition" minOccurs="0">
     <xsd:complexType>
      <xsd:choice minOccurs="0">
       <xsd:element ref="xpdl:And"/>
       <xsd:element ref="xpdl:Or"/>
       <xsd:element ref="xpdl:SimpleCondition"/>
      </xsd:choice>
      <xsd:attribute name="Type">
       <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
         <xsd:enumeration value="Required"/>
         <xsd:enumeration value="Supporting"/>
         <xsd:enumeration value="Sufficient"/>
        </xsd:restriction>
       </xsd:simpleType>
      </xsd:attribute>
     </xsd:complexType>
    </xsd:element>
   </xsd:complexType>
  </xsd:element>
  <xsd:element name="GoalWeights" maxOccurs="1">
   <xsd:complexType>
    <xsd:element name="GoalWeight"
minOccurs="0">
     <xsd:complexType>
      <xsd:attribute name="Name" type="xsd:string"/>
       <xsd:attribute name="Weight"
type="xsd:integer"/>
     </xsd:complexType>
    </xsd:element>
   </xsd:complexType>
```

```
    </xsd:element>
  </xsd:complexType>
</xsd:element>

<xsd:element name="PostSection" maxOccurs="1">
  <xsd:complexType>
    <xsd:element name="Operation" minOccurs="0">
      <xsd:choice minOccurs="0">
        <xsd:complexType>
          <xsd:element ref="xpdl:Function"/>
          <xsd:attribute name="Type">
            <xsd:simpleType>
              <xsd:restriction base="xsd:NMTOKEN">
                <xsd:enumeration value="Function"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:attribute>
        </xsd:complexType>
        <xsd:complexType>
          <xsd:all>
            <xsd:element name="LValue">
              <xsd:element ref="xpdl:Parameter"/>
            </xsd:element>
            <xsd:element                    name="RValue"
type="ValueType" />
          </xsd:all>
          <xsd:attribute name="Type">
            <xsd:simpleType>
              <xsd:restriction base="xsd:NMTOKEN">
                <xsd:enumeration value="Is"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:attribute>
        </xsd:complexType>
      </xsd:choice>
    </xsd:element>
  </xsd:complexType>
</xsd:element>

<xsd:element name="And">
  <xsd:complexType>
    <xsd:element ref="xpdl:And" minOccurs="0"/>
    <xsd:element ref="xpdl:Or" minOccurs="0"/>
    <xsd:element              ref="xpdl:SimpleCondition"
minOccurs="0"/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="Or">
  <xsd:complexType>
    <xsd:element ref="xpdl:And" minOccurs="0"/>
    <xsd:element ref="xpdl:Or" minOccurs="0"/>
    <xsd:element              ref="xpdl:SimpleCondition"
minOccurs="0"/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="SimpleCondition">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="LValue" type="ValueType" />
      <xsd:element name="RValue" type="ValueType" />
    </xsd:all>
    <xsd:attribute name="Type">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="Eq"/>
          <xsd:enumeration value="Ne"/>
          <xsd:enumeration value="Lt"/>
          <xsd:enumeration value="Le"/>
          <xsd:enumeration value="Gt"/>
          <xsd:enumeration value="Ge"/>
          <xsd:enumeration value="Contains"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="ValueType">
  <xsd:choice minOccurs="0">
    <xsd:complexType>
      <xsd:choice minOccurs="0">
        <xsd:element ref="xpdl:Parameter"/>
        <xsd:element ref="xpdl:Function"/>
      </xsd:choice>
      <xsd:attribute name="Type">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="Parameter"/>
            <xsd:enumeration value="Function"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="DataType">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="String"/>
            <xsd:enumeration value="Number"/>
            <xsd:enumeration value="Boolean"/>
            <xsd:enumeration value="DateTime"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType>
      <xsd:simpleContent>
        <xsd:extension base="xs:string">
          <xsd:attribute name="Type">
            <xsd:simpleType>
              <xsd:restriction base="xsd:NMTOKEN">
                <xsd:enumeration value="Value"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:attribute>
          <xsd:attribute name="DataType">
            <xsd:simpleType>
              <xsd:restriction base="xsd:NMTOKEN">
                <xsd:enumeration value="String"/>
                <xsd:enumeration value="Number"/>
                <xsd:enumeration value="Boolean"/>
                <xsd:enumeration value="DateTime"/>
              </xsd:restriction>
            </xsd:simpleType>
```

