



Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Computer science	Spring semester, 2012 Open
Writer: Artur Latifov (Artur Latifov)
Faculty supervisor: Chunming Rong External supervisor(s): Jochen Müller, Bouvet	
Titel of thesis: Dynamic Enterprise Architecture - From Static to Dynamic Models	
Credits (ECTS): 30	
Key words: Enterprise architecture, dynamic models, static models, Petri net, BPMN	Pages: 82 + enclosure: CD Stavanger, June 14/ 2012 Date/year

Frontpage for master thesis
 Faculty of Science and Technology
 Decision made by the Dean October 30th 2009

Abstract

Nowadays the most commonly used technologies in Enterprise Architecture modeling are static. BPMN is one of such technologies, which is used widely for drawing models of business processes. Limitation of BPMN is that it gives a static image, or drawing, of business process without simulation capability. Petri net is one of the modeling tools with dynamics functionality. This functionality is mostly applied in simulation of discrete-event dynamic systems. Business process can be represented as such a system. BPMN and Petri net are completely different tools with different purposes. In the paper we suggest an original approach of transformation of BPMN to Petri net, in addition to it we propose an extension to BPMN diagram which will make further simulation with Petri net more powerful. As a proof for the suggested concept, we made a partial implementation of it.

Contents

Faculty of Science and Technology	1
MASTER'S THESIS	1
1 Introduction	7
1.1 Enterprise architecture.....	7
1.2 Definition of the problem	10
1.3 The goal and structure of the paper.....	11
2 Technologies.....	13
2.1 Overview over enterprise architecture modeling technologies.....	13
2.1.1 Frameworks.....	14
2.1.2 Methods and techniques.....	15
2.2 BPMN.....	18
2.3 Petri net.....	22
3 Theoretical description of suggested approach BPMN-to-PetriNet mapping.....	26
3.1 Overview of suggested approach	26
3.2 Transformation to Petri net model	27
3.2.1 Creation and format of BPMN diagrams.....	27
3.2.2 Mapping BPMN diagrams to Petri net	28
3.3 Business relevant simulation with Petri net.....	36
3.4 Extension to BPMN diagram.....	42
4 Proof of concept implementation.....	53
4.1 General overview of implementation.....	53
4.2 Architecture of Java program.....	54
4.3 Testing	56
4.4 Evaluation	68
5 Conclusion and future work	69
Appendix A. BPMN model updated with probability information	71
Appendix B. BPMN model updated with time delays information.....	72
Appendix C. How to use the program: prerequisites.....	73
Appendix D. GPenSIM: installation guide	73
Appendix E. How to use the program: steps	74
References	77

List of figures

Figure 1. Enterprise architecture domains.....	8
Figure 2. Business process	9
Figure 3. BPMN diagram example	11
Figure 4. Data flow diagram.....	16
Figure 5. Elements of activity diagram	17
Figure 6. IDEF0 box	17
Figure 7. Marked Petri net.....	23
Figure 8. Transition T0 before firing	24
Figure 9. Transition T0 after firing.....	24
Figure 10. Overview of suggested approach	27
Figure 11. BPMN pools	31
Figure 12. Petri net form of pools.....	31
Figure 19. Case 3	32
Figure 20. Case 2	32
Figure 18. Case 1	32
Figure 16. Case 1 realized with simple replacement	33
Figure 17. Petri net form of end events.....	33
Figure 18. Element has outbound connection to previous element	33
Figure 19. Representation in Petri net by simple replacement from Table 1	34
Figure 20. Suggested approach to solve the problem of cycles.....	34
Figure 21. Subprocess in BPMN	35
Figure 22. Subprocess in Petri net	35
Figure 23. Forks in BPMN	36
Figure 24. Forks in Petri net	37
Figure 25. Probability density function of exponential distribution.....	38
Figure 26. Simple delivery process model	39
Figure 27. Petri net for simple delivery process model	39
Figure 28. Sample BPMN model with resource sharing	40
Figure 29. General sketch of Petri net with resource sharing	40
Figure 30. Petri net with resource sharing.....	41
Figure 31. Gateway.....	43
Figure 32. Probability for gateways.....	44
Figure 33. Activities with time consumption	44
Figure 34. Annotations with time information in activities	45
Figure 35. Annotations with resource-related information in BPMN	45
Figure 36. Resource related information in BPMN (with collapsed pools).....	46
Figure 37. Petri net with resource sharing (big number of annotations case).....	47
Figure 38. Petri net with resource sharing (with collapsed pools case).....	47
Figure 39. Resource sharing with capacity in BPMN (full form)	49
Figure 40. Petri net for resource sharing with capacity (full form).....	50
Figure 41. Petri net for resource sharing with capacity (simplified form).....	50
Figure 42. Overview of BPMN-to-PetriNet practical realization.....	53
Figure 43. Architecture of Java program	54
Figure 44. Structure of BPMNModel class	55
Figure 45. Structure of PetriNet class	56
Figure 51. BPMN model of recruitment process.....	57
Figure 52. Petri net model for recruitment process	59
Figure 48. Main simulation file	60
Figure 49. Simulation results for simple BPMN model.....	61
Figure 50. Simulation results with probabilities (a)	62
Figure 51. Simulation results with probabilities (b)	63

Figure 52. Simulation results with probabilities(c)	64
Figure 53. Activity with time delay information.....	64
Figure 54. Main simulation file, firing times for transitions.....	65
Figure 55. Start event with exponential distribution.....	67
Figure 56. Main simulation file with random firing time	67
Figure 57. Problem 3 simulation results (a)	68
Figure 58. Problem 3 simulation results (b).....	68

List of tables

Table 1. Basic BPMN elements	20
Table 2. BPMN elements and corresponding Petri net elements	30
Table 3. Rules of building Petri net from BPMN model	36
Table 4. Petri net usage for business relevant simulation	42
Table 5. Extensions to BPMN diagram	52

1 Introduction

1.1 Enterprise architecture

Enterprise is researched within the theory of enterprise. The theory of enterprise could be studied from two perspectives – economic theory and social theory.

Within economic theory we can find following approaches to enterprise research:

1. In neo-classical theory a firm is depicted as “black box”, which has resources on input and goods or services on output [1].
2. In a stakeholder’s perspective a firm consists of several entities (stakeholders) which try to realize own goals. Enterprise can be private, state owned or hybrid. Sometimes their goals can turn into conflict with each other, but more often there is a certain level of consensus between these entities. So as a result of this consensus, the common goal of organization is achieved. This can be done by agreement between different entities or by domination of one entity (for example, biggest stakeholder)[2].

As a conclusion from economic perspectives we can say, that enterprise is a business organization. The latter means “an organization, involved in the trade of goods, services, or both to consumers”[3].

In social theories we can find definitions of enterprise as the institution which serves the function of satisfying basic needs of society (Marxist perspective).

According to network perspective enterprise functions as an actor in social networks (which can include single person or groups of people, united in one organization).

So after combining both economic and social approaches, we would like to present the definition of enterprise, which we will use further in the paper. Enterprise – is “an organization (or cross-organizational entity) supporting a defined business scope and mission. An enterprise includes interdependent resources (people, organizations, and technology) which must coordinate their functions and share information for support of a common mission (or set of related missions)”[4].

Term enterprise architecture consists of two words – first of them was defined above. The remaining is architecture. There are different definitions of architecture. According to one of them, architecture is “the structure of components, their interrelationships, and the principles and

guidelines governing their evolution and design.”[5] We consider that architecture is a simplified representation of complex systems.

As it is known from systems theory, all complex objects consist of simpler ones. We call these simple entities elements or components. Components are needed to have connections between each other (or relations). The whole picture with elements and relations between them is structure.

Architecture is a model of complex system. In case of enterprise architecture, the complex system is represented by enterprise. As every system, enterprise has elements. They are represented by:

1. Business processes and people.
2. Information.
3. Technology.

According to these elements there are several types of architecture domains (layers):

1. Business architecture.
2. Information systems architecture (Data and application).
3. Technical architecture.

This relation is shown in the picture below.

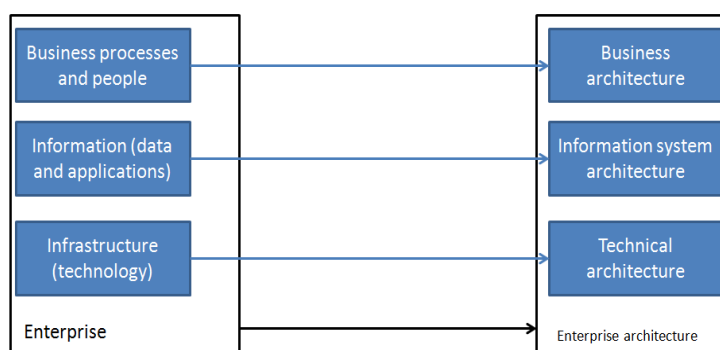


Figure 1. Enterprise architecture domains

First domain is *business architecture*. It encompasses governance, business processes and business information[6]. This is the most crucial domain among others, which defines strategic goals for the remaining layers. Business strategy means general direction, in which organization

is moving, including its mission, objectives, way of handling resources, behavior in external environment.

The key element in the architecture layer is business process. Business process can be defined as activities, which have input and output. This is shown in the picture below.

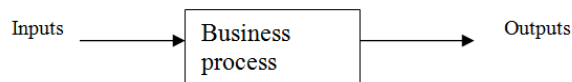


Figure 2. Business process[7]

Outputs can be products or services.

Second domain is *information systems architecture*. There are two layers which lie in the scope of information – applications architecture and data architecture. Applications architecture covers connections between applications and business processes. So applications architecture serves as a medium between the upper layer (business processes) and lower levels.

Data architecture is focused on data which is stored in enterprise system. When we talk about information modeling, it is important to study the structure of the data and relationships between collections of data.

Third domain is *technical architecture*. It deals with technological infrastructure of the enterprise. It can include networks, computers, hardware, servers, etc. Technical architecture encompasses hardware and software, which serves as a basis for information system, which lies on it.

As some authors argue, technical architecture should not be too specific, in contrast, it should be more general. Otherwise this description of architecture can become quickly out of date, as technologies always change rapidly[8].

Modeling of enterprise architecture gives following benefits[9]:

1. Modeling of entire enterprise hides from us complexities and make the business simple to analyze and understand. Also different types of stakeholders can have their specific views of enterprise. This is also very convenient, because it can help to have a shared vision of the business.

2. Generalization creates a viewpoint at the highest level. We are able to see the whole system, instead of concentrating too much on sub-systems. So when we are analyzing any sub-system, we understand the role it plays in the whole system.
3. Enterprise architecture allows to formulate principles, which will be used in the long-term perspective when we are working on projects. These principles will govern the project work in order to make it comply with enterprise objectives.
4. If there are regulative acts and legislation in the country our enterprise is functioning, then enterprise architecture will help us to make our business comply with this legislation.

In overall, purpose of depicting enterprise architecture - to improve the effectiveness and efficiency of the business itself. “A well-defined and well-maintained enterprise architecture is also proven to be a key factor in an organisation’s agility, effectiveness and ability to respond to risk, opportunity and change.”[10]

Enterprise architecture can be studied as an aspect of IT governance[5]. IT governance is defined as “the leadership and organizational structures and processes that ensure that the organization’s IT sustains and extends the organization’s strategies and objectives”[11]. IT governance is aimed at making sure, that IT serves for enterprise’s goals. There are international standards in IT governance, for example, ITIL and COBIT.

1.2 Definition of the problem

Most commonly used modeling technologies of enterprise architecture include BPMN, UML, Flowcharts, Data flow diagrams. For modeling business processes BPMN is used most of all. In the paper main accent is put on the business related modeling, because business process is the highest level of enterprise architecture. That is the reason why the highest interest for us is BPMN technology.

Most of currently used enterprise modeling technologies, and BPMN as one of them, are static. Static models depict state of the process at certain time point, and during the timeline this state does not change. In real life scenarios business processes are not static. That’s why demand for non-static models appears. Business process, or corresponding model of it, can have two types of dynamics:

1. The structure of business process, or the process itself, may change due to change in external and internal drivers.
2. The process has input and output, or beginning and end. There is a movement, or change, in process from one point to another (from input to output, from beginning to end).

First type of dynamics is dependent on internal (Business strategy, organizations infrastructure, IT) and external (market, regulating acts by government, changes in technologies) drivers [12]. Static models, used nowadays, do not encompass these internal and external possibilities of change. They are created at certain moment, when we have a current set of internal and external drivers.

Second type is connected with process execution. BPMN model is just a static picture of a business process.

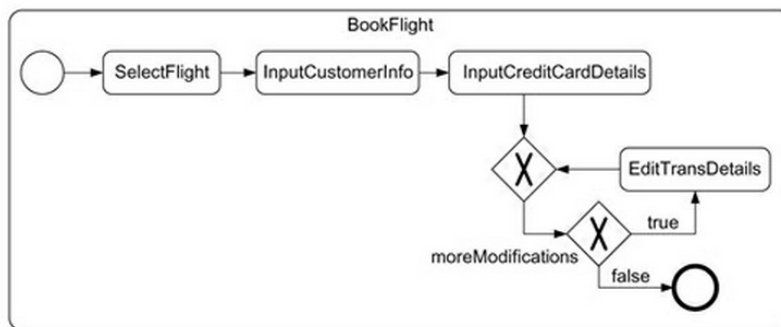


Figure 3. BPMN diagram example[13]

This picture can not show the action (how quickly the process goes through stage “SelectFlight”, how the decision at “moreModifications” gateway is done). The main disadvantage of the picture like this is that it gives no possibility for simulation, i.e. no way to see how the process is executed. Business may need simulation because it helps to get better knowledge of simulated object (business process), and to improve its efficiency and performance.

In the paper we are going to research the way of adding dynamics to models in its second interpretation (moving from beginning to end in the process). Only after this kind of dynamics is added to the model, we get an ability to simulate.

1.3 The goal and structure of the paper

The goal of the paper is to suggest an approach of converting static Enterprise architecture models (BPMN diagrams) into dynamic ones (Petri net models). In the paper we are aimed at validation of following hypothesis:

Hypothesis 1. BPMN, which is a static model of Enterprise Architecture, can be transformed into Petri net, which is a dynamic model.

Hypothesis 2. Transformation to dynamic model can bring useful values for business.

Structure of the paper is the following:

1. In the chapter 2 an overview of enterprise architecture modeling tools is given. This overview includes general facts about commonly known standards (TOGAF, Zachman, UML, etc.) and more detailed description of technologies which we are going to use in our project (BPMN, Petri net).
2. In chapter 3 we suggest our own approach of transforming BPMN to Petri net. The chapter contains theoretical description of the approach. The approach itself consists of
 - mapping rules between BPMN and Petri net, which are completely different standards;
 - ways of Petri net simulation, which bring business value;
 - new extensional rules for making BPMN diagram, which will enhance and make more powerful Petri net simulation.
3. In chapter 4 we present an partial implementation of the approach, developed in chapter 3. The aim of implementation is to show that the suggested approach can be realized and that it can give benefits for business.

Concepts, studied in chapter 2, are based on wide range of literature, describing such standards as BPMN, Petri net, TOGAF, etc [14], [15], [16], [17].

Chapter 3 partially relies on existing works, but the main part of the chapter is our own approach. Existing researches were used for the definition of some mapping rules between standards BPMN and Petri net [18], [19], [20]. But these works do not cover the full scope of BPMN. They are focused on element-to-element mapping (and not all BPMN elements are considered). These works do not describe how the full integration of transformed elements into one Petri net diagram should be done. These lacks do not allow to establish a full BPMN-to-PetriNet transformation approach. Chapter 3 suggests additional rules which will make this approach complete.

In addition to mapping rules chapter 3 defines ways of how Petri net simulation can be used for business goals. These ways are derived from general Petri net techniques. While many works describe general Petri net simulation methodology, we made it compatible with business-relevant goals.

Besides this, in chapter 3 we worked out our own extension for BPMN diagram, which will enhance simulation with it, when it is transformed into Petri net. The extension is a suggested combination of BPMN elements + annotations, which is based on following considerations:

1. The extension should not change the workflow of a BPMN model (independently if we have or do not have this extension, a static picture of BPMN model has the same flow logic).
2. The extension should make possible all available ways of business-relevant simulation with Petri net afterwards.

Chapter 4 do not contain any reference literature, as it describes our implementation of the suggested BPMN-to-PetriNet transformation approach.

2 Technologies

2.1 Overview over enterprise architecture modeling technologies

Modeling is the process of creating models. It is widely used in scientific as well as in other activities. In general, model is a simplified representation of an empirical object, existing in the real world.

In case of enterprise architecture modeling, models are created for the complex system. This complex system is represented by enterprise.

There are different types of classification of enterprise architecture models. They can be classified by the layer, where they are used. For example, process models (on business process layer), information models (on information layer), technical models (on technology layer). [14]. On business process layer following technologies are used:

1. Flowcharts.
2. Data flow diagrams.
3. IDEF0.
4. UML and BPEL/BPMN.
5. FMC Compositional structure diagram

Information layer includes entity-relationship and object-oriented modeling. First is realized in SQL and FMC (Fundamental Modeling Concepts) Entity Relationships Diagrams, second - in UML.

The last layer is technical architecture. Different technologies and tools can be used for modeling it, for example:

1. UML.

2. FMC Compositional Structure Diagram.

Another classification divides enterprise architecture models into three hierarchical levels – Frameworks (on top), methods (in the middle) and techniques (in the bottom) [21]

In the third approach enterprise models are of two kinds – descriptive and formal [22]. Formal models are based on mathematical formalism, while descriptive models lack this formalism.

Fourth approach states, that there are two kinds of models – static and dynamic. Word static is derived from “state”. State means a configuration of an object at certain time point. So static model is a simplified representation of the complex object at certain time point. Dynamic models can change, during the life-cycle or because of simulation.

In the paper we are going to study models, created for the upper-layer of enterprise architecture (business processes).

2.1.1 Frameworks

Enterprise architecture framework describes structure and composition of views in enterprise on a strategic level. It serves more as theoretical description of enterprise rather than implementation of it.

There are two main types of architecture frameworks. First is application-class. In this approach enterprise architecture focuses more on IT systems. Second is enterprise-class. In this view enterprise architecture framework models the enterprise as a whole entity. All components of this large system are integrated in a holistic view [23].

Most used enterprise architecture frameworks are Zachman, TOGAF (the Open Group Architecture Framework), CIMOSA (Computer Integrated Manufacturing Open System Architecture), FEAF (Federal Enterprise Architecture Framework), IAF (the Integrated Architecture Framework) [15].

Zachman’s framework was developed by J. Zachman in 1980s. Since that time it was constantly evolving. Core element of this framework is 6x6 dimensional matrix[24].

TOGAF (The Open Group Architecture Framework) uses the concept of 4 domains – business, applications, data, technology. On the top level there is a business domain (concerns business strategy, processes, etc.). On the lowest level there is a technical domain (hardware, network infrastructure, etc.).

CIMOSA (Computer Integrated Manufacturing Open System Architecture) uses 4 views on enterprise – function view, information view, resource view, organization view. One of advantages of CIMOSA is that it suggests modeling language for implementing its concepts. It makes CIMOSA model executable.

FEAF (Federal Enterprise Architecture Framework) was developed in USA, initiated by Office of Management and Budget. The purpose was to unite architectures of different governmental structures in one entity on federal level. The central thing in FEAF is reference model. Reference model describes one specific set of elements in enterprise domain (for example, technical or business operations). The structure of FEAF includes 4 layers – business, data, application, technology. Also it helps to make a strategy of coming from current to target state of the system[4].

IAF (Integrated Infrastructure Framework) was developed by Capgemini. Basically IAF is an architecture method. Many of its principles are incorporated in TOGAF (which is the Open Group standard).

