A Scientific Workflow Engine Design and Implementation

Zhichao Zhang[1,2,3,a], Meng Guo*[1,2,b], Tingting Guo[4,c]

[a]nyzhangzc@foxmail.com, [b]guomeng@sdas.org, [c]catree98@qq.com

[1]Key Laboratory of' Computing Power Network and Information Security of Ministry of Education, Shandong Computer Science Center (National Supercomputing Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan, Shandong 250013, P. R. China
[2]Shandong Provincial Key Laboratory of Computer Networks, Shandong Fundamental Research Center for Computer Science, Jinan, Shandong 250013, P. R. China
[3]Inspur General Software Co. Ltd., Jinan, Shandong 250101, P. R. China
[4]Shandong Carbon Manager Group Co. Ltd., Jinan, Shandong 250101, P. R. China

Abstract. This paper designs and implements a scientific workflow engine that follows the BPMN 2.0, enriching existing research on scientific workflow engine specification and theory. The engine is implemented using the Java language and adopts a layered architecture, including a process virtual machine, a BPMN model engine and a scientific workflow model engine from the bottom up, which has good scalability and is able to provide powerful support for the design and operation of scientific workflows so as to process scientific research data more efficiently.

Keywords: Scientific workflow; workflow; workflow engine; BPMN 2.0

1 Introduction

The current data overload such as experimental, sensor, satellite and biology data[1] makes it almost impossible to handle complex computational processes manually any more, and thus scientific workflow engines have become an indispensable tool, which plays an important role in the study of geophysics of the atmosphere, oceans, seismic sounding, cosmology, and other geophysical sciences.

There are already some popular Scientific Workflow Management Systems (SWFMS) in various research fields, among which the more representative ones are Kepler[2], which is oriented to the fields of biology and astronomy, and Taverna [3], which is mainly used in the field of molecular biology, in addition to Galaxy[4] and Nextflow[5]. However, since each scientific workflow engine is more oriented to its own research area and started relatively late, there is no unified theory and specification, and the self-contained system is not easy to be extended and integrated. In addition, the modelling of workflow activities is more in the way of Petri nets[6][7] or UML diagrams[8], and the model elements are more obscure and not easy to understand.

To address these issues, this paper implements a set of highly scalable and highly available scientific workflow engine to meet the needs of multiple aspects, which makes the following improvements compared with the existing scientific workflow engine. Firstly, the engine

PMIS 2024, March 15-17, Changsha, People's Republic of China
Copyright © 2024 EAI
DOI 10.4108/eai.15-3-2024.2346547

needs to strictly follow the BPMN 2.0 specification [9][10][11], which has the constraints of a unified theory and specification, and can form a specification of the engine within the scientific workflow domain. Secondly, this engine as an easily portable and extensible component with few third-party dependencies. Finally, the engine is implemented using a layered architecture, with the model-independent process virtual machine PVM extracted from the bottom layer, and the BPMN model engine BPMN Engine extended based on the PVM. Based on this engine can continue to extend the scientific workflow engine SciWf Engine, including this paper, applicable to a variety of business scenarios workflow engine, the engine also has the ability to support other process modelling languages such as jPDL, BPEL[12].

2 Process virtual machine design and implementation

2.1 Process Virtual Machine Overview

The Process Virtual Machine (PVM), which draws on the design concepts of the open source workflow engine Activiti[13], is a set of abstract APIs with no concrete implementation, and is used to describe the various possibilities in terms of workflow through the API. PVM just provides the execution environment and related APIs needed by the process activity nodes, and actually does not implement any activities, which makes PVM can be better adapted to a variety of different workflow domain languages. However, because of the simplicity of the definition , PVM in the actual implementation of the transformation into a lot of additional features in order to complete the framework requirements . PVM using the Java language implementation , does not rely on any third-party libraries or frameworks .

2.2 Key Classes for PVM

The classes in the design period are mainly used to describe the properties of the process model, the base class is PvmElement with the ElementId attribute to serve as a unique identifier, and its subclasses are PvmProceessDefinition, PvmActivity, and PvmTransition. The behaviour of each activity is stored in the PvmActivity through the activityBehavior property whose type is IPvmActivityBehavior.

Runtime classes are mainly used for the process of running the process, the base class is the execution of PvmExecution, its subclasses are PvmProcessActivity, PvmActivityInstance and PvmTransitionInstance, each subclass is responsible for providing a specific process scheduling, for example, PvmActivityInstance will execute the behaviour of the activity recorded in PvmActivity.

