ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Grigorian** Jméno: **Bogdan** Osobní číslo: **452746**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Sémantický nástroj pro BPMN modely podporující STPA analýzy

Název diplomové práce anglicky:

Ontology-based BPMN tool to support STPA analysis

Pokyny pro vypracování:

BPMN is a language to express processes within an organization, while STAMP is an accident causation model based on systems theory. The goal of the thesis is to create an extension of a selected software that would enable managing BPMN diagrams extended with STAMP concepts. Such a combination of models would be possible to use for STPA analysis inferred from processes of an organization. The developed software will be tested on existing BPMN diagrams developed within the Civil Aviation Authority in Czech Republic in cooperation with domain experts. The software will use existing STAMP ontology developed at KBSS group [1].
Instructions:
1) get familiar with Semantic Web technologies RDF, OWL, SPARQL and Unified Foundation Ontology [2]
2) review existing BPMN ontologies [4]
3) design BPMN model and integrate it with provided STAMP model in cooperation with domain experts
4) specify requirements and scenarios to work with the extension
5) design and implement the extension
6) test the extension including user tests on selected scenarios with at least 3 participants

Seznam doporučené literatury:

[1] Kostov, et al., STAMP ontology, available at http://onto.fel.cvut.cz/ontologies/stamp
[2] Guizzardi, Giancarlo, et al. "Towards ontological foundations for the conceptual modeling of events." International Conference on Conceptual Modeling. Springer, Berlin, Heidelberg, 2013.
[3] Suchánek, Marek, and Robert Pergl. "Mapping UFO-B to BPMN, BORM, and UML Activity Diagram." Workshop on Enterprise and Organizational Modeling and Simulation. Springer, Cham, 2019.
[4] Dijkman, Remco, Jorg Hofstetter, and Jana Koehler, eds. Business Process Model and Notation. Vol. 89. Springer, 2011.
[5] Aagesen, Gustav & Krogstie, John. (2015). BPMN 2.0 for Modeling Business Processes. Handbook on Business Process Management 1: Introduction, Methods, and Information Systems. 219-250. 10.1007/978-3-642-45100-3_10.
[6] Leveson, N. G., and J. P. Thomas. "STPA handbook. 2018." URL https://psas. scripts. mit. edu/home/get_ file. php (2019).

Jméno a pracoviště vedoucí(ho) diplomové práce:

Mgr. Miroslav Blaško, Ph.D., skupina znalostních softwarových systémů

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **27.07.2021** Termín odevzdání diplomové práce: **04.01.2022**

Platnost zadání diplomové práce: **19.02.2023**

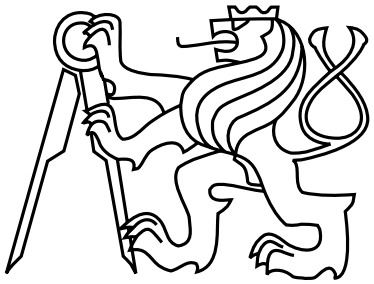
Mgr. Miroslav Blaško, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studenta

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering



Master's Thesis

Ontology-based BPMN tool to support STPA analysis

Bc. Bogdan Grigorian

Supervisor: Mgr. Miroslav Blaško, Ph.D.

Study Programme: Open Informatics, Master

Field of Study: Software Engineering

January 4, 2022

Declaration

I declare that I elaborated this thesis on my own and that I mentioned all the information sources and literature that have been used in accordance with the Guideline for adhering to ethical principles in the course of elaborating an academic final thesis.

Prague, date ....

Abstract

In today's complex systems such as aviation, errors like the wrong approach, human error, or lack of comprehensive understanding can result in significant economic losses. Typically organizations have their processes expressed in some way, often using the BPMN standard. The goal of this thesis was to create an extension to the selected software that enables the managing BPMN diagrams extended with STAMP concepts. This combination of models can be used to analyze STPA inferred from an organization's processes.

Keywords: RDF, SPARQL, JOPA, Ontology, STAMP, STPA, BPMN, Semantic Web

Abstrakt

V dnešních složitých systémech může dojít k chybám, jako je nesprávný přístup, lidská chyba nebo nedostatečné porozumění, což může mít za následek značné ekonomické ztráty. Obvykle mají organizace své procesy nějakým způsobem vyjádřeny, často pomocí standardu BPMN. Cílem této práce bylo vytvořit rozšíření vybraného softwaru, které umožní správu diagramů BPMN rozšířených o koncepty STAMP. Tuto kombinaci modelů lze použít k analýze STPA odvozených z procesů organizace.

Klíčová slova: RDF, SPARQL, JOPA, Ontologie, STAMP, STPA, BPMN, Sémantický web

Contents

1 Introduction **1**

2 Background **3**
2.1 Ontology . . . 3
2.1.1 Upper ontology . . . 4
2.1.2 Unified Foundational Ontology (UFO) . . . 4
2.2 Semantic web . . . 4
2.2.1 Resource Description Framework (RDF) . . . 5
2.2.2 Web Ontology Language (OWL) . . . 5
2.2.3 SPARQL . . . 6
2.3 STAMP and STPA . . . 6
2.3.1 STAMP ontology . . . 6

3 Analysis **9**
3.1 Business Process Modeling Notation . . . 9
3.1.1 BPMN 2.0 . . . 10
3.1.2 BPMN conformance and diagram types . . . 10
3.1.3 Basic BPMN elements . . . 11
3.1.4 BPMN extensions . . . 13
3.1.5 Diagram interchangeability . . . 14
3.1.6 Example SAG meeting . . . 14
3.2 Bonita Studio . . . 15
3.2.1 BPMN diagram editor . . . 16
3.2.2 Organization structure editor editor . . . 16
3.2.3 Actor mapping configuration . . . 17
3.2.4 Example SAG meeting . . . 17
3.2.4.1 CAA organization structure . . . 18
3.2.4.2 Actor mapping in SAG meeting . . . 19
3.3 Comparison of existing BPMN ontologies . . . 20
3.3.1 Natschläger's BPMN 2.0 ontology . . . 22
3.3.2 BPMN 2.0 compatible upper ontology by Penicina . . . 22
3.3.3 BPMN Based Ontology (BBO) . . . 23
3.3.4 Summary . . . 25
3.4 BBO and UFO compatibility . . . 25
3.5 Analysis of the existing converter . . . 26

4 Design **29**
4.1 Requirement analysis . . . 29
4.1.1 MoSCoW prioritization . . . 29
4.1.2 Client application . . . 30
4.1.2.1 Functional requirements . . . 30
4.1.2.2 Non-functional requirements . . . 31
4.1.3 BPMN to STAMP converter library . . . 31
4.1.3.1 Functional requirements . . . 31
4.1.3.2 Non-functional requirements . . . 32
4.2 Extending BBO ontology . . . 33
4.3 Conversion stages . . . 34
4.3.1 BPMN to BBO mapping . . . 35
4.3.1.1 Organization structure to BBO mapping . . . 35
4.3.2 BBO to STAMP mapping . . . 37
4.3.3 Enriching basic control structure . . . 38
4.3.4 Import conventions . . . 38

5 Implementation **41**
5.1 Technology stack . . . 41
5.1.1 Mapstruct . . . 41
5.1.2 JOPA . . . 42
5.1.3 Apache Commons CLI . . . 42
5.2 Problems and solutions . . . 43

6 Testing **45**
6.1 Testing results . . . 45

7 Conclusion **47**

A Organization structure model in Bontia Studio (Complete) **51**

B Contents of the enclosed CD **53**

List of Figures

2.1 Main concepts in UFO. From [13] 5

3.1 Pool in Bonita Studio 11
3.2 Pool with two Lanes in Bonita Studio 11
3.3 Example of the process with two User Tasks in Bonita Studio 12
3.4 Diverging XOR Gateway with conditional and default Sequence Flows in Bonita Studio 13
3.5 SAG meeting BPMN diagram in Bonita Studio 15
3.6 Validation window in Bonita Studio 16
3.7 Organization structure model in Bonita Studio (compact) 18
3.8 Actor mapping using Membership in Bonita Studio (from XML files) 19
3.9 Organization structure (roles and groups) of the CAA, defined in Bonita Studio 20
3.10 Groups in organization structure without assigned Roles 21
3.11 Actor mapping of the SAG meeting process, defined in Bonita Studio 22
3.12 Groups in organization structure with assigned Roles 23
3.13 BBO concepts related to a BPMN process. Image taken from [22] 24
3.14 BBO concepts related to an Agent. Image taken from [22] 25

4.1 The sequence of stages in the BPMN to STAMP conversion process 34
4.2 Ontology import convention 39

5.1 Dependency usage and module structure of the BPMN to STAMP converter library 42

A.1 Organization structure model in Bontia Studio (Complete) 52

List of Tables

3.1 STAMP to UFO class and relation mapping 26

4.1 BPMN to BBO class mapping 36
4.2 BPMN to BBO relation mapping 36
4.3 Org. structure to BBO class mapping 36
4.4 Org. structure to BBO relation mapping 36
4.5 BBO to STAMP class mapping 37
4.6 BBO to STAMP relation mapping 37

Chapter 1

Introduction

Nowadays, in complex systems such as aviation, errors, such as wrong approach, human errors or lack of comprehensive understanding can lead to significant economic losses. This makes safety one of the most important aspects, therefore a traditional approach of hazard analysis is being challenged by the new analysis methods.[27]

Systems-Theoretic Accident Model and Processes (STAMP) is an accident causation model, based on systems theory and systems thinking. STAMP by the meaning is not a method of analysis, it is a set of assumptions about how accidents happen. Examples of such factors are software, human decision-making, human factors, as well as new unproven technologies.[11][10]

STPA is a STAMP-based hazard analysis method. STPA is an abbreviation of a System-Theoretic Process Analysis. The scope of this method includes a variety of complex systems, including aviation. In contrast with the traditional approach of hazard analysis, STPA can be started even in the early concept analysis.[11][10]

Today, most organizations use the BPMN standard to describe their processes and models. These process descriptions can be used as a basis for defining the control structure of the organization, which is a foundation of the STPA analysis.

The goal of the thesis is to create an extension of a Bonita Studio software, which will enable the managing of BPMN diagrams, extended by STAMP concepts. The first part 3 of the work is devoted to the analysis of the mentioned BPM tool, its capabilities to express the organization structure along with the processes within this organization. The analysis is done in cooperation with domain experts from the Civil Aviation Authority of the Czech Republic, as well as ontology engineers from the KBSS group. Chapter 4 describes the process of integrating the BPMN model into the STAMP analysis using the STAMP-based ontology created by the KBSS group. The proposed method is covered by the software designed and developed as a result of this thesis. Chapter 5 describes the process stack used in the proposed software. In addition, several problems encountered during the implementation phase are discussed. The last chapter 6 is dedicated to the testing phase, where the methodology and test results are summarized.

Chapter 2

Background

2.1 Ontology

Ontology is a formal representation of some knowledge, a group of concepts, and relationships between them. Ontology has many definitions in various domains. In the software engineering area ontologies are very close to conceptual modeling. In other words, an ontology is a group of axioms and logical theories designed to describe a specific conceptual model. The representation of such axioms allows to accept assumed models and reject models that are not assumed in the described field. [1] [2]

There are many existing ontologies, developed for all kinds of domains such as medicine, information technology, etc. For example, Dun Bradstreet and the United Nations Development Program created an ontology for Product Classification Systems[1] containing vocabulary for products and services. An ontology provides structured, machine-interpretable information of basic concepts within a domain, as well as terminology for sharing such information.[3]

Developing an ontology brings the following benefits:[3]

- Sharing a common understanding of the structure of information among people or machines (including software). For example, by representing some domain as an ontology an agent can aggregate or request some information satisfying certain conditions.
- Provides the possibility to reuse domain knowledge. For example, in many areas, the concept of time is used. This concept consists of many other concepts such as time intervals, points in time, etc. By providing a description of such an ontology, many domain experts in other fields can reuse it. Another example is the UFO ontology, which will be described in the section 2.1.2.
- Make domain assumptions explicit. Allows to easily change the assumptions as a reaction to changes in domain knowledge.
- Separates domain and operational knowledge. Basically means separation of the specific data from their descriptions. For example, an ontology containing a description of PC components can be separated from the knowledge of the specific components, used in a concrete order.

[1] http://www.ebusiness-unibw.org/ontologies/pcs2owl/

- Analyzing domain knowledge. Formal analysis of terms is important both in the case of reusing an already defined ontology and in the case of extending it.

2.1.1 Upper ontology

There are several levels of abstraction, in which ontologies can exist. Ontology, existing at the most abstract level is called upper ontology. Upper-level or upper ontology exists on the highest level of abstraction, describes the most general aspects, and is domain-independent, allowing other domain-specific ontologies to be derived from it. Upper ontology is a description of general concepts that are the same across all domains.[2]

2.1.2 Unified Foundational Ontology (UFO)

The Unified Foundational Ontology or UFO is an ontology, developed by Giancarlo Guizzardi and associates and further extended at the Ontology and Conceptual Modelling Research Group (NEMO)[2]. UFO consists of three main parts, divided by the concepts they represent:[12]

- UFO-A is an ontology of endurants, covering the aspects of structural conceptual modeling. In other words, UFO-A describes the structural properties of the objects.
- UFO-B is an ontology of perdurants, which mainly covers events and processes.
- UFO-C is an ontology of social and intentional aspects.

"The combination of UFO-A, B and C has been used to analyze, redesign and integrate reference conceptual models in a number of complex domains, for instance, Enterprise Modeling, Software Engineering, Service Science, Petroleum and Gas, Telecommunications, and Bioinformatics." [12]

The fundamental concepts of the UFO, including classes and relations between them, are outlined in the figure A.1.

