FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF MASTER’S THESIS

Title:	Evolvability of Business Process Models
Student:	Bc. Stanislav Mikeš
Supervisor:	Ing. Marek Skotnica
Study Programme:	Informatics
Study Branch:	Web and Software Engineering
Department:	Department of Software Engineering
Validity:	Until the end of winter semester 2019/20

Instructions

Majority of state-of-the art enterprises are using business process management systems (BPMS) to bridge the gap between the business needs and IT implementations. But as enterprises do change over time due to government regulations and market requirements, these systems need to change over time. A goal of this thesis to investigate what types of changes usually occur and what is their impact on enterprise information systems based on BPMS.

Steps to take:
1. Explore state-of-the-art BPMS
2. Explore how BPMS handle versioning
3. Create a proof-of-concept case study with minor and major changes
4. Propose best practices how to handle these changes

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague June 2, 2018

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master’s thesis

Evolvability of Business Process Models

Bc. Stanislav Mikeš

Department of Software Engineering
Supervisor: Ing. Marek Skotnica

January 9, 2019

Acknowledgements

I would like to thank my thesis supervisor, Marek Skotnica, for his valuable advice and guidance throughout the process of writing and searching for resources. I would also like to thank my parents for their continuous support.

Declaration

In Prague on January 9, 2019 . . . .

Citation of this thesis

Mikeš, Stanislav. *Evolvability of Business Process Models.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

Podnikové procesy jsou zásadní součástí každé společnosti. Vzhledem k tomu, že společnosti rostou, procesy se stávají stále složitějšími a náklady spojené s jejich změnami rostou. Podle některých publikací je až 6 % obratu společnosti vynaloženo pouze na dodržování regulací a předpisů.

Abychom se vypořádali s těmito stále se měnícími procesy, používají se systémy pro správu business procesů (BPMS). Tyto systémy umožňují společnostem překlenout propast mezi implementací softwaru a obchodními potřebami.

Hlavním cílem této práce je analyzovat, jak efektivně BPM systémy řeší měnící se podnikové procesy.

První kapitola poskytuje přehled o tom, jak se stávající BPM systémy řeší tyto změny a porovnává je s jinými přístupy, jako jsou například Low-code platformy.

Druhá část analyzuje, jaké změny nastávají v podnikových procesech, a vysvětluje na příkladu, jak s nimi pracovat.

Nakonec jsou popsány postupy a doporučení pro řízení změn v podnikových procesech.

Klíčová slova BPMN, evolvabilita, migrace, procesní řízení, změnové řízení, BPR

Abstract

Business processes are an essential part of every company. As the companies grow, the processes are becoming more complex and cost a lot of money to maintain. According to some reports, 6 % of company turnover is spent only on compliance processes alone.

To deal with these ever-changing processes, a Business Process Management Systems (BPMS') are being used. The BPMS' allow companies to bridge the gap between software implementation and business needs.

The main goal of this thesis is to analyze how effectively the BPMS' deal with the changing business processes.

The first chapter provides an overview of how existing BPMS' are dealing with the changes and compares them to other approaches such as Low-code platforms.

The second part analyzes what kind of changes occur in business processes, and explains on an example of how to deal with them.

Finally, best practices to deal with changes in business processes are described.

Keywords BPMN, evolvability, migration, business process management, change management, BPR

Contents

List of Figures

List of Tables

Introduction

Motivation

Processes are generally a part of all tasks. They can be found in corporate structures and the public sector but are also used by many individuals. The complexity of processes can increase based on the company growth or continuously changing regulations.

The main goal of this thesis is to provide recommendations for change management of business processes. It is reflected in the competitiveness of the market. In this case, there is the frequent use of Business Process Management (BPM) that specifies entities and their part that they will engage in the processes. It also specifies the inputs and outputs of a process.

An important role in this effort is business process modeling, which allows processes to be captured in the visual form to facilitate their creation, analysis, and subsequent changes. A unified notation (BPMN) is used for this visualization.

Nowadays, all spectrums of business are supported by some type of information system. Each company uses different information systems based on their own business types and company structures. A majority of enterprises have decided to take advantage of Business Process Management Systems (BPMS') to bridge the gap between the business needs and IT implementations. BPMS' allow the aforementioned functionality to model business processes as well as orchestrate sub-steps from the point of view of lower-level systems. The purpose of these systems is, for example, to manage different types of recourses or customer relationships.

However, as enterprises change over time, due to government regulations and market requirements, these systems need to change as well. Companies claim that they spend around 6% of their turnover on complying regulations [5].

Objectives

The goal of this work is to investigate what types of changes are most often performed and what the impact is on BPM-based corporate information systems.

This thesis is divided into theoretical parts. They include the exploration of popular solutions for process management and an output in the form of best practice recommendations for change management. That is based on the practical part of creating a case study with change interventions.

Explore state-of-the-art BPMS'

The first step is to explore the BPMS' that are used these days and are the most popular in the market. It is also important to investigate an alternative solution called Low-code Application, which replaces BPMS' for less complex processes, mostly for smaller companies.

Explore how BPMS' handle versioning

The next step is to learn about the procedures that BPMS' have to deal with during model versioning in case of necessary repairs or updates. The focus in this chapter is mainly on BPMs.

Create a proof-of-concept case study with minor and major changes

After the versioning exploration, the proof-of-concept (with an open-source solution) is created based on a more complex business process where it is possible to present interventions in the form of minor and major changes.

Propose best practices how to handle these changes

Based on the two previous steps and their outputs it is possible to propose methods how to handle upcoming changes in complex process models.

Chapter 1

Business Process Management Platforms

There are two main research and advisory companies that release their research papers nearly every year. This is no different with BPM systems. The first well known company is Gartner, with their magical quadrants [6]. Another research company that this chapter will use as a source is called Forrester. They started using the term DPA [7], instead of BPMS, which means Digital Process Automation. Since these terms were used for the first time, the BPM market grew and there are many more specialized software solutions. Some vendors aim at the enterprise sphere or larger companies while others aim at smaller businesses. This will affect the number of features, price, time for implementation, and the need for specialists. Software for business process management can be divided into two categories - BPMs and low-code platforms. Main vendors have already recognized the demand for low-code solutions and most of them offer these solutions within their BPMS.