2.3 Process Scheduling for PVM

PVM's process scheduling is quite flexible, it records where the current process is executing through the PvmProcessActivity and enables the process to flow correctly by moving the PvmProcessActivity. The PVM defines a series of atomic operations to make the process work.

Atomic operation is one of the most basic single-step operations during the execution of process instances, which is essentially an event producer that can notify the listener to execute event processing logic during the execution of each single-step operation. Atomic operation adds control logic to event notifications, such as starting an activity, and ending a process, and so on.

Figure 1 depicts the PVM's execution of atomic operations during the forward flow of a process, with light-coloured nodes representing objects related to process instance and the methods they contain, and blue-coloured nodes representing atomic operations. Figure 2 depicts the execution of atomic operations during a process interruption.

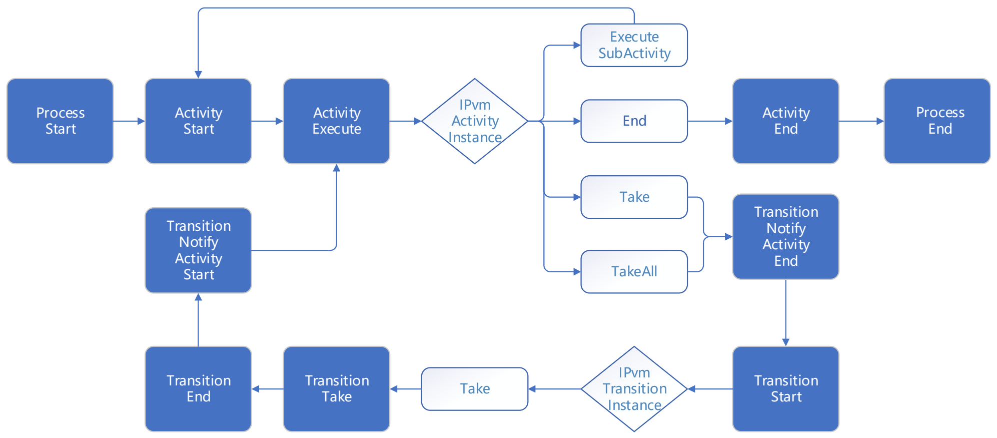


Figure 1. Atomic operations for process forward flow scenarios.

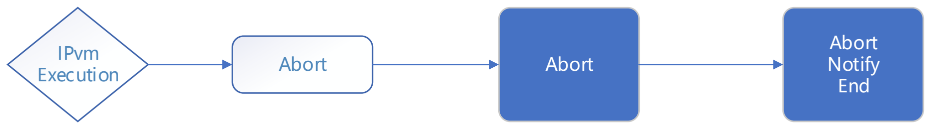


Figure 2. Atomic operations for process interruption scenarios.

3 Bpmn engine design and implementation

BPMN engine is a process resolution engine for the BPMN 2.0 specification, a specific implementation of PVM under the BPMN2.0 specification. The BPMN engine uses the Java language like the PVM. The BPMN engine relies only on PVM and a few libraries related to JSON operations.

3.1 Formal definition of the BPMN model

In this paper, we use directed graph for process model construction, based on the relevant specification of BPMN 2.0, the process model is a directed model graph constructed by the model elements such as start event, end event, task, exclusion gateway, parallel gateway, sequential flow, sub-process, etc., which is finally converted into a process model file in xml format.

(1) The complete process model contains start and end events, with the start event as the process entry and the end event as the process end marker.

(2) The task is a node of activity that cannot be subdivided.

(3) The sub-process can be refined into another process.

(4) Sequence flow describes the process operation and data flow direction, which is used to connect tasks, gateways, events and sub-processes with inputs and outputs.

(5) Tasks, gateways, and sub-process nodes should have input and output streams to ensure that processes are connected and reachable.

In response to the above model definition, an example flow diagram is given in this paper, as shown in figure 3.