2.2 Semantic web

The semantic web is a set of information, represented and interconnected in such a way as to be processable by machines. The main purpose of the Semantic Web is to be structured and easily interpreted by machines in order to perform different tasks, such as finding, combining, and acting upon the information across the web.[4]

W3C defines “Semantic Web” as a “Web of data”. which ultimate goal is to allow the computers to do more useful tasks and to develop systems that support trusted interactions over the network. Technologies related to the semantic web allow people to create data stores, develop vocabularies and define rules for data handling. [5]

The semantic web technology stack includes the following technologies:

[2]https://nemo.inf.ufes.br/en/projetos/ufo/

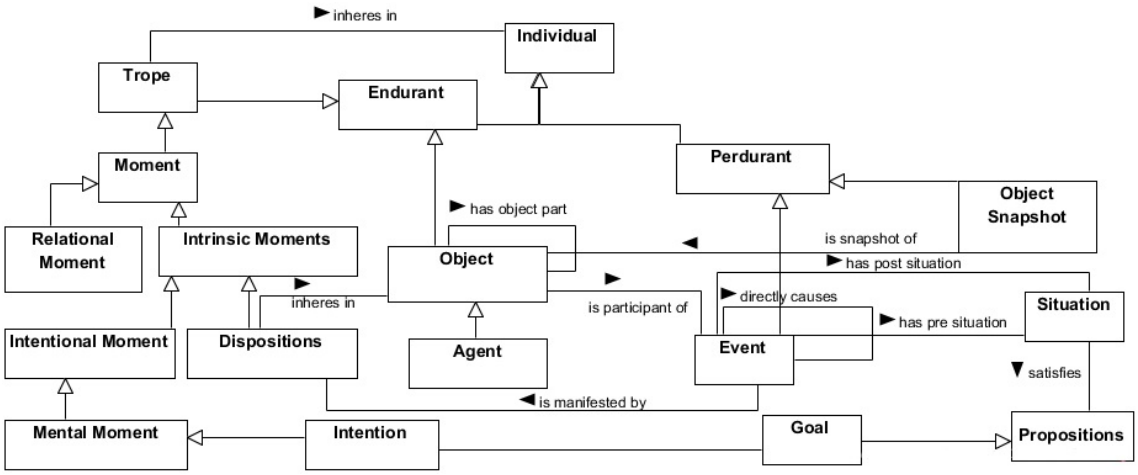


Figure 2.1: Main concepts in UFO. From [13]

- RDF
- OWL
- SPARQL
- etc.

2.2.1 Resource Description Framework (RDF)

RDF stands for Resource Description Framework, it is a framework for expressing the data (documents, files, meta-information), shared across the web. The RDF Data model represents triples, statements consisting of three elements: subject, predicate, and object. Subject and object are resources, related to each other, while predicate represents the nature of the relationship. Any graph can be represented as a set of triples.[6]

2.2.2 Web Ontology Language (OWL)

OWL (Web Ontology Language) is an ontology language, designed to support the Semantic Web technology. OWL is an international standard that was created to share and exchange ontologies between different domains. OWL was originally created by the W3C Web Ontology Working Group[3] for the representation of complex knowledge, i.e. about various kinds of concepts and relations. The current version of OWL is called "OWL 2", which was introduced in 2009, with a Second Edition published in 2012.[7] [8]

[3]https://nemo.inf.ufes.br/en/projetos/ufo/

2.2.3 SPARQL

SPARQL, or SPARQL Protocol and RDF Query Language, is a query language, designed for querying data, stored in an RDF data model. SPARQL is similar to the SQL query language, which is used both for reading and modifying data, although the data are stored in a different model. The result of the SPARQL is either a constructed knowledge graph or a table, containing the result data. A simple example of a SPARQL query can be seen in the source code below.[9]

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX ms: <https://example.com/music-store>

SELECT ?artist ?music
WHERE {
 ?artist rdf:type artist
 ?music ms:written-by ?artist
}
```

2.3 STAMP and STPA

STAMP (Systems-Theoretic Accident Model and Processes) is an accident causation model. This model is based on systems theory and systems thinking and includes numerous factors that can subsequently be critical in more complex systems. STAMP by the meaning is not a method of analysis, it is a set of assumptions about how accidents happen. Examples of such factors are software, human decision-making, human factors, as well as new unproven technologies. [10] [11]

STPA is a STAMP-based hazard analysis method. STPA is an abbreviation of a System-Theoretic Process Analysis. The scope of this method includes a variety of complex systems, such as aviation, space, defense, railroads, etc. In contrast with the traditional approach of hazard analysis, STPA can be started even in the early concept analysis. [10] [11]

2.3.1 STAMP ontology

STAMP-based ontology or STAMP ontology is a formal representation of a knowledge model, related to the STAMP analysis, mentioned in the previous section. In this section, the main concepts of the ontology will be outlined as they will be referenced in this thesis. It is worth mentioning that in this section all “stamp:” prefixes reference to the STAMP ontology, while all “ufo:” prefixes reference to the UFO ontology.

The following classes represent different aspects of the accident causation model:

- stamp:behavioral-constraint (‘process constraint’) - a constraint of a controlled process. The constraint mitigates a specific set of hazards that may occur in the controlled process and it is enforced by a given control.
- stamp:capability - a general category of behaviors assigned to elements in the control structure.

- stamp:controlled-process - a process that is subject to control, which reports status using feedback.
- stamp:controller - a controller is an ufo:agent which has the capability to control processes by providing control actions.
- stamp:process - a process that is not subject to control, therefore does not have a defined controller.
- stamp:structure - represents a stamp control structure, which is usually a complex concept, constituted by two or more structural parts.
- stamp:control-structure-component - the components of a complex control structure, e.g. formed of controller, sensor, and actuator.
- stamp:too-late - a special case of a stamp hazard that violates the temporal constraint by providing a control action too late.
- stamp:feedback-capability - capability to provide feedback to a controller.

The following concepts represent relations between objects in STAMP ontology:

- stamp:derived-from - relation between factor and capability. Usually used to represent that some hazard is derived from some capability.
- stamp:has-capability - relation between a stamp control structure component and its capability.
- stamp:has-control-structure-element-part - represents a relation between a control structure and its elements.
- stamp:is-part-of-control-structure - inverse version of the relation has-control-structure-element-part
- stamp:mitigates - typically denotes the relationship between hazard and constraint. For example, constraint A mitigates hazard B.
- stamp:stamp:action-control-connection - a specific connection between a controller and a sub-ordinate controller or a controlled process.

Chapter 3

Analysis

This chapter contains the analysis of a Business Process Modeling standard of the current version 2.0.2, including the description of the existing BPM tools and the basic BPMN elements. For ease of reference, this standard will be referenced as “BPMN 2.0” in this document and a diagram, made by such standard as “BPMN 2.0 diagram” or simply “BPMN diagram”. The processes, described in the Business Process Modeling Notation will be further used as an input for the BPMN to STAMP converter. Next, the idea of the BPMN interchangeability will be explained. The majority of the discussed in this chapter concepts will be illustrated by a practical example. This example represents a process named “Jednání SAG” (SAG meeting), which is also described in this chapter. Afterward, the research of the existing ontologies, covering the BPMN concepts, will be presented. As the result, the most suitable ontology will be chosen and its main idea will be outlined. In the end, the analysis of the existing BPMN to STAMP converter solution will be discussed.

The analysis was carried out as a part of the project "Digitalization of integrated aviation safety oversight"[1] with the regular meetings and immediate participation of ontology engineers from KBSS group. The discussion mainly concerned STAMP-based ontology, UFO compatibility, and analysis of the BPMN Based Ontology (BBO). Other information regarding STAMP analysis in CAA, along with an explanation of the organization structure certain processes in the organization were provided by domain experts from CAA.

3.1 Business Process Modeling Notation

This section contains a brief overview of what is a Business Process Modeling Notation (BPMN), what core structure elements it has in the latest version of the specification, and how they can be extended. After that, the BPMN diagram interchangeability will be explained, as well as what limitations it has. Then, an example of the process diagram SAG meeting will be introduced along with the explanation of its XML format. Afterward, a brief description of the relevant BPM tool called Bonita Studio BPM tool from Bonitasoft[2] will be presented. Then, in the same section, the organization structure definition aspects will

[1] https://starfos.tacr.cz/en/project/CK01000073
[2] https://www.bonitasoft.com/

be discussed along with the CAA[3] organization structure, including actor mapping for the before mentioned process.

3.1.1 BPMN 2.0

BPMN stands for Business Process Modeling Notation. BPMN provides tools to describe business processes within an organization. Business Process Modeling Notation is a standard for modeling various business processes, therefore the main goal is to provide an easily understandable graphical notation for all, including business stakeholders, business analytics, managers, technical developers, etc. The latest available version of the BPMN specification is 2.0.2[4] released in January of 2014.[14][15] In this thesis for sake of readability, the latest version of the BPMN standard (BPMN 2.0.2) is also referenced as BPMN 2.0.

3.1.2 BPMN conformance and diagram types

According to the BPMN 2.0 specification[5], there are four types of conformance: Process Modeling Conformance, Process Execution Conformance, BPEL Process Execution Conformance, and Choreography Modeling Conformance. Conformance defines a set of requirements, which software should follow to claim compliance or conformance with one of the mentioned conformance types. Requirements in every conformance type can be classified into five categories:[17]

1. A visual representation of BPMN Diagram Types.
2. BPMN Diagram Elements that need to be supported.