```
        </xsd:attribute>
       </xsd:extension>
     </xsd:simpleContent>
   </xsd:complexType>
  </xsd:choice>
</xsd:complexType>

<xsd:element name="Parameter">
  <xsd:complexType>
    <xsd:attribute name="Type">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="String"/>
          <xsd:enumeration value="Number"/>
          <xsd:enumeration value="Boolean"/>
          <xsd:enumeration value="DateTime"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="Name" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="FunctionParameter">
  <xsd:complexContent>
    <xsd:extension base="ValueType">
      <xsd:attribute name="Name" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Function">
  <xsd:complexType>
    <xsd:element name="FParameters" maxOccurs="1">
      <xsd:complexType>
        <xsd:element name="FParameter" minOccurs="0"
base="FunctionParameter" />
      </xsd:complexType>
    </xsd:element>
    <xsd:attribute name="Name" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

3 Conclusion

The above material contains the results of studies related to the automated modeling and construction of the dynamic business processes. The presented process definitions supplemented with the *pre* and *post* sections allow to increase the automation level of the process of selecting subsequently executed actions, based on the set process goal. The application of the mechanisms for defining rules of availability of a given action in terms of the specific process progress and runtime environment and its surrounding allow to automate or support self-defining and self-organizing business processes. The presented mechanism has also adaptive properties, since - in case of the target set for the process, its implementation may be each time different. The above depends on the progress of the process, variable set of available activities, changing status of the workflow environment as well as changing status of the runtime environment. The described mechanism for defining and implementing dynamic business processes refers to the area of process modeling and supporting decisions made with respect to the automatic selection of subsequent actions aimed at realizing the target defined in the process. In the following steps, the research should be directed at verification and potential modification of the method for defining dynamic processes, development of formal method for determining selection of further steps, including, among other things, reduction of costs and time needed to implement the process. Another additional and significant aspect is the dynamic and automated selection of some executive services, available at a given time for the performed activities, including their cost, resources and time. Such a solution would allow to automate the area of the business process modeling and definition as well as the selection of executive services of such processes.

References

1. 1-M. Weske, *Business Process Management. Concepts, Languages, Architectures*, Berlin, Springer-Verlag. (2007).
2. 2-M. Sałaciński, *Modelowanie procesów biznesowych. Praktyczne wykorzystanie BPMN*, Software Developer's Journal, vol. **04** (2010).
3. 3-OMG, BPMN - standard documents, [Online] http://www.omg.org/spec/BPMN/2.0/ (2011)
4. 4-A. Arsanjani, N. Bharade, M. Borgenstrand, P. Schume, J. K. Wood i V. Zheltonogov, *Business Process Management Design Guide. Using IBM Business Process Manager*, USA, IBM Corporation. (2015).
5. 5-WFMC, XML Process Definition Language – standard XPDL documents, [Online] http://www.xpdl.org/ (2012).
6. 6-W. van der Aalst, K. van Hee, *Workflow Management. Models, Methods, and Systems,* The MIT Press Cambridge, Massachusetts London, (2002).
7. 7-G. Bliźniuk, M. Chmielewski, T. Gzik, J. Koszela, Warehouse of processes, *Studia Informatica*, vol. **33-2A**, 111-127. (2012).
8. 8-W. van der Aalst, *Process Mining*, Springer-Verlag, Berlin. (2016).
9. 9-M. Kiedrowicz, *Zarządzanie informacjami wrażliwymi - wybrane aspekty organizacyjne, prawne i techniczne ochrony informacji niejawnych* WAT Warszawa, (2015).
10. 10-J. Koszela, Business process modeling for processing classified documents using RFID technology in *20th International Conference on Circuits, Systems, Communications and Computers*, MATEC Web of Conferences, vol. **76**, (2016)
11. 11-M. Kiedrowicz, J. Koszela, *Secret office model for the processing of classified documents using RFID technology*, Collegium of Economic Analysis Annals, pp. 67-81, vol. **42**. (2016).
12. 12-R. Waszkowski, M. Kiedrowicz, T. Nowicki, Z.