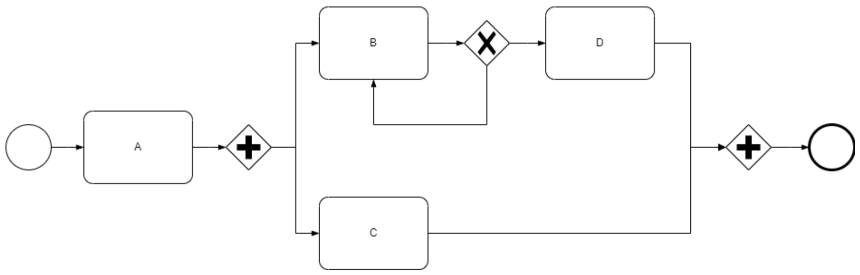
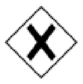
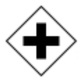


Figure 3. Example of BPMN modelling process.

According to Figure 3, it can be seen that a standard process model definition consists of multiple model elements, including process entry and termination, data flow control, process routing branches and task nodes and other elements, through the organic combination of the model elements to build a complete, practical use of the process model definition. A detailed description of each of the main model elements is given below, with specific data shown in Table 1.

Table 1. Process Model Element Data Interpretation.

Modelling element	Description	Symbol
StartEvent	It indicates the start of the process and is the entry point for the execution of the process.	○
EndEvent	It indicates the end of the process and is the terminating element of the process.	◯
Task	Processing of specific operations.	▢
SequenceFlow	Connector between two model elements.	→

Modelling element	Description	Symbol
ExclusiveGateway	It is used to model the decisions in the process.	
ParallelGateway	It is used for process modelling of concurrent tasks.	

3.2 Structured design of BPMN models

Workflow model is the foundation and core of workflow technology, and an unreasonable workflow model will cause irreparable losses once it is put into actual operation. Therefore, designing and building a standardised workflow model is the key to using workflow technology.

The standardised process model should contain a StartEvent, an EndEvent and a SequenceFlow at least, and the data can flow to all task nodes. If there is a branching structure, it needs to be implemented by gateways, and the process data must be closed-loop. The complexity of the process model is mainly reflected in the construction of mutually exclusive and synchronous branches and sub-processes, so we give a standard specification for the use of mutually exclusive branches, parallel branches and sub-processes.

(1) Standardization of mutually exclusive structures

Process models that use mutually exclusive structures must be constructed using an ExclusiveGateway, and execution expressions should be created for the output streams of the ExclusiveGateway so that the value of the expression can be computed to select a branch when routing is performed. The example of the mutually exclusive structure is shown in Figure 4. There are two branches that can be executed after the execution of node A, and the engine needs to calculate the expression in the output stream based on the input variables, in order to select the execution node B or node C.

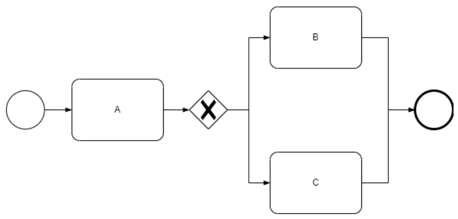


Figure 4. Example of mutually exclusive structure.

(2) Standardization of parallel structure

If a parallel execution structure is required in the process model, a ParallelGateway is used as a way to execute branches in parallel. It should be noted that ParallelGateway can split and merge branches, i.e., a parallel structure should have two ParallelGateway nodes. The example of parallel structure is shown in Figure 5. After the execution of node A, the ParallelGateway node will split the process into two branches so that node B and node C can be executed concurrently, and then the second ParallelGateway will merge the branches and flow to the end node.

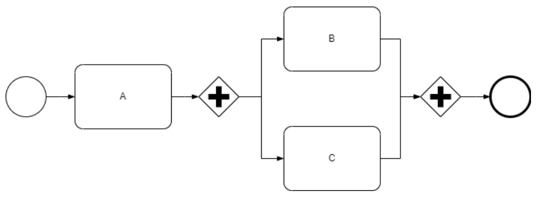


Figure 5. Example of parallel structure.

(3) Standardization of sub-process

Sub-processes are used to solve complex embedded processes. BPMN 2.0 requires that the internal elements of the SubProcess should contain complete model elements, while the output streams of the nodes in the process should not be directly connected to the nodes in the SubProcess, but should be connected to the SubProcess nodes. The normalized sub-process structure is shown in Figure 6.