3. Import/Export of diagram type.
4. Support for Graphical syntax and semantics.
5. Support for Execution Semantics.

BPMN 2.0 standard also defines several types of diagrams, such as Choreography, Process, and Collaboration. A process, or an orchestration diagram, is the most common type of diagram, describing actions and choices, which lead a process actor to reach a specific objective. A choreography diagram describes interactions between entities in some structure. For example, such diagrams can represent how individual components cooperate in some software or represent interactions between groups in some organization. A collaboration type diagram is meant to illustrate the communication of different processes. [18]

[3] https://www.caa.cz/en/
[4] https://www.omg.org/spec/BPMN/2.0.2/PDF
[5] https://www.omg.org/spec/BPMN/2.0.2/PDF

3.1.3 Basic BPMN elements

For a better understanding of the business process examples used in the thesis, the purpose of the basic BPMN elements, according to the BPMN 2.0 specification, will be summarized in this section.

Pool

A *Pool* can represent either a process or a "black box", which is just an abstraction for a participant. Hence, a *Pool* does not require a reference to a specific process. A *Pool* can have a vertical or a horizontal layout, however, it is not necessary for BPM tool to support both layout types, since the meaning for both variants stays the same. A *Pool*, basically, is a container for a process, or rather is a flow, containing flow elements. The figure 3.1 shows how a simple "black box" pool without any defined elements looks like in Bonita Studio.

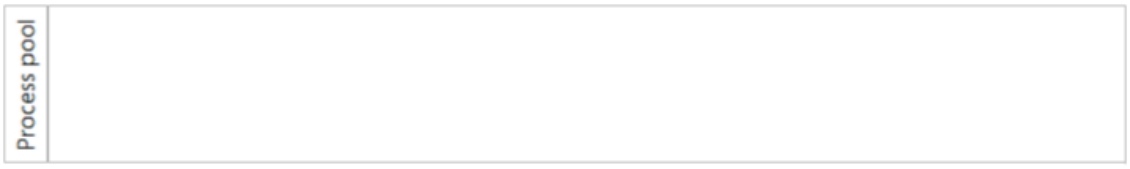


Figure 3.1: Pool in Bonita Studio

As a majority of the BPMN elements, a Pool has a label, or rather a name.

Lane

A *Lane* splits the Pool into multiple parts, which are sometimes called "swimlanes". Each part can contain flow elements. *Lane* layout is inherited from the pool process, in other words, a pool with a vertical layout will be split vertically, and a pool with a horizontal layout - horizontally. Figure 3.2 represents a Pool having two lanes.

Figure 3.2: Pool with two Lanes in Bonita Studio

Performer

BPMN 2.0 defines resources, which are responsible for some tasks within a process. Such resources are called Performers. For instance, a customer is responsible for the task "make an order" in the "Order the pizza" process. In this case, a customer is a *Performer*, which is responsible for the User Task "make an order".

User Task

User Task is a special case of the class Task, which is an atomic Activity within a process. User Task is typically carried out by a human. An example, depicted in figure 3.3 shows modeled in the Bonita Studio process of account registration on a website.

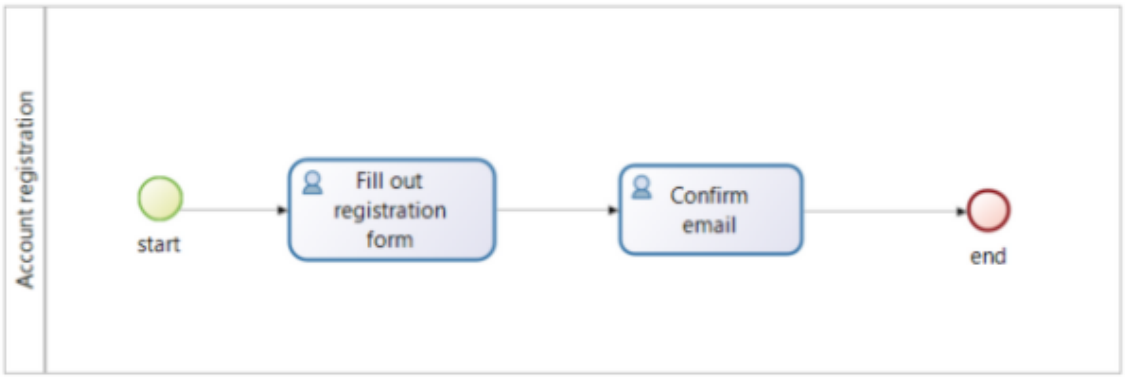


Figure 3.3: Example of the process with two User Tasks in Bonita Studio

In the presented example a website user is responsible for the execution of the two User Tasks: filling out the online form and email confirmation.

There are other task types, such as Manual Task, Service Task, Script Task, Abstract Task, Service Task, etc., however, they will not be covered in this thesis.

Sequence Flow

Sequence Flow is an ordered transition between Flow Elements within a process. *Sequence Flow* is required to have at least one source and one target element. Besides that, such transitions can connect elements from different lanes, but not from different Pools. Basically, *Sequence Flow* defines the process execution order, by providing the next step for each Flow Element (besides the End Events). There are three types of Sequence Flows:

1. (Normal) Sequence Flow.
2. Conditional Sequence Flow, used for Gateway outputs, having defined expression which is a condition for a gateway check result.
3. Default Sequence Flow. Represents a transition, which is taken when all other conditional sequence flows are failed (conditions are not satisfied).

Gateway

Gateways are the controlling points, providing mechanisms for merging (converging) the multiple processes flows and splitting (diverging) them within a process. *Gateways* can have a specific type, such as Exclusive (XOR), Inclusive (OR), Parallel (AND), Event-Based, or Complex. Typically a gateway has multiple incoming (converging) or outgoing (diverging) Sequence Flows. The behavior of a gateway is defined by its gateway type. For example, diverging XOR gateway requires the outgoing Sequence Flows to be conditional since only one of the transitions can be taken. The described case is depicted in the figure 3.4.

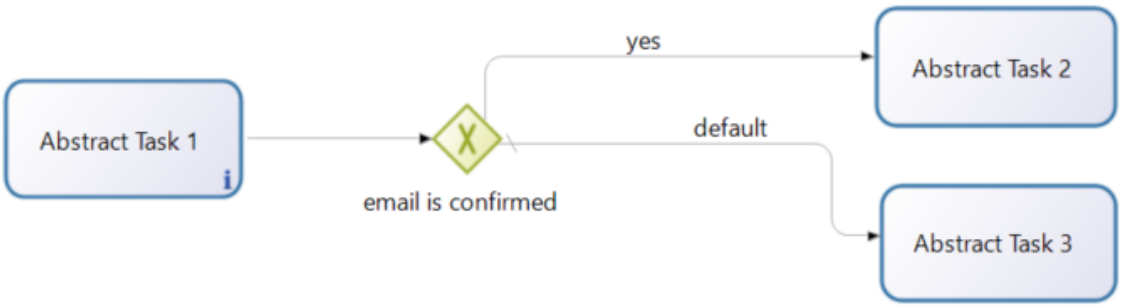


Figure 3.4: Diverging XOR Gateway with conditional and default Sequence Flows in Bonita Studio

Call Activity

Call Activity is a type of Activity, which refers to an already defined independent process. In other words, *Call Activity* is a reference to a process, either self or a different one. This concept provides reusability for the processes by providing the ability to execute the second process within the first one. Once a referenced process is finished, the callee process continues.

Event

Event is an abstraction, meaning that something happened in the process. An event can act as an *Intermediate Event* (connected by Sequence Flows) or a *Boundary Event* (attached to some Activity). There are multiple types of events, such as *Start Event*, *End Event*, Intermediate Event. *Start event* indicates the corresponding process starting point, while *End event* indicates its ending. *Intermediate events* are all other events, that happen between the *Start* and *End* events.

Timer event

Timer Event is a special case of the Event, which has a timer definition. Essentially, a *Timer Event* is something that can happen inside a process if a timer condition was met.

3.1.4 BPMN extensions

The common BPMN model can be extended with additional aspects. This is a useful feature in case we want to extend modeling with additional concepts such as those related to STAMP. However, this will also require the corresponding modifications of the used BPM tool, which in some cases may be expensive or even impossible.

In terms of transformation compatibility (including backward compatibility), it is worth mentioning that extended elements are additional concepts, uncovered by the interchangeability specification. In other words, extended concepts are software-specific and such information will be lost during importing the diagram to another software.

3.1.5 Diagram interchangeability

In order to provide instruments for exchanging BPMN diagrams, the BPMN Model Interchangeability Working Group (BPMN MIWG)[6] has developed a standard for the exchanging of BPMN diagrams between modeling tools from different developers. Interchangeability was achieved by defining a unified XML schema for saving BPMN diagrams in XML format. Defining the unified exchanging format allowed BPM tool users to get rid of vendor-locking. That means if the BPMN designing tool is BPMN 2.0 complaint and supports exporting of the diagrams in such format, another application can import that diagram without the need to manually replicate it.

However, not all information can be exchanged without losses. According to the BPMN 2.0 interchangeability specification, diagram interchange has some limitations. The following aspects of the diagrams may be lost during exchanging between different BPM software:[16]

1. Some visual aspects, like color and decoration of the BPMN elements or text formatting. Nevertheless, the layout of the elements in diagrams stays the same. There is a good example of this behavior in chapter 5.1 of BPMN 2.0 interchangeability specification[16].
2. Some semantic aspects. Those aspects are present in form of scripts (for Script tasks) and implementations of the User tasks.
3. Vendor-specific extensions. This, in general, means any element properties, which are specific only to a particular BPM software.

Such information, usually, gets lost not during exporting of a diagram, but during importing to another software.

3.1.6 Example SAG meeting

As an example of a BPMN diagram, a process named "SAG meeting" (originally "Jednání SAG"), performed within the CAA organization will be introduced. This process is a subject of the "STAMP-based Safety Data Collection and Processing in Civil Aviation Authority" master's thesis by Bc. Kateřina Grötschelová [19] and was modeled by domain experts using the BPM tool Bonita Studio, which is a part of the Bonita application platform. The example will be described in all phases of BPMN to STAMP transformation and every created artifact during that conversion will be represented in the 4 section.

The diagram is presented in figure 3.5. The process SAG meeting starts at the element with the name "Start12" of type Start Event and goes through five activities, represented by elements of type User Task. After the last activity, the process ends at the element with the name "End18" of type End Task. This is a small but useful example for representing the whole process of BPMN to STAMP transformation. According to the OMG specification, every User Task has an actor - a performer, which is responsible for the task execution.[17]

The structure of the XML format is plain and self-explaining. Every BPMN element has its XML representation and references to other elements by id. There is a small excerpt from the SAG meeting process XML file:

[6] https://www.omgwiki.org/bpmn-miwg/doku.php

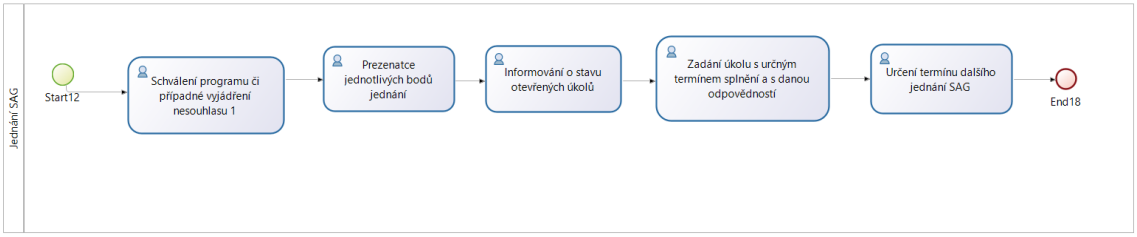


Figure 3.5: SAG meeting BPMN diagram in Bonita Studio