1.1 Key Factors

The importance of key factors can vary depending on the category or size of a company.

One of the main factors is if the solution is cloud-based or on-premise. It is crucial for enterprises that their processes are complex and include their know-how. One of the requirements is to keep the solution on their servers because of lack of trust or a wider variety of adjustment. On the other hand, the cloud-based software can be useful for smaller companies or companies that are not strictly process-centric. It allows them to start modeling and using the software at that exact moment without spending unnecessary time or money during the implementation.

Another important factor is the capability of integration with the existing systems. Most of the solutions that are on the market feature that. The integration is mainly available via REST API, Java integration, Soap, prepared adapters, and connectors.

One of the integrations could be called Single Sign-On (SSO) as well. SSO becomes more important with the growth of a company and the need of employee directories to manage role/person access to their systems. In this case, SSO is a mandatory feature to allow users to log in with one set of credentials.

Another key factor could be mobile-ready apps so the users can interact with models from wherever they need to.

Creating models is the main purpose of these solutions. There are a variety of standards for process models, the most popular being BPMN. It should be used in leading technology rather than proprietary notations. It leads to easier understanding for all of the users that are involved, such as the end customer, team member or project manager.

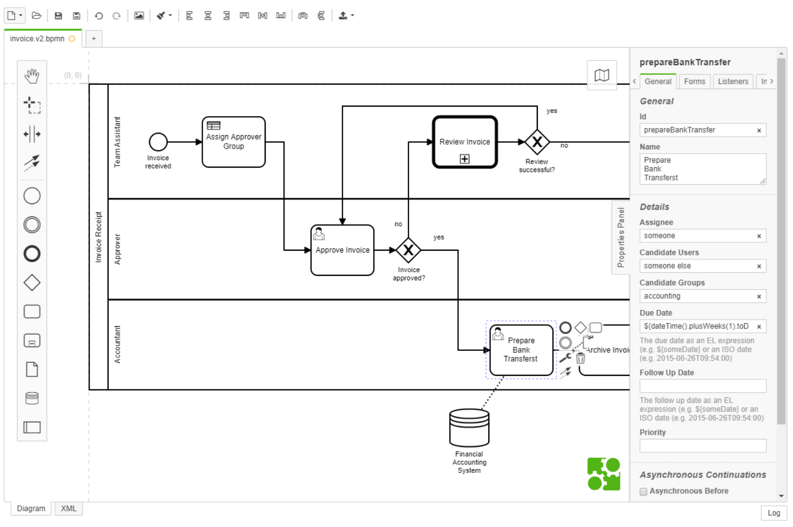
Performance also plays a key role. The software performs differently depending on the number of users at one moment. Scalability is important during the growth of a business.

1.2 Comparison of BPMS and Low-code Approaches

The main difference between traditional BPM and new Low-Code approaches is a complexity of the setup process. Any BPMS requires coding skills (mostly in Java) to develop and integrate automated business processes. As a result, this means that the company needs an outsourced team to set up the environment or assemble its own team. All of this coding and related concept preparation can take months to prepare and many more months to reach positive numbers of ROI. Also, maintenance can be as expensive as the initial deployment.

On the other hand, there is the Low-code approach. It can not be said that these are two different things since a low-code platform is more likely to be an extension of a traditional BPM solution. It provides prepared templates, connectors, and even database schemas that allow users to easily create automated tasks, test them, and deploy them in a matter of days/weeks. There is very little need for the coding and creation process of UI. This is done via drag and drop designers, as you can see in table 1.1. Due to its simplicity, the low-code solution cannot be used for more complex problems. An example of these advantages and a simple way for how to deploy an app can be seen in the pictures 1.2.

More differences can be seen in the table 1.2 and they can have a major role in the decision-making process for choosing a solution for a company.



prepareBankTransfer

General | Forms | Listeners | In

General

Id: prepareBankTransfer

Name: Prepare Bank Transferst

Details

Assignee: someone

Candidate Users: someone else

Candidate Groups: accounting

Due Date: ${dateTime().plusWeeks(1).toD

The due date as an EL expression (e.g. ${someDate} or an ISO date (e.g. 2015-06-26T09:54:00)

Follow Up Date:

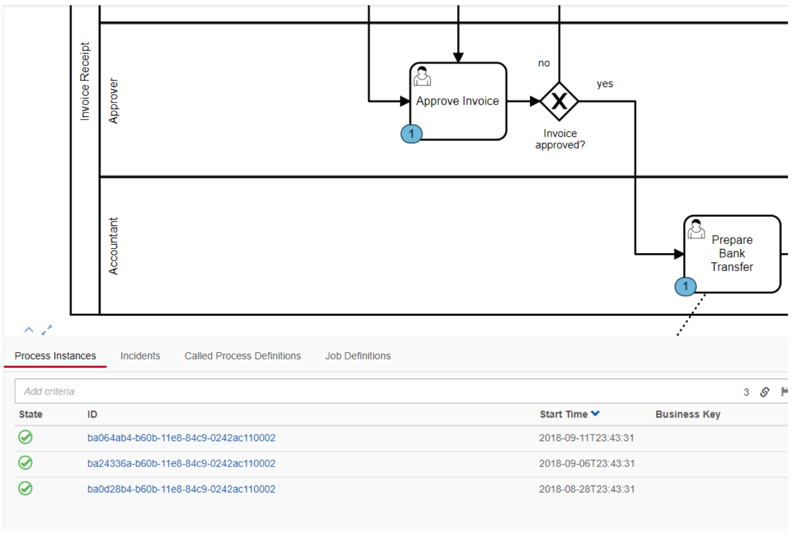
The follow up date as an EL expression (e.g. ${someDate} or an ISO date (e.g. 2015-06-26T09:54:00)

Priority:

Asynchronous Continuations

☐ Asynchronous Before

(a) Camunda Modeler for creating and editing process models



State	ID	Start Time	Business Key
✓	ba064ab4-b60b-11e8-84c9-0242ac110002	2018-09-11T23:43:31	
✓	ba24336a-b60b-11e8-84c9-0242ac110002	2018-09-06T23:43:31	
✓	ba0d28b4-b60b-11e8-84c9-0242ac110002	2018-08-28T23:43:31	

(b) Managing instances of a process in Camunda Cockpit

Figure 1.1: Example of creating and managing business processes in Camunda BPM platform [1]

The Low-code software is perfect for straightforward processes that require less customization. Due to its simplicity, the low-code software offers a lower learning curve that can speed up the entire setup process. As a result, this is a preferred option for smaller (and maybe medium) sized enterprises.