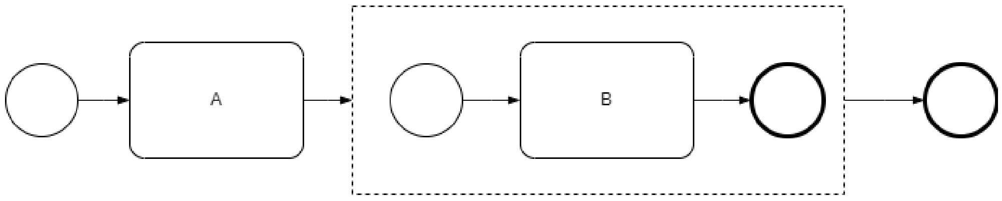


Figure 6. Example of sub-process structure.

By following standardised process model rules, we construct a unified model base that provides solid support for model validation methods. This modelling foundation ensures consistency and accuracy of models, allowing various model validation methods to be manipulated and compared on the same platform. In this way, we provide a more rigorous and standardised methodology for the design and implementation of the scientific workflow engine, thereby improving the reliability and efficiency of model validation. This unified model foundation not only helps to improve the performance of the scientific workflow engine, but also provides clearer and more specific guidance for future model validation efforts.

3.3 Analytical design of BPMN models

This paper uses the process definition tool and build the process model based on the XML specification defined by BPMN 2.0 for the process description file, and after the process model design is completed, it will be saved as a bpmn file and stored in the database. Process model parsing is mainly used to parse model elements and related attributes, such as start and end events, task nodes, gateways, sequence flows, and sub-processes, during process deployment, and store them in the database to avoid repeated parsing of model files when the process is running, so as to improve the efficiency of process operation. The process model file follows the XML specification defined by BPMN 2.0, and each element contains the name and id attributes, and different elements also contain their own unique attributes.

3.4 Implemention of the BPMN model

BPMN Engine consists of two main parts, one is the BPMN model library and the other is the BPMN behaviour library. The model library declares the main activity elements of the BPMN Engine, and the behaviour library defines the specific execution logic of each main activity. The main elements of this engine are described above, but in order to ensure that the entire scientific workflow engine is reasonable and extensible, and also to strictly follow the requirements of the BPMN 2.0, a more complex internal implementation is used in the BPMN model library. The following are the key classes in the model library.

(1) BpmnModel is used to describe a complete flowchart, which has two key properties, defaultProcess and defaultDiagram. The defaultProcess records the activity nodes, sequence flows and their corresponding attribute configurations in the flowchart, and the defaultDiagram records the geometry information corresponding to the elements in the defaultProcess.

(2) BpmnModelElement is the base class for all elements in a BPMN model, and it has only one internal attribute Model, of type BpmnModel, which means that any element in the whole model can be accessed via the Model attribute inside a model element.

(3) Process represents a BPMN process and is the type of the defaultProcess attribute of the BpmnModel class, which inherits from the root element class RootElement, and the Key attribute is a unique identifier of the Process object, which also holds a collection attribute of process elements. Process is the core of a bpmn model, and the process engine will convert the Process object into a structure that the engine can recognize during process execution, and then make the process instance run correctly in accordance with the design graph.

(4) FlowElement has two main subclasses FlowNode and SequenceFlow. FlowNode represents the flow object in the BPMN 2.0, which is the main graphical element that defines the process, and can be extended by the FlowNode class for activities such as StartEvent, EndEvent, Task, ExclusiveGateway, ParallelGateway, and so on, which are supported by this engine. SequenceFlow is represented in the flowchart as a link between nodes, which allows process instances to flow in the order of the link.

(5) OmgdiElement and OmgdcElement and their subclasses are classes used to describe image features, for example Bounds is used to describe the coordinates and size of a node, recorded in the bounds property of the Node class. The Point class is used to record the path points of a connected line, recorded in the Edge class.

3.5 Realisation of BPMM behaviour

The BPMN model in Section 3.4 is only a static description of the process and does not represent an actual process run at a time, so it is also necessary to define behaviours for the BPMN model that correspond to each of its activities, so that a process is able to achieve a real flow based on the behaviours of the different activities.

The most central class in the BPMN behaviour library is FlowNodeBehavior, which is a concrete implementation class of the IPvmActivityBehavior interface in the PVM layer, defining the default execution logic for all sub-classes of FlowNode. In the PVM's atomic operation ActivityExecute, the Execute method of the IPvmActivityBehavior interface is called to execute the specific business logic of the different activity nodes. And the Execute method in FlowNodeBehavior is the default implementation for all subclasses of FlowNode.

In FlowNode, the outgoingGatewayType attribute is used to identify the node exclusive gateway or parallel gateway. According to the different value, the Leave method will enter different branches. The exclusive gateway approach leaves a node with the option of having a default link flowing to the next node, whereas the parallel gateway approach flows the data along all the going links to each succeeding node.