Enterprise Architecture Frameworks are theoretical concepts. They do not suggest practical technologies for implementation of these concepts (besides CIMOSA).

2.1.2 Methods and techniques

Among those technologies, which can be used in software modeling of enterprise architecture, are Flowcharts, Data flow diagrams, UML, IDEFX, GRAI net, Petri net, BPMN.

Flowcharts base on the assumption, that every process could be described as sequence of activities. Graphical part consists of symbols – terminator, process, data input, decision, flow indicators, connectors[25].

One of types of flowcharts is EPC (Event-driven process chain), developed by ARIS. EPC is based on the same logic as other flowcharts with more stress on events. Also it uses logical operators (OR, AND, XOR).

Data flow diagrams show relations between components instead of sequence of activities. There are different notations of data flow diagrams. Below there is an online example of Data flow diagram.

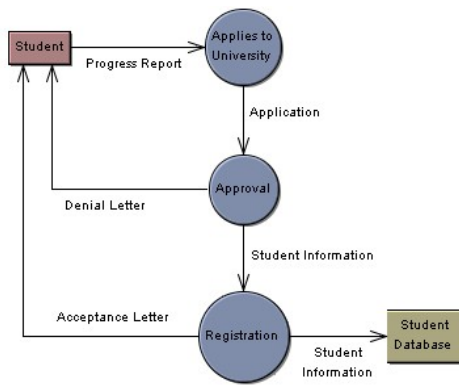


Figure 4. Data flow diagram[26]

We can see that arrows here do not represent any sequence of actions. Main focus of data flow diagrams is on relations. Relations are of two kinds:

1. Between components in the architecture.
2. Between components in the architecture and external entities.

UML (Unified Modeling Language) is a modeling object-oriented tool [27]. It is particularly successful used in software modeling. When object oriented languages became widespread, UML helped to model classes and connections between them.

The core element in UML are diagrams. There are two main types of UML diagrams – structural and behavioral. Structural diagrams are targeted at depicting structure and components. Behavioral – at processes and changes in the system. For business purposes one subset of behavioral diagrams is used – activity diagram.

Activity diagram allows to depict business processes in the enterprise. It shows sequence flows, conditions and objects [28]. Core elements of activity diagram are shown below.



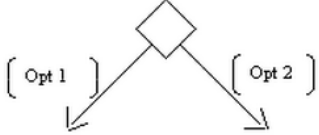
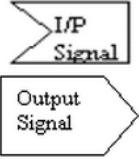
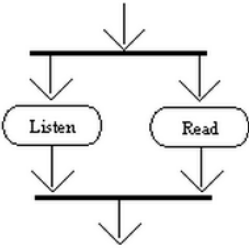

Element and its description	Symbol
Initial Activity: This shows the starting point or first activity of the flow. Denoted by a solid circle. This is similar to the notation used for Initial State.	
Activity: Represented by a rectangle with rounded (almost oval) edges.	
Decisions: Similar to flowcharts, a logic where a decision is to be made is depicted by a diamond, with the options written on either sides of the arrows emerging from the diamond, within box brackets.	
Signal: When an activity sends or receives a message, that activity is called a signal. Signals are of two types: Input signal (Message receiving activity) shown by a concave polygon and Output signal (Message sending activity) shown by a convex polygon.	
Concurrent Activities: Some activities occur simultaneously or in parallel. Such activities are called concurrent activities. For example, listening to the lecturer and looking at the blackboard is a parallel activity. This is represented by a horizontal split (thick dark line) and the two concurrent activities next to each other, and the horizontal line again to show the end of the parallel activity.	
Final Activity: The end of the Activity diagram is shown by a bull's eye symbol, also called as a final activity.	

Figure 5. Elements of activity diagram [29]

IDEF (Integration definition) is a set of modeling languages. There are IDEF0, IDEF1X, IDEF2, etc. For modeling business processes IDEF0 and IDEF3 are used.

IDEF0 is used for modeling of functions. The core element of IDEF0 diagram is a simple box, shown below.

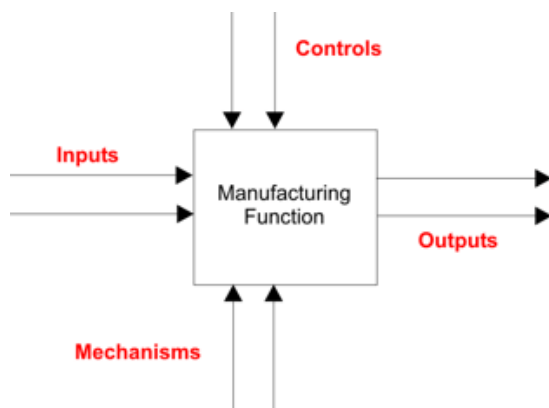


Figure 6. IDEF0 box [30]

IDEF3 expands IDEF0 and targeted at business process modeling. IDEF3 has 2 main focuses – on object state or process. Core element in object state diagram is Object State label (depicted as a circle), in process diagram – Unit of behavior label (depicted as a rectangle)[31]. These two elements can be combined.

GRAI net method is focused on controls in the system. GRAI net consists of 4 subsystems – information, decisional, operational, physical.

Petri net are mathematical tool for modeling of discrete-event dynamic systems (it is studied more thoroughly further in the chapter).

Another modeling tool, which is also used for modeling of business processes, is BPMN (Business Process Model and Notation, also studied further in the chapter).

2.2 BPMN

BPMN (Business process model and notation) is aimed at visualizing business processes. In this sense, it is similar to UML activity diagram. BPMN was adopted as a standard by Object Management Group in 2006. The current version of this standard is BPMN 2.0.





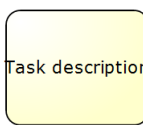
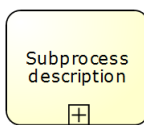




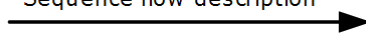
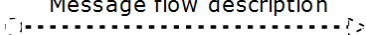

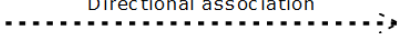
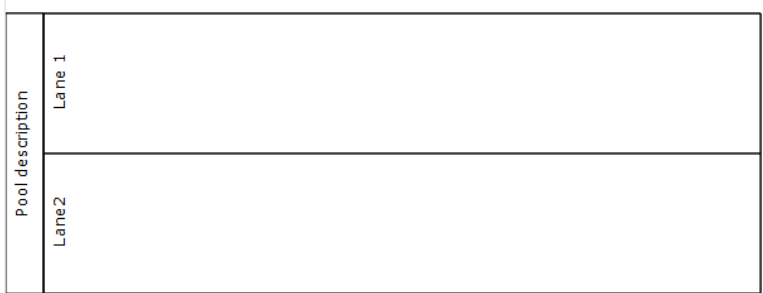

Business process can be defined as the set of connected procedures, which realize the business goal [16]. Business process is one of the most important components in enterprise architecture. Business level is above other lower layers of enterprise architecture (Data, Applications and Technology).

Visualization in BPMN is done by means of diagrams, supported by this standard. There are several element categories in BPMN:

1. Flow objects.
2. Connecting objects.
3. Swim lanes.
4. Artifacts.

Table below shows basic BPMN objects.

Category	Elements	Notation

Flow Objects	Event	 Start event  End event  Message event  Timer event
	Activity	 Task description  Subprocess description
	Gateway	 Join\Split exclusive gateway  Parallel gateway  Event-based gateway  Inclusive gateway
Connecting objects	Sequence flow	Sequence flow description 
	Message flow	Message flow description 
	Association	 Directional association 
Swim lanes	Pool and lanes	
Artifacts	Data object	 Data object description

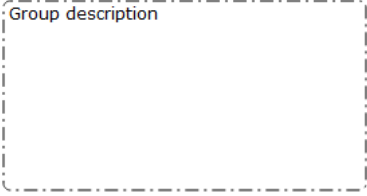
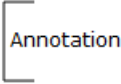
	Group	
	Annotation	

Table 1. Basic BPMN elements [32]

Elements, listed above, are only basis ones. In BPMN standard there are more extension elements.

Event is triggered by something, there are two basic types of event – start and end. *Activity* defines the work which is executed. It can be atomic, consisting of only one work (task) or compound (subprocess). *Gateways* are used for splitting and joining. Exclusive gateway implies that only one outgoing sequence flow will be executed. It can join two sub-diagrams into one or split one into two. Parallel gateway means that outgoing activities will be executed in parallel.

Flows are used to connect elements of BPMN diagram with each other. *Sequence flow* shows the order of activities. *Message flow* is used for showing exchange of messages between different pools (never within one pool). *Associations* are used to link artifacts with other elements.

Pool separates different participants from each other. Usually one pool represents an organization. It can contain some processes, or it can be empty to symbolize black box. *Lanes* are in the pool, they separate activities in one pool from each other.

Artifacts are extension to elementary BPMN objects. They enrich the BPMN model with new information. One of artifact is *Data Object*. Data Object is used for showing an information entity (it can be electronic or physical, for example sales order). They do not change the process, but instead give more information to user. It is possible to attach data object to message, sequence flow, activity, etc. For example, in case of activity data object can represent input or output of the activity. Data object can represent both single object and set of objects. *Group* does not change the process (the same as data object), but it has a nominal, informal meaning. Group unites similar activities under one category. *Annotations* are also informal element, which does not affect the whole process. Annotations can be used to give some text explanation to user. But

BPMN specification does not limit us in ways of using annotations. They can be applied to all BPMN objects.

Besides these basic BPMN elements there are less significant symbols. For example, complex gateways (which allow to make more than 2 output flows); inclusive gateways; event-based gateways. The full set of BPMN objects and their description can be found in BPMN specification [32].

BPMN is implemented in several software tools, most popular are BizAgi, Intalio, ITP process modeler, Oracle BPMN Studio, SparxSystem EA, Visual Architect BP-VA, ARIS Express [16].

In the previous chapter we analyzed several technologies of enterprise architecture modeling, both frameworks and methods/techniques. BPMN is the technology which was selected for this paper (in implementation part). And this has a certain reason.

First, frameworks. Such a framework as Zachman is very useful for an overall theoretical analysis and modeling of enterprise, because it suggest a holistic view of the company. But the main disadvantage for Zachman's and other frameworks is that they are too theoretical. For the purpose of the paper they do not bring much value.

Second, methods and techniques. One of technologies is GRAI net. GRAI net has a main focus on decisional aspect of enterprise models. It lack functionality to model the flow of work, business processes [22]. By this reason, this technology is not used in the paper.

Second technology is IDEF. As Ryan [22] argues, IDEF lacks two main things – control capabilities and resource modeling. Control functions (conditions, limitations on elements) are not realized in IDEF at high level. Resources (like data objects , etc. in BPMN) are not covered by IDEF either. Also this technology is quite abstract. This makes it non efficient in implementation.

Another technology, UML, is more powerful than IDEF when we talk about practical implementation of Enterprise Architecture. Another advantage of UML over IDEF is its simplicity [34]. There are different types of diagrams in UML and this allows to model enterprise from different perspectives.

The paper is aimed at making dynamic models of business processes (which is the highest layer in enterprise architecture structure). That's why, we are more interested in one type of UML diagrams – Activity diagrams. UML Activity diagrams and BPMN are quite similar technologies

[34]. Almost all processes can be modeled in BPMN and UML in a similar way (with slight difference in shapes and names). But for the purpose of this paper BPMN is more suitable. This is because UML covers all layers of enterprise architecture (with the main focus on software models), while BPMN is focused specifically at business processes. It makes BPMN more expressional and convenient when we model business processes. Researchers [35] argue that BPMN and UML Activity diagrams are quite similar, but BPMN has more representational power and external interactions are realized much better in BPMN. It is reached by additional tools, available in BPMN (one of them is more convenient work with business exceptions).

2.3 Petri net

Petri net is a “graphical and mathematical tool of modelling discrete event dynamic systems” [36]. Dynamic system is the system in which time is included. System dynamics is called a dynamic process, or time behavior. Discrete process is defined in following way [17]

If a finite set is given

$$\Sigma = \{e_1, e_2, \dots, e_n\}$$

Σ is called the event set. We assume that an event $e_{i(k)} \in \Sigma$ occurs at the time point $\tau_{i(k)}$. Let a sequence of events be given as

$$\sigma = e_{i(1)}, e_{i(2)}, \dots, e_{i(k)}, \dots, e_{i(N)}$$

where $e_{i(1)} \in \Sigma$ occurs in the discrete time point $\tau_{i(1)}$, $e_{i(2)} \in \Sigma$ in time point $\tau_{i(2)}$, etc., $e_{i(k)}$ in time point $\tau_{i(k)}$, etc., and $e_{i(N)}$ in time point $\tau_{i(N)}$, $\tau_{i(1)} < \tau_{i(2)} < \dots < \tau_{i(k)} < \dots < \tau_{i(N)}$.

Then the sequence σ is called a discrete process.

Business process, depicted in BPMN, could also be studied as a discrete process, because it has a finite set of events (or states) and they occur at discrete time points (time points can be defined in BPMN). So, if BPMN process can be represented as a discrete process, then it can be simulated with Petri net. This assumption we will use further in our work.

Magalhaes [37] also gives formal definition of Petri net:

PN can be defined as a 5-tuple

$$(\mathbf{P}, \mathbf{T}, \mathbf{F}, \mathbf{w}, \mathbf{M}_0),$$

where:

$P = \{P_1, \dots, P_m\}$ is a finite set of places;

$T = \{t_1, \dots, t_n\}$ is a finite set of transitions;

$F \subset (P \times T) \cup (T \times P)$ is a set of arcs;

$w: F \rightarrow \{1, 2, \dots\}$ is a weight function;

$M_0: P \rightarrow \{0, 1, 2, \dots\}$ is the initial marking;

with $(P \cap T) = \emptyset$ and $(P \cup T) \neq \emptyset$

Basic elements of Petri net are:

1. Places.
2. Transitions.
3. Arcs (or connections).

Mark for place P_i (m_i) is number of tokens contained in P_i . Marking is a set of marks for all places. For example, in the Petri net

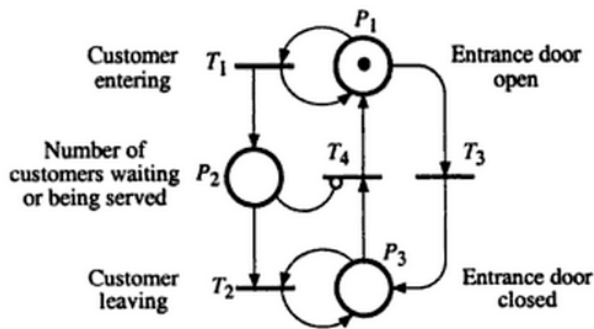


Figure 7. Marked Petri net[38]

marking $M = (1, 0, 0)$.

Transition in Petri net can fire in 2 cases [17]:

1. For each pre-place of this transition, its marking is \geq weight of the arc from pre-place to the transition.
2. This transition has no inbound places.

If transition is fireable, it does not necessarily mean that it will fire. There could be two fireable transitions, but only one can fire at a time. Also there could be a probability imposed on transition that it fires or does not fire.

Every place p , after transition t fires, will have following marking $m'(p)$:

1. If p is simultaneously an input and output place for t . Then $m_{\text{new}}(p) = m_{\text{old}}(p) - \text{weight}(p, t) + \text{weight}(t, p)$.
2. If p is only an input place for t , then $m_{\text{new}}(p) = m_{\text{old}}(p) - \text{weight}(p, t)$.
3. If p is only an output place for t , then $m_{\text{new}}(p) = m_{\text{old}}(p) + \text{weight}(t, p)$.
4. If p is neither an input, nor an output place for t , then marking will be unchanged $m_{\text{new}}(p) = m_{\text{old}}(p)$.

Example of transition firing is shown below.

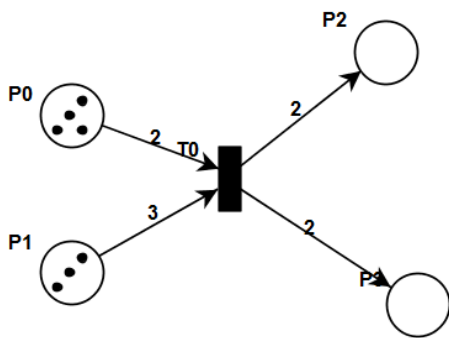


Figure 8. Transition T0 before firing

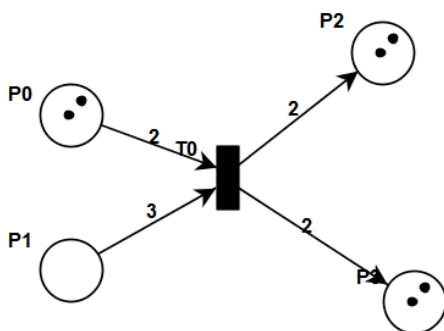


Figure 9. Transition T0 after firing

It is possible to introduce time into Petri net. It can be done in several ways (time is introduced to places or to transitions or to both). But the most common approach is to associate time with transitions. The result of this approach is that transitions have their time delays. In non-timed

Petri net every transition can fire immediately after it has been enabled. In timed Petri net firing of a transition will require a certain amount of time.

Timed Petri net can be defined as [39]

$$\text{TPN} = \{P, T, F, w, M_0, \tau\}$$

where τ is a collection of time delays for transitions or distribution functions.

Depending on which approach we use (fixed time delays or distribution functions), there are 2 types of Petri net:

1. Deterministic.
2. Stochastic.

In deterministic Petri net time delays for transitions are fixed constant values. In stochastic - time intervals are random variables, which are exponentially distributed.

In addition to time, color can be also introduced to Petri nets. Colors can be assigned to tokens. Colors serve for several purposes:

1. Color brings individuality to tokens. Every token can have its specific color, which differentiate it from others.
2. Token can have properties which change during execution of the process, in this case colors are changed and overwritten in transitions.

In colored Petri nets transition can do following operations with color [40]:

1. Color creation.
2. Color revision (changing one of existing colors).
3. Color addition (color is added to existing one).
4. Color deletion.

Time and color can be combined and used together in one Petri net.

Petri nets are also classified in relation to their complexity, there are two main groups of Petri nets:

1. Low-level.
2. High-level.

Low-level Petri net has a simple structure and therefore it is easy to analyze. But drawback of these nets is that very often it does not reflect the real-life object. Simple Petri nets are good for testing the model, for example, check that the process will work correctly and pass through all necessary stages. But low-level net is not suitable for performance analysis. To enable this functionality we need to use time and color.

It is considered that, when time and color are added to Petri net, then it becomes a high-level one. The name for such nets is Timed Colored Petri net. Petri nets with time and color functionality are “the most integrated approach available for complex real-time systems”[41].

Further in paper we are going to focus mainly on timed and colored Petri nets and describe ways of their usage in business process modelling.

There are different existing tools, which support Petri net modelling:

1. SPNP.
2. GreatSPN.
3. INA.
4. CPN Tools.
5. GPenSIM.

In the paper we will use GPenSIM as Petri net simulation tool. It can be downloaded and installed from <http://www.davidrajuh.net/gpensim/>. It is integrated in MATLAB platform. Reason for this choice is that GPenSIM can suggest functionality of mathematical formulas available in MATLAB. This makes simulation more powerful, the only limitation is that it does not have a graphical view.

3 Theoretical description of suggested approach BPMN-to-PetriNet mapping

3.1 Overview of suggested approach

The general aim of our approach is to show how the transformation of static diagrams into dynamic ones can be done and which benefits we can get from dynamic modeling. General picture, describing suggested approach is shown below.