Another factor from the comparison table 1.2 is pricing. The low-code software does not require hiring external specialists or building an in-house team. It is the same with the initial costs. The low-code platform can decrease

Table 1.1: Comparison of tools

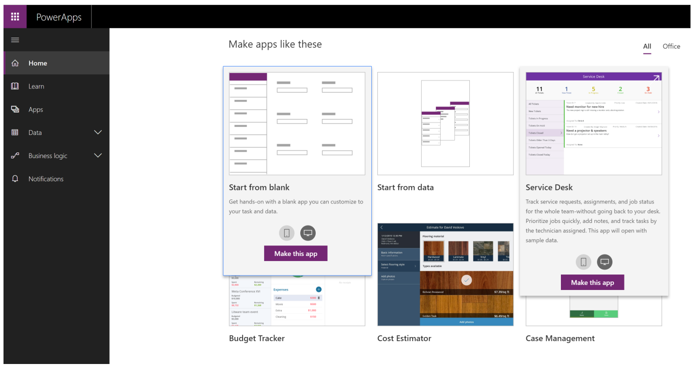
	BPMS	**Low-Code**
Designer	BPMN modeler, but no designer for creating UI	Doesn't support BPMN (or any other standard notations - only proprietary ones), basic flowcharts, drag and drop designer for UI
Connectors	Most common products or services are provided or has to be customly developed with SDK	Only common products or services are provived, lack of these connectors
Monitoring	Tool for monitoring running instaces of processes, also analyzing bottlenecks or potential money saving	Just limited monitoring

these costs for small and medium companies as well.

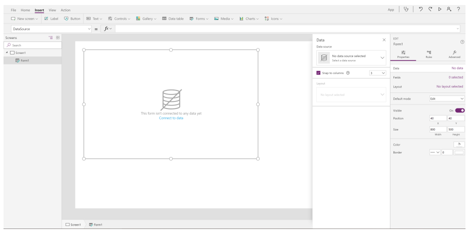
However, simplicity is not always an advantage for BPM platforms. The main reason is that developers are not able to customize forms or automate complicated workflows that are based on requirements from c-level executives or business analysts.

Table 1.2: Comparison of key features and capabilities

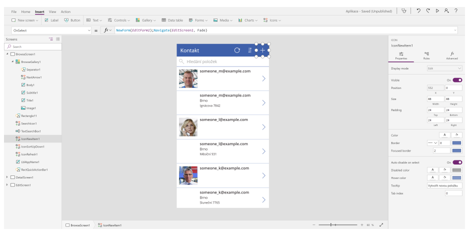
	BPMS	**Low-Code**
Process setup	Has to be programmed and takes more resources to make a later changes	Drag and drop designer, pre-built app templates and connectors
Setup time	In range of couple of months with specialists	In range of couple of minutes/hours
Deployment type	Mostly on-premise	Mostly cloud-base
Integration	Some already provide connectors or add-ons	Many connectors for 3rd party services/databases, data-driven development
Pricing	Thousands of dollars (depends on the size of the project) for deployment and monthly for maintenance or help-desk	Average $10/month/user
IT skills	IT specialists to setup and training for end-users	Minimal or no coding skills needed


PowerApps
Home
Learn
Apps
Data
Business logic
Notifications
Make apps like these
All
Office
Start from blank
Get hands-on with a blank app you can customize to your task and data.
Make this app
Start from data
Service Desk
Track service requests, assignments, and job status for the whole team-without going back to your desk. Prioritize jobs quickly, add notes, and track tasks by the technician assigned. This app will open with sample data.
Make this app
Budget Tracker
Cost Estimator
Case Management

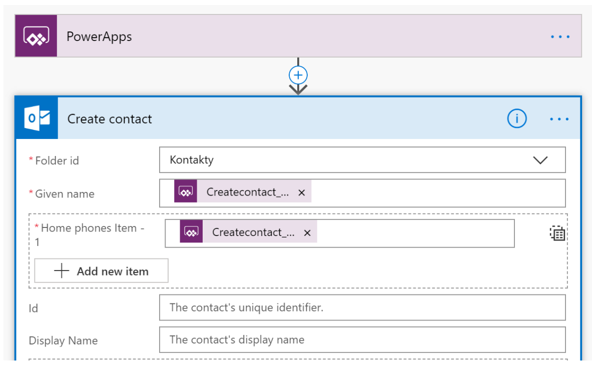
(a) Gallery of prebuilt apps



(b) Blank app ready to connect to data sources



(c) Creating form and gallery of contacts



* Folder id	Kontakty
* Given name	Createcontact_...
* Home phones Item - 1	Createcontact_...
Add new item	
Id	The contact's unique identifier.
Display Name	The contact's display name

(d) Using Microsoft Flow to cover integration of services

Figure 1.2: Example of creating app with PowerApps [2]

1.3 BPM Solutions

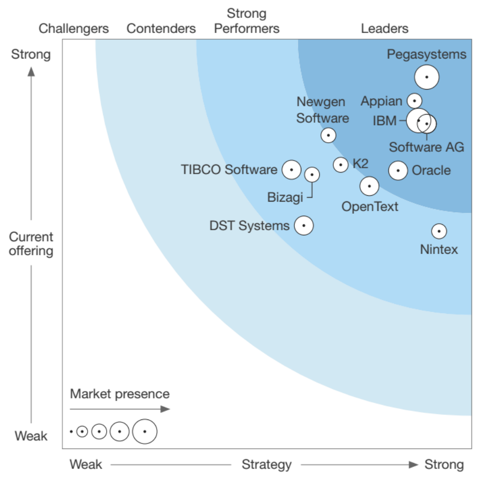
The main advantages were mentioned in comparison section 1.2 but there are also other reasons why companies should decide to use a BPM solution. The most important factor is the efficiency of business processes: it can help with productivity thanks to parallel processing or the removal of bottlenecks. The monitoring feature improves the process of finding unnecessary steps or the potential risk of fraudulent activity.