```
...
<model:process id="_k8C2AJZuEemBJoioCbYPCA" name="Jednání SAG">
    <model:userTask id="_2b1mYLqlEeq4RbAg2SWqZQ" name="Schválení
    programu či případné vyjádření nesouhlasu">
      <model:performer id="_Cr6xQEAwEey_ut8uqk4npA">
        <model:resourceRef>_8nP_wLqlEeq4RbAg2SWqZQ</model:resourceRef>
      </model:performer>
    </model:userTask>
...
</model:process>
...
```

Element of the type User Task with id "_2b1mYLqlEeq4RbAg2SWqZQ" (id is generated by the BPMN modeling tool, in this case in Bonita Studio) has a name in the corresponding attribute. User Task element is a part of the parent element Process, with id "_k8C2AJZuEemBJoioCbYPCA", which means that the activity belongs to that Pool. Inside the User Task activity is a definition of an actor, or a performer, according to BPMN 2.0 standard. This performer is responsible for the task execution, has their own id "_Cr6xQEAwEey_ut8uqk4npA" and references the corresponding activity. It is obvious, that the Process element, can have multiple flow elements as well, as have multiple performers.

3.2 Bonita Studio

Nowadays there are many BPM software applications worldwide. Some of them are complete and robust business process management systems, capable to control multiple aspects of the process management within an organization, some of them are only for graphical business process modeling. In the master's thesis by Bc. Kateřina Grötschelová[19] a comparison of some existing BPM software was provided. In the scope of this thesis, the Bonitasoft software will be used as it is a source of diagrams created by domain experts in CAA.

Bonita Studio is a desktop application, a part of the Bonita application platform, developed and maintained by the french company Bonitasoft. The application has two edition types:[26]

- Community edition - free version of the application, available for Windows, Linux, and macOS. Bonita Community is distributed under GPL v2 license and has an open-source code, available on GitHub [7].
- Enterprise edition, which, compared to Community edition, is a commercial version of the platform, with provided users global support, customization of the interface, built-in dashboards, and other features.

Bonita Studio provides to users various sets of tools for describing the business processes within an organization from different perspectives. The next sections will describe the Bonita Studio tools useful to understand the BPMN to STAMP transformation process.

3.2.1 BPMN diagram editor

BPMN diagram editor is a business process modeling tool, providing most of the core elements of the BPMN specification. Users can create multiple diagrams within one project. However, the editor's user interface allows them to work with at most one diagram simultaneously. Every diagram can contain multiple Pools, which, in turn, can be split with multiple Lanes.

A useful editor feature is a BPMN diagram validation, which can validate the diagram and inform the user about inconsistencies or warnings. As can be seen in the figure 3.6 validation can be run manually by clicking the Refresh button in the "Validation status" tab of the editor.

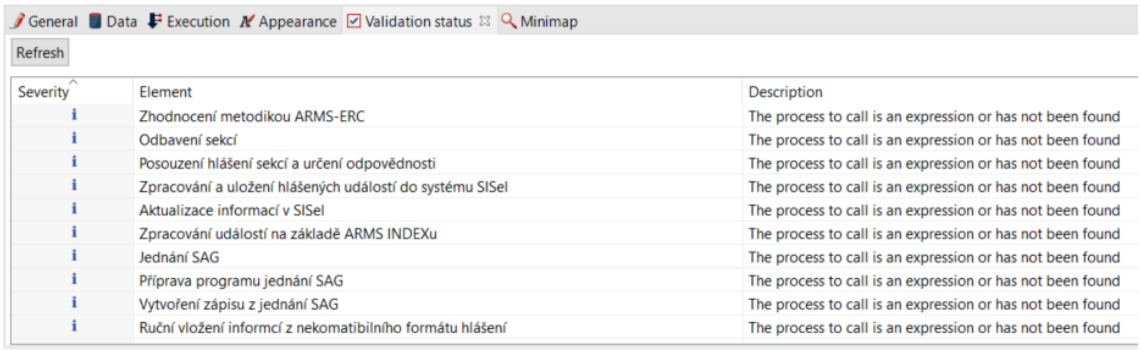


Figure 3.6: Validation window in Bonita Studio

3.2.2 Organization structure editor editor

Organization structure in Bonita is a set of three types of entities:[20]

- Group - usually is a section or department in the organization. The organization can have multiple groups. Every group can have multiple sub-groups. A sub-group is a group with having a parent group. By using group and sub-groups the user can define

[7] https://github.com/bonitasoft/bonita-studio

the department hierarchy within the organization. The group is identified by its name, or exactly by a group path. Group path is a sequence of the parent group names separated by the slash "/" symbol. This means that mentioned separation symbol is not recommended to use in the group names.

- Role - a position in an organization. The role is typically a position within some department. However, due to the fact, that a role can not be explicitly assigned to a specific department, the role is typically a position within the whole organization, not within the department. The role can be specific, for example, a Manager, or be abstract, for example, Member.

- User - is a specific user, a person, within an organization. In Bonita Studio, a User is a specific Person, identified by name and surname. This entity will not be used in this thesis, since provided by domain experts examples contain only definitions of type Group and Role.

- Membership - specifies the groups that the user belongs to, and what role they have in each group. [20] It is confusing, that although Membership in an organization structure consists of the same pair of entity types as a Membership in actor mapping. However, Membership from the organization structure can not be used in the actor mapping.

Entities, explained above are depicted in the figure 3.7. The diagram shows how entities are related to each other along with the carnality of such relationships. The full diagram, including entity attributes, is contained in Appendix A.

3.2.3 Actor mapping configuration

For the created BPMN diagram, users can define actors. An actor is a participant, which performs some tasks within the process, or specifically, within a pool, in which they are defined. Basically, in Bonita Studio, an actor is a placeholder that defines some component of the organizational structure. Actor mapping is a concept that decreases the coupling between the processes of an organization and its elements. An actor can be mapped to one of the following entities: Group, Role, User, or a Membership. Membership is a pair of two elements: Group and Role.

Bonita Studio allows exporting the actor mapping of a specific pool as an XML document. An exported XML contains a list of the defined actors, identified by actor name. Every actor can have multiple mapping targets. The diagram in the figure 3.8 shows how actors can be mapped to groups and roles using membership pairs. This type of mapping will be mainly used in this thesis.

3.2.4 Example SAG meeting

In this section, an example of the CAA organization structure along with the actor mapping for the SAG meeting process will be presented. This example complements the BPMN diagram for the mentioned process, previously introduced in the 3.1.6 sections.

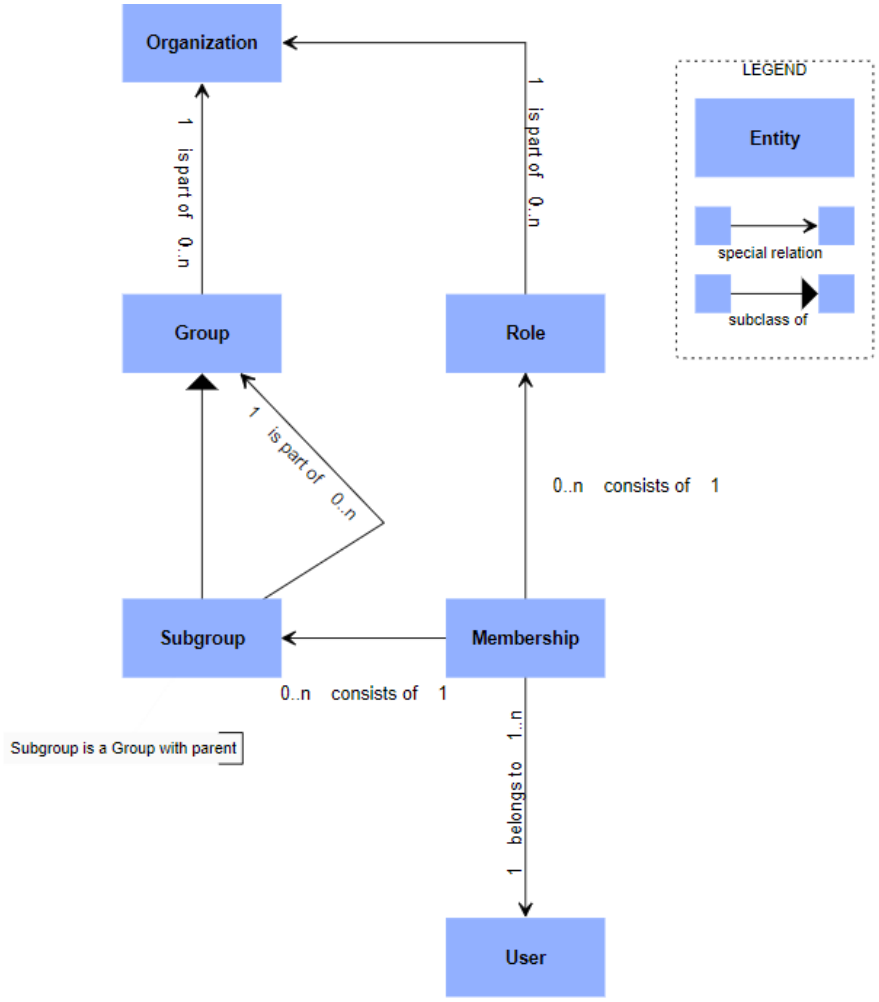


Figure 3.7: Organization structure model in Bonita Studio (compact)

3.2.4.1 CAA organization structure

As mentioned in the documentation section of the Organization management in Bonita Studio, it is suggested not to copy the whole organization to the project but to define only roles and groups that are useful in the processes.[20] The figure 3.9 represents a part of the CAA organization structure, defined by domain experts using Bonita Studio.

There are three top-level groups, ÚCL (Civil Aviation Authority), the organization in which described processes occur, ÚZPLN (Air Accidents Investigation Institute), and Policie ČR (Police of the Czech Republic). The ÚCL has four sub-roups: SAF, SAG, Porada vedení (The management meeting), and Sekce (Section), which also has a sub-group Odbor (Department). The relevant groups to the SAG meeting process example are SAG, SAF, and Sekce. The following list enumerates the relevant roles, defined in the CAA organization:

1. Člen SAG (SAG member).

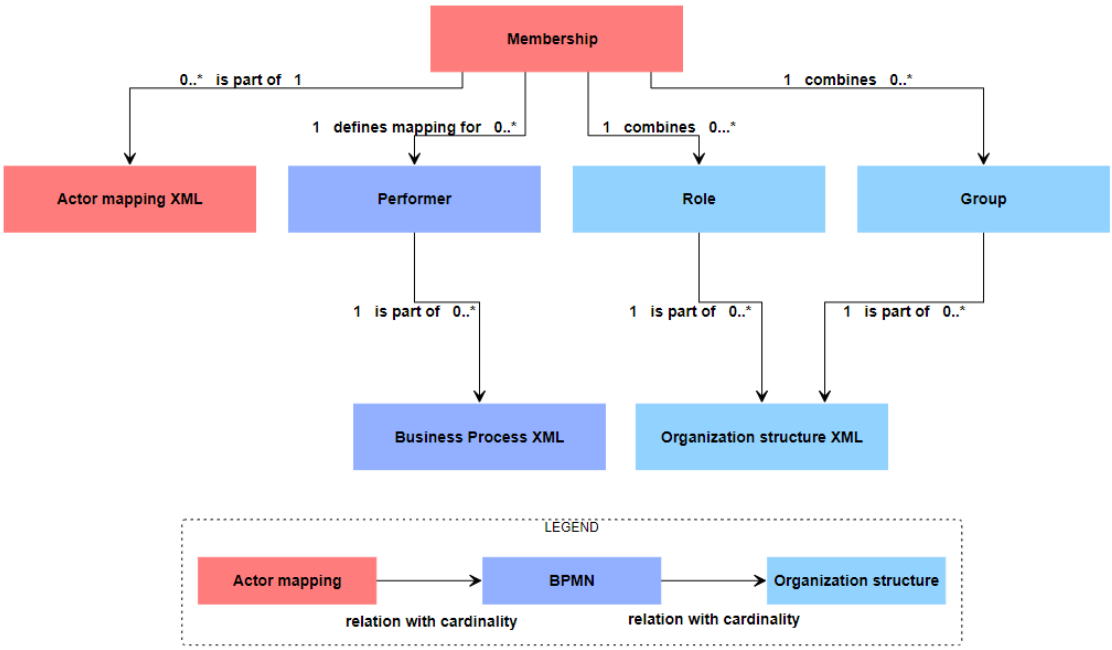


Figure 3.8: Actor mapping using Membership in Bonita Studio (from XML files)

2. Vedoucí SAG (SAG manager).
3. Ředitel sekce (Division director).
4. Manažer systému řízení (Control system manager).

As it can be seen in figure 3.9, the relations between presented groups and roles are not visualized within the Bonita Organisation structure view. This is caused by the Bonita Studio limitations, although this information can be extracted from the actor mapping, described in the next section. The diagram of the groups from the example above is presented in figure 3.10.

3.2.4.2 Actor mapping in SAG meeting

As described in the previous sections an actor mapping in Bonita Studio defines the connections between the process performers and organization structure components. The figure 3.11 represents actor mapping for the SAG meeting process, made in Bonita Studio. Mapping is realized with the Membership mapping type, which, basically, is made through pairs of roles and groups, where both entities are required.

From the mapping depicted in figure 3.11 the following information about the organization structure can be extracted:

- Actor "každý člen SAG" is defined by pair of group SAG and role SAG member.
- Actor "vedoucí SAG" is defined by pair of group SAG and role SAG manager.

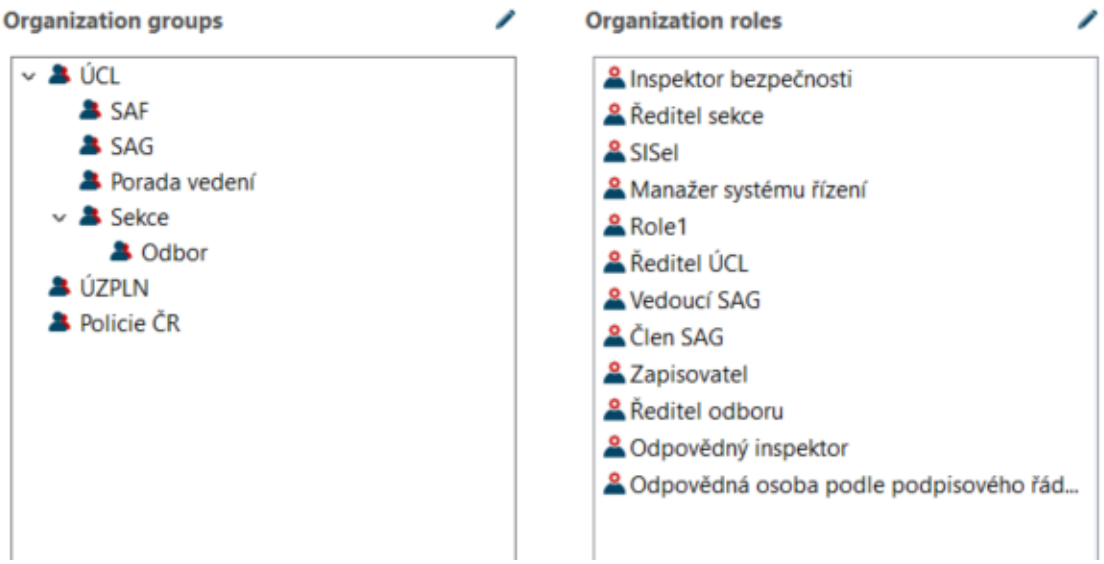


Figure 3.9: Organization structure (roles and groups) of the CAA, defined in Bonita Studio

- Actor "ředitel sekce/manažer systému řízení" is a generalization, therefore an actor is defined by two memberships:
  - Group SAF and role Control system manager.
  - Group Section and role Division director.
- Role SAG member belongs to the group SAG.
- Role SAG manager belongs to the group SAG.
- Role Division director belongs to the group Sekce.
- Role SAG member belongs to the group SAG.
- Role Control system manager belongs to the group SAF.

This example also shows how information about role to group assignment can be retrieved from an actor mapping of a particular process. In contrast to the diagram in figure 3.10, the next diagram contained in the figure 3.12 shows how the group hierarchy looks after the information from the actor mapping was extracted.

3.3 Comparison of existing BPMN ontologies

This section is dedicated to the research of the existing ontologies, which can cover most of the BPMN 2.0 concepts. This ontology will be used in this work as a middle step in BPMN to STAMP transformation. By choosing a good BPMN ontology many problems, which can lead to incompatibilities during the transformation, can be avoided. The searched ontology should comply with the following requirements:

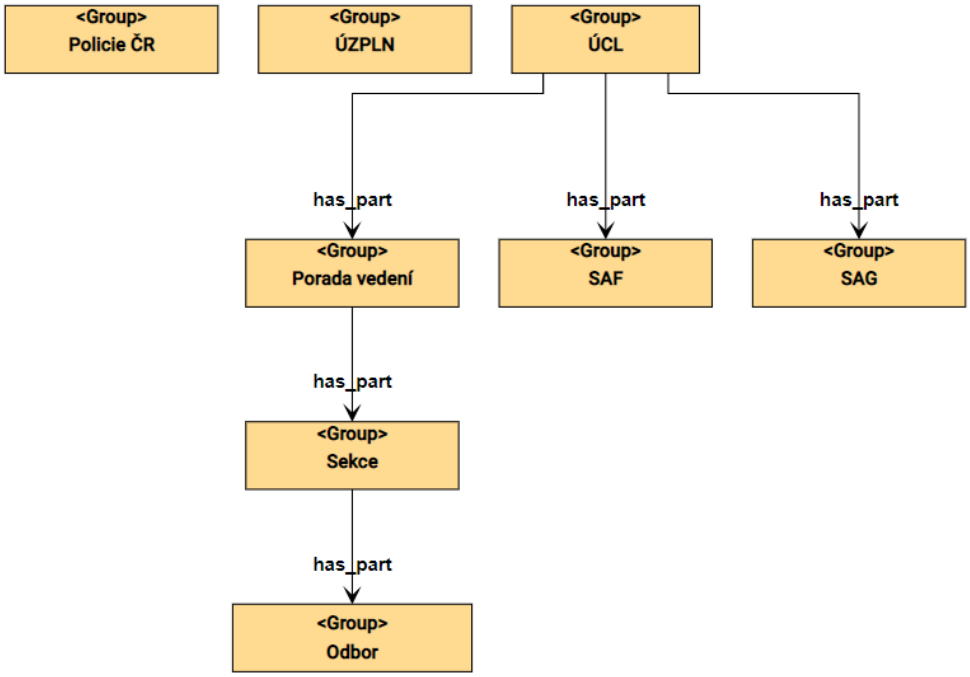


Figure 3.10: Groups in organization structure without assigned Roles

I. The ontology should cover basic business process modeling aspects, used in processes within the organization. The minimal set of such elements is mentioned in the requirements analysis section 4.1. Along with the semantic elements, the ontology should cover properties for describing relations between those elements.

II. The ontology should cover the latest version of the BPMN standard, which is BPMN 2.0.