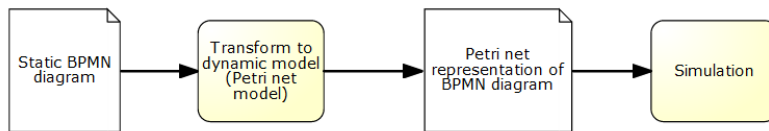


Figure 10. Overview of suggested approach

We see a static BPMN diagram on the input. This kind of diagrams are very popular and used in many companies as existing solution. This is a static modeling of enterprise architecture. But we suggest to extend the current approach and transform static diagram into dynamic one.

The transformation is done in the first yellow block. The result of it is dynamic model, expressed with the help of Petri net standard.

The second yellow block from the figure gets this Petri net model and makes simulation with it.

So our main focus in next part of this chapter will be to describe theoretically how first (Transformation to Petri net model) and second (Simulation) yellow blocks can be implemented.

In order to describe the principles of static-to-dynamic transformation of models we will answer following questions:

1. In which form will we store static BPMN diagram and which technologies will be used for representing this diagram?
2. What are the rules of mapping BPMN diagram to Petri net?
3. The way how Petri net could be used in business relevant simulation.
4. How static BPMN diagram can be extended in order to enable additional advantages of Petri net models (timed Petri net, resource sharing)?

Further in this chapter we are going to answer each of these 4 questions.

3.2 Transformation to Petri net model

3.2.1 Creation and format of BPMN diagrams

First question, form of representing BPMN diagrams and technologies used for this, is important because BPMN is only a graphical representation of business processes. It contains notation for graphical elements specifications. In most cases BPMN diagram is stored in jpeg, png or in file

formats, specific for the used tool (for example, ADF is a file format used in ARIS Express tool). But the limitation of graphic formats are that they are designated for viewing models, not for making any manipulations with them. Tool-specific formats also have disadvantage that they can be utilized only within specific tool.

For our work we need such a tool, which will be able to export created BPMN models into operable file format. This file format could be XML, because it gives following advantages [42]:

1. Usability.
2. Compatibility.
3. Easy to process in Java program.

XML is widespread and commonly used standard. This will make it easier for users to export their currently existing BPMN diagrams into XML format.

Also we need to decide which software tool we will use in order to create static BPMN diagrams. Main requirements to this tool are that it will be open source, it will give a possibility to export BPMN model into XML, it will be easy to use. Based on these requirements, we selected Signavio Process Editor (<http://www.signavio.com/en.html>). According to some authors [16] Signavio is one of the most popular open source software tools for BPMN diagrams. This conclusion was done after the survey among users. But we are not limited by Signavio, because any tool, which can export BPMN to XML, could be used as well.

So answer to the first question, raised earlier, is that we will store BPMN diagrams in the form of XML and use Signavio tool for this purpose.

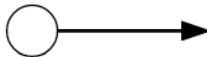
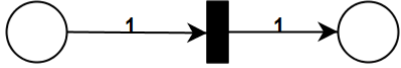
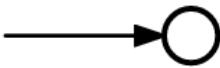
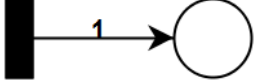



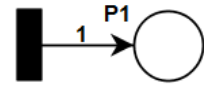
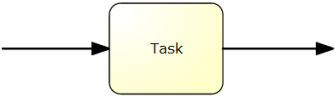
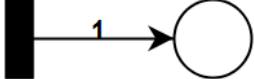
3.2.2 Mapping BPMN diagrams to Petri net

This part is one of the most complicated ones. This is because BPMN and Petri net are completely different standards, which are aimed at diverse things. But we will try to suggest our own approach of how the problem of mapping between them could be solved. In our approach we will rely on some existing works about BPMN-to-Petri net mapping ([18], [19], [20]). Our suggestion is to combine some ideas, suggested in these researches, propose our own enhancements to them and describe the practical way of implementing them.

In this chapter we are going to describe BPMN-Petri net transformation on conceptual level. This will be done in two parts:

1. First, we will make suggestions of how to representing BPMN single elements (events, activities, gateways, connecting objects, pools, lanes) into corresponding Petri net form.
2. Second, we will describe the way of integrating all these single elements into one big picture (result Petri net model).

Basic elements in BPMN should correspond to certain sets of places, transitions and connections in Petri net. In table below we tried to find corresponding Petri net set for each of BPMN elements. Information in the table was partly combined from existing sources ([18], [19]), and partly proposed by us (event-based gateways, events).

Name of BPMN element	BPMN representation	Petri net equivalent
Events		
Start Event		
End event		
Intermediate message event	 <p>Message event</p>	<p>Message event</p> 
Timer event -firing time for transition TO should be set according to timeout in timer event	 <p>Timeout 10 hours</p>	<p>TO</p> 
Activities		
Task		
Gateways		

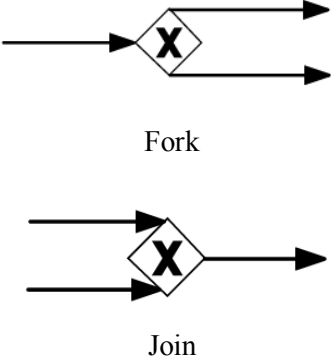
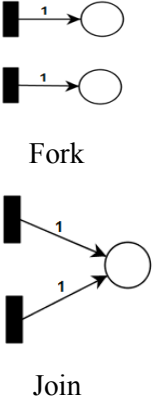
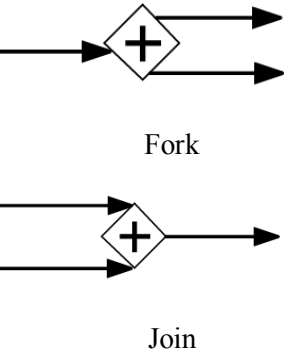
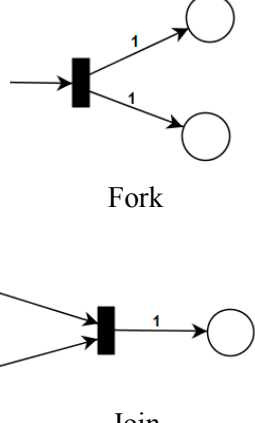
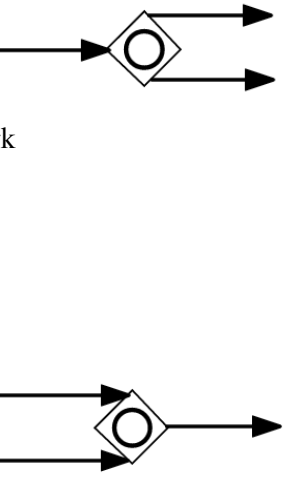
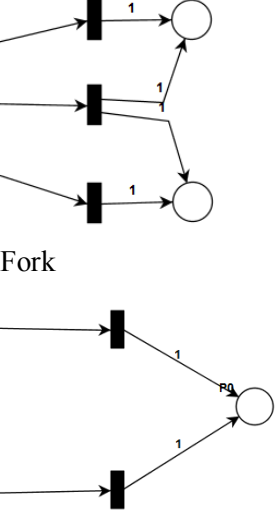
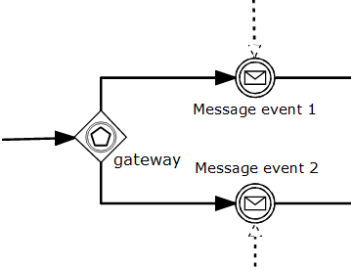
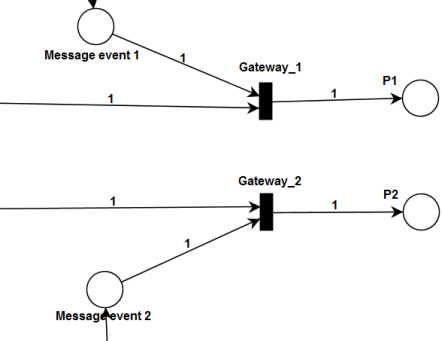
<p>Exclusive (XOR) gateway</p>	 <p>Fork</p> <p>Join</p>	 <p>Fork</p> <p>Join</p>
<p>Parallel gateway</p>	 <p>Fork</p> <p>Join</p>	 <p>Fork</p> <p>Join</p>
<p>Inclusive gateway</p>	 <p>Fork</p> <p>Join</p>	 <p>Fork</p> <p>Join</p>
<p>Event-based gateway</p> <p>-gateway is represented in the same way as exclusive gateway</p> <p>-The only difference is that events became input for gateway transitions instead of output</p>	 <p>Message event 1</p> <p>Message event 2</p> <p>gateway</p>	 <p>Message event 1</p> <p>Gateway_1</p> <p>P1</p> <p>Message event 2</p> <p>Gateway_2</p> <p>P2</p>

Table 2. BPMN elements and corresponding Petri net elements

Sequence flows correspond to connections in Petri net.

Another BPMN element which is often used in diagrams is swim lane. It includes pools and lanes. Their main purpose is to visually group set of elements into one entity. So depicting pools or lanes is just a question of visualization, it does not influence the workflow. Petri net does not suggest many graphical tools, that's why it is not possible to directly interpret pool into Petri net. Instead we can group elements and link them with connection as it is shown below. First picture is a sample with BPMN pools.

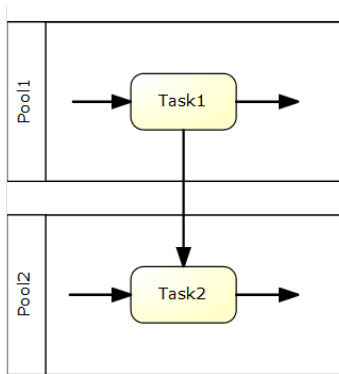


Figure 11. BPMN pools

It can be represented in Petri net as:

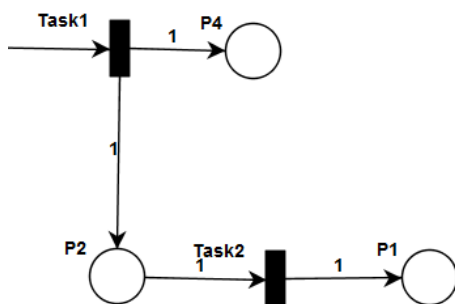


Figure 12. Petri net form of pools

So as we see, representation of pools is just 2 sets of transition-place (this is standard for task, see above) and a new place “P2”, which is used for connection between two tasks. The same approach can be applied for lane.

After we mapped every single BPMN element to corresponding Petri net set of places and transitions, we need to solve the problem of integrating all sets into one diagram.

Before we present our own way of integrating elements into one Petri net model, we would like to present *patterns* in Petri net form of BPMN elements from table above.

1. First pattern. For each BPMN element (with exception of message event) there is at least one transition. Conclusion from it for our work is that each transition in our approach will symbolize corresponding BPMN element (or several transitions will stand for one BPMN element).
2. Second pattern. Every Petri net representation of BPMN element has one or several places to the right of it. Place, which has incoming connection from transition, will mean that the BPMN element (represented by this transition in Petri net) is done.

Based on everything, said above about single BPMN element-to-Petri net transformations and patterns, we will derive our own rules of building the final Petri net diagram out of given BPMN model.

Building the complete Petri net model. It is not possible to build the final Petri net model by simple replacement of all elements in the BPMN model by Petri net equivalents. One of the reasons is that we can have a cycle in BPMN (sequence flow from one of elements could go back to previous element). That's why we will make rules for BPMN to Petri net transformation part by part.

First is start events. They are more or less simple, because they do not have incoming sequence flows. That's why start event is inserted into Petri net diagram by simple replacement from Table 1.

Second, end events. The difficulty here is that we can have 3 cases with them in BPMN:

1. One end event and all sequence flows coming to it.
2. Several end event and each of them has one incoming sequence flow.
3. Combination of case 1 and 2.

Examples of these cases are shown in figures below.

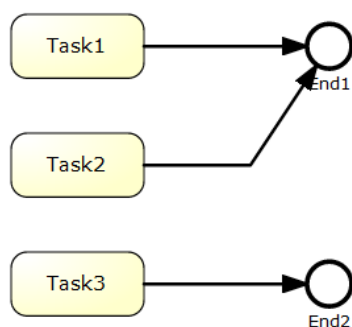


Figure 15. Case 1

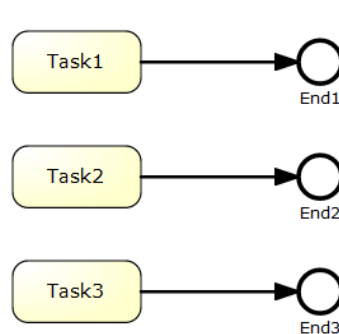


Figure 14. Case 2

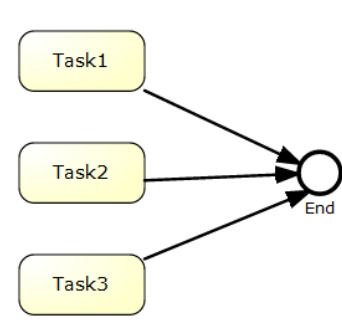


Figure 13. Case 3

We think that for the purpose of our work it will be better if we create separate transition+place for each of these cases. So it will be possible to see from which task did the token came to end. If we just insert equivalent for end event from Table 1 it will be unclear which task triggered end event (especially in cases 1 and 3). Picture shows how Petri net will look like if we try to realize case 1 by simple replacing from the Table 1, see figure below.

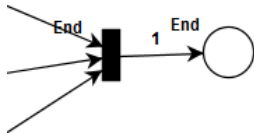


Figure 16. Case 1 realized with simple replacement

This does not reflect from which transition or task the token came to place “End”. So we suggest the following way of dealing with this problem (the same for all 3 cases).

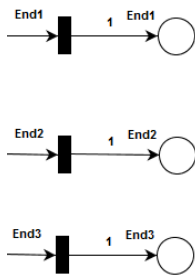


Figure 17. Petri net form of end events

The picture shows that we are going to create a set of one transition and one place for each sequence flow which goes from any of BPMN element to end event.

Third, timer events, tasks and gateways. Simple replacing from Table 1 will be suitable only if we do not have cycles and if no task or gateway has an outbound sequence flow to one of previous elements. If we have such a situation, then replacement from table will not work, because we will have two inbound places in transition, which will change a logic of model. This situation is shown in figure below.

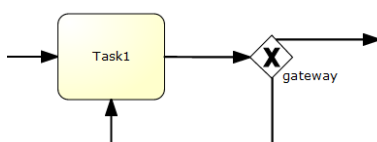


Figure 18. Element has outbound connection to previous element

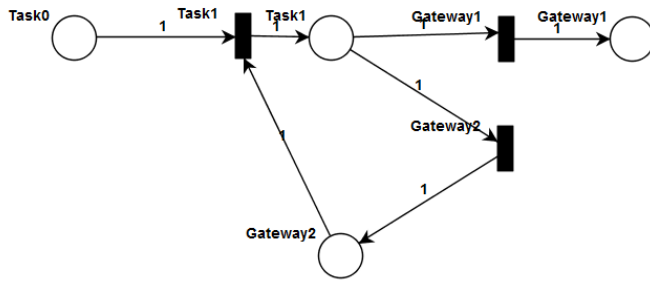


Figure 19. Representation in Petri net by simple replacement from Table 1

So, as we see from figure above, transition “Task1” has two inbound places, but it means that it can run only when both place “Task0” and “Gateway2” are filled. In real BPMN model Task1 can run when just one of its inbound elements (gateway or Task0) is done. That’s why in order to solve this problem (which also occurs when we have cycles) we suggest to exclude the right place of the Petri net set. In figure above we will place “Gateway2” will be excluded and instead connect transition “Gateway2” to the inbound place of the transition “Task1” – “Task0”. See picture below.

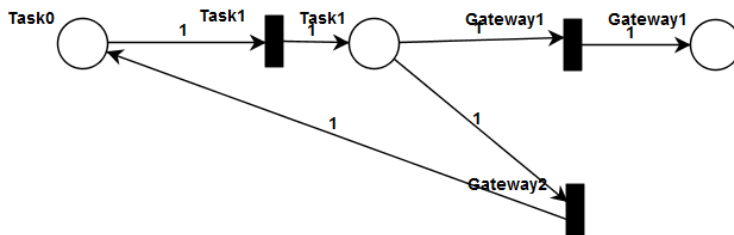


Figure 20. Suggested approach to solve the problem of cycles

The rule will sound as follows: if the current BPMN element has outbound connection to one of the previous elements, then exclude the right place in the simple representation of the current element (from Table 1) and connect transition in its representation to inbound place of the transition, representing the previous element. This rule will work for all types of gateways and tasks.

BPMN diagram can also include subprocesses, which create additional complexity in mapping rules. Subprocess in BPMN can be depicted with a special symbol.

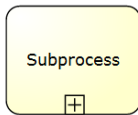


Figure 21. Subprocess in BPMN

The graphic symbol can be expanded by clicking “+”. After expansion the whole process will be seen. So we have to deal with 2 alternative ways of depicting subprocess – single element or the whole process. We suggest following mapping rules for subprocesses:

1. When we want to depict a single element, represent subprocess in Petri net with one transition-one place, see Table 1. Apply the same rule as for the task.
2. When we want to have a complete subprocess visible, include a whole Petri net sub-diagram into super-diagram. In this case start event in the sub-diagram will consist only from one transition and one place (not 2 places, 1 transition), see picture below.

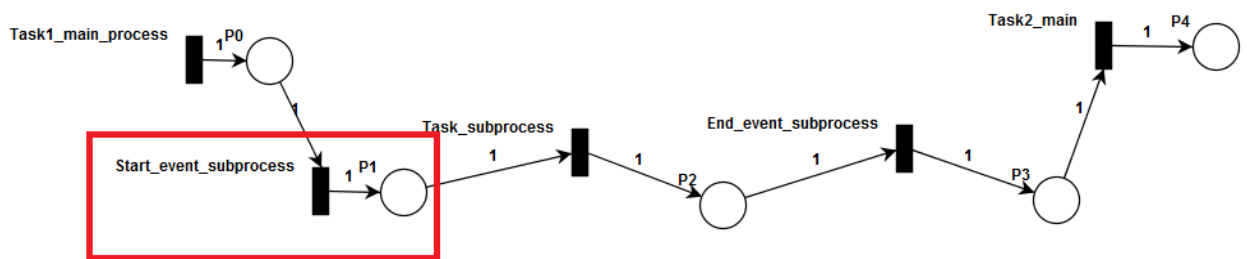


Figure 22. Subprocess in Petri net

As a conclusion, we would like to present a summary table of integration rules, applied when building a Petri net diagram from BPMN model.

BPMN element	Description of problem	Rule
Subprocess	A process can be included in the main process as a single element or as a whole process	<p>Represent a single element as a task.</p> <p>Represent the whole subprocess as usual Petri net process, integrated in the main process. Start event should have one transition and one place.</p>
End event	There could be several end events and different amount of inbound sequence flows to each of them	Represent each connection from BPMN element to end event with a separate transition-place set.
BPMN element A, which has connection to another element B, when B has	Element E2 can have inbound connections from more than 1 element, for example from E1 and E3. It can constitute a cycle	Represent the E1 element without outbound place on the right side. Find transition A in the E1, which will have outbound connection. Find place B, which is inbound to E2.

several inbound sequence flows		Connect transition A to place B.
Swimlanes, pools	Serve the purpose of visualizing, can not be represented in Petri net	Ignore
Other cases		Replace BPMN element with corresponding Petri net set from Table 1

Table 3. Rules of building Petri net from BPMN model

3.3 Business relevant simulation with Petri net

In this chapter we are going to describe how Petri net could be used for business process simulation, so that it will bring some value for enterprise.

First way of using Petri net is connected with *counting probability* that the system is in one or another state. Computing probability is valuable for business when we deal with BPMN diagrams. The need for probabilities arises when we deal with exclusive or inclusive gateways in BPMN. Each of these gateways creates a situation when we have “fork” – several possible alternative outcomes from the gateway. “Forks” in the diagram give alternative ways of going through the process. State of the system is one of such ways in the business process. See picture below.