Nowadays, companies have to be able to manage new regulations and market demands. BPM systems can ensure that regulatory requirements will be implemented properly and quickly. Furthermore, that prevents associated fines.

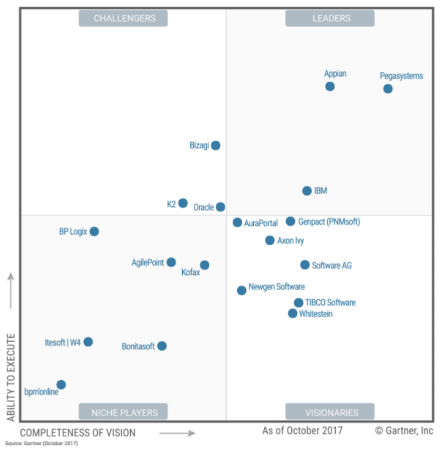
The requirements also come from the business structure and its continuous change (growth or adaptation on new markets). All these aspects are tightly intertwined with the versioning of preexisting business process models (more information about how BPMS' handle versioning can be found in chapter 2).

The main leaders usually offer their own solution for dealing with business processes. For companies that wonder about adopting this, it is hard to decide what solution to get and from which vendor. These BPMS' can look similar because most of them are based on visual modeling. However, there are differences between them that can have more weight in decision making.

As it can be seen from the graphs in 1.3, the research from Gartner [6] is based on strategy, vision, and offers. Forrester [7] even includes the market presence, which is also important for talking about leaders. A majority of solutions are Java-based.



(a) Forrester Wave: Digital Process Automation Software, Q3 2017 [7]



(b) Gartner's Magic Quadrant for BPMS', October 2017 [6]

Figure 1.3: Comparison of two leading research graphs about BPM suites

1.3.1 Pega Platform

The first one is Pega, from Pegasystems [8], with the highest ranking from both sets of research. One of the reasons is that Pegasystems have an extensive history of developing CRM applications that offer support services for sales, marketing, and customer service.

The platform offers low-code development to accelerate building process models through visual tools. Continuous modification and improvement of processes are allowed thanks to Agile Studio and their DevOps solution, which are part of the Pega platform. The software supports robotic automation, which means that it is possible to create bots to take care of repetitive communication with a customer.

1.3.2 IBM

IBM has a long history of developing enterprise solutions and continues to show this in the case of their BPMS. It is widely used by many companies and is available in on-premises and cloud configurations.

The solution [9] is designed to support mobile devices, featuring case management capabilities. Its tooling is ready to design, execute, monitor, and optimize business processes. A crucial highlight should be aimed at the capability to be deployed on a single process server or in a federated topology. Thanks to the Process Portal, users are provided with a collaborative work environment. As usual, the pricing is available upon request.

1.3.3 Appian

Appian only offers a BPM solution [10] with the support of case management that allows building process-centric and case-centric applications.

Their solution includes every key factor that was mentioned in chapter 1.1. For instance, the key factor - cloud/on-premise - is covered with both of these options. Users are able to manage the creation of mobile or web applications with the low-code approach. Appian BPMS provide real-time monitoring and management tools to keep track of running processes. It is possible to reassign tasks and track the progress of the tasks in different parts of the company. Team members can define critical policies and procedures.

1.3.4 Bizagi

One of the main leaders is Bizagi, whose solution [11] is divided into three components - Bizagi Engine, Bizagi Studio, Bizagi Modeler. These products are also available in the cloud version. The Studio and Modeler are available for free so the process can be ready before even buying the Engine.

The solution is similar in the key capabilities to its alternatives. It provides users with process modeling, automation, low-code development, and case management.

The learning path is supported by the extensive catalog of courses. Another useful catalog is full of templates of process models, process apps, and widgets to achieve fast results. When compared to the other solutions, Bizagi supports JEE and .NET platforms and all server operating systems.

1.3.5 AuraPortal

Another solution that is slightly dependent on Microsoft technology is AuraPortal [12]. Its main difference from the others is that it uses Microsoft SharePoint as its document manager.

It also offers no-code development, which can be a dealbreaker for smaller companies with only a basic IT department.

AuraPortal consists of Core BPM and optional modules. These modules are notable benefits that are not seen in their competitors' solutions. The main module is Deep Business Intelligence and allows the insight of business processes with a connection to Microsoft Power BI. The next modules provide a management of business rules, having three synchronized environments (development, production, and testing), and external users workflow that allows users to send tasks to each other without requiring an intermediary employee.

1.3.6 K2

K2 [13] offers a nearly identical portfolio of components and capabilities when compared to its competitors. It also provides the low-code approach to create apps without any need to have coding skills, or just creating simple forms. For its costumers, K2 solution has pre-build applications to take a shortcut to process automation.

The basic product K2 runs on the cloud, however, there are two other products (K2 Five and K2 Blackpearl) that are ready for on-premise solutions.

1.3.7 Oracle

Oracle, one of the largest software makers in the world, also has its own BPM solution (Oracle BPM Suite) [14]. This also indicates that the pricing is not published and final, it depends on an individual plan made for user's needs.

The solution offers basic features as the web-based Process Composer to create models, Web Form designer, application adapters for web services or databases or Desktop BPM studio.

When compared to the other vendors, Oracle BPM also includes capturing business goals, objectives, and strategies. This allows users to get reports, such as KPI Heat Map, which helps with the evaluation of the quality of implemented business models.

The entire solution is provided as a cloud service.