III The ontology formal description, or rather ontology artifacts, should be publicly available, so it can be accessed by external systems.

IV. Possibility to simply express an organization structure, including particular group hierarchy within the organization, and connect its components to performers in the BPMN process. At least the following aspects should be covered:

   a. Organization group component, representing any department or division within the organization.
   b. Relations between groups to express group hierarchy.
   c. Roles in an organization, representing structure components.
   d. Actors or performers, responsible for the task execution within the process.
   e. Association of the organization role with the organization group, to which this role belongs.

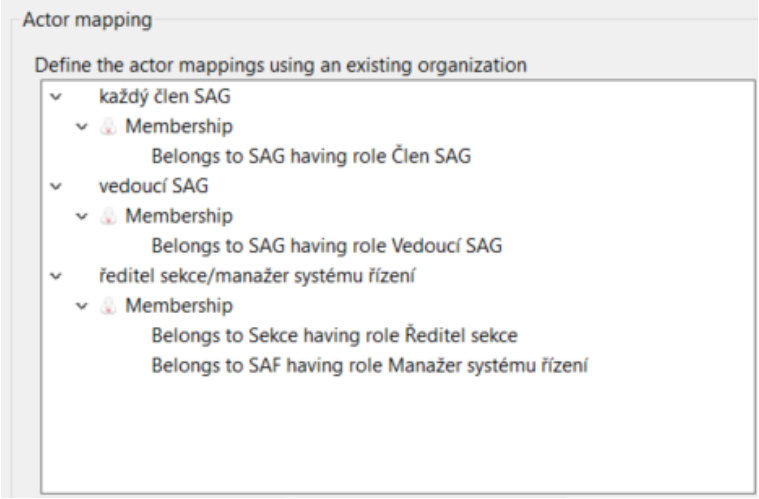


Figure 3.11: Actor mapping of the SAG meeting process, defined in Bonita Studio

V. The ontology should be UFO-compatible (or at least part of it) in order to implement mapping to the STAMP ontology.

3.3.1 Natschläger's BPMN 2.0 ontology

In the paper Towards a BPMN 2.0 ontology (2011)[21], the author proposed a new BPMN ontology called BPMN 2.0 Ontology. This ontology consists of two sub-ontologies:

- bpmn20base ontology, containing base elements taken from the BPMN 2.0 specification,
- bpmn20 ontology, extending the base one.

The consistency and correctness of the provided ontology were evaluated using Reasoner and Syntax checking methods. Unfortunately, this ontology is not available for the community [22], therefore can not be used in this thesis.

3.3.2 BPMN 2.0 compatible upper ontology by Penicina

In this work[2] author's main idea was to find the most suitable existing upper-level ontology, which covers most of the BPMN 2.0 aspects. The authors provided a comparison between the following upper-level ontologies: BFO (Basic Formal Ontology), Sowa, BWW (Bunge-Wand-Weber Ontology), SUMO (Suggested Upper Merged Ontology), and Cyc.

As a result of the comparison, the best-suited ontology BWW was chosen. BWW has covered the listed BPMN concepts: Process, Subprocess, Activity, Compensation, Task, Event, Throw Event, Catch Event, Intermediate Throw Event, End Event, Start Event, Intermediate Catch Event, Boundary Event, Message, Message Flow, Pool, Lane, Association, Data Store, Data Object, Sequence Flow, Gateway. The only semantic element missing is the

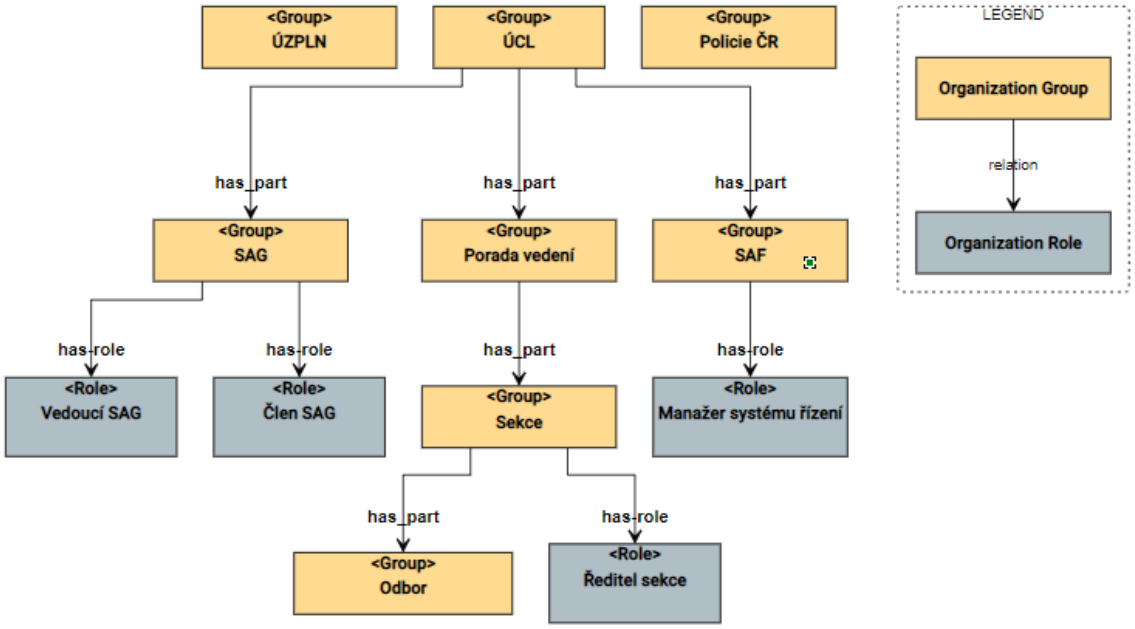


Figure 3.12: Groups in organization structure with assigned Roles

Performer, or an actor, responsible for the task execution. Sequence flow, which represents transitions between activities and events has an associated concept, however, there is no concrete information about the compatibility of other relations between BPMN elements, e.g. responsibility for the Task or transition conditions. Another disadvantage is that BWW ontology is an only upper-level ontology, which is not specifically dedicated to BPMN, so some aspects are abstractions, which would have to be extended to provide missing details. In the conclusion, the author points out that researched BPMN compatibility is purely theoretical and was not tested on a real example.[2]

3.3.3 BPMN Based Ontology (BBO)

The authors of this work[22] proposed an ontological version of the BPMN 2.0 called BPMN 2.0 Based Ontology. The ontology was implemented using fragments from already existing ontologies and meta-models.

To define the scope of the proposed ontology two knowledge sources were used:

1. Technical documents describing industrial business processes.
2. Collected competency questions.

BBO covers five concepts of business processes:

1. Process - a key concept, representing the activities, events, and other flow elements within a process.

2. Input/output specifications, which are the actual data, required by the activities or other elements.

3. Agent. This concept covers the performers, or actors, which perform some activities.

4. Work product. Intends to specify the processes that are required to be executed in order to produce a product.

5. Manufacturing facility. Representing the place, or rather a facility, where the processes take place and the activities are performed.

However, in the scope of this thesis, only Process and Agent concepts are relevant, since they provide enough information to satisfy the functional requirements of the BPMN to STAMP conversion. The Process conceptual model is represented in figure 3.13. All the classes can be used to fully describe most of the BPMN 2.0 processes.

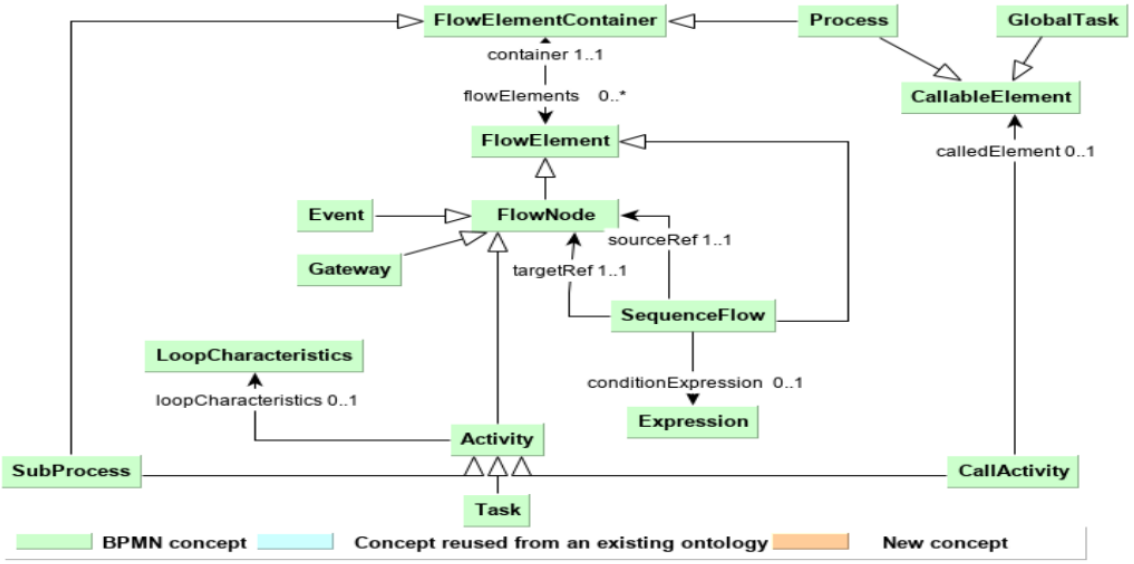


Figure 3.13: BBO concepts related to a BPMN process. Image taken from [22]

Classes and properties, proposed in the Agent concept can be used to represent actors within the processes and the organization structure, in which the processes are executed. However, some concepts needed for the STAMP analysis are missed. The problem and a proposed resolution will be discussed in the 4 chapter. The diagram in figure 3.14 represents the conceptual model of the Agent class:

Finally, the proposed ontology was evaluated using schema metrics and competency questions, mentioned previously. In conclusion, the BBO ontology meets almost all requirements:

I. The proposed ontology covers all BPMN 2.0 elements, listed in the requirement analysis section 4.1, including relations between them.

II. BBO was proposed in 2019 and is an ontological version of the BPMN 2.0.

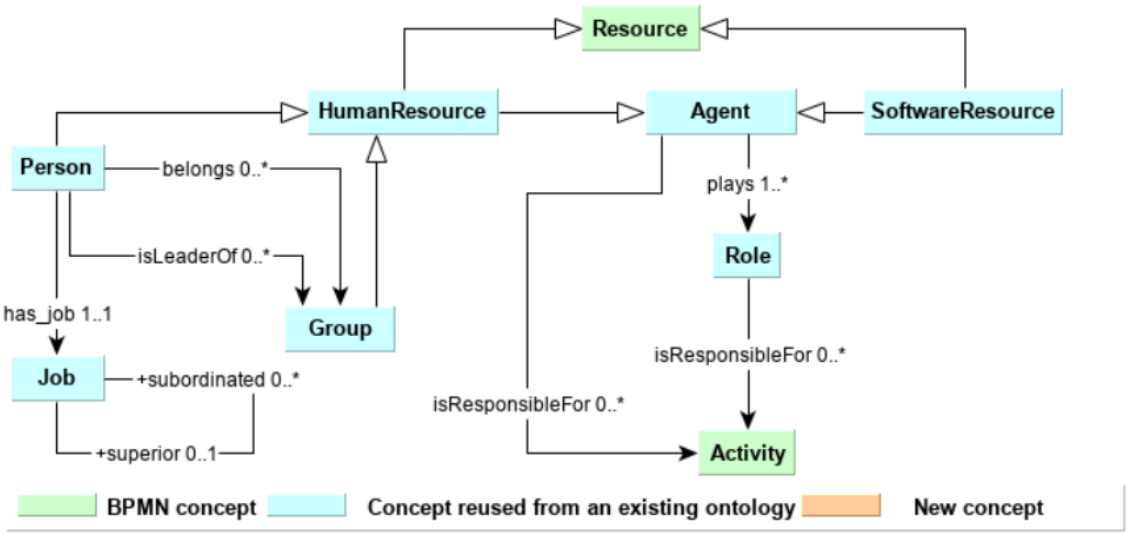


Figure 3.14: BBO concepts related to an Agent. Image taken from [22]

III. The ontology is publicly available[8] in form of OWL files.

IV. BBO provides classes to represent an organization structure, including roles, groups, and actors. However, there is no direct connection between the Roles and Groups. This problem will be discussed later in the section 4.

V. There is no information about UFO compatibility, thus it will be discussed in the next 3.4 section.

3.3.4 Summary

Based on the information obtained during the research, the BBO ontology was chosen as the most modern, suitable for representing BPMN 2.0, and meets most of the specified in the section 4.1. However, the research did not provide any information on compatibility with UFO, as well as the BBO ontology does not cover some aspects of the organization structure. Other works either did not propose a concrete ontology, which meets the requirements, or the proposed ontology is not available to the community.

3.4 BBO and UFO compatibility

In the previous section, the ontology, covering the BPMN elements model was chosen. In the chapter 2 explained the main concepts of the STAMP ontology. Since STAMP ontology is UFO compatible and the goal is to map BPMN to STAMP, it is useful to align BBO to UFO as well. As a result, the representation of the BPMN elements in the STAMP ontology will be possible. However, it requires a lot of work, therefore, the UFO compatibility will be

[8] https://github.com/AminaANNANE/BBO_BPMNbasedOntology

BBO ontology	UFO ontology
Role	ufo:agent
StartEvent	ufo:event-type, ufo:disposition
EndEvent	ufo:event-type, ufo:disposition
UserTask	ufo:disposition, ufo:event-type
Pool, Lane	ufo:event-type
Group	ufo:agent
is_responsibleFor	ufo:has-inherent-moment
is-role-in	ufo:is-object-part-of
has_part	ufo:has-object-part

Table 3.1: STAMP to UFO class and relation mapping

considered partially. Tady bych spomenul konzultaci/spolupraci s "ontology engineer" (nebo muzete i explicitne jmenovat Bogdan Kostov) During the consultations with the ontology engineer Ing. Bogdan Kostov from the KBSS[9] group the following table 3.1 was produced. The table table contains the BBO to UFO mapping for the most relevant for this thesis elements.

The table comprises the ontology elements, i.e. classes and object properties. The first column represents the BBO elements, the second column represents the corresponding UFO concepts. Some of the BBO elements do not require any mapping, since they are irrelevant in the STAMP. This mapping will serve as the basis for the BBO to STAMP mapping, which is described in the chapter 4.

3.5 Analysis of the existing converter

This chapter describes the analysis of the existing BPMN to STAMP converter called bpmn2stampo. The application was written in Java by Ing. Bogdan Kostov, Knowledge Based Software Systems Group (KBSS), FEE CTU in Prague, 2019 and is available on github[10].

The application was developed within the project[11], whose main goal was to create a tool, capable of transforming the BPMN processes along with the organization structure, as well as producing a corresponding STAMP model, based on the given information.

The converter can accept the diagrams exported from Bizagi and Adonis applications in their application-specific formats. The Bizagi-specific format represents an archive file, composed of multiple files, such as user preferences, BPMN diagrams, and others. In contrast with Bonita Studio, the available information about organization structure was implicitly contained in the BPMN file, thus the converter should extract that information from the given processes. On the other hand in Adonis, the application-specific format represents an XML file, that includes the organization structure model. Additionally, the exported

[9] https://kbss.felk.cvut.cz
[10] https://github.com/kbss-cvut/STAMP-Based-Investigation-Tool/tree/master/bpm2stampo
[11] https://starfos.tacr.cz/cs/result/RIV%2F68407700%3A21230%2F19%3A00341289

BPMN files from both applications have a format, which is not interchangeable with other BPM tools. Moreover, the structure of exported data is version-specific, which may lead to compatibility issues in the future.

The analyzed bpmn2stampo converter has some limitations, which have to be taken into account during the implementation of a new solution. The main limitation is that the converter is application-specific and can process only BPMN diagrams from the mentioned software. Moreover, the format of the files is outdated, since both of the BPM tools were actively updated. To resolve the diagram interchangeability problem standardized BPMN 2.0 format should be used. Besides that, the output STAMP model is not complete and lacks certain concepts, such as hazards, constraints, etc.

Chapter 4

Design

The previous chapter was dedicated to the analysis, which serves as a basis for the BPMN to STAMP converter. The analysis covered the explanation of the BPMN 2.0 standard, along with the description of the main components, an overview of the artifacts, complementing the BPMN process, such as organization structure and actor mapping. Finally, the BBO ontology was chosen as a meta-model for the BPMN 2.0. Although the BBO ontology covers most of the necessary concepts, there are some missing parts, which will be discussed and supplemented in this chapter.