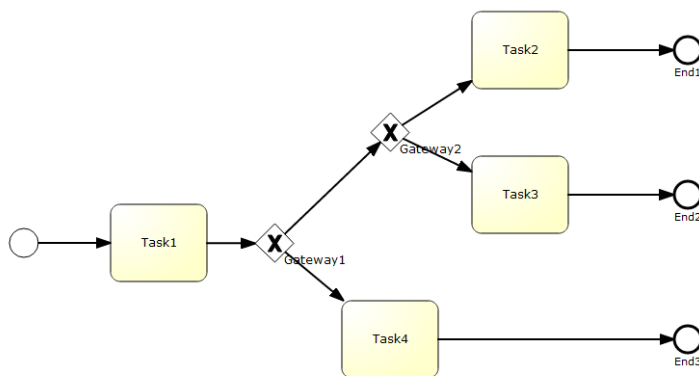


Figure 23. Forks in BPMN

In the case shown above we have three alternative states of system: StartEvent-Task1-Gateway1-Gateway2-Task2-End1; StartEvent-Task1-Gateway1-Gateway2-Task3-End2; StartEvent-Task1-Gateway1- Task4-End3. By transforming this process into Petri net we will be able to simulate the probability that Task2 is triggered, that the process results in End2 or End3, etc.

To make such a simulation we need probabilities for gateways (what is the probability that from Gateway1 process will go to Task4, etc.). And here the challenge appears. Usually in most

current BPMN models gateways do not reflect any probabilities, instead they just show possible outcomes. But this problem can be solved:

1. If gateway in BPMN model refers to technical process, for example “operation in back-end system is performed successful”, the probability information is often accessible in technical description. This information includes reliability of the system, availability, its fault-tolerance [43].
2. Sometimes gateway depicts a subjective decision making, for example “user decided to reject the invoice”. Assigning probability to it is more challenging task. Possible way of solving this problem could be gathering statistical information from company. In many enterprises there is data about patterns of user behavior. For example, in online book shops there is available statistics that a typical client while making a purchase will open one of the “related” books, suggested by system, with certain probability. This statistical information can be brought into gateway.
3. If enterprise does not have enough statistical data, then we can not compute probability properly, but we can avoid the problem by switching probability from 1 to 0, see picture below.

The picture shows Petri net equivalent for BPMN model, presented earlier in this chapter (“Forks in BPMN”).

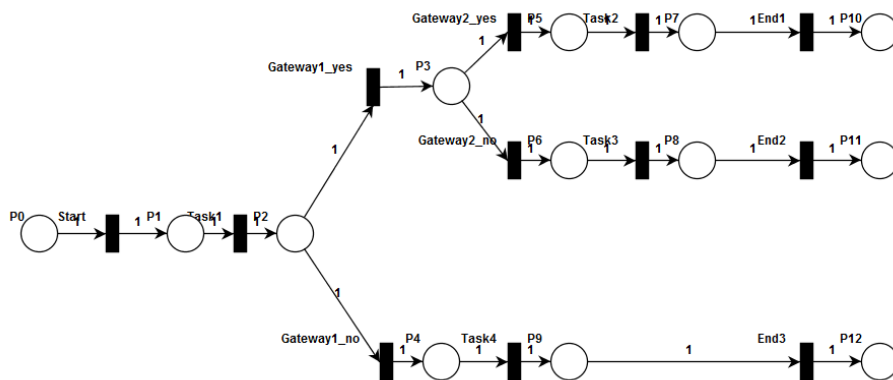


Figure 24. Forks in Petri net

As we can see, there are two pairs of transitions representing gateways – Gateway1_yes, Gateway1_no and Gateway2_yes, Gateway2_no. Our purpose is to compute probability that the process will end in End1, End2 or End3 events. Places P10, P11, P12 symbolize that these end events are accomplished. For example, we have probability data for Gateway2 but do not have for Gateway1. Then we can compare between each other probability that process ends in P10 or

P11, but we can not compare P10 and P12, P11 and P12. Simulation with Petri net for finding probabilities could be performed in the following way. First, define the probability that transitions “Gateway2_yes” and “Gateway2_no” fire (these probabilities are available to us, as a sample, we can set 70% for Gateway2_yes and 30% for Gateway2_no). Second, make initial marking for P0 big enough to get good statistical information (let’s say 10 000 tokens). Third, switch between transitions Gateway1_yes and Gateway1_no by setting their probability 1 or 0. Then we can make a simulation and get corresponding probabilities for End1 and End2. The result would be around 7000 tokens for P10 and 3000 for P11, when Gateway1_yes set to 1. This sample may seem too simple, because number of tokens in P10 and P11 is defined by the probability for Gateway2 what is already known for us. In this case we do not need a simulation, we took this example only for illustration of procedure, but in real life models we can have a lot of gateways, cycles, so counting result probabilities is no longer a trivial task.

Another way of using Petri net in business process simulation is *time-related simulation*. One of the most often used functionalities in Petri net is estimating time delays and the total time, which is required for the process to complete.

This functionality is achieved through assigning firing times to transitions. Based on the type of transition firing time, we can deal with two kinds of Petri net – deterministic and stochastic. In deterministic approach time delays are assigned as fixed values to transitions. In stochastic Petri net transitions “have exponentially distributed firing delays with rate λ ” [44]. Exponential distribution is used when we deal with time between events of Poisson process. Probability density function is

$$f(x; \lambda) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0, \\ 0, & x < 0. \end{cases}$$

Figure 25. Probability density function of exponential distribution

In order to assign firing times we need to do:

1. If time delay for transition is fixed, then define its firing time in Petri net definition in the beginning.
2. If time delays are stochastic, then use mathematical functions to generate time delays from a given distribution. In this case generation of time delays will happen at runtime during execution of the model.

To illustrate time simulation in Petri net, which is valuable for business purposes, we made a sample BPMN model.

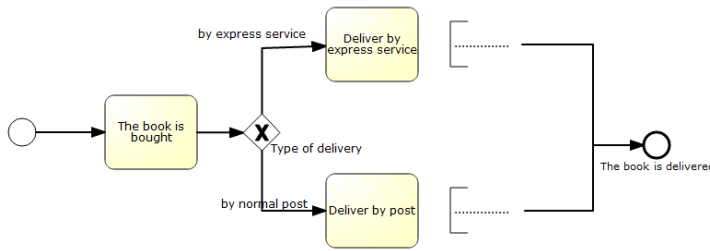


Figure 26. Simple delivery process model

There are two delivery opportunities in the model –by normal post or by express service. It is obvious that express service will require less time and book will be delivered earlier. In our simulation we want to compute how much time the whole process will require if delivery is done by post and by express.

In order to make such a simulation we will need time delays for tasks – “Deliver by express service” and ‘Deliver by normal post’”. For example, delivery by normal post requires 12 days and by express service 1 day. Then after transformation to Petri net the model will look like

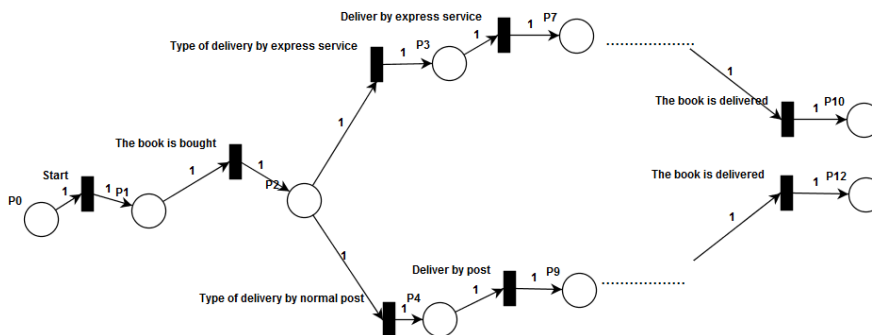


Figure 27. Petri net for simple delivery process model

Transition “Deliver by express service” will have 1 as time delay, transition “Deliver by post” – 12. To compare results with each other we can make two simulations: first, by putting 100% firing probability for transition “Type of delivery by express service”, second, by putting 100% firing probability for transition “Type of delivery by normal post”. Initial marking should include 1 token in P0. After making a simulation we will get results, i.e. finishing time, when token will be in P10 and when in P12.

Third way of applying Petri net methodology in business-relevant simulation is simulation with *resources*. Resource could be a person, a free machine, etc. The need to utilize resources in the

simulation appears when elements or sets in the process (such as tasks or chain of BPMN elements) require similar resources to be accomplished. The model below shows such case.

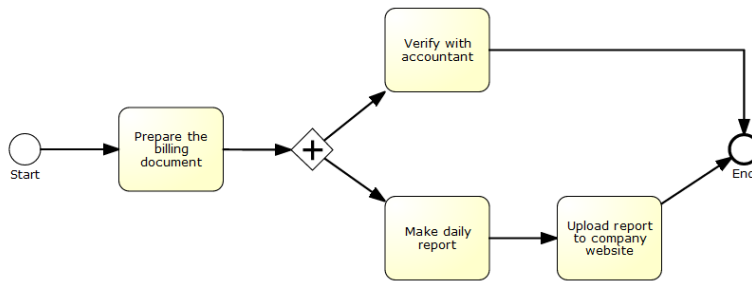


Figure 28. Sample BPMN model with resource sharing

Tasks “Verify with accountant” and “Make daily report” both require accountant for accomplishing them. If a company has only one available accountant, then these tasks will share the resource (accountant) between each other.

General model of a Petri net subset, which handles resources, presented below.



Figure 29. General sketch of Petri net with resource sharing

Process here could be a single transition or a set of Petri net elements (but first element in this set should be transition). The transition will fire only in case when there is a need for it (for the task or subprocess) and there is available resource.

When we simulate resource sharing, Petri net with colors could be useful. Because often resources have capacity (for example, amount of working hours), during the processing of such tokens in transitions capacity can be changed. To illustrate this, we will transform BPMN model with resource sharing (see figure above in this chapter) into colored Petri net.

There are two tasks, requiring the same resource – “Verify with accountant” and “Make daily report”. Let’s suppose, that verification requires 1 hour work, making report 4 hours, preparing the billing document – 3 hours, upload report to company’s website – 0,1 hour. The company has only one accountant with 8 working hours available. The corresponding Petri net model for this process is shown below.

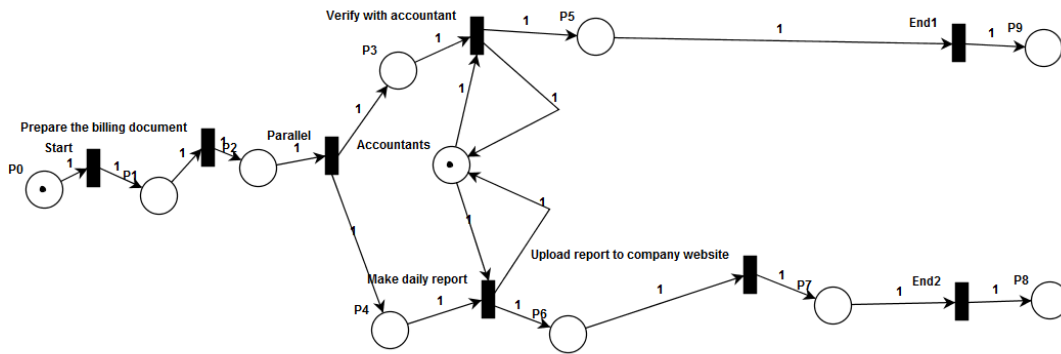


Figure 30. Petri net with resource sharing

Resources are located in place “Accountants” (1 token stands for one accountant). So one resource is represented with one token. Both transitions (“Verify with accountant” and “Make daily report”) have the inbound connections from place “Accountants”. Only one of this transitions can fire at a time, as inbound place has only one token. The resource (accountant) has 8 available hours, we can put this number as a color to token. If transition “Verify with accountant” fires, then it changes the color of token from 8 to 7 and after one time point, puts the token back to place “Accountant”. Then transition “Make daily report” fires and changes the color of token from 7 to 3 and returns the token back after 4 time points. After the simulation we will see that accountant was busy during 5 hours and still has 3 more hours, completing the whole process took 8,1 hours instead of 7,1 hours (because in fact “Verify with accountant” and “Make daily report” were not accomplished in parallel).

In order to make the simulation colored and timed Petri net is needed. These technologies belong to the so called high-level net unlike a simple Petri net, which has no time or color.

Application of Petri net, described in this chapter, is valuable for business, because by help of it we can make performance analysis. In the beginning of simulation we input some parameters into Petri net (gateways’ probabilities, time delays for activities, number of available resources, their capacity). Then after simulation, we get results, which we can compare with each other. Based on it, best input parameters could be selected or the whole BPMN model could be changed in order to improve its performance.

As a conclusion, there is a table summing up ways of Petri net usage for business simulation.

Business purpose	Realization	Type of Petri net
Probability that business process will go this or that way	-assign probabilities to gateways’ transitions; -insert a statistically big number of tokens in initial place;	Simple Petri net

	-make simulation and count number of tokens in target places (this step will give the probability that that process will go through certain target place)	
Time delays, total time of business process to complete	-assign firing times to transitions (as fixed values once in a program or as random values from exponential distribution at runtime); -make simulation and compute time delays, total time for the process	Timed Petri net
Resource (people, machines. etc.) usage by elements in BPMN scheme	-create additional places in Petri net for keeping resources; -assign color to tokens if we want to measure their capacity; -during simulation change colors in tokens if they are used; -make a simulation and count how much time resources were occupied, resources were free, how much capacity left in resources (if colors were used)	Timed and color Petri net

Table 4. Petri net usage for business relevant simulation

3.4 Extension to BPMN diagram

In previous chapter we described how Petri net could be used in simulation, valuable for business. We decided that for this purpose high-level (timed and color) Petri nets should be used. BPMN technology by default does not suggest such functionality, as probability for gateways, time delays for tasks or other elements, resources. In this chapter there are suggestions of how currently used BPMN diagrams could be changed so that these functionalities would be enabled.

Besides changing the BPMN diagram, following techniques can be applied in order to utilize time, color, resources, probabilities in Petri net:

1. Add all necessary additional information directly to Petri net after transformation from BPMN diagram. Then additional places, time delays for transitions, etc. can be added manually to Petri net. It requires analysis made by the person, good understanding of Petri net technology.
2. The intermediate layer program (i.e. the one, which makes transformation from BPMN to Petri net) should suggest opportunities to user to define time delays, resources for BPMN diagram which is being processed.

The first way of solving the problem with additional functionality in Petri net is not good enough. It has limitations related to subjective factor. After an user-friendly, easily

understandable BPMN diagram is transformed to Petri net, it loses its simplicity. The user is required to understand the structure of Petri net, its syntaxes and manually enter all necessary information in Petri net file (so as a result huge part of mapping between BPMN and Petri net will be done manually).

Second suggestion is more user-friendly, although again it requires some work from user when doing mapping. But in this case work becomes much simpler, as the program will ask the user to write additional information for BPMN model (not to a Petri net, which is more complicated). User does not need to understand the syntaxes and the whole Petri net technology in order to update Petri net model with new information. Mapping program will update Petri net by itself (create new places, new colors, firing times for transitions, etc.).

But the preferred way of enabling useful Petri net functionality is to make changes directly in BPMN, which is more known for users. Therefore editing BPMN diagram will be a simpler task and then mapping between a new enhanced BPMN diagram and Petri net will be done automatically.

The next issue is to define how additional information can be added to BPMN model without changing its structure and workflow. One of possible solution to this problem could be annotations. Annotations does not change the flow of the BPMN model, instead they are needed for clarification, explanation for user. But in Petri net these features are not needed and can not be reflected, because main purpose of Petri net is simulation. In Petri net annotation is not represented and is discarded while transformation. So there is minimal risk for us when we are editing annotations, that this will make any damage to BPMN or Petri net model.

Table in the end of previous chapter summarizes which functionalities should be brought into Petri net. One of them is *probability*. Probabilities should be assigned to gateways. Exclusive gateways can have one of two possible outcomes, see the picture.

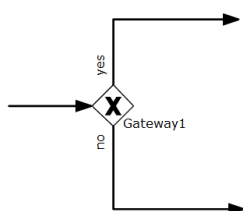


Figure 31. Gateway

In Petri net such a diagram will be represented with two transitions (“Gateway1_yes”, “Gateway1_no”) and, occasionally, two places. Each of the transitions should have a probability that it fires (what is the probability that outcome of the gateway will be yes or no?). Our propose is to add special text to sequence flows outgoing from gateway.

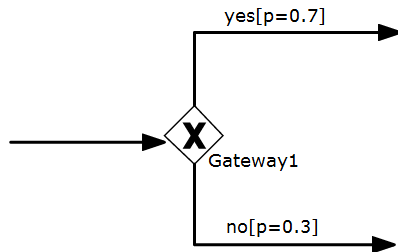


Figure 32. Probability for gateways

$p=0,7$ means that probability of the sequence flow to be executed is 0,7. Later when transforming this BPMN diagram into Petri net element, this number will be interpreted as a probability that transition “Gateway1_yes” fires. Transition “Gateway1_no” will have 0,3 probability.

Second functionality in Petri net is *computing time*. Activities in BPMN diagram consume time and it is reflected in time delay for transitions. Example of such activities is shown below.

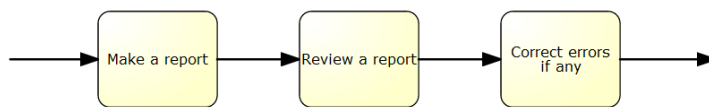


Figure 33. Activities with time consumption

There are three activities – “Make a report”, “Review a report”, “Correct errors if any”. Each of them requires some time to be completed. For example, company has statistical data and knows that usually making a report takes 3 hours, reviewing a report – 2 hours. But time delay for the last activity is not fixed value, it can be 0 hours (if no errors) or half an hour or 1 hour, etc. Suppose, we have an information that it is exponential distribution with parameter $\lambda = 2$. Time delays for this activity will be random variables from exponential distribution with parameter 2. In Petri net this chain of BPMN activities will be represented with 3 transitions (“Make a report”, “Review a report”, “Correct errors if any”) and 3 places. Firing times for first and second transitions will be 3 and 2, third transition has random firing times, which will be generated at runtime.

In order to insert additional data about time delays we propose to use annotations for these activities, starting with “t=”. If time delay is fixed, then we should write in annotation “t=v”,

where v is time delay. If time delay is random variable, we can write “ $t=ev$ ”, where v is parameter for exponential distribution. After these corrections diagram will have following look.

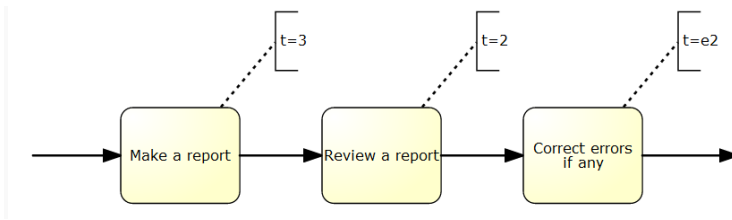


Figure 34. Annotations with time information in activities

Third functionality in Petri net is *resource sharing*. We also propose to use annotations in BPMN in order to enable resource sharing in corresponding Petri net diagram. In order to implement resource sharing in the model, following information is needed:

1. Number of available resources and their types.
2. For each activity: how many and what type of resources does it require.

First information can be given in annotation to start event. Second – in annotations to activities. Type of required resource can be depicted as “ $r_t=v$ ”, where v is name of the resource type. Number of resources “ $r_n=v$ ”, where v – the number. These expressions are separated from each other with “&”, from other expressions with “;”. Example of annotations used for resource related information is shown below.