1.4 Low-code Platforms

Low-code platforms are based on products and services (mostly not provided on-premise), which are intended for application development that employs a visual and declarative approach instead of programming. They are supposed to have a big advantage in low or almost no cost for training time and building apps right away. Some vendors offer low-code or no-code solutions but there is still a small gap that has to be filled by coding to integrate access to older applications for reporting or customized user interface.

As it can be seen in graph 1.4, made by Forrester [15] and Gartner [16], there are again some companies (Appian, Bizagi, K2) that were already mentioned in chapter 1.3 about BPM systems. Their products also offer low-code development, but there is no need to list them again.

Low-code platforms are often cloud-based and correspond with issues for integration in an internal business network. It does not affect web services, which are public, so these platforms can connect. Support for integration with databases is done by JDBC connectors. However, it is not usually possible to be directly connected.

For example, the PowerApps platform 1.4.3 has its own data storage (Common Data Service) or it offers many connectors for widely used database systems. These systems can be reached by PoweApps proprietary gateway, which has to be installed on the server system that has access to the given database. The gateway creates an outbound connection to Azure Service Bus, where it is possible to reach the data from the database.

Another solution that Appian uses is its own Data Store. This allows them to insert, update, query, and delete data in the format needed by the applications without writing structured queries. Then, tables and keys have to be created in the business database. It is recommended to perform bulk operations for retrieving external data.

1.4.1 Salesforce

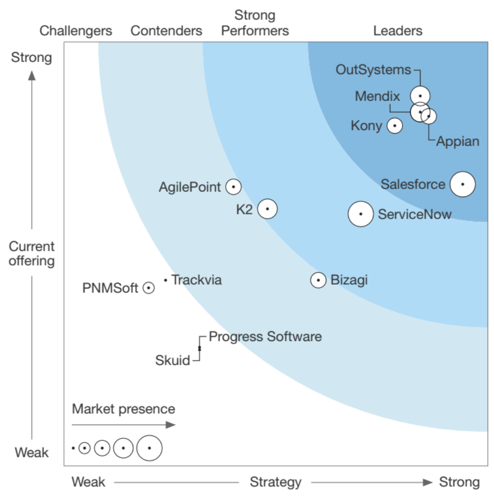
Salesforce is a well-known cloud-based customer service platform that is used widely around the world. They offer solutions for customer relationship management, sales, and marketing. It was a logical move to extend this portfolio [17].

It has been offered for years and the product developed into the cutting-edge solution on the market with features to build code-less apps. To achieve this, the platform provides the component marketplace and visual interfaces to build apps or create business models.

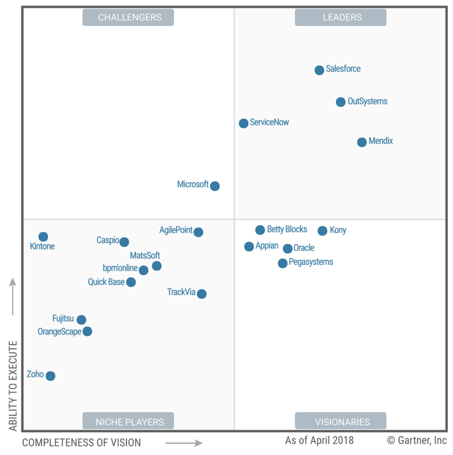
The pricing plans are almost the highest when compared to competitors. The prices range from $25 to $100 per user per month (or $4,000+ per company) and vary in the number of available objects that can be used to build apps.

1.4.2 OutSystems

This product is mainly enterprise-focused for large companies. That is why OutSystems [18] can give their platform with limited features (single private



(a) Forrester Wave: Low-Code Development Platforms for AD&D Pros, Q4 2017



(b) Magic Quadrant for Enterprise High-Productivity Application Platform as a Service, April 2018

Figure 1.4: Comparison of two leading research graphs about low-code platforms

development environment, limited scalability only-cloud) for free to organizations with less than 100 users. On the other hand, the pricing tiers start at $2,100.

An unique feature is publishing the apps to iOS App Store and Google Play store.

The OutSystems solution is not fully cloud-based and it is necessary to install a desktop integrated development environment. However, it offers the component marketplace or connectors like the competition.

1.4.3 Microsoft

Microsoft, as a major vendor, also decided to have its own solution for the low-code platform. In comparison to the other large competitors, it doesn't offer a full-fledged BPM suite (their BizTalk product can be used for the orchestration of business services and supporting their processes) but does have two other products that can help to develop apps and support basic processes.

The first product for building apps is called PowerApps [2], which starts at $ 7 per user per month or can be included in Microsoft Office 365 and Dynamics 365. The second product for covering task automation (known as IFTTT - If This Then That) is Microsoft Flow. Both of these products are developed to cooperate and offer connectors right in the visual interface.

The PowerApps provide access to Microsoft's Common Data Service, which is a file storage and database to support newly created apps. It also offers connectors for regular databases or services as well.

Many others also include pre-built apps to get started right away. The entire interface is made with a look and feel of other Office applications.

1.4.4 Mendix

Mendix [19] offers the same common features as its competitors. However, it is more IT-focused, includes testing, and also has customization and options for analytics.

The pricing starts from 10 users (less is for free with the Community edition). The enterprise editions start at $ 1,875 per month for a single app and goes up to $ 7,825 with a wider range of features as an advanced deployment, continuous integration, private cloud deployment, scaling, or apps on-premise.

Mendix makes users create a design first and continue with creating a model, app logic, and workflows.

1.5 Open-source

BPM software is solely for a commercial sphere because it aims at cost saving, greater efficiency and better organization of work, productivity, and process performance. This will project in its prices. For companies with a large revenue, it could be a profitable investment. However, for small and medium enterprises the BPMS' are basically unavailable. These companies have a choice in using open-source software as it can lower the initial costs. However, these solutions mostly do not include as many features as the commercial BPMS'. Open-source does not mean free software since the costs are connected with the support and maintenance for the products.

1.5.1 Activiti

Activiti is a Java-based open-source BPMN engine [20]. It is owned by the company Alfresco. They also have their enterprise BPM solution and the Activiti engine that is used as the core.

The Activiti is designed for Spring Cloud, docker, and kubernetes. It includes all the basic tools to create models and workflows or manage processes. All the tools are web-based with a designer, which is an Eclipse plug-in for developing the workflows.