As an additional part of the analysis, an example of the SAG meeting process within the CAA organization was introduced. The example itself consists of three parts: the BPMN representation of the process, the necessary part of the organization structure model, and the actor mapping, which is intended for connecting the two first parts, as well as providing additional information on how roles are related to the groups within an organization.

It is worth mentioning that in this chapter all "ufo:" prefixes reference to the UFO ontology.

4.1 Requirement analysis

This section contains requirements for the applications implemented in the scope of this thesis. All requirements were composed and defined during the analysis phase and meetings with the domain experts from the KBSS group. Furthermore, every requirement has its priority, which was classified using the MoSCoW method for the priority classification.

4.1.1 MoSCoW prioritization

MoSCoW is a widely known prioritization methodology. Upper-case letters in the name are an acronym for the set of priorities, used for the classification. The order of the letters corresponds to the priority scale: M is for the highest priority and W is for the lowest priority. MSCW stands for:[23]

- **M**ust have. M priority classifies the requirements, which the final product must meet. MUST is an abbreviation as well, which stands for Minimum Usable SubseT. The im-

plemented application guarantees that these requirements will be delivered, otherwise the project can not be functional or useful.

- **S**hould have. Requirements categorized with S are important for the final product, but still could be left out, even though it could be painful for the project.

- **C**ould have. C classification is used for the requirements, which are less important for the final product. If those requirements are not met, the negative impact will not be great. The main difference between S (Should have) and C (could have) is in "degree of pain" in case if the requirement is omitted.

- **W**on't Have this time. Category W is for the requirements, which are agreed, but most likely will be avoided, for example, due to insufficient time for the implementation.

4.1.2 Client application

4.1.2.1 Functional requirements

This section is dedicated to the functional requirements for the implementing client application for the BPMN to STAMP converter. Requirements are arranged into groups and every functional requirement has its identification number for ease of reference. Every requirement priority is evaluated using the prioritization method mentioned before.

- **M FR1**: The system must provide a command-line interface for interaction with the user.

- **S FR2**: The system must validate the input parameters and notify the user in case of the wrong parameter usage.

- **M FR3**: The system must accept as an input the valid BPMN file and perform a conversion to the BBO ontology output file in Turtle format.

- **M FR4**: The system must accept as an input the valid organization structure file, exported from Bonita Studio, and perform a conversion to BBO ontology output file in Turtle format.

- **M FR5**: The system must accept the following input files and perform conversion to STAMP ontology output file in Turtle format:

  - a valid BBO ontology files*
  - a valid actor mapping files, exported from Bonita Studio

* The provided BBO ontologies might be partial and will be merged during the conversion process.

4.1.2.2 Non-functional requirements

- **M NFR1**: The system should use the BPMN to STAMP converter library, described in the next section.
- **S NFR2**: Source code should be documented (e.g. using Javadoc)
- **S NFR3**: All messages and documentation should be written in English
- **M NFR4**: The system should be tested. At least user testing should be performed with at least three participant.
- **M NFR5**: Minimal set of the requirements technologies: Java

4.1.3 BPMN to STAMP converter library

4.1.3.1 Functional requirements

This section is dedicated to the functional requirements for the implementing BPMN to STAMP converter library. Requirements are arranged into groups and every functional requirement has its identification number for ease of reference. Every requirement priority is evaluated using the prioritization method mentioned before.

- FR6: BPMN diagram processing
  - **M FR6.1**: The system must accept as an input the valid BPMN (XML) file[1].
  - **M FR6.2**: The system must inform the user if the provided BPMN input file has an invalid format.
  - **FR6.3**: The system must be able to perform a conversion of the following BPMN elements to BBO ontology:
    - **M FR6.3.1**: Pools and Lanes
    - **M FR6.3.2**: User Task
    - **M FR6.3.3**: Transition (sequence flow), including conditions
    - **M FR6.3.4**: Gateway
    - **M FR6.3.5**: Boundary Timer event, including timer definition
    - **M FR6.3.6**: Relationships between related components
- **FR7**: Bonita's organization structure file processing
  - **M FR7.1**: The system must accept as an input the valid[2] organization structure XML file, exported from Bonita Studio.
  - **M FR7.2**: The system must inform the user if the provided organization structure input file has an invalid format.

[1] valid BPMN file complies with the schema https://www.omg.org/spec/BPMN/20100501/Semantic.xsd
[2] valid file complies with the schema https://github.com/bonitasoft/bonita-studio/blob/master/bundles/plugins/org.bonitasoft.studio.identity/model/bos-organization.xsd

  - **FR7.3**: The system must be able to perform a conversion of the following organization structure elements to BBO ontology:
    * **M FR7.3.1**: Group
    * **M FR7.3.2**: Role

- FR8: Bonita's actor mapping file processing
  - **M FR8.1**: The system must accept as an input the valid[3] actor mapping XML file, exported from Bonita Studio.
  - **M FR8.2**: The system must inform the user if the provided actor mapping input file has an invalid format.
  - **M FR8.3**: The system must accept an actor mapping file and use it to enrich the organization structure data by assigning roles to the groups.

- FR9: Process represented in BBO ontology
  - **M FR9.1**: The system must accept as an input a valid[4] BBO ontology file.
  - **M FR9.2**: The system must inform the user if the provided BBO ontology input file has an invalid format.
  - **FR9.3**: The system must be able to perform a conversion of the following BBO ontology elements to STAMP ontology:
    * **M FR9.3.1** Role
    * **M FR9.3.2** EndEvent
    * **M FR9.3.3** UserTask
    * **M FR9.3.4** Process
    * **M FR9.3.5** Group
    * **M FR9.3.6** NormalSequenceFlow
    * **M FR9.3.7** TimeExpression
    * **M FR9.3.8** TimerEvent (boundary)
    * **M FR9.3.9** TimerEventDefinition
    * **M FR9.3.10** Relationships between related components
  - **C FR9.4**: The system must be able to validate the constraints in the provided BBO ontology input file.

- **W FR10**: The system should be integrated into Bonita Studio software, providing user access to the library directly from the Bonita Studio application.

4.1.3.2 Non-functional requirements

- **C NFR6**: Source code should be documented (e.g. using Javadoc)
- **M NFR7**: Minimal set of the requirements technologies: Java, JOPA

[3] valid file complies with the schema https://github.com/bonitasoft/bonita-studio/blob/master/bundles/plugins/org.bonitasoft.studio-models/actorMapping.xsd

[4] having correct syntax of the used file format

- **S NFR8**: All messages and documentation should be written in English
- **M NFR9**: The system should be tested
- **S NFR10**: Source code should be versioned using Github
- **W NFR11**: The system should be published in Maven central repository
- **M NFR12**: Modularity: The system should be able to work as a standalone library, which means it can be used as a dependency from any other application.

4.2 Extending BBO ontology

In the section 3.4 was outlined that BBO ontology did not comply with some of the requirements. Firstly, there is no information about UFO compatibility, hence the BBO to UFO mapping of some elements was presented in the previous sections. This problem will be addressed in the 4.3.2 section. Secondly, BBO does not provide the object properties, to represent the direct relation between organization Group and organization Role, since, according to the UML diagram in the figure 3.14, Role and Group both are subclasses of the class Agent. To resolve this problem the new object properties were introduced:

- has-role:
  - Domains (from): Group
  - Ranges (to): Role
- is-role-In:
  - Domains (from): Role
  - Ranges (to): Group
  - Inverse of: has-role
- has-role_part
  - Domains (from): Role, organization structure component
  - Ranges (to): Role, actor in a process
- is-role_partOf
  - Domains (from): Role, actor in a process
  - Ranges (to): Role, organization structure component
  - Inverse of: has-role_part

Object property “has-role” connects a Group within the organization to a specific Role. This property is more precise than the already existing property “has_part”, since “has_part” in BBO does not have any defined constraints or boundaries. The introduction of the property “has-role” allows expressing how Roles are related to Groups. It has an inverse property

"is-role-In", having the same semantic interpretation but opposite orientation, i.e. from Role to Group.

Since an actor, performing specific tasks in a process, and a role within an organization has the same representation in BBO through the class Role, it was necessary to add a new object property to represent the mapping from actor to the concrete organization role. Newly added property "has-role_part" (with an inverse version "is-role_partOf") is capable of connecting instances of the class Role (organization structure component) to another instance of the same class (actor in a process) to represent such a relationship.

4.3 Conversion stages

In this section, an overview of the whole conversion process from BPMN to STAMP will be explained. For transparency and better understanding, examples will be presented in the following sections.

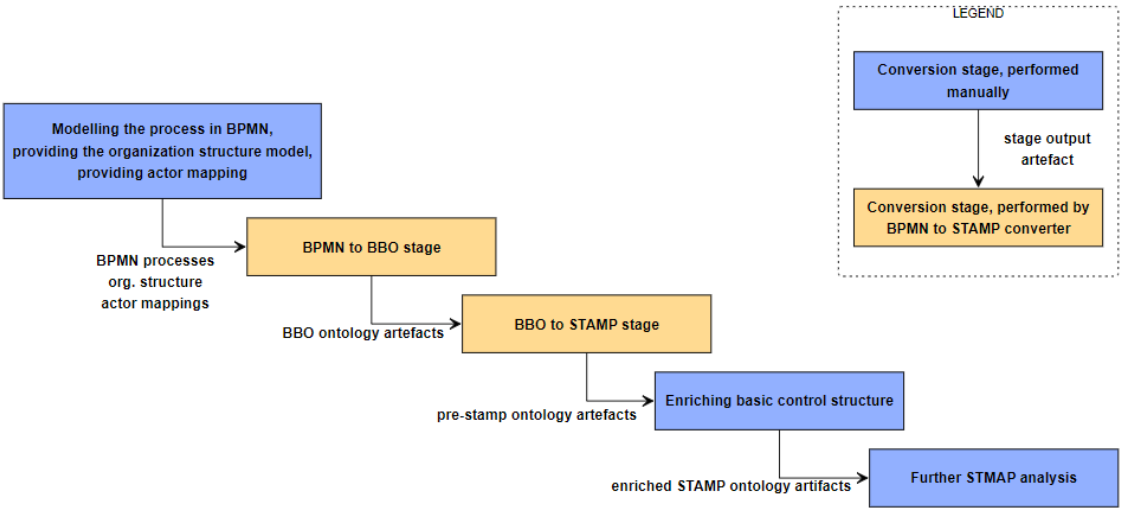


Figure 4.1: The sequence of stages in the BPMN to STAMP conversion process

As depicted in figure 4.1 the whole process starts with the initial stage, where domain experts of a specific organization provide artifacts of three types:

1. The XML format of the process, modeled in any BPM tool capable of exporting the BPMN 2.0 diagrams in interchangeable format.
2. Organization structure model or at least parts used in the provided processes. Actor mapping for each provided process (in which at least one performer is defined).
3. Actor mapping for each provided process (in which at least one performer is defined).

As long as the required data is prepared, the BPMN to STAMP converter can be used to automate the BPMN to STAMP transformation. The transformation, or a conversion, has two major steps:

1. Conversion of the input to the BBO ontology artifacts. This includes the process, the organization structure and the actor mapping.

2. Conversion of the BBO ontology artifacts to the STAMP ontology artifacts. For ease of reference, the output of this stage will be sometimes referenced as "pre-stamp ontology artifacts" or "pre-stamp data" which is, in essence, a STAMP ontology artifact, containing the basic control structure.

The next stage is out of the scope of the BPMN to STAMP converter, however, the converter output provides a foundation for further STAMP analysis. The domain experts can enrich the pre-stamp data with additional information to improve the knowledge about the STAMP control structure model.

4.3.1 BPMN to BBO mapping

This stage is done by a BPMN to STAMP converter. The required information about the process, organization structure, and actor mapping will be extracted from the provided artifacts and applied to the BBO ontology model. As a result, the information will be represented in BBO ontology, having all the necessary information about process execution, performers, responsible for the task activities, and the corresponding organization structure units.

The table 4.1 describes the mapping between the classes of concrete elements. The table 4.2 describes the mapping between the relations of the mapped classes. Since BPMN elements and BBO classes, in the majority of the cases, represent the same concepts, in the mapping description elements from the BPMN will have a prefix "T", for example, TUserTask.

As a future work, it is proposed to design the conversion of additional BPMN elements related to STAMP analysis.

4.3.1.1 Organization structure to BBO mapping

As mentioned in the section "7.2 BPMN Scope" of BPMN 2.0 specification, the Business Process Modeling Notation applies to concepts of a process, hence the following aspects are out of the scope of the specification:[17]

- Definition of organizational models and resources,
- Modeling of functional breakdowns,
- Data and information models,
- Modeling of strategy,
- Business rules models.

In order to model a basic control structure for the STAMP analysis, it is necessary to retain a model of an organization structure. In the tables 4.3 and 4.4 the organization structure to BBO class mapping and relation mapping respectfully are provided.

BPMN	BBO
TParticipant*	Performer
TProcess	Process
TEndEvent	EndEvent
TSartEvent	StartEvent
TUserTask	UserTask
TSequenceFlow	NormalSequenceFlow
TBoundaryEvent**	InterruptingBoundaryEvent
TTimerEventDefinition	TimerEventDefinition
TExpression**	TimeExpression***

* having no referenced process, since according to BPMN 2.0 specification TParticipant can represent either a Process or a Performer.
** BPMN 2.0 specification specifies the multiple types of a class. For simplicity, only one target type is chosen for mapping. However, the mapping can be improved as a part of future work.
*** The result expression contains the specific value in a form of the ontology data property.

Table 4.1: BPMN to BBO class mapping

BPMN	BBO
flowElement (TProcess)	has_flowElements
resourceRef (TUserTask)	is_responsibleFor
sourceRef (TSequenceFlow)	has_sourceRef
targetRef (TSequenceFlow)	has_targetRef
attachedToRef (TBoundaryEvent)	has_boundaryEventRef
timeCycle (TExpression)	has_timeCycle

Table 4.2: BPMN to BBO relation mapping

Organization structure	BBO
TGroup	Group
TRole	Role

Table 4.3: Org. structure to BBO class mapping

Organization structure	BBO
[derived from membersips mapping]	is-role-in
[extracted from the group path]	is_partOf

Table 4.4: Org. structure to BBO relation mapping

BBO	STAMP	UFO
Process	controlled-process	ufo:agent
Role	structure-component or controller*	ufo:agent
StartEvent	process and capability	ufo:event-type, ufo:disposition
EndEvent	process and capability	ufo:disposition, ufo:event-type
UserTask	process and capability	ufo:event-type
Group	structure-component or structure**	ufo:agent

* in case the Role has defined responsibilities
** in case the Group has no parent groups

Table 4.5: BBO to STAMP class mapping

BBO	STAMP	UFO
is_responsibleFor (Role)	has-capability	ufo:has-inherent-moment
is-role_partOf (Role)	is-part-of-control-structure	ufo:is-object-part-of
has_part (Role)	has-control-structure-element-part	ufo:has-object-part

Table 4.6: BBO to STAMP relation mapping

The fact that the described mapping is done for the Bonita Studio-specific format brings some limitations. By providing an additional level of abstraction for the organization structure to BBO mapping, it becomes possible to avoid locking to a concrete BPM tool. In other words, in the case of using another BPM tool, the user will be able to use the BPMN to STAMP converter by providing their own mapping implementation. The described idea is a ground for future work, so will not be realized in the scope of this thesis.

4.3.2 BBO to STAMP mapping

This step is the last in the conversion process, performed by the BPMN to STAMP converter. As a result, a basic control structure will be extracted and become available for further STAMP analysis. In order to implement a BBO to STAMP mapping its necessary to look at the foundational concepts, described in the UFO ontology. By using the UFO concepts, corresponding to the mapped elements, the BBO to STAMP mapping was produced. The class mapping is depicted in the table 4.5, while relation mapping is shown in the table 4.6. Both tables consist of three columns, representing BBO concepts, STAMP concepts, and foundational concepts provided by UFO upper-level ontology.

It is obvious, that some of the concepts are not covered by the proposed mapping. Some of them were omitted, because they are irrelevant to the STAMP analysis, e.g. “NormalSequenceFlow”. Others require certain rules to be defined since the semantics of the source (BBO) and the target (STAMP) elements do not match. For such mapping certain rules are declared:

- For the individual of “InterruptingBoundaryEvent” class in BBO a new individual of the STAMP class “hazard” will be created.

- For the individual of "TimerEventDefinition" class in BBO a new individual of the STAMP class "constraint" will be created.
- The relation "has_boundaryEventRef" between the "InterruptingBoundaryEvent" and a corresponding activity will be represented by the "derived-from" property.
- The relation "has_timeCycle" between the "TimerEventDefinition" and the "InterruptingBoundaryEvent" will be represented by the "mitigates" property.

4.3.3 Enriching basic control structure

This stage is out of the scope of the BPMN to STAMP converter. This stage is dedicated to the enrichment of the pre-stamp ontology artifacts, which is a product of the converter. Domain experts, by using the organization-specific knowledge are capable of providing additional information, such as declaring system-level hazards, losses, etc.

4.3.4 Import conventions

During the whole BPMN to STAMP converting process, the following artifacts are produced:

- BBO ontology artifact, describing the organization structure
- BBO ontology artifact, describing processes and performers
- pre-stamp ontology artifact
- STAMP ontology artifact

Thus, in order to define rules of how the mentioned artifacts should import each other, the import conventions were created, which are graphically represented in figure 4.2.

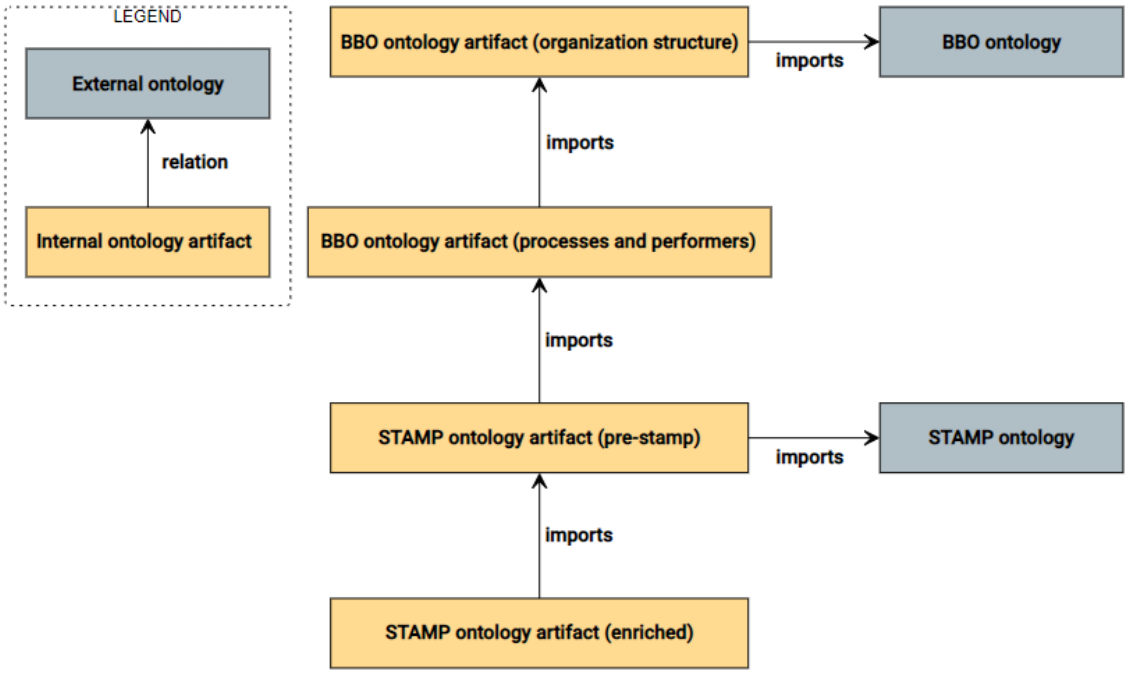


Figure 4.2: Ontology import convention

Chapter 5

Implementation

This chapter describes the implementation phase. Since the scope of this thesis includes two applications, this chapter contains information about each of them. The first section in this chapter describes the technology stack, used in the implemented solution. Additionally, the dependencies used in the applications are mentioned with a short description. Finally, the problems encountered during the implementation process are discussed.

5.1 Technology stack

Based on the requirements, described in the chapter 4.1, it was decided to divide the application into two modules. Both of them are written in Java using Maven for building processes and dependency management.

The first module is the converter library "bpmn2stamp_converter" which can be reusable in any other Java project. The library provides methods for different types of mapping described in the Design section 4. For the mapping process, the library uses the Mapstract library. To enable the validation of the structure of the XML files, such as the BPMN process description, the description of the org. structure, the mapping of actors, the JAXB library was used. JAXB was allowed to generate Java classes from the XSD files for each of the models, which were subsequently used to perform operations on the input data. Persistence and ontology data reading is done by JOPA API.

The second module “bpmn2stamp_console” is a CLI application, which provides a command line interface for access to the converter library functions. This module is lightweight and does not use many dependencies. For the construction of the CLI options, the Apache Commons CLI library was used.

The diagram in the figure 5.1 depicts the dependency usage of each sub-module.

5.1.1 Mapstruct

MapStruct[1] is a simple and powerful library, which generates code, based on the rules in form of the Java annotations. It uses an annotation processor, plugged into the Java compiler,

[1] https://mapstruct.org/

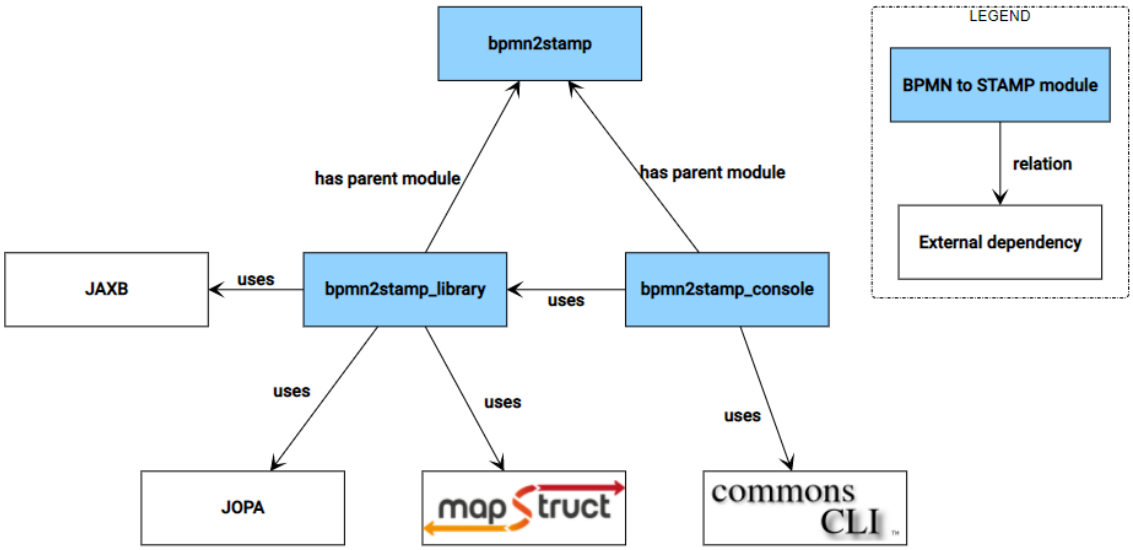


Figure 5.1: Dependency usage and module structure of the BPMN to STAMP converter library

which during the compilation phase parses the declared mapping rules, stated by the user. As a result implementations of the mapping methods are generated.[25]

5.1.2 JOPA

JOPA[2] is an abbreviation for the Java OWL Persistence API. JOPA provides an API to access the ontologies in an Object-oriented way. Architecturally, JOPA is divided into two main parts:[24]

- OOM, which stands for Object-ontological Mapping, takes care of the object-oriented data representation along with providing persistence for the application.