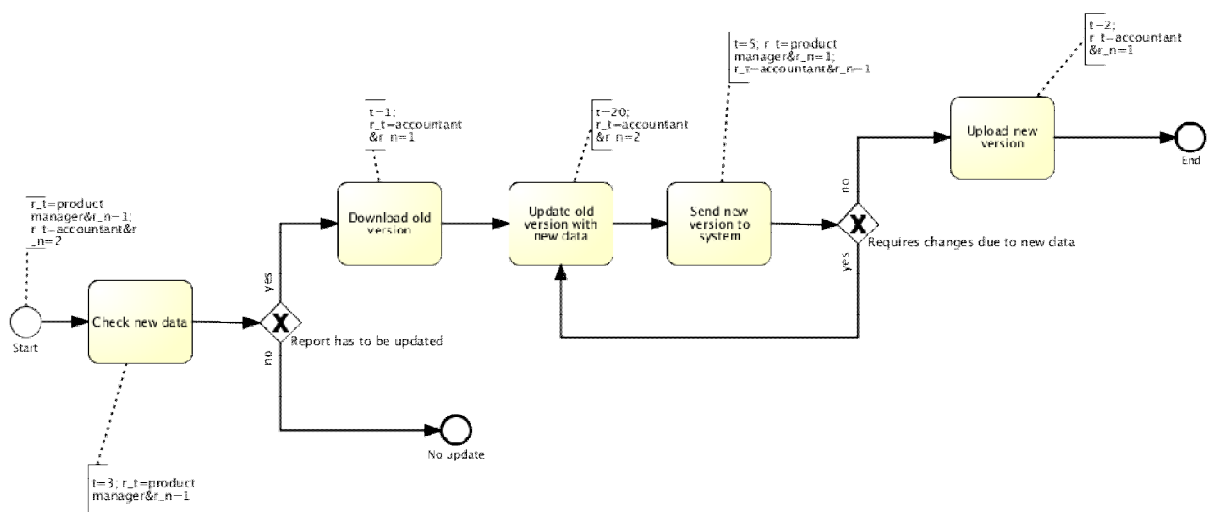


Figure 35. Annotations with resource-related information in BPMN

These annotations give to us following information:

1. Activity “Check new data” requires 1 resource called “product manager”.

2. “Download old version” requires 1 accountant.
3. “Update old version with new data” requires 2 accountants.
4. “Send new version to system” requires 1 product manager and 1 accountant.
5. “Upload new version” requires 1 accountant.

Available resources in BPMN process include 1 product manager and 2 accountants (this information is given in start event).

But writing annotation to each activity is not always convenient, because it requires a lot of work especially when type and number of required resources is the same for a big set of activities. Solution to this problem could be use of collapsed pool which has connections to several activities.

For example, in the above BPMN model we will make slight changes: now for one activity (“Check new data”) it is required 1 product manager; other activities (“Download old version”, “Update old version with new data”, “Send new version to system” and “Upload new version”) require 1 accountant. Then we can reduce the number of annotations for this model and final BPMN model will look as shown.

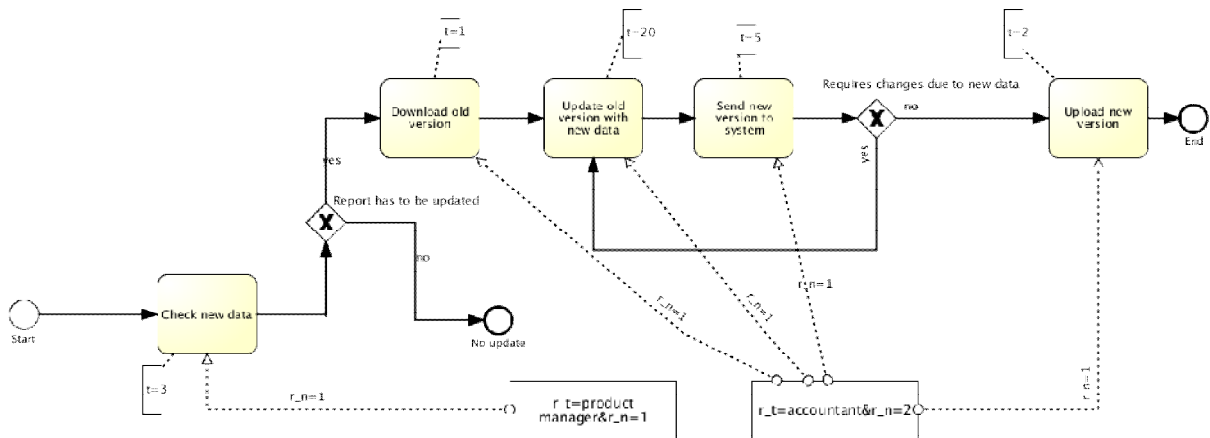


Figure 36. Resource related information in BPMN (with collapsed pools)

We can see from the picture that for the group of activities (“Download old version”, “Update old version with new data”, “Send new version to system” and “Upload new version”) there is only one collapsed pool. Each collapsed pool has connections to activities. Connections have signs “r_n=1” which means that this activity requires 1 resource from the collapsed pool. Now there are no annotations in start event.

After transformation of the first BPMN diagram (with a lot of annotations) to Petri net we will get following model.

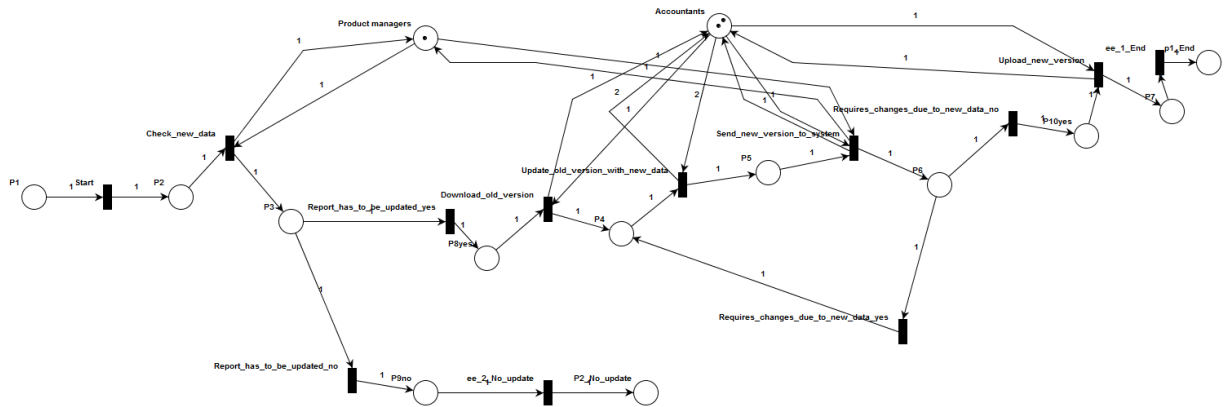


Figure 37. Petri net with resource sharing (big number of annotations case)

The second BPMN diagram in its Petri net form will have a following look.

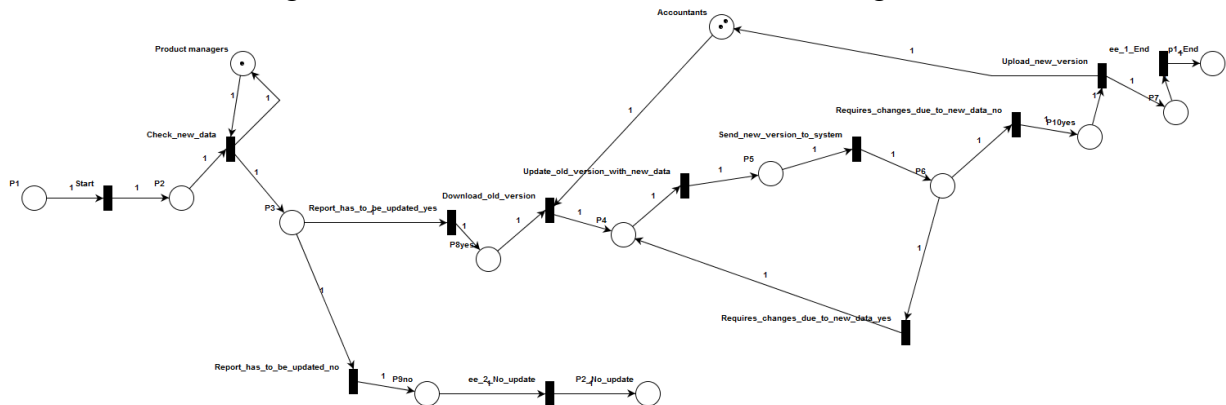


Figure 38. Petri net with resource sharing (with collapsed pools case)

By this enhancement (usage of one collapsed pool for several activities) we get Petri net with a simpler and more understandable structure. It is important to notice that for each type of resources (product managers or accountants) there is one place in Petri net, number of tokens in each place corresponds to number of resources of the given type. This information (type and number of available resources) was defined earlier in BPMN diagram in annotations and collapsed pool.

We would like to compare these two approaches – writing annotations with resource information separately for each activity and using collapsed pool instead. In the first approach there is annotation to start event which describes number and type of available resources. Annotations to other elements (activities) symbolize type and number of required resources. This is a difference between annotations in start events and other BPMN elements. In the second approach there are

no annotations for start event and activities, but instead collapsed pools and connections from them to activities are used.

We consider that the second approach has several advantages:

1. Structure of BPMN diagram becomes more similar to Petri net. This similarity between BPMN model and Petri net diagram will make transformation from one to another easier. For example, collapsed pool in BPMN symbolizes resource pool in Petri net. Expression “ $r_n=2$ ” in collapsed pool will mean 2 tokens in Petri net resource pool.
2. There is less work required from user to enter all resource related data in BPMN, because there is no need to write annotation for each activity.
3. Amount of annotations and text on BPMN model becomes less.

So our suggestion is to use collapsed pools in order to express resource related data.

Together with resource sharing we can utilize use of *color* in Petri net. Color will stand for resource capacity. In our case, see previous 2 BPMN models, resources are product manager and accountants. Each of them can have its own capacity, for example, working hours. This capacity can be expressed through colors. Suppose, product manager has 4 available working hours, then token in place “Product managers” will have color “4”. After transition “Check_new_data” fires, it will change token color to “1” and return this token back to place “Product managers”. The same will happen with tokens in “Accountants” place.

In order to implement color handling in Petri net it is also needed to use annotations together with collapsed pools. For this purpose following information is needed:

1. Which capacity does every available resource have in the beginning of the process?
2. How does the capacity of each resource change during the process? Or, if put in another way, what every activity does with used resource’s capacity during execution?

First information is given in collapsed pool or in annotation to start event (if we are using this approach). Second - in associations from collapsed pools to activities. There are two approaches to resource handling in the model, full and simplified. In a full-functional resource approach resource capacity can be expressed in whatever units of measure (time, kilograms, etc.). In a simplified model resource capacity is always available time points. This means that if resource of type “consultant” has capacity equals to 10 hours, then the overall available time of this resource is 10 hours, during exploitation of “consultant” in process execution this time will be decreased.

First, we would like to describe how the full version approach will work (i.e. when capacity can have whatever units of measure). In the collapsed pool we can introduce additional expression “ $r_c=v$ ” where “ v ” is amount of capacity. This expression should go together with “ $r_t=...$ ” as it relates to the certain type of resource. If this expression is used in collapsed pool, for example “ $r_t=secretary&r_n=1&r_c=10$ ”, then it means that we have 1 available secretary for our process with capacity equals to 10. Also we have connections from collapsed pool to activities. Each connection shows how the activity changes capacity (it can be increased or decreased). This change is defined in “ $r_c=f(r_c)$ ”, where $f(r_c)$ is function of r_c . For example, expression “ $r_n=1&r_c=r_c-2$ ” in connection means that the activity decreases capacity of the resource “secretary” by 2 (i.e. after this activity one secretary will lose 2 hours from his capacity).

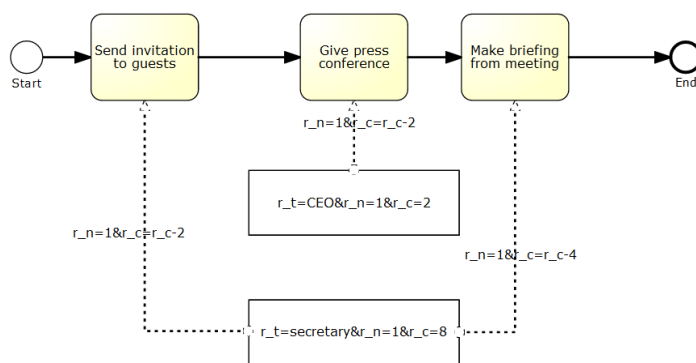


Figure 39. Resource sharing with capacity in BPMN (full form)

In the picture two collapsed pools stand for available resources (1 “CEO” and 1 “secretary”). “CEO” has 2 available hours and “secretary” – 8. Activity “Give press conference” requires 1 “CEO” and decreases “CEO’s” available hours by 2. Activities “Send invitation to guests”, “Make briefing from meeting” requires correspondingly 2 and 4 hours from secretary. After transformation of this model to Petri net we will have two resource pools (two places) with one token each.

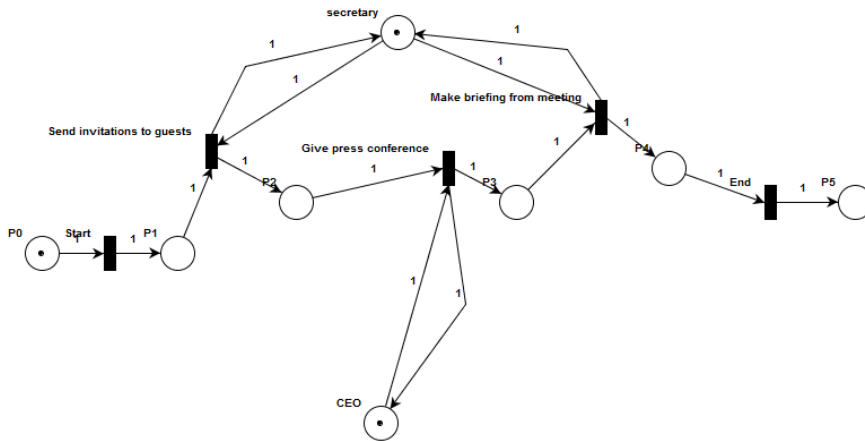


Figure 40. Petri net for resource sharing with capacity (full form)

Token in place “CEO” will have 2 as color and token in “secretary” will have 8 as color. After the execution of the process is completed, token in “CEO” will have color equal to 0 and token in place “secretary” will have color equal to 2. It means that the whole process took 2 hours from CEO and 6 hours from secretary. What is important here is that, instead of working hours, color can symbolize whatever measurement. For example, resource can be a ship, color is its cargo and during different activities cargo will change (decrease at unloading and increase at stevedoring).

If capacity stands for time points (working hours, working days, etc.), then instead of “ $r_c=f(r_c)$ ” in connections we should use time delay information from activities. This allows to eliminate resource related text in connections. This is simplified form of resource handling in BPMN. Example of this approach is shown below.

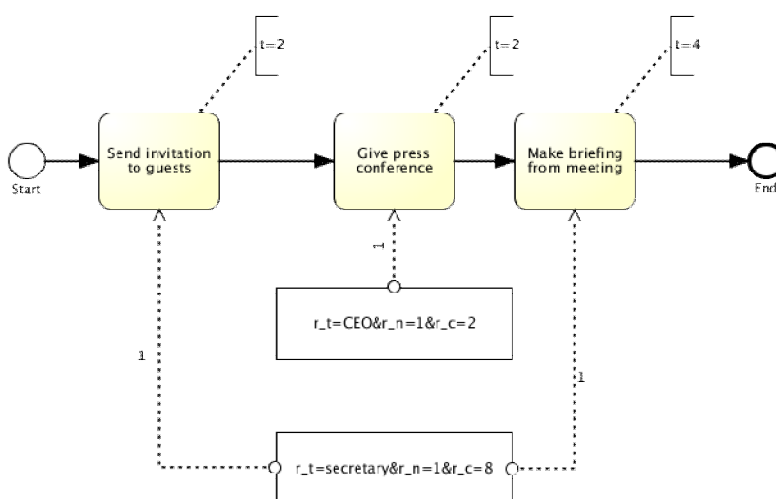


Figure 41. Petri net for resource sharing with capacity (simplified form)

In the picture above connections from collapsed pools to activities have even less text – only number (which symbolizes number of required resources from the pool). We do not explicitly define how colors should be handled in activities, it is implied that they are all decreasing correspondingly to delay times for activities. For example, after activity “Send invitation to guests”, which has 2 time points as accomplishing period, is finished, then the token “secretary” will lose in its color 2 points (it will decrease from 8 to 6).

The simplified form is easier to implement in BPMN, because requires less work from user, but is limited in functionality (color can stand only for available time and can only decrease).

In the current chapter we proposed features which can enhance BPMN model and enable several useful things in Petri net – probabilities computing, time delays, resource sharing and color handling. Table below summarizes all proposed features, rules of their implementation in BPMN and rules of their interpretation during transforming to Petri net.

Feature	Implementation in BPMN	Transformation to Petri net
Probability	Sequence flows outgoing from gateways have additional text in their title “[p=v]”, where v – is the probability that this sequence flow will be activated	“p=v” will be interpreted as probability for transitions, which represent gateway
Time delays	-if time delay for activity is fixed, then write “p=v” in annotation where v is amount of time -if time delay is random variable from exponential distribution, then write “p=ev” in annotation where v is parameter λ	-if time delay is fixed, then define its firing time in Petri net definition in the beginning. -if time delays are stochastic, then use mathematical functions to generate time delays from a given distribution. In this case generation of time delays will happen at runtime during execution of the model.
Resource sharing	-In order to describe number and type of available resources for the process, use collapsed pools. Each collapsed pool should have a title in the form “r_t=name&r_n=v”, where name is the type of resource and v – number of available resources -Each type of resources should have separate collapsed pool (i.e. only one resource type in one collapsed pool) -In order to describe type and number of required resources for the task, use connections from corresponding collapsed pools to activities. Connections should have titles in the following form “r_n=v” where v – is the number of required	-Each collapsed pool should be interpreted as place (which has a meaning of resource pool) in Petri net. name – is the name of the place, v – is number of tokens in the place. -For each activity Y, which has a connection with number V_a between collapsed pool X and activity Y: 1. If activity Y’, which is preceding to Y does not have any connection from X, then create a new

	resources for the activity.	<p>connection from place X to transition Y with weight=V_a.</p> <ol style="list-style-type: none"> 2. If Y' has connection B with $V_b < V_a$, then create a new connection in Petri net from place X to transition Y with weight=$V_a - V_b$. 3. If activity Y'', which is subsequent to Y, has connection C from X with $V_c < V_a$, then create a new connection from transition Y to place X with weight=$V_a - V_c$. 4. If activity Y'' does not have any connections from X, then create a new connection from transition Y to place X with weight=V_a. 5. If activity Y', which is preceding to Y, has connection B from X with the $V_a \leq V_b$, or if activity Y'', which is subsequent to Y, has connection C from X with $V_c \geq V_a$, then do nothing.
Color	<p>1.For a full version:</p> <ul style="list-style-type: none"> -Use additional expressions in collapsed pools' titles. Expressions should have form "$r_c = v_1 * v_2 \dots$" where v_1, v_2 are capacity of the resources. For example, "$r_t = \text{professor} \& r_n = 2 \& r_c = 10 * 5$" says that there are 2 professors each of them has 10 and 5 capacity correspondingly. -Use additional expressions in titles of connections from collapsed pools to activities. These expressions should have a form "$r_c = f(r_c)$" where $f(r_c)$ is a function, showing, how color (i.e. r_c) will change in the activity. For example, "$r_c = r_c - 10$" says that color should be decreased by 10. <p>2.For a simplified version.</p> <ul style="list-style-type: none"> -Titles in collapsed pools remain the same as in full version. -Titles of connections have only number, which means number of required resources 	<p>1.For a full version</p> <ul style="list-style-type: none"> -Create colors in tokens, kept in resource pools, in accordance with "$r_c = \dots$" expressions -Each transition should handle colors in accordance with title of connection of the form "$r_c = f(r_c)$" <p>2.For a simplified version.</p> <ul style="list-style-type: none"> - Create colors in tokens, kept in resource pools, in accordance with "$r_c = \dots$" expressions -Transition decreases the color of token accordingly to its time delay. <p>new color = old color – time delay</p>

Table 5. Extensions to BPMN diagram

4 Proof of concept implementation

4.1 General overview of implementation

In the previous chapter it was shown that BPMN transformation to Petri net is possible and the rules of transformation were described in more details. Further we would like to show the implementation of these rules in order to prove that described approach will work.