1.5.2 Camunda

Camunda is an open-source BPM platform for workflow and business process automation [1].

The models can be created with BPMN in the modeler. Camunda also supports other standard definitions, such as CMMN or DMN. Users can manage reports via Comunda Optimize and create dashboards for business monitoring or searching for bottlenecks of models.

As with most of the solutions, Camunda is written in Java. This means that it supports development in Java EE and extending functionality as well as REST API. Additionally, it can be added to Java applications as a library.

1.5.3 Coverage of BPMN 2.0 Standard

Most of the BPM solutions cover a major part of elements defined by the standard BPMN 2.0. This also applies on the Camunda BPM platform, which declares a coverage of most elements of BPMN 2.0 [21]. The coverage is one of the reasons why Camunda will be used is for this thesis as the main BPMS to implement and test evolvability of business process models.

1.6 Vendor Lock-in

During circumstances in which low-code platforms (1.4) are used, the dependency on a vendor is almost 100% and results in the issue of vendor lock-in. The vendor lock-in means that the source code is closed and the vendor does not provide any API or SDK to enhance the product functionality. This problem is still manageable, and for most smaller companies with less complex processes, is relatively safe.

However, a more significant threat than the extensibility is the future of the vendor. The company that delivers the product as is can be acquired or become insolvent. This problem becomes more critical as deeper automation processes are involved in the core business.

The vendor lock-in is not only a problem for low-code platforms but also for traditional BPM solutions. The company that developed the solution or its partners are the only ones who have developers with the required skills. The lack of knowledge for the end users is caused by a closed system and, for example, expensive training and certifications. These subjects offer custom deployment with configuration or support. However, the risk is still present in the case of implementation of the BPM suite as a core business system.

The way that a company can go is an indoor solution. This approach has more risks than the previous ready-to-go solutions. The main risk is that the development team has to be experienced enough to produce at least the same quality product as the normal off-the-shelf application. Another risk could be insecurity, time which was consumed for developing, or the financial aspect.

The most common cloud vendor lock-in issues are these: (based on the analysis in Journal of Cloud Computing [22])

Data lock-in This is a problem of low-code platforms and BPM solutions offered on the cloud (also known as SaaS). Data is stored only at one cloud provider.

Data breach This problem is also related to the platforms mentioned above. The company using the service cannot ensure the security of their data.

Proprietary data format Hard to go to another vendor because of the format of data, which is specific for the current vendor.

Lack of integration Using another vendors‘ services with the current one is hard or impossible because there is no prepared interface.

An example of the vendor lock-in is in forms of other services from the same vendor. In this case, a company will buy a traditional BPM suite. It is working properly the entire time and the amount of business processes are increasing as the company grows. The board of the company wants to connect the BPM system with other (new or older) systems to be more flexible in growing or to automate more tasks to save financial resources.

This thesis will open the opportunity for a better understanding of the process engine and more extensive variety of modifications. The Camunda BPM solution (1.5.2) is a proper product for the study of evolvability of business process models.

Chapter 2

Handeling of Business Process Changes

Whenever a company decides for a BPMS, it is because their processes can no longer continue being handled by a human. These business processes become more complex and can run long term. In a single instance of a process, the participants have to deliver documents or have to approve some steps of the process; having the ability to significantly impact the length of the instance run. The instance can last for weeks or months. The requirements for the process model can vary throughout time and have to be changed, even if some instances are still running. If the change is crucial and fundamental, the requirement can be more complicated for the already running instances and they have to be migrated to the newest version.

2.1 Business Process Reengineering

For the continuously evolving business environment, the competitiveness of individual companies is crucial for survival. Improving processes is an essential part of maintaining or improving market position. Such improvements can be faster customer clearance, approval projects, or product manufacturing. From the process view, an optimal path is investigated by considering time or resources consumed in that process.

The definition of BPR also includes these three points [4]:

Aiming at processes
Focusing on basic business processes that affect customers and not purely internal processes. Recognition of these processes is crucial and identified as critical.

Radical change
A distinctive target for radical change is to improve competitiveness or

to dominate the market. The result is to switch from the old functional grouping of departments to the process-driven departments.

Dramatic improvement
BPR has a radical effect over small gradual improvements. BPR is performed on core business processes, which are mandatory in improving competitiveness. First, the strategic objectives are set. Next, the process is transformed to achieve these goals.

The BPR can be divided into three types:

Mild - the executive staff works similarly but with better support

Medium - some activities are added at the operational level

Heavy - complete reorganization

When optimizing processes, it is good to deal with various aspects like searching for places that break the optimum path, changing the organizational structure or dividing competencies. It is also helpful to outsource the process or reduce bureaucracy in the process. Some businesses also incorporate new technologies and practices.

The most common weaknesses of processes whose removal leads to improved process performance are stated in table 2.1.

2.2 Low-code Versioning

In section 1.4, the most popular low-code platforms were mentioned. Low-code platforms can be seen as BPM systems even though they do not necessarily use the standard BPM notation (some of them have their own proprietary notations or just support simple flowcharts). In this section, there are slightly different platforms - the first one is more simple and the second one is more advanced. It is important to include these platforms as well because they can provide valuable insight into the concept of versioning process models.

2.2.1 Versioning in Microsoft PowerApps

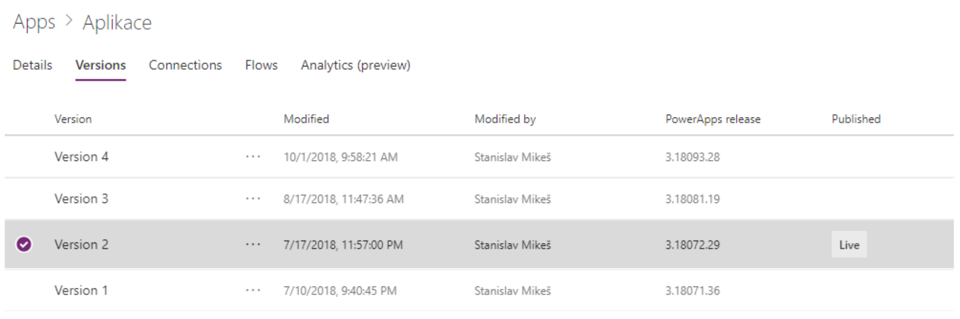
The PowerApps solution from Microsoft can be used as the first example for low-code platforms. The concept of saving changes of created apps is simple and straightforward. In picture 2.1 is the version number, date, author and flag if the version is published. This table does mot describe versions of process models created with Microsoft Flow tool.