Wesolowski, and K. Worwa, Business processes in the RFID-equipped restricted access administrative office, in *20th International Conference on Circuits, Systems, Communications and Computers*, MATEC Web of Conferences, vol. **76**, DOI: 10.1051/matecconf/20167604003 (2016).
13. 13-R. Dijkmana., *Similarity of business process models: Metrics and evaluation*, Information Systems, Elsevier, vol. **36** (2011).
14. 14-W. van der Aalst & others, *Process Mining Manifesto*, IEEE Task Force on Process Mining, [Online] http://www.win.tue.nl/ieeetfpm/downloads/Process%20Mining%20Manifesto.pdf
15. R. Antkiewicz, M. Dyk, R. Kasprzyk, A. Najgebauer, D. Pierzchała, Z. Tarapata, *Criminal Procedure Management Based on BPM Simulation*, Information Systems in Management, pp. 87 – 99, vol. **2**, (2013).
16. D. Pierzchała, R. Antkiewicz, M. Dyk, R. Kasprzyk, A. Najgebauer, Z. Tarapata, *Modelling, simulation and computer support of the Polish criminal procedure*, (in:) Information Systems Architecture and Technology. The Use of IT Technologies to Support Organizational Management in Risky Environment, (Eds.) Z. Wilimowska, L. Borzemski, A. Grzech, J. Świątek, pp. 51-60, (2014).
17. M. Kiedrowicz, T. Nowicki, R. Waszkowski, Z. Wesolowski, and K. Worwa, *Software simulator for property investigation of document management system with RFID tags*, 20th International Conference on Circuits, Systems, Communications and Computers, MATEC Web of Conferences, vol. **76**, DOI: 10.1051/matecconf/20167604012 (2016).
18. R. Kasprzyk, M. Zabielski, P. Kowalski, G. Oksiuta, K. Rzempołuch, *GUARDIAN – Emergency Response System with Incremental Information Delivery Model*, Biuletyn Systemów Informatycznych, pp. 1-6, vol. **11**, (2013).
19. M. Kiedrowicz, *Objects identification in the information models used by information systems*, GIS ODYSSEY 2016, pp. 129-136, (2016).
20. M. Kiedrowicz, *Uogólniony model danych w rozproszonych rejestrach ewidencyjnych*, Roczniki Kolegium Analiz Ekonomicznych, pp. 209-234, vol. **33**, (2014).
21. M. Kiedrowicz, *Location with the use of the RFID and GPS technologies - opportunities and threats*, GIS ODYSSEY 2016, pp. 122-128, (2016).
22. M. Kiedrowicz, *The importance of an integration platform within the organisation*, Zeszyty Naukowe, pp.83-94, vol. **46**, (2014).
23. M. Kiedrowicz, J. Stanik, *Selected aspects of risk management in respect of security of the document lifecycle management system with multiple levels of sensitivity*, (in:) Information Management in Practice, (eds) B.F. Kubiak and J. Maślankowski, pp. 231-249, (2015).
24. M. Kiedrowicz, T. Nowicki, R. Waszkowski, Z. Wesolowski, and K. Worwa, *Method for assessing software reliability of the document management system using the RFID technology*, 20th International Conference on Circuits, Systems, Communications and Computers, MATEC Web of Conferences, vol. **76**, DOI: 10.1051/matecconf/20167604009 (2016).
25. M. Kiedrowicz, T. Nowicki, R. Waszkowski, *Business process data flow between automated and human tasks*, 3rd International Conference on Social Science (ICSS 2016) December 9–11 2016, pp. 471-477, (2016).
26. M. Kiedrowicz, *Rejestry i zasoby informacyjne wykorzystywane przez organy odpowiedzialne za wykrywanie i przeciwdziałanie przestępczości*, (in:) Jawność i jej ograniczenia, (ed.) G. Szpor, Monografie Prawnicze, pp. 170-264, vol. **IX**, (2015).
27. M. Kiedrowicz, *Dostęp do publicznych zasobów danych - Big data czy Big brother*, (in:) INTERNET. Publiczne bazy danych i Big data, (ed.) G. Szpor, pp. 15-39, (2015). M. Kiedrowicz, R. Waszkowski, *Business rules automation standards in business process management systems*, (in:) Information Management in Practice, (eds) B.F. Kubiak and J. Maślankowski, pp. 187-200, (2015).
28. M. Kiedrowicz, J. Koszela, *Business processes modelling for the processing of classified documents using RFID technology*, Collegium of Economic Analysis Annals, vol. **42,** pp. 53-66, (2016).