4 Design and implementation of sciwf engine

4.1 Overview of SciWf Engine

Both the PVM and BPMN Engine in the previous section are highly abstract process resolution engines, usually with no code logic associated with a specific business and few third-party dependencies. In this chapter, we implement a scientific workflow engine (SciWf Engine) in conjunction with a scientific workflow business scenario.

The SciWf Engine is an extension of the BPMN Engine, which also consists of a model library and a behavioural library. The model library is inherited from BPMN Engine's model library, while the behaviour library is based on BPMN Engine's behaviour library and imports necessary third-party dependencies to achieve complex and variable functional requirements.

4.2 Architectural design of SciWf Engine

The SciWf Engine adopts a layered architecture to complete bottom-up encapsulation as shown in figure 7.

The core interface layer, SciWfCore, has been defined by PVM.

The basic component layer, SciWfComp, provides basic components and services such as session context, persistent storage, event listening, and cache management.

The core implementation layer, SciWfImpl, defines the key model classes of the BPMN specification and the concrete implementation of FlowElementBehavior.

The command layer, SciWfCmd, uses the Command Pattern to decouple functions. It provides abstract interfaces that allows most client-related requirements to be implemented outside the engine in the form of concrete command implementation classes, such as Pass, Back, Suspend, Resuse, Enable, Deactivate, etc.

The command interception layer, SciWfCmdncptr, adopts the Chain of Responsibility Pattern. The layer is responsible for creating conditions for the execution of commands, such as starting transactions, creating command contexts, logging, and so on.

The business interface layer, SciWfBiz, is designed for specific businesses, and provides various interfaces, such as RuntimeService, RepositoryService, HistoryService, TaskService, etc.

The process engine layer, SciWfEngine, is the gateway to all interfaces. Clients can invoke the apis provided by SciWfEngine to implement specific functions.

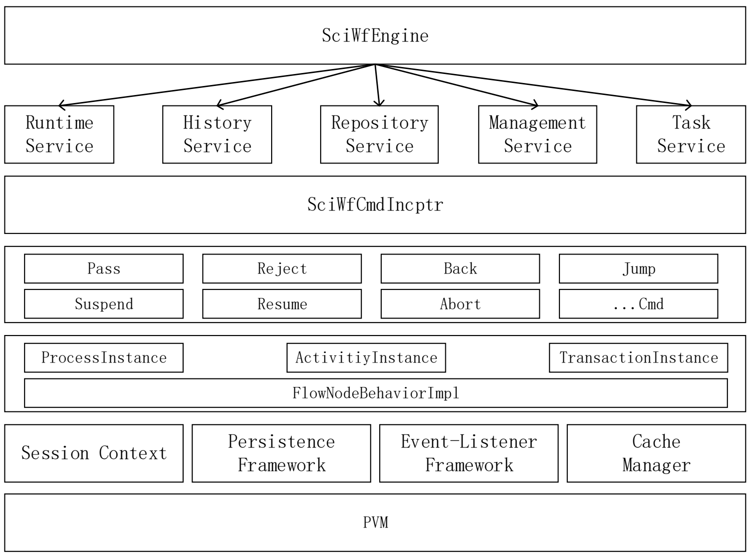


Figure 7. Layered Architecture Design for SciWf Engine.

4.3 Main activities of the SciWf Engine

In order to meet the needs of scientific computing, SciWf Engine is extended based on BPMN2.0, which implements a richer activity model that can meet the needs of computation, transformation, storage and notification.

(1) ComputeTask is the most commonly used activity in SciWf Engine, which is used to execute a computational process, and after the computation is completed, the result is stored in the process instance context and flows to the next node. ComputeTask supports HTTP, direct calls and other execution methods, the input parameter can be variables, constants, etc. in the context of the process instance. Direct call means configuring a Java file for the compute task and dynamically executing the file with the help of Java's reflection mechanism.

(2) MappingTask is used to implement the mapping and conversion between two entity structures, and is usually used to connect two ComputeTask nodes, converting the output of the former task into the parameter structure required by the latter.

(3) ScriptTask is an activity that executes the corresponding JavaScript script when the process execution arrives. The return value of a ScriptTask can be set to a process variable via the resultVar attribute, which can be an already existing or a new process variable. If specified as an existing process variable, the value of the process variable is overridden by the result value of the script execution. If you do not specify a result variable name, the script result value is ignored.