There can be only one running version of the app, which can interact with one version of Flow model implicitly. However, from the view of the Flow environment, it is possible to create a conditional calling of other flows, which

Table 2.1: Weaknesses in processes [4]

Type	**Description**
Spacial	Different locations of tasks in one process. It is time-consuming and hard to manage.
Time	Tasks are poorly coordinated on the time scale.
Organizational	Tasks are executed by different departments. The communication can slow down or the priority of the particular process is not on the same level throughout the different departments.
Informational	Information can get missing through a process. The data is not compatible or accessible.
Descriptional	Tasks are not described properly and it can cause a misunderstanding.
Applicational	This problem is connected with the informational point. Different applications are used in a company where data transformation is used, resulting in possible delays in time.
Sequence	Some processes are executed sequentially, even though they can be executed parallellly.
Loops	Complex processes often include many conditions and it can cause them to return to previous tasks.

can be seen as different versions. The main issue comes with already running processes, which have to be finished or canceled before changing the flow.



	Version		Modified	Modified by	PowerApps release	Published
	Version 4	···	10/1/2018, 9:58:21 AM	Stanislav Mikeš	3.18093.28	
	Version 3	···	8/17/2018, 11:47:36 AM	Stanislav Mikeš	3.18081.19	
✓	Version 2	···	7/17/2018, 11:57:00 PM	Stanislav Mikeš	3.18072.29	Live
	Version 1	···	7/10/2018, 9:40:45 PM	Stanislav Mikeš	3.18071.36	

Figure 2.1: PoweApps - table of versions of an app

2.2.2 Versioning in Mendix

The second example is a solution called Mendix, which was mentioned in section 1.4.4. Projects made by this platform consist of many parts (java codes, web pages, web service, connectors, images, flows etc.). The flow is basically a type of process model with two options of complexity - microflow and nanoflow.

Microflow Represents the model which can perform actions like creating and updating objects, showing pages, or making choices as it is showed in picture 2.2. Microflow supports error handling with predefined error variables for inspection.

Nanoflow It is prepared for more simple tasks with no expectation of failure. The error handling is not supported and also nanoflow does not support transactions. If an error occurs, it will not roll back to any previous changes. The actions are directly executed without waiting for results.

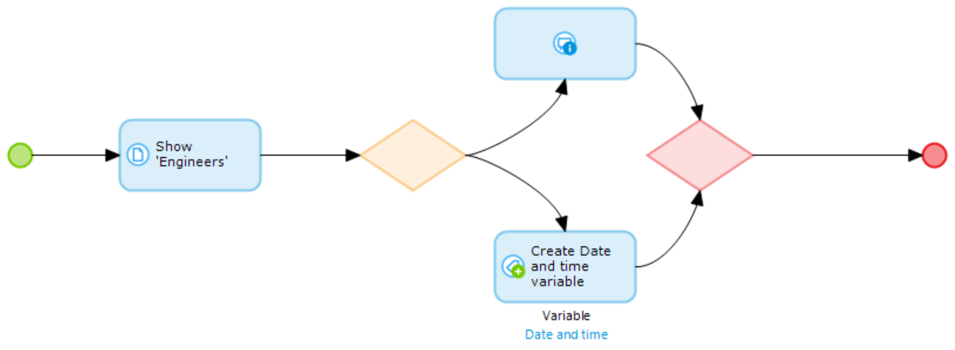


Figure 2.2: Business process model as a microflow in Mendix Modeler

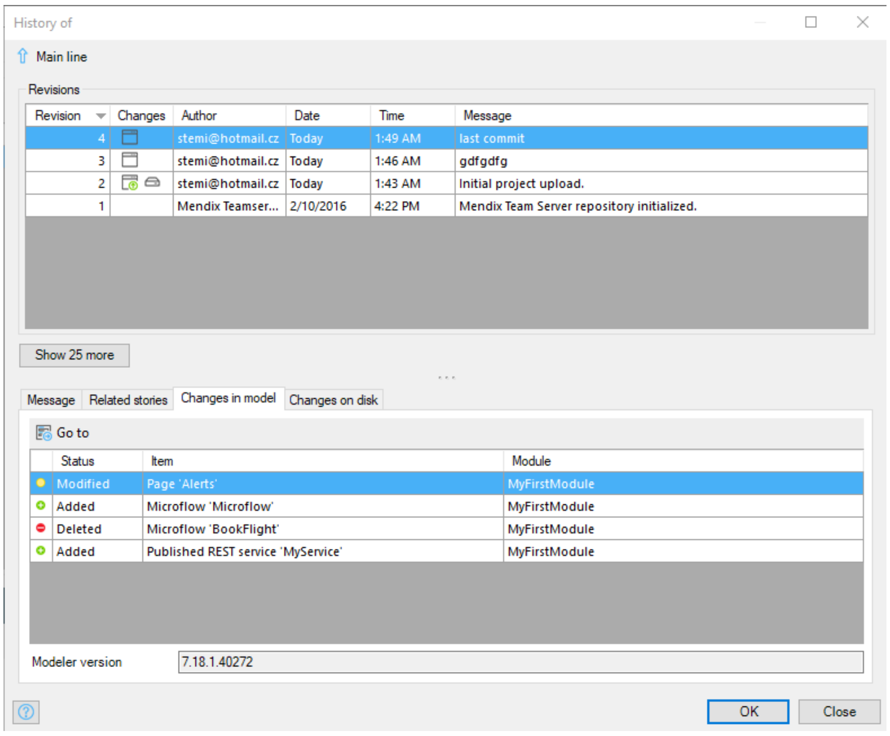
The whole process of creating the project is done in Mendix Modeler - making flows, services, pages or handling versioning. This platform is more advanced with versioning than PowerApps.