While the previous chapter contains theoretical description of suggested approach, the current chapter is focusing on practical realization of the approach. Following tools were used:

1. Signavio Process Editor. BPMN model was created in this tool, the exported to XML file. Additional features for BPMN were also realized here.
2. Own Java program was created in NetBeans. The program interprets xml file and transforms BPMN diagram into Petri net.
3. GPenSIM was used in order to make simulation with Petri net model, extracted in the previous step.

Picture below shows the overview of practical realization of the approach, described in the previous chapter.

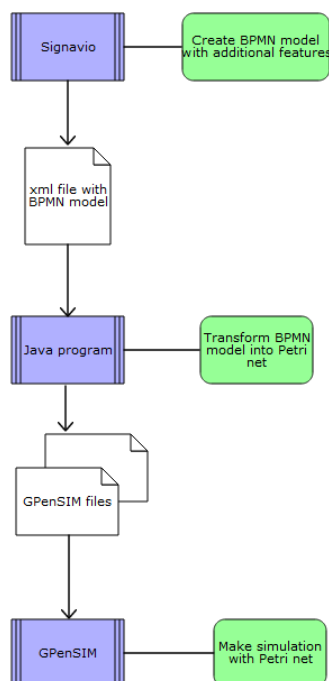


Figure 42. Overview of BPMN-to-PetriNet practical realization

Main goals of the above-mentioned steps shown in the figure:

1. To prove that transformation from BPMN diagram to Petri net can be accomplished and the result Petri net model will be a right equivalent of the initial BPMN model.
2. To show that by adding new information to BPMN diagram the final Petri net model becomes more powerful and useful.
3. To demonstrate that Petri net diagram, extracted from BPMN, has valuable functionality, which can be simulated and give useful results.

4.2 Architecture of Java program

Java program is the second block in the figure “Overview of BPMN-to-PetriNet practical realization”. The program has the architecture, shown below.

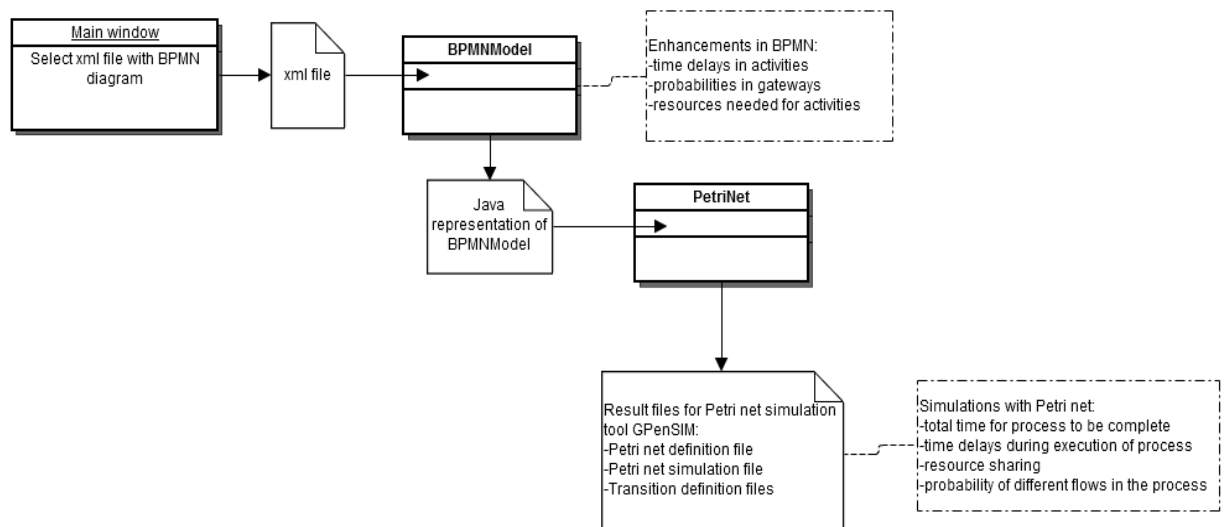


Figure 43. Architecture of Java program

There are three main classes in the program – MainWindow, BPMNModel and PetriNet.

1. MainWindow is a GUI which helps user to select xml file with BPMN diagram, stored locally on PC. Also it can deal with certain enhancements in BPMN diagram (shown earlier).
2. BPMNModel is a class which creates representation of BPMN model. It extracts data from xml and creates instances of class in java program, representing xml data.
3. PetriNet is the class where the main work of transforming BPMN to Petri net is done. As a result of the program GPenSIM files are created (Petri net definition, Petri net simulation and transition definitions). Further these files can be used in GPenSIM

software to make simulations of time delays, total time of process, probabilities, resource sharing.

While *MainWindow* class is just a GUI, we do not concentrate on it too much. The main purpose of *BPMNModel* class is to contain in a convenient way BPMN diagram (all its elements, connections between them, annotations, etc.). Then it will be easier to work with them when doing transformation to Petri net. *BPMNModel* class has following structure.

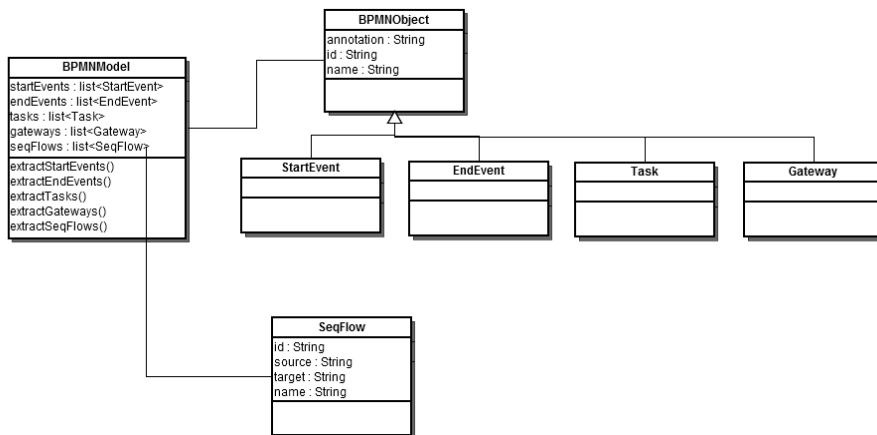


Figure 44. Structure of *BPMNModel* class

Following BPMN elements can be interpreted by the program:

1. Exclusive gateway.
2. Start event.
3. End event.
4. Activity (Task).
5. Sequence flow.

Each of these elements has corresponding java class. Class *BPMNModel* stores lists of these elements (startEvents, endEvents, etc.).

At the input to *BPMNModel* there is an xml file. This file is queried with the help of Java library `org.w3c.dom`. Queries are based on assumption that xml file was created in Signavio Process Editor and all elements have certain names (for example, “textAnnotation” for annotations). But if the BPMN model was created in another software and xml file has different element names, then they could be easily changed in *BPMNModel* code.

There are number of such methods as “extract...”. In this method all necessary information about certain BPMN element (id, name, source and target for sequence flows) is extracted from xml

file and stored into the relevant list in *BPMNModel* class. Annotations are elements in xml file which are stored separately, that's why they are extracted in method of class *BPMNModel* "extractAnnotations".

Another java class used in program is *PetriNet*. UML diagram of it is shown in the figure below.

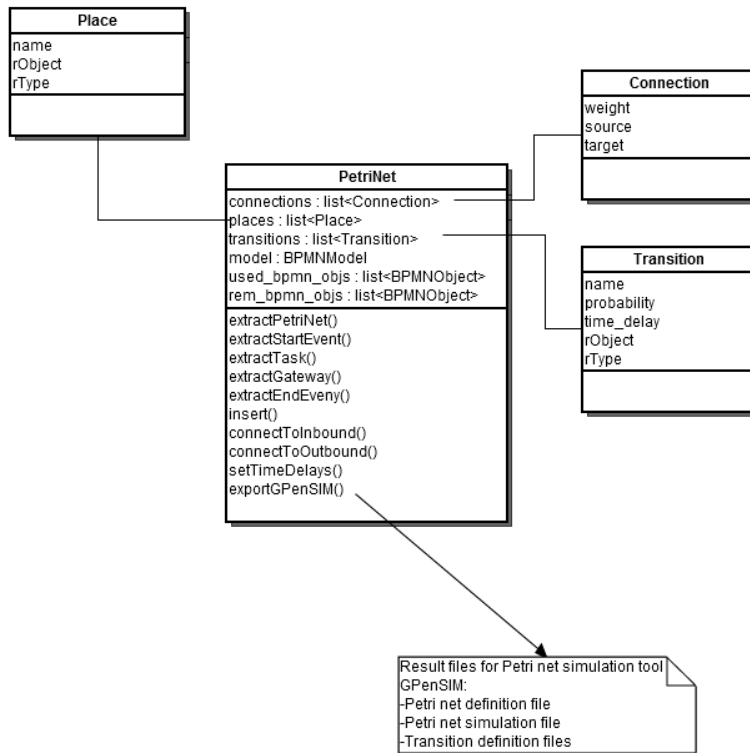


Figure 45. Structure of *PetriNet* class

Methods in *PetriNet* class convert from BPMN to Petri net strictly according to rules suggested in chapter 3.

4.3 Testing

In this chapter we are going to test the approach, suggested by us, of transforming static Enterprise Architecture diagrams into dynamic ones. For testing of the approach we will use instruments, mentioned in chapter 4.1. Our own Java program is a part of these instruments. First task is to take a sample BPMN model or create our own.

BPMN model for recruitment process was created in Signavio tool. Model shows how the recruitment in company occurs since the moment it gets CV until candidate is rejected or accepted.

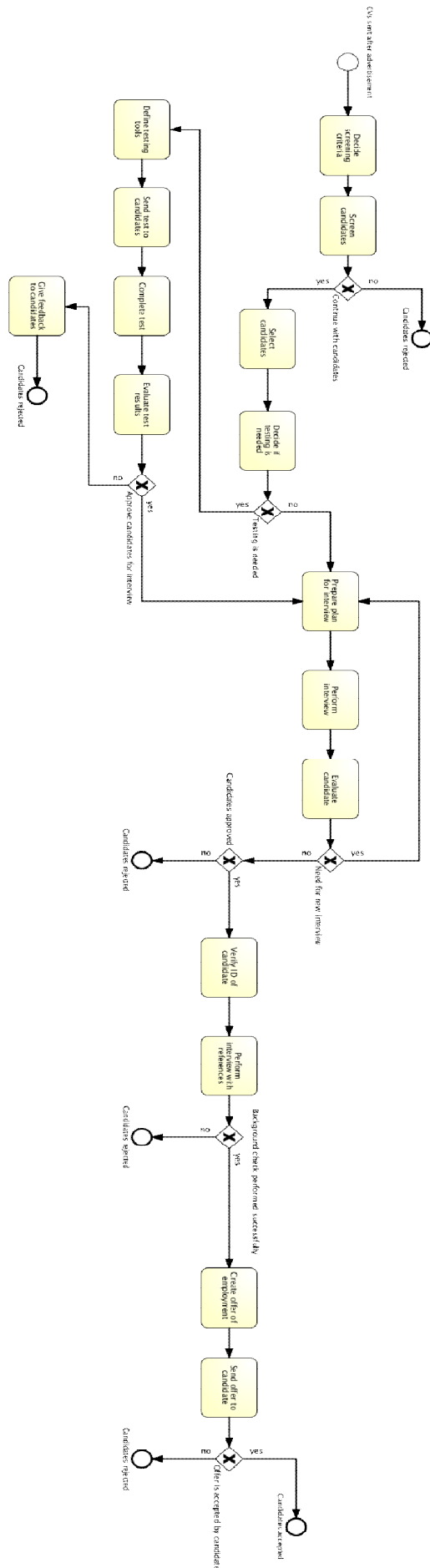


Figure 46. BPMN model of recruitment process

Then this model is transported to XML file. Java program translates xml file and extracts all the data which is needed for building a Petri net out of it. All rules, mentioned in chapter 3 concerning mapping between these two standards, are implemented. The result of Java program are files in GPenSIM standard, representing corresponding Petri net. These files contain primarily information about Petri net structure, also about simulation procedures for it. To visualize the structure we created a picture with Petri net graph (the same Petri net is described in GPenSIM main definition file “model_def.m”, but not in graphical format).

As it can be seen from BPMN diagram, model has 6 end events, they correspond to 6 places in Petri nets:

1. p1_Candidates_rejected. It is triggered after activity “Give feedback to candidates”.
2. p2_Candidates_rejected. It is triggered after gateway “Continue with candidates”.
3. p3_Candidates_rejected. It is triggered after gateway “Candidates approved”.
4. p4_Candidates_rejected. It is triggered after gateway “Background check performed successfully”.
5. p5_Candidates_accepted. It is triggered after gateway “Offer is accepted by candidate”.
6. p6_Candidates_rejected. It is triggered after gateway “Offer is accepted by candidate”.

Then in GPenSIM the extracted Petri net can be simulated. For example, we can define probability for each gateway as 0.5 (for example, gateway “Offer is accepted by candidate” has 0.5 likelihood that it is accepted and that it is rejected, so both sequence flows from it have equal probability). Firing times for transitions (i.e. time consumption by each activity) are all equal to 0. By inputting in initial place X number of tokens, we would be able to perform X number of simulations. Firing times and number of initial tokens are defined in main simulation file, probabilities – in transition definitions:

```
png = petrinetgraph('model_def');
dynamicpart.initial_markings={'p1', 100};
dynamicpart.firing_times={'CVs_sent_after_advertisement',0.0,'Decide_screening_criteria',0.0,'Screen_candidates',0.0,'Select_candidates',0.0,'Decide_if_testing_is_needed',0.0,'Prepare_plan_for_interview',0.0,'Evaluate_test_results',0.0,'Complete_test',0.0,'Send_test_to_candidates',0.0,'Define_testing_tools',0.0,'Give_feedback_to_candidates',0.0,'ee_1_Candidates_rejected',0.0,'Evaluate_candidate',0.0,'Perform_interview',0.0,'Verify_ID_of_candidate',0.0,'Perform_interview_with_references',0.0,'Create_offer_of_employment',0.0,'Send_offer_to_candidate',0.0,'Continue_with_candidates_no',0.0,'Continue_with_candidates_yes',0.0,'ee_2_Candidates_rejected',0.0,'Testing_is_needed_no',0.0,'Testing_is_needed_yes',0.0,'Approve_candidates_for_interview_yes',0.0,'Approve_candidates_for_interview_no',0.0,'Need_for_new_interview_yes',0.0,'Need_for_new_interview_no',0.0,'Candidates_approved_yes',0.0,'Candidates_approved_no',0.0,'ee_3_Candidates_rejected',0.0,'Background_check_performed_successfully_yes',0.0,'Background_check_performed_successfully_no',0.0,'ee_4_Candidates_rejected',0.0,'Offer_is_accepted_by_candidate_yes',0.0,'Offer_is_accepted_by_candidate_no',0.0,'ee_5_Candidates_accepted',0.0,'ee_6_Candidates_rejected',0.0};
sim = gpensim(png, dynamicpart, global_info);
print_statespace(sim);
plotp(sim, {'p1_Candidates_rejected', 'p2_Candidates_rejected', 'p3_Candidates_rejected', 'p4_Candidates_rejected', 'p5_Candidates_accepted', 'p6_Candidates_rejected'});
```

Figure 48. Main simulation file

After 100 simulations with Petri net we got following results for 6 places, mentioned above (these results are drawn in standard GPenSIM function “plotp”, axe Y – number of tokens, axe X – firing times).

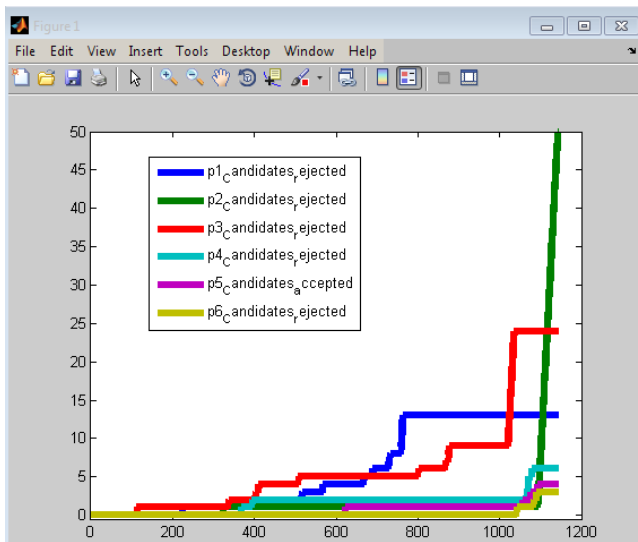


Figure 49. Simulation results for simple BPMN model

Place “p1_Candidates_rejected” has 13 tokens in the end; “p2_Candidates_rejected” – 50; “p3_Candidates_rejected” – 24; “p4_Candidates_rejected” – 6; “p5_Candidates_accepted” – 4; “p6_Candidates_rejected” - 3. These results can be explained by location of gateways, “Continue with candidates” is the first gateway, “Offer is accepted by candidate” – the last in the chain, that’s why outcomes of it have the lowest probability.

The presented BPMN model does not have any additional information, recommended in chapter 3 (probabilities, time delays, resources). That’s why problem with simulation for this model becomes trivial. In chapter 3.2.4 four ways of simulation in Petri net were studied (probability, time delay, resource sharing, color).

First is probability. As it was described earlier, BPMN model can be updated with probability information for gateways. Updated BPMN model (with probability information) is shown in Appendix A. Every sequence flow from gateway in the model has its signature with probability. After java program runs, each gateway is mapped to 2 transitions. Every transition is assigned a probability. In overall we get 14 transitions out of gateways in the examined BPMN model:

Continue_with_candidates_no,	p=0,7
Continue_with_candidates_yes,	p=0,3
Testing_is_needed_yes,	p=0,5
Testing_is_needed_no,	p=0,5
Approve_candidates_for_interview_no,	p=0,5
Approve_candidates_for_interview_yes,	p=0,5
Need_for_new_interview_yes,	p=0,1
Need_for_new_interview_no,	p=0,9
Candidates_approved_yes,	p=0,5
Candidates_approved_no,	p=0,5
Background_check_performed_successfully_no,	p=0,99

Background check performed successfully yes,	p=0,01
Offer is accepted by candidate yes,	p=0,95
Offer is accepted by candidate no,	p=0,05

These values can be taken from business statistics, for example if company knows how likely or unlikely is that background check of candidate is ok, etc. There are also some subjective gateways, which depends on company's decisions, for example, company may make requirements less strict and take 70% of candidates to interview instead of 50%. In this case of subjective probability, it is possible to experiment by inserting different values in transitions and see how the results would change.