(4) MessageActivity can send messages to a specified user in order to notify the user that a scientific computing process or a session has been completed. The subject and content of the message can be dynamically set through JavaScript scripts, and when the process execution arrives at the message activity, the message sending tool will be called to send the dynamic content to the specified user's mailbox.

5 Conclusions

This paper introduces some concepts related to traditional workflow, scientific workflow and BPMN specification, and designs and implements a set of scientific workflow parsing engine based on Java language. The engine is implemented using a layered architecture, the bottom layer is the PVM process virtual machine, the BPMN model engine is the extension and implementation of the PVM based on following the BPMN 2.0, and the top layer of the SciWf Engine is the further extension and implementation of the BPMN model engine in combination with the business scenarios of scientific computing. This engine can be used as a middleware for a scientific workflow management system, responsible for parsing process models and controlling the startup, running and termination of process instances. The SciWf Engine is highly versatile and extensible, and work that can be continued as follows:

(1) Implementing a parsing engine using other process languages based on PVM.

(2) Developing a complete scientific workflow management system in conjunction with the SciWf Engine.

(3) Implementing embedded sub-processes parsing and running.

Acknowledgements. This work was financially supported by the Shandong Provincial Key Research and Development Program (No. 2022CXGC020106) and the Pilot Project for Integrated Innovation of Science, Education and Industry of Qilu University of Technology (Shandong Academy of Sciences) (No. 2022JBZ01-01).

References

[1] Grossman, RL.: Ten lessons for data sharing with a data commons. Sci Data. Vol. 10, No. 1, pp. 120 (2023)

[2] Yang, P. C. , Purawat, S. , Ieong, P. U. , Jeng, M. T. , Demarco, K. R. , & Vorobyov, I. , et al.: A demonstration of modularity, reuse, reproducibility, portability and scalability for modeling and simulation of cardiac electrophysiology using Kepler Workflows. PLoS Comput Biol. Vol. 15, No. 3, pp. e1006856 (2019)

[3] Gou, C. , Li, J. , Li, Y. , Liu, J. , Zhao, S. , & Xiao, Y. , et al.: Construction of a specialized integrated simulation platform for molecule screening based on scientific computing workflow engine. Sci Rep. Vol. 13, No. 1, pp. 15549 (2023)

[4] Teichman, G. , Cohen, D. , Ganon, O. , Dunsky, N. , Shani, S. , & Gingold, H. , et al.: RNAlysis: analyze your RNA sequencing data without writing a single line of code. BMC Biol. Vol. 21, No. 1, pp. 74 (2023)

[5] Li, Y. , Molik, D. , & Pfrender, M. E.: EPPS, a metabarcoding bioinformatics pipeline using Nextflow. Biodiversity Science. Vol. 27, No. 5 (2019)

[6] Ji, W.: Research on Computer Software Engineering based on Scientific Workflow. pp. 698-703. Proceedings of 3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019), (2019)

[7] Du, N. , Li, Q. , & Liang, Y.: Actor Petri net model for scientific workflows. Proceedings of the 4th International Conference on Ubiquitous Information Management and Communication, ICUIMC 2010, Suwon, Republic of Korea (2010)

[8] Pllana, S. , Qin, J. , & Fahringer, T.: UML based Grid Workflow Modeling under ASKALON. Distributed & Parallel Systems. pp. 191-200 (2006)

[9] OMG.: Business process model and notation (BPMN) version 2.0. (2010)

[10] Park, M. , Na, H. , Ahn, H. , & Kim, K. P.: A scientific workflow model designer based on scientific information control nets. IEEE, (2015)

[11] Chinosi, M. , & Trombetta, A.: BPMN: An introduction to the standard. Computer Standards Interfaces. Vol. 34, No. 1, pp. 124-134 (2011)

[12] Yongchareon, S. , Liu, C. , & Zhao, X.: UniFlexView: A unified framework for consistent construction of BPMN and BPEL process views. Concurrency and Computation: Practice and Experience. Vol. 32, No. 11, (2020)

[13] Huang, W. , Zhu, J. , Gao, Y. , Liu, G. , & Yuan, Y. , et al.: Design and implementation of customized workflow configuration platform for electric power company. Energy Reports. Vol. 7, No. S7, pp. 230-241 (2010)