- OntoDriver, serves as a software layer that provides access to the underlying storage.

5.1.3 Apache Commons CLI

Apache Commons CLI[3] is a Java library, which allows to easily implement a command-line interface (CLI) in Java. The library provides tools for defining options, helping messages, and other attributes of a typical CLI application. Besides the Java-like properties, Apache Commons CLI offers other types, such as GNU and POSIX-like options.

[2]https://github.com/kbss-cvut/jopa
[3]https://commons.apache.org/proper/commons-cli/

5.2 Problems and solutions

During the implementation several problems occurred, each of them will be briefly discussed in this section.

Relax participation constraint validation of inferred attributes on entity persist

The problem is related to JOPA when all constraints of the saving entities are being verified, which leads to the "CardinalityConstraintViolatedException" exception. Since the inferred values can not be resolved, due to the inexistence of the saved ontology, the solution[4] was proposed by Ing. Martin Ledvinka, which extended the JOPA API by adding the possibility to disable the constraint checking during the saving phase.

Initialization of ontologies

JOPA does not provide methods for ontology initialization. Thus a workaround solution was used by using the OwlApi[5] library and saving the resulting ontology in Turtle format with all required initial triples and prefixes.

Updating of the mapping file

JOPA initializes the mapping of the ontology IRI and its location once an instance is created. This means that the mapping cannot be updated during runtime. Moreover, the file locations should be in absolute path format. To address these problems a temporary solution was implemented, which is that a new mapping file is created during runtime and used for the JOPA persistence manager initialization.

Name conflict of the actor mapping XML schema

This problem was caused by the malformed XSD file, containig a schema for actor mapping XML files from Bonita Studio. The file contained two elements with the same name but different meanings. This issue prevented the JAXB class generator to generate the corresponding Java classes from the mentioned XSD schema. The problem was resolved simply with manual updating the file, i.e. changing the duplicated element names.

[4] https://github.com/kbss-cvut/jopa/issues/98
[5] https://github.com/owlcs/owlapi

Chapter 6

Testing

This chapter is dedicated to testing the application implemented in this thesis. The BPMN to STAMP converter was tested on examples of processes, provided by domain experts from CAA.

The main purpose of the testing phase was user testing. User testing was performed by a group of individuals. In order to evaluate the tested software, all participants received a post-test questionnaire to describe their impressions.

6.1 Testing results

During the user testing phase, three participants completed two test scenarios. Each of these scenarios was focused on the specific functionality of the application. The converter was evaluated using a pre-prepared set of questions. Each question asked the user to verify whether all relevant elements in the converter input could be found on the converter output. Since not all participants could be familiar with either the BPMN 2.0 standard or the Bonita Studio software, input files were explicitly provided for testing (e.g. the BPMN process XML along with an exported organizational structure from Bonita Studio). Finally, a post-test questionnaire was offered to each participant to collect user experience information.

The provided responses indicated shortcomings of the software under test, mainly due to the unfriendly user interface. Although everyone reported that the documentation was clear and all the participants were satisfied with the converter output.

Chapter 7

Conclusion

This thesis was focused on the analysis of the latest version of the BPMN standard (2.0.2) and Bonita Software, which is a BPM tool used in the Civil Aviation Authority (CAA) organization to model business processes. The process of transforming BPMN diagrams into a control structure model using the STAMP ontology (created by the KBSS group) had to be defined. During the analysis phase several ontologies serving as ontological representation of BPMN were considered. The most appropriate one, namely BBO, was chosen as the ontology for the representation of the organizational structure as well as the processes within the organization. This ontology was used as a transitional step in the process of converting BPMN to STAMP. The selected ontology was analyzed and partially aligned with the upper-level UFO ontology in order to be able to convert BBO elements into corresponding concepts of the STAMP-based ontology. The analysis process was carried out in regular meetings with ontology engineers from the KBSS group. The discussion mainly focused on STAMP-based ontology, UFO compatibility, and BBO analysis. Other information concerning STAMP analysis in CAA, as well as an explanation of the organizational structure of some processes were provided by CAA domain experts.

As a result of this work, a prototype transformation tool capable of converting the basic elements of BPMN into a control structure formally described in a STAMP-based ontology was designed and implemented. This tool was realized in the form of a Java library and command-line tool. The prototype was successfully tested on the SAG meeting process provided by domain experts from the Civil Aviation Authority organization. The designed converter software supports the interchangeability of the BPMN 2.0 format, therefore it is compatible with other BPM tools that also support it.

Bibliography

[1] Barjis, Joseph & Pergl, Robert. (2014). Enterprise and Organizational Modeling and Simulation: 10th International Workshop, EOMAS 2014, Held at CAiSE 2014, Thessaloniki, Greece, June 16-17, 2014, Selected Papers. 10.1007/978-3-662-44860-1.

[2] Penicina, L. (2013). Choosing a bpmn 2.0 compatible upper ontology. In The 5th International Conference on Information, Process, and Knowledge Management (pp. 89-96).

[3] Noy, N. & Mcguinness, Deborah. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Knowledge Systems Laboratory. 32.

[4] Elnaggar, Ahmed. (2015). The Semantic Web. 10.13140/RG.2.1.3622.0888.

[5] W3C. (2015). Semantic Web - W3C. Semantic Web - W3C. Retrieved December 17, 2021, from <https://www.w3.org/standards/semanticweb/>

[6] Dave Beckett, Brian McBride. (2004). RDF/XML Syntax Specification (Revised). W3C. Retrieved November 1, 2021, from <https://www.w3.org/TR/REC-rdf-syntax/>

[7] OWL Working Group. (2012). OWL - Semantic Web Standards. W3C. Retrieved November 28, 2021, from <https://www.w3.org/OWL/>

[8] Heflin, J. (2007). An Introduction to the OWL WEB Ontology Language. <http://www.cse.lehigh.edu/~heflin/IntroToOWL.pdf>

[9] Nečaský, M. (2021, January 25). Série Znalostní grafy: Díl 3: SPARQL - Portál otevřených dat České republiky. Data.GOV. Retrieved October 8, 2021, from <https://data.gov.cz/%C4%8Dl%C3%A1nky/znalostn%C3%AD-grafy-03-sparql>

[10] Reykjavik University. (n.d.). What is STAMP/STPA? Retrieved October 1, 2021, from <https://en.ru.is/stamp/what-is-stamp/>

[11] N. G. Leveson And J. P. Thomas. (2018). STPA Handbook. <https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf>

[12] Suchánek, M. (2018). UFO — OntoUML specification documentation. OntoUML.ReadTheDocs. Retrieved December 4, 2021, from <https://ontouml.readthedocs.io/en/latest/intro/ufo.html>

[13] Ahmad, J., Kostov, B., Lalis, A., & Kremen, P. (2018). Ontological Foundation of Hazards and Risks in STAMP.

[14] Object Management Group. (2021). BPMN Specification - Business Process Model and Notation. Retrieved October 5, 2021, from <https://www.bpmn.org/>

[15] Business Process Model and Notation. (2021, November 12). In Wikipedia. <https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation>

[16] Kurz M, Menge F, Misiak Z (2014): Diagram Interchangeability in BPMN 2. <http://www.omg.org/oceb-2/documents/BPMN_Interchange.pdf>

[17] Object Management Group, Oracle, SAP AG, TIBCO, Unisys. (2014). Business Process Model and Notation Version: 2.0.2. OMG.ORG. Retrieved December 1, 2021, from <https://www.omg.org/spec/BPMN/2.0.2/>

[18] Corradini, Flavio & Morichetta, Andrea & Polini, Andrea & Re, Barbara & Tiezzi, Francesco. (2020). Collaboration vs. choreography conformance in BPMN.

[19] Kateřina Grötschelová, K. (2011). STAMP-based Safety Data Collection and Processing in Civil Aviation Authority [Master's Thesis, CZECH TECHNICAL UNIVERSITY IN PRAGUE]. <https://dspace.cvut.cz/bitstream/handle/10467/88211/F6-DP-2020-Grotschelova-Katerina-DP_Grotschelova.pdf>

[20] Bonitasoft. (2021a). Organization management in Bonita Studio | Bonita Documentation. Retrieved November 2, 2021, from <https://documentation.bonitasoft.com/bonita/2021.1/organization-management-in-bonita-bpm-studio>

[21] Carpella, Christine. (2011). Towards a BPMN 2.0 ontology. 95. 1-15. 10.1007/978-3-642-25160-3_1.

[22] Annane, Amina & Aussenac-Gilles, Nathalie & Kamel, Mouna. (2019). BBO: BPMN 2.0 Based Ontology for Business Process Representation. 10.34190/KM.19.113.

[23] Agile Business Consortium. (2021). Chapter 10: MoSCoW Prioritisation | The DSDM Handbook | Agile Business Consortium. Retrieved December 21, 2021, from <https://www.agilebusiness.org/page/ProjectFramework_10_MoSCoWPrioritisation>

[24] Ledvinka, Martin & Kremen, Petr. (2015). JOPA: Accessing Ontologies in an Object-oriented Way. ICEIS 2015 - 17th International Conference on Enterprise Information Systems, Proceedings. 2. 212-221. 10.5220/0005400302120221.

[25] MapStruct. (2021). MapStruct – Java bean mappings, the easy way! Retrieved September 19, 2021, from <https://mapstruct.org/>

[26] Bonitasoft. (2021). Bonita pricing. Retrieved September 16, 2021, from <https://www.bonitasoft.com/pricing>

[27] MIT Partnership for Systems Approaches to Safety and Security. (n.d.). MIT Partnership for Systems Approaches to Safety and Security (PSASS). PSAS Scripts MIT. Retrieved July 8, 2021, from <http://psas.scripts.mit.edu/home/>

Appendix A

Organization structure model in Bontia Studio (Complete)

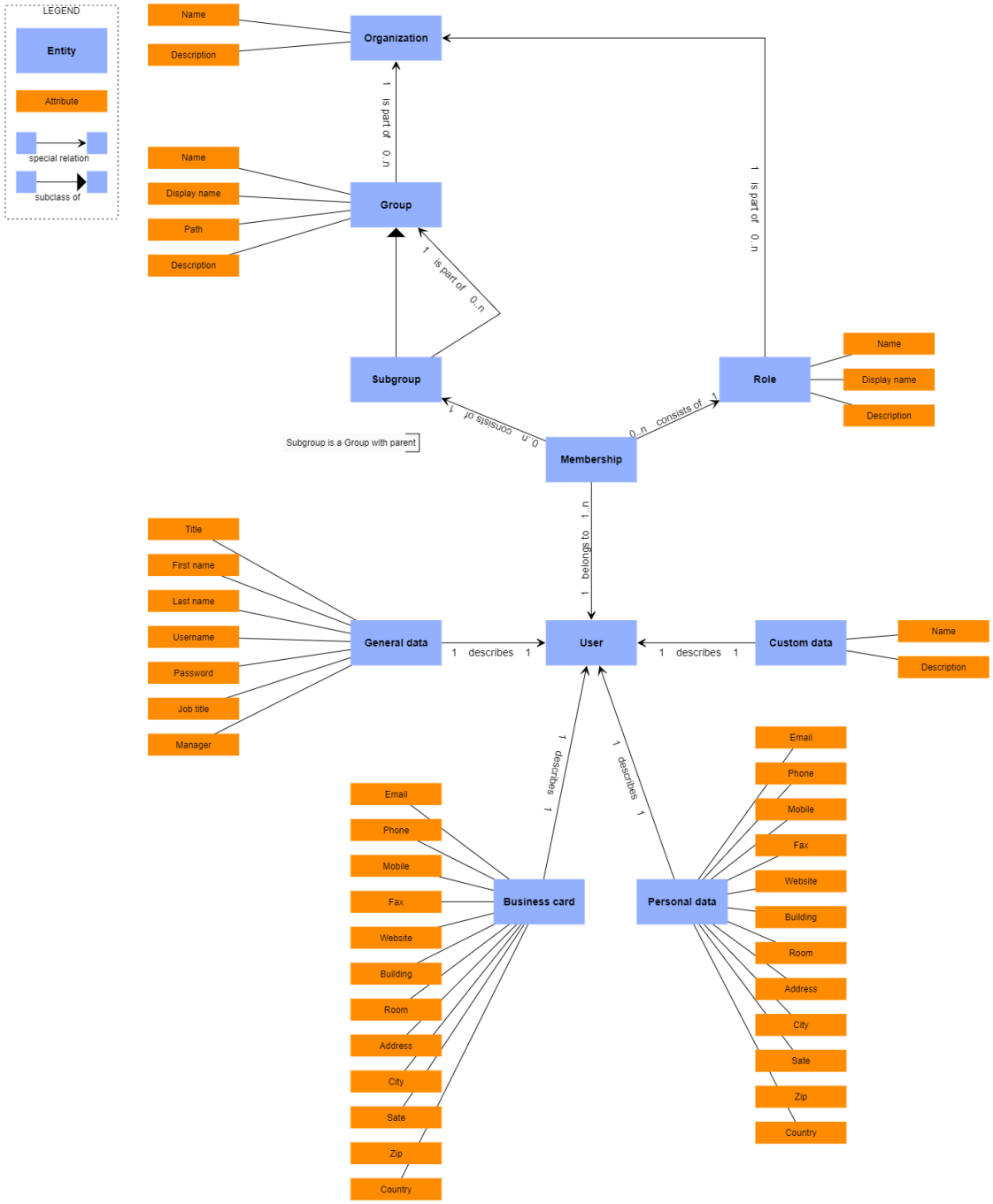


Figure A.1: Organization structure model in Bontia Studio (Complete)

Appendix B

Contents of the enclosed CD

```
.
|--- bpmn2stamp_source.zip.........Source code, packed in a zip format
|--- bpmn2stamp.jar................Executable JAR file of the CLI tool
|--- readme.txt....................Readme file
|--- cli_documentation.txt.........CLI tool documentation
|--- grigobog_thesis.pdf...........Thesis PDF
```