After simulation with the probability values shown in Appendix A, following results were gotten (number of initial tokens was again 100).

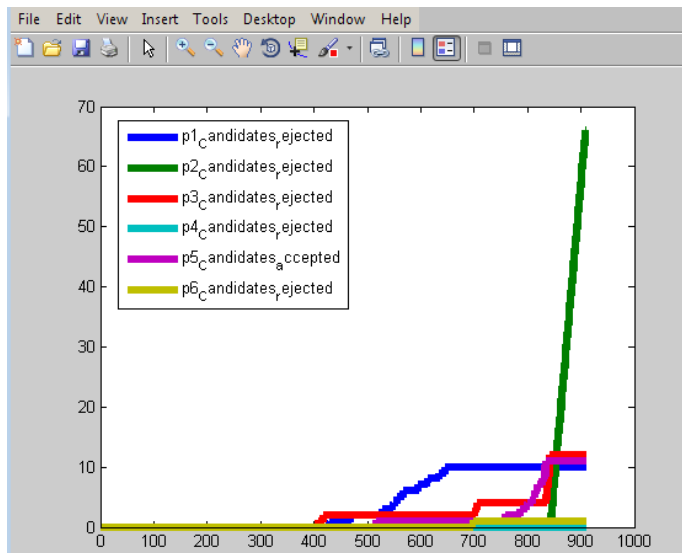


Figure 50. Simulation results with probabilities (a)

Place “p2_Candidates_rejected” has 66 tokens in the end, place “p5_Candidates_accepted” – 11. Result values allow us to judge about probability that each new candidate will end up in this or that end event. As a sample, business may need to solve following issues by mean of simulation:

Problem 1. If gateways have the probabilities, defined earlier, then how likely is that candidate is accepted and that candidate is rejected after interview stage?

For making a relevant simulation we can insert a relatively big number of tokens into initial place (for example, 1000). Then the result of simulation is

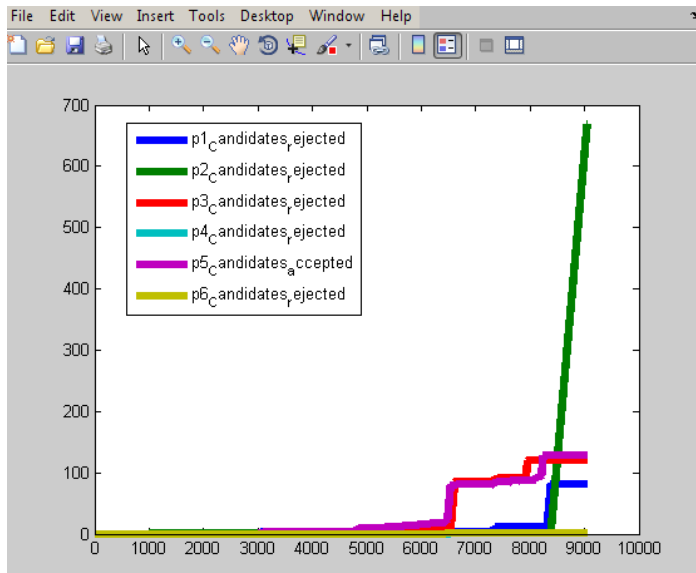


Figure 51. Simulation results with probabilities (b)

End places have following number of tokens

p1_Candidates_rejected	81
p2_Candidates_rejected	669
p3_Candidates_rejected	120
p4_Candidates_rejected	1
p5_Candidates_accepted	127
p6_Candidates_rejected	2

So the probability that candidate is accepted $\approx 0,127$, the probability that candidate is rejected after interview stage (place “p3_Candidates_rejected”) $\approx 0,120$. The same procedures can be applied for all places in Petri net, for example, counting the probability that it is needed to give feedback to candidate after testing stage (place “p11”), etc.

The result can be interpreted in another way. If there are 100 CVs, then around 12 candidates will be accepted.

Problem 2. How the probability that candidate is accepted will change, if we make requirements more strict on the interview and approve 20% of candidates instead of 50% how we do now.

For this simulation we will change the probability for transitions corresponding to gateway “Candidates approved”. Transition “Candidates_approved_yes” will have now 0,2 probability and “Candidates_approved_no” – 0,8. This change can be done either directly in BPMN model or in Java program, which allows us to change probabilities dynamically.

Number of initial tokens is 500. Results of simulation shown below.

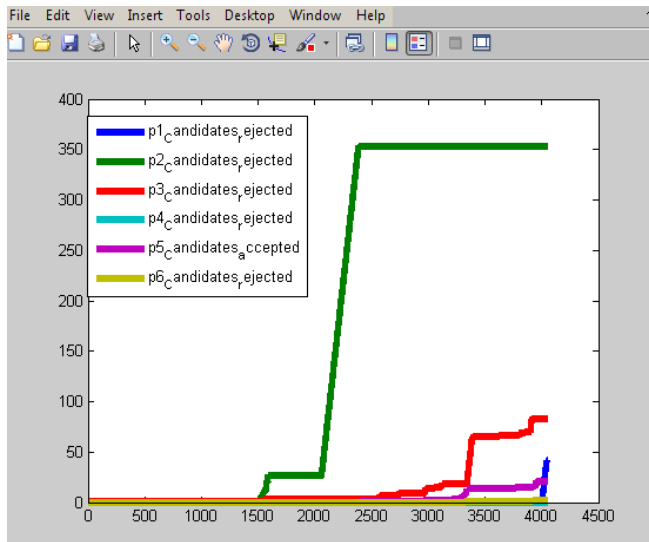


Figure 52. Simulation results with probabilities(c)

Now place “p5_Candidates_accepted” has only 21 token out of 500, this is $\approx 0,042$ probability. So if a company is going to get 100 CVs, only around 4 people would be accepted (this is 3 times less than it was before).

Next way of using Petri net in business is time delay computing. We are going to combine time delays with probabilities (to get the more real picture). Time delays were defined for activities in the original BPMN model (Appendix B). For example, activity “verify ID of candidate” takes 5 hours, etc.

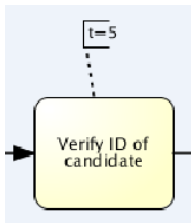


Figure 53. Activity with time delay information

After transforming this new BPMN model into Petri net, we get new values in array “dynamicpart.firing_times”, instead of all 0s now it has different time points.


```
dynamicpart.firing_times={'CVs_sent_after_advertisement',0.0,'Decide_screening_criteria',0.0,'Screen_candidates',0.5,'Select_candidates',0.1,'Decide_if_testing_is_needed',0.3,'Prepare_plan_for_interview',1.0,'Evaluate_test_results',0.5,'Complete_test',1.0,'Send_test_to_candidates',0.5,'Define_testing_tools',0.0,'Give_feedback_to_candidates',0.3,'ee_1_Candidates_rejected',0.0,'Evaluate_candidate',2.0,'Perform_interview',1.5,'Verify_ID_of_candidate',5.0,'Perform_interview_with_references',3.0,'Create_offer_of_employment',1.0,'Send_offer_to_candidate',0.2,'Continue_with_candidates_no',0.0,'Continue_with_candidates_yes',0.0,'ee_2_Candidates_rejected',0.0,'Testing_is_needed_no',0.0,'Testing_is_needed_yes',0.0,'Approve_candidates_for_interview_yes',0.0,'Approve_candidates_for_interview_no',0.0,'Need_for_new_interview_yes',0.0,'Need_for_new_interview_no',0.0,'Candidates_approved_yes',0.0,'Candidates_approved_no',0.0,'ee_3_Candidates_rejected',0.0,'Background_check_performed_successfully_yes',0.0,'Background_check_performed_successfully_no',0.0,'ee_4_Candidates_rejected',0.0,'Offer_is_accepted_by_candidate_yes',0.0,'Offer_is_accepted_by_candidate_no',0.0,'ee_5_Candidates_accepted',0.0,'ee_6_Candidates_rejected',0.0};
```

Figure 54. Main simulation file, firing times for transitions

Below are sample problems, which can be relevant for business and which can be solved by means of simulation.

Problem 1. How much time will the whole process take in cases when candidate is accepted. For this purpose we will input only 1 token in initial place and make relatively big number of simulations (in this example, it was 200). In each case, when place “p5_Candidates_accepted” is filled with a token, we will show the current time. It is done by adding one line of code in the relevant transition definition file:

```
disp(PN.current_time);
```

So when this transition fires, it will print out the current time. Results are below

```
16.6000
14.6000
16.6000
16.6000
14.6000
16.6000
14.6000
14.6000
19.1000
14.6000
14.6000
16.6000
16.6000
14.6000
16.6000
14.6000
14.6000
14.6000
```

The average is $\approx 15,62$.

Problem 2. Company decided to use testing less and instead put more focus on interview, the interview in this case should become more time-consuming. For example, testing should be

applied only in 20% of all cases, but time of “Prepare plane for interview” should increase up to 1,5 hours and “Perform interview” to 2 hours. Following information should be updated in BPMN diagram to reflect these changes:

1. Probability that testing is needed should be set to 0,2.
2. Time delays for two activities (Prepare plan for interview, Perform interview) should be changed to 1,5 and 2 correspondingly.

Changes in these places will affect only that part of the model which stands before gateway “Candidates approved”. So to make conclusion about time it is sufficient to trace time at transition “Candidates_approved_no”. First, we will test the model without changes. Firing times for this transition after 100 testings were.

5.4000
9.9000
7.4000
7.4000
5.4000
5.4000
7.4000
5.4000
5.4000
5.4000
5.4000
5.4000

Average time is 6,3545.

Then the same testings (100) will be performed for a changed BPMN model (updated probabilities and time delays for some transitions). Firing times of transition “Candidates_approved_no” for a changed BPMN diagram are the following.

11.9000
6.4000
6.4000
6.4000
11.9000
6.4000
13.9000
6.4000
6.4000
6.4000

6.4000

Average time is 8,0818.

The conclusion is that if the changes are implemented in the company, then the process will take more time than it takes now.

Problem 3. CVs are sent to company randomly (time between 2 CVs is a random variable from exponential distribution with parameter $\mu=15$ hours). Company may be interested in 2 questions:

1. How many CVs the company may need to hire 10 new employees.
2. How long will it take for the company until it hires 10 new employees.

First, to reflect the problem we will add annotation to start event to show that it can fire in random time units.

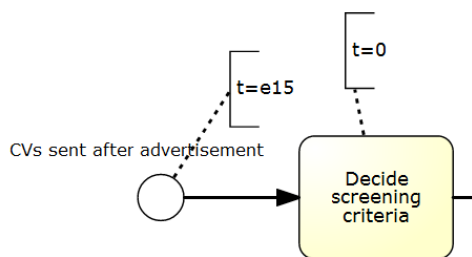


Figure 55. Start event with exponential distribution

Then the java program was executed and it transformed the BPMN into Petri net and set firing time for transition as random variable.

```
dynamicpart.firing_times='CVs sent after advertisement', 'exprnd(15.0)';  
sim = gpsim(png, dynamicpart, global_info);
```

Figure 56. Main simulation file with random firing time

Then we will make test with 50 CVs (i.e. 50 tokens in initial place). The result is below.

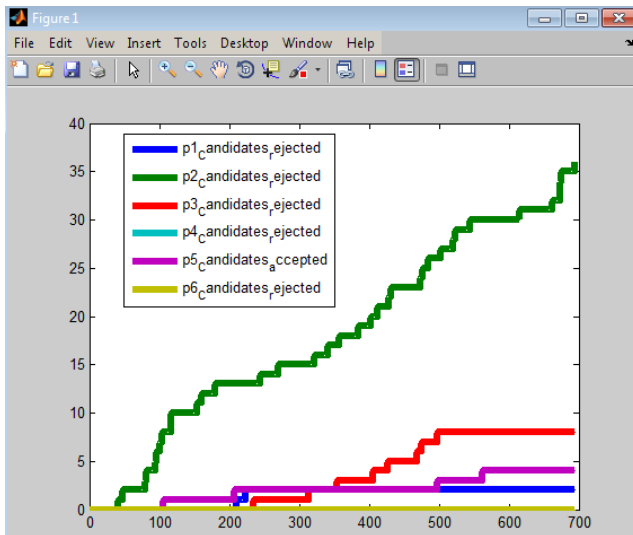


Figure 57. Problem 3 simulation results (a)

The number of tokens in “p4_Candidates_accepted” place is 4. It is not enough. Second testing will be done with 100 tokens in initial place.

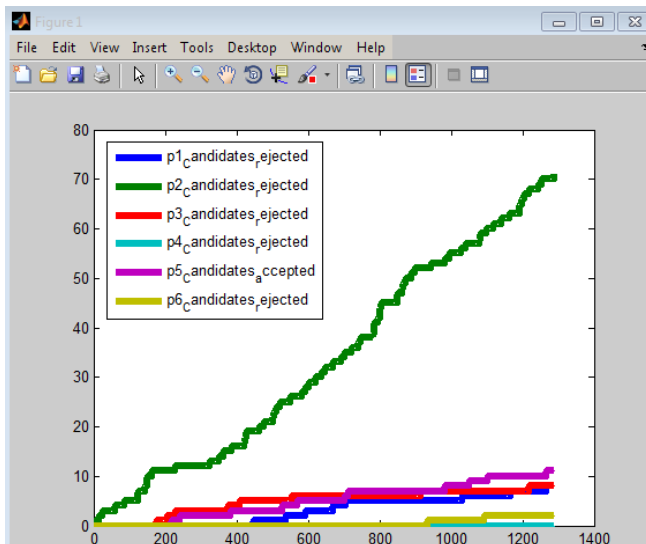


Figure 58. Problem 3 simulation results (b)

Now there are 11 accepted candidates, which is ok. All 100 CVs were handled at time unit 1287. So the initial conclusion could be that it is required to handle 100 CVs before company hire 10 employees and that all this process will take up to 1287 hours. By making more simulations we will get better statistical approximation.

4.4 Evaluation

As the proposed approach BPMN-to-PetriNet transformation is designed mainly for business goals, we would like to present a short SWOT analysis for it:

1. **Strengths.**

- Easy way to make transformation from BPMN to Petri net (all we need is a BPMN tool which exports diagram into xml file).
 - Powerful simulation abilities, which appear out of static pictures.
 - Petri net simulation brings such benefits to business, as performance analysis.
2. **Weaknesses.**
- Simulation with Petri net requires some knowledge in Petri net technology and simulation tool used for it.
 - GPenSIM does not allow to see the graphical picture of Petri net diagram.
3. **Opportunities.**
- Further development and popularization of dynamic Enterprise Architecture concept
4. **Threats.**
- Development of another approach which is simpler and user-friendly

5 Conclusion and future work

The goal of the paper was to suggest our approach of transforming dynamic models of Enterprise Architecture into dynamic ones. As a static technology of modeling, BPMN was selected, because it is one of the most used tools in companies. Petri net was chosen for dynamic model, because it suggests good simulation capabilities combined with mathematical functions in MATLAB.

In chapter 3 we developed and theoretically described our own approach of transformation from BPMN to Petri net. Some parts of this approach were based on few existing researches in this area and others were developed completely by us. The resulting table of mapping rules between BPMN and Petri net is presented in chapter 3.2.2 (Table 2 and Table 3) and in chapter 3.4 (Table 5). Also we studied the ways how Petri net model of business process can give useful simulation for the company. These ways are summarized in Table 4.

In section 4 of the work we made a partial testing of our approach. For this purpose we created a Java program, that transforms BPMN model (with certain set of elements) into Petri net. After testing of the program we proved that BPMN can be mapped to the equal Petri net, i.e. Petri net, which completely corresponds to the properties of the original BPMN diagram. This allows us to accept hypothesis 1, proposed in chapter 1.3, as true.

Second, we made simulation with extracted Petri net model in GPenSIM. The results of this simulation were discussed in chapter 4.3. It was shown that based on simulation, made with final Petri net model, company can extract useful information for it and accept important decisions. Based on these results, we believe that hypothesis 2 is also true.

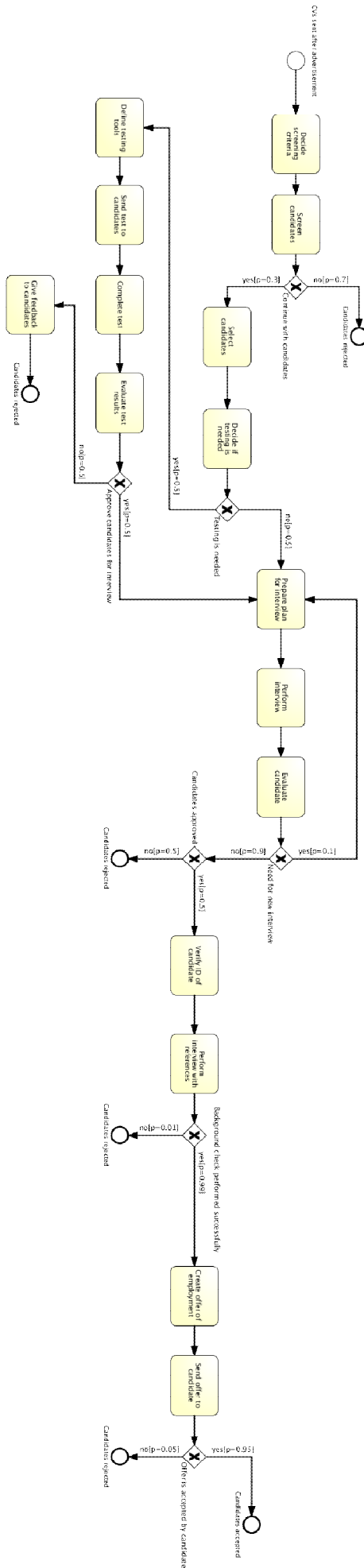
While writing the paper we faced following challenges:

1. Mapping rules between BPMN and Petri net become a complicated problem, when all elements are integrated together into one Petri net model. In some existing works [18], [19] it was described how to map a single BPMN element into corresponding set of Petri net elements, but these works do not focus on the connecting everything together into one model.
2. BPMN does not suggest any functionality to represent time consumption, probabilities or resource usage in diagram. At the same time these features are widely used in Petri net and make Petri net simulation more powerful. It was challenge for us to find a way, how BPMN can be changed or enhanced in order to reflect these features.

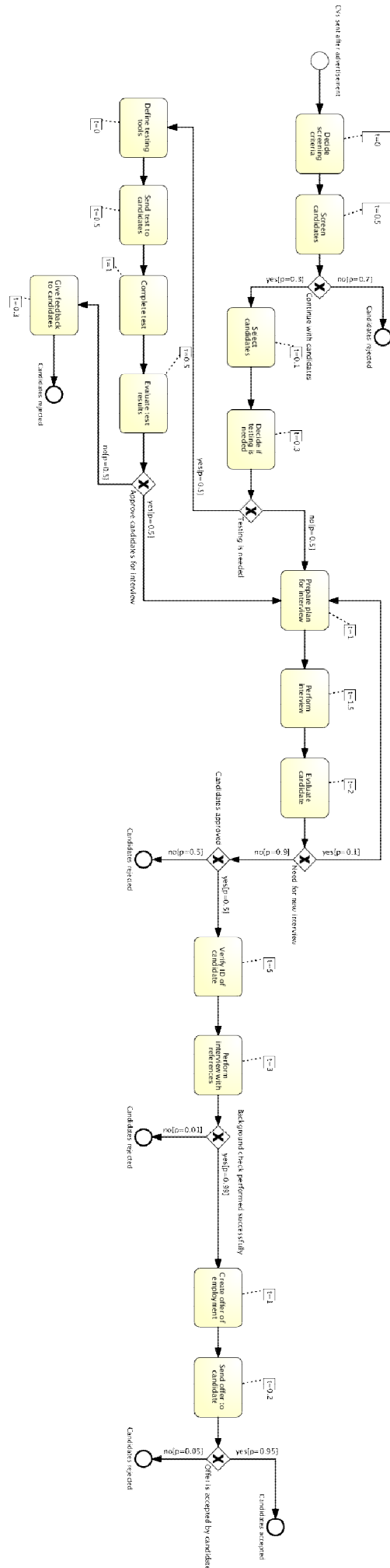
We can conclude that the goal of the paper was reached, but there are several considerations for future works:

1. The program of transforming BPMN to Petri net can be developed further. It can get the ability to handle the remaining elements from BPMN standard (all types of gateways, events, etc.). Now it handles only exclusive gateways, tasks and events.
2. In the paper we showed how to extend BPMN diagram so that it will enable some additional Petri net features (probabilities, time delays, resource sharing). Probabilities and time delays were realized and presented in chapter 4. Resource sharing can be a challenge for further works. In chapter 3 we suggested the way and described the steps how resource sharing can be realized. The challenge for future works is to implement proposed steps.
3. As a modeling tool for Petri net, GPenSIM was chosen. It is good in simulation and suggest powerful mathematical tools, as it is integrated in MATLAB. But it does not suggest any graphical tool, so user can not see the picture with Petri net model itself. To make the Petri net simulation more user-friendly, it is recommended to develop a mapping tool between GPenSIM and PNML (Petri net Markup Language). The last is an xml-based language which is easily converted to graphical picture in special Petri net software like PIPE or others.

Appendix A. BPMN model updated with probability information



Appendix B. BPMN model updated with time delays information



Appendix C. How to use the program: prerequisites

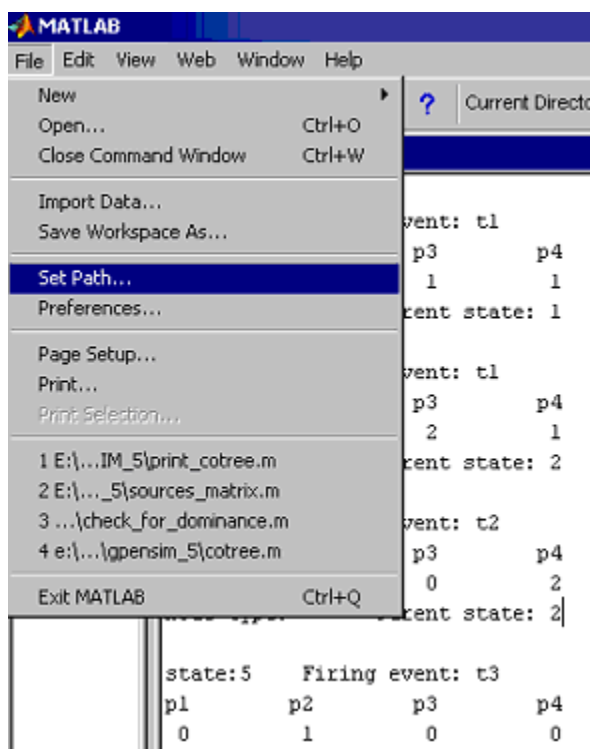
In order to test the program following tools are needed to be installed:

1. Java Runtime Environment.
2. MATLAB.
3. GPenSIM.

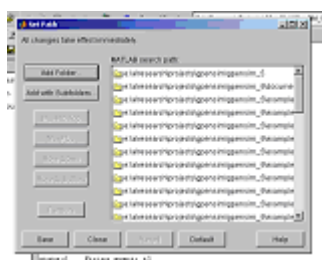
Appendix D. GPenSIM: installation guide

1. Source files of GPenSIM (M-files in ZIP pack) can be downloaded from <http://davidrajuh.net/gpensim>
2. **Unzip the GPenSIM pack.** Unzip the GPenSIM toolbox functions under a directory, say “d:\GPenSIM_40”
3. **Path Command.**

Go to the file menu and select “set path” command:



4. **Select “Add with folders”:**

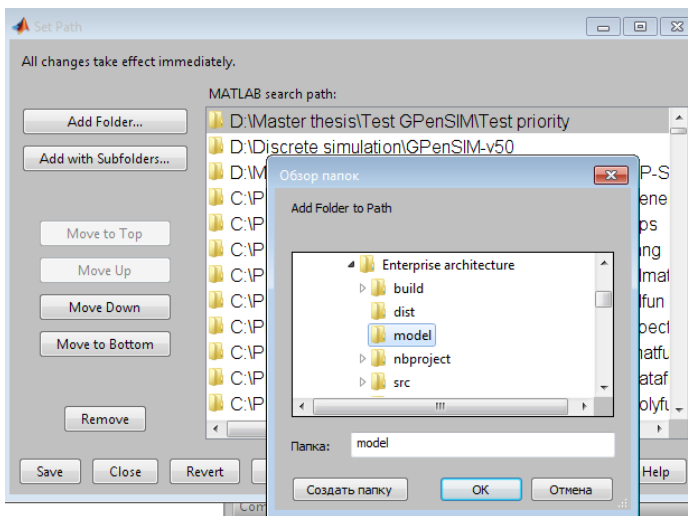


5. **Add GPenSIM Directory**

A new dialog box will appear. Browse through the directories and select the directory where you have unzipped the GPenSIM toolbox functions.



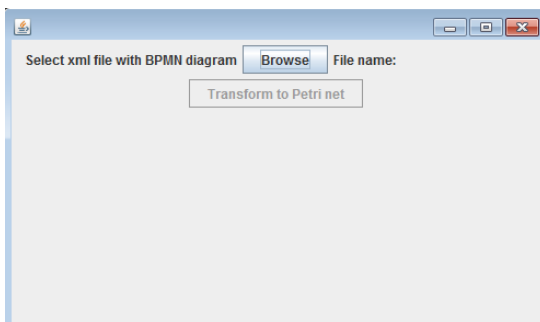
6. Go again to **Set path** in file menu.
7. Select **Add folder**.
8. Add model directory (in root folder there is a “model” directory).



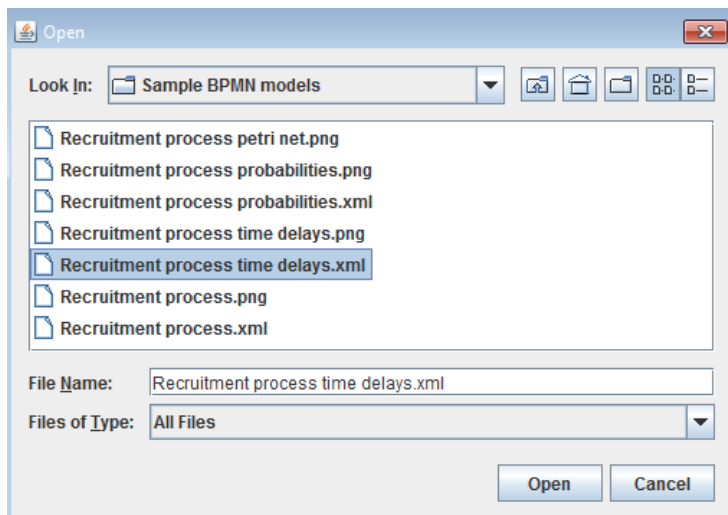
9. Press **Save**.

Appendix E. How to use the program: steps

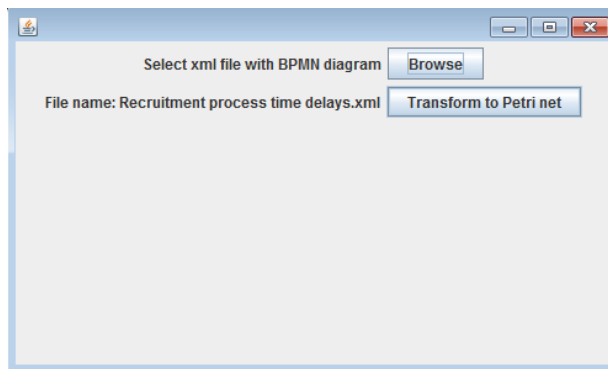
1. Run the jar file “Enterprise architecture.jar”, located in the root directory.
2. Window of selecting an xml file with BPMN model will appear.



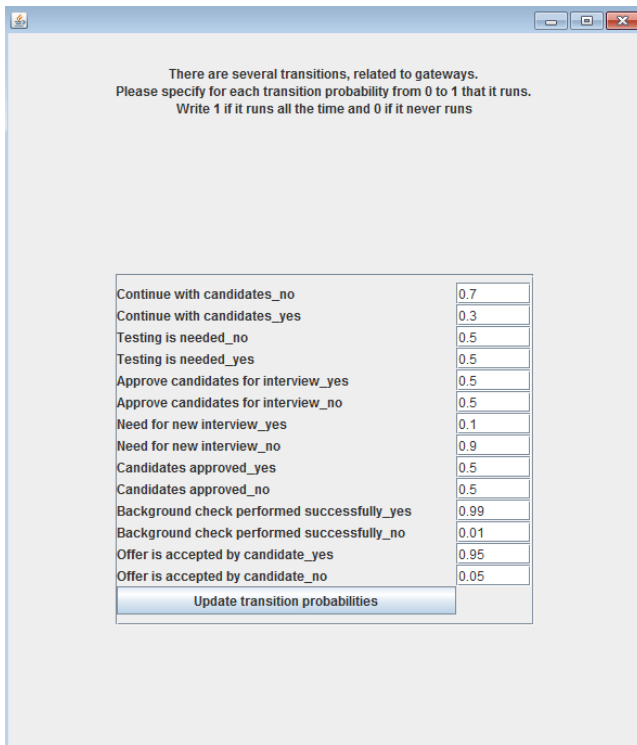
1. Press **Browse**. Window for selecting file will appear. Select one of xml files, located in directory “Sample BPMN models”, for example “Recruitment process time delays.xml”.



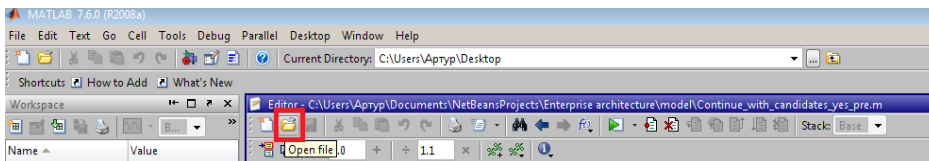
2. Click **Transform to Petri net**



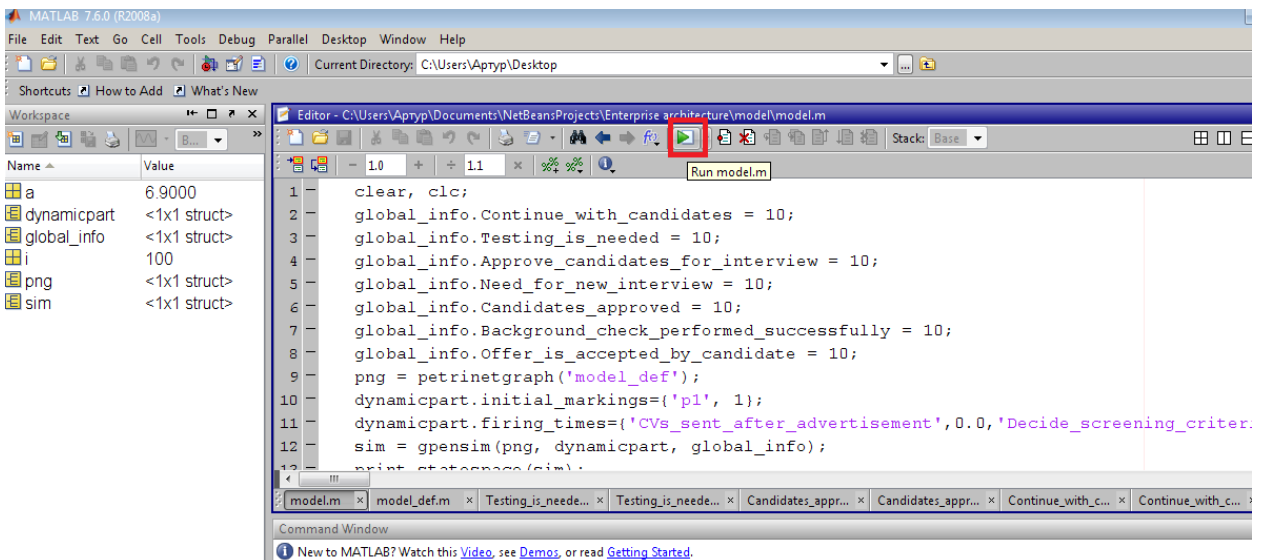
3. A new window will appear with possibility to write probabilities for gateway transitions (Each exclusive gateway is transformed into 2 transitions, every transition should be assigned a probability, for example, if we have gateway “Agree” with output sequence flows “Yes” and “No”, then probabilities should be assigned to 2 transitions – “Agree_yes”, “Agree_no”).



4. After probabilities are assigned, press **Update transition probabilities**.
5. New GPenSIM files are created in folder “model”.
6. Now open MATLAB with GPenSIM and open all files from the folder “model”.



7. Go to the file “model.m” and press **Run** button.



8. Results will appear. It is also possible to change the code in “model.m” or in transition definition files to change the firing times, the probabilities, the output, etc.

References

- [1] C. Vibert. "Theories of Macro-Organizational Behavior : A Handbook of Ideas and Explanations", February, 2010
- [2] R. Garud, A.H. Van de Ven. "Strategic Change Processes." In Handbook of Strategy and Management, 2001
- [3] A. Sullivan, Steven M. Sheffrin. "Economics: Principles in action", 2003
- [4] "A practical guide to Federal enterprise architecture". <http://www.gao.gov/special.pubs/eaguide.pdf>
- [5] T. Graves. "Enterprise Architecture : A Pocket Guide", 2009
- [6] "Business architecture overview". http://bawg.omg.org/business_architecture_overview.htm
- [7] D. Ensor. "Developing systems to improve business". <http://www.davidensor.com/2009/11/developing-systems-to-improve-business/>
- [8] "Technical architecture". <http://netmation.com/exp0004.htm>
- [9] Ronald E. Giachetti "Design of Enterprise Systems: Theory, Architecture, and Methods", 2010
- [10] J. Ross, P. Weill and D. Robertson. "Enterprise Architecture as Strategy", 2006
- [11] "Board briefing on IT Governance". http://www.isaca.org/Knowledge-Center/Research/Documents/BoardBriefing/26904_Board_Briefing_final.pdf
- [12] Lankhorst "Enterprise architecture at work: Modelling, communication and analysis", 2009
- [13] Vitus S.W. Lam. "Formal analysis of BPMN models: A NuSMV-based approach", *International Journal of Software Engineering*, pp. 987-1023, Vol. 20, No. 7, 2010
- [14] F. Vernadat "Enterprise modeling and integration: principles and applications", 1996
- [15] Schekkerman "A comparative survey of enterprise architecture frameworks". http://www.google.ru/url?sa=t&rct=j&q=&esrc=s&source=web&cd=7&ved=0CHwQFjAG&url=http%3A%2F%2Fwww.enterprise-architecture.info%2FImages%2FDocuments%2FComparative_Survey_of_EA_Frameworks.pps&ei=J358T7r8L87E4gSatNTmDA&usq=AFQjCNHwfMPKPGj3GHeHK5Wqao8xwe3Pgg&sig2=kM9GNUktyhcc5PKG0bFWw
- [16] Chinosi "BPMN an introduction to the standard", *Computer Standards & Interfaces*, pp. 122-134, Volume 34, Issue 1, 2012
- [17] B. Hruz, M. Zhou "Modeling and control of discrete-event dynamic systems: with Petri Nets and Other Tools", 2007
- [18] R.M. Dijkman, M. Dumas, C. Ouyang. "Semantics and analysis of business process models in BPMN", *Information and Software Technology* 50, pp.1281-1294, 2008
- [19] R. Koniewski, A. Dzielinski, K. Amborski. "Use of Petri Nets and Business Processes Management Notation in Modelling and Simulation of Multimodal Logistics Chains". <http://www.scs-europe.net/services/ecms2006/ecms2006%20pdf/84-cs.pdf>
- [20] R. Hassan, M. Ramadan. "BPMN Formalisation using Coloured Petri Nets", in *Proceedings of the 2nd GSTF Annual International Conference on Software Engineering & Applications (SEA 2011)*, December, 2011
- [21] M. V. Fuente, L. Ros, A. Ortiz. "Enterprise modelling methodology for forward and reverse supply chain flows integration", *Computers in Industry*, 61, pp. 702-710, 2010

- [22] J. Ryan, C. Heavey. "Process modeling for simulation", *Computers in Industry*, 57, pp. 437-450, 2006
- [23] M.G. Mykityshyn. "Supporting strategic enterprise processes: An analysis of various architectural frameworks", *Information Knowledge Systems Management*, Vol. 6 Issue 1/2, p145-175, 2007
- [24] J.P. Zachman "The Zachman Framework Evolution". <http://www.zachman.com/ea-articles-reference/54-the-zachman-framework-evolution>
- [25] Flowchart shapes. <http://www.quality-assurance-solutions.com/flowchart-shapes.html>
- [26] "Introduction to data flow diagrams". <http://visualcase.com/tutorials/about-data-flow-diagram.htm>
- [27] "UML infrastructure". <http://www.omg.org/spec/UML/2.0/Infrastructure/PDF/>
- [28] "Activity diagrams". <http://www.uml-diagrams.org/activity-diagrams.html>
- [29] M. Chitnis "Activity Diagram in UML". <http://www.developer.com/design/article.php/2247041/Activity-Diagram-in-UML.htm>
- [30] "Overview of IDEF0". <http://www.idef.com/IDEF0.htm>
- [31] "Overview of IDEF3". <http://www.idef.com/IDEF3.htm>
- [32] "Business process model and notation". <http://www.omg.org/spec/BPMN/2.0/PDF/>
- [33] O.S. Noran. "Business Modelling: UML vs. IDEF", 2000
- [34] D.C.C. Peixoto. "A Comparison of BPMN and UML 2.0 Activity Diagrams". http://homepages.dcc.ufmg.br/~cascini/SBQS_2008.pdf
- [35] L. Eloranta, E. Kallio, I. Terho. "A Notation Evaluation of BPMN and UML Activity Diagrams", 2006
- [36] L. Salum "Petri nets and time modelling", *International Journal of Advanced Manufacturing Technology*, Vol. 38 Issue 3/4, p377-382, 2008
- [37] L.P. Magalhaes. "Animation modeling with Petri nets", *Computers & Graphics*, Vol. 22, issue 6, pp. 735-743, 1998
- [38] R. David, A. Hassane. "Discrete, Continuous and Hybrid Petri Nets", 2005
- [39] F.D.J. Bowden. "A Brief Survey and Synthesis of the Roles of Time in Petri Nets", *Mathematical and Computer Modelling*, 31, pp. 55-68, 2000
- [40] S. Ha. "A timed colored Petri nets modeling for dynamic workflow in product development process", *Computers in Industry*, Volume 59, Issues 2-3, pp. 193-209, 2008
- [41] V.C. Gerogiannis. "Comparative study and categorization of high-level petri nets", *Journal of Systems and Software*, Volume 43, Issue 2, pp. 133-160, 1998
- [42] T. Bray. "Extensible Markup Language (XML) 1.0 (Fifth Edition)". <http://www.w3.org/TR/REC-xml/#sec-origin-goals>, 2008
- [43] M. Vijai, R. Mittal. "Algorithm-based fault tolerance: a review", *Microprocessors and Microsystems*, Volume 21, Issue 3, pp. 151-161, 1997
- [44] S. Haddad. "Product-form and stochastic Petri nets: a structural approach", *Performance Evaluation*, Volume 59, Issue 4, pp. 313-336, 2005