The versioning is supported by Mendix Team Server [23], which works as the central place for all projects. There are repositories for every application with process models.

The repository stores a specific version of the project, which is called revision. This revision includes process models, java codes, custom widgets, etc. The number of revision incrementally increases every time when someone commits a set of changes to the repository. This way, the project can be identified at a certain point in time.

Every team member can download the most recent revision and has their own copy of the project, called the working copy. It is the lowest level of versioning that is done locally so that the changes that are made by the person are not yet reflected on the server. Once the changes are ready, the person can commit the project to the server, which creates a new revision. The history of revisions with other info is in picture 2.3.

Every single part of the project is tracked and marked with a proper graphical status that describes if the part was added, modified, moved, deleted or if it is in conflict. As an example, the mark can be seen in the history window in picture 2.3.



Revision	Changes	Author	Date	Time	Message
4		stemi@hotmail.cz	Today	1:49 AM	last commit
3		stemi@hotmail.cz	Today	1:46 AM	gdfgdfg
2		stemi@hotmail.cz	Today	1:43 AM	Initial project upload.
1		Mendix Teamser...	2/10/2016	4:22 PM	Mendix Team Server repository initialized.

Status	Item	Module
Modified	Page 'Alerts'	MyFirstModule
Added	Microflow 'Microflow'	MyFirstModule
Deleted	Microflow 'BookFlight'	MyFirstModule
Added	Published REST service 'MyService'	MyFirstModule

Modeler version: 7.18.1.40272

Figure 2.3: History of revisions in Mendix Modeler

There is also an update request that retrieves the latest changes from the repository. This process does not only mean updating some parts of the project because the individual will get the newest revision. During the update process, the new working copy and local changes are compared and combined. Consequently, two types of conflicts can arise. The first one is a document conflict, which means that two people changed the properties of the same data view (when the changes are too close to each other). The second one is a project conflict when someone deleted a flow and the other person was changing it or just moving files to different locations in the project tree.

The Mendix versioning system is more useful thanks to the development lines (in picture 2.4). The repository can contain a number of these lines. This concept is well known in the GIT world. There can be more development lines (branches) for developing new features or the main one for fixing bugs.

Whenever changes are made in the main (for example, in production) development line, it is possible to merge these changes to the feature branch. This concept is useful to have bug fixes in every branch.

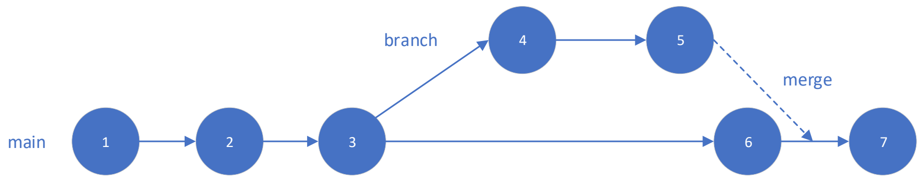


Figure 2.4: Braching and merging concept in Mendix

Overall, there is no option for how to migrate running instances of process models.

2.3 BPM Versioning

Unlike low-code platforms, BPM systems have a unified pattern (which can vary in some details) to handle the versioning of process models and the migration of their running instances.

2.3.1 Common Pattern

The versioning of process models is done by simply uploading a new process model to the system under a single process that is incrementally marked. The new instance of the process is created according to the latest valid version of the process model. Running instances, which were created according to previous versions of the process model, must stay active.

There are two scenarios of how the system can handle the versioning. The first scenario is simple. The instance continues uninterrupted, according to the previous version of the model that it was created from. The second is more advanced and based on migrating instances into the latest version of the model process. The migration means that the unfinished part of the running instance of the process will continue on the basis of the new version of the model.

Table 2.2 represents the definition of the process of versioning model. There is the definition with its three different versions, including dates of creation. This table is used in picture 2.5, which describes the process the versioning along with the process of creating the instances out of the model templates (definitions).

Without migrating The first instance is created based on version 1. Later on, the second instance is created and both of them will finish independently of new templates. Version 2 is still defined while the second instance is running and from now on, new instances will be created based on it. The same will happen to the last instance, based on version 3.

Table 2.2: Example of versioning

Name	Version	Date
A	1	01/10
A	2	08/10
A	3	12/10

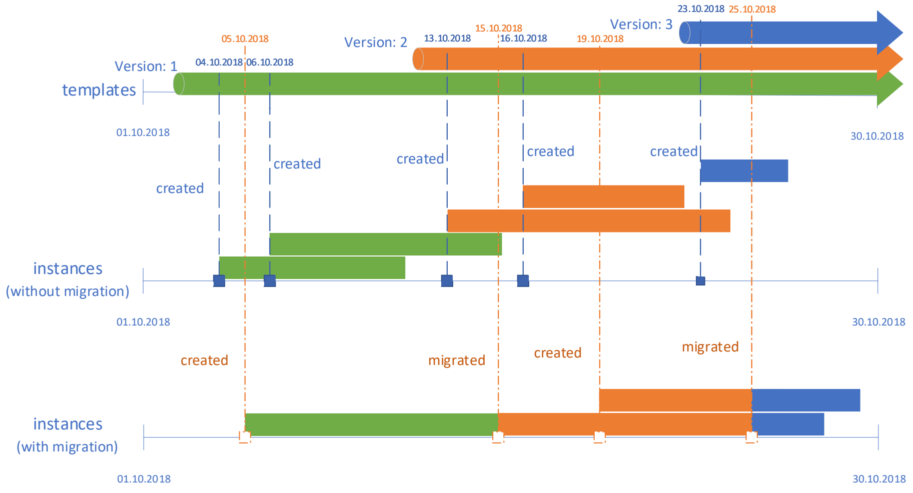


Figure 2.5: Visualisation of running processes from different definitions

This scenario is the basic one where attention is not focused on migrating already running instances.

With migrating The first instance is created from version 1 of the definition. After the second version is published, the running instances are required to migrate. The first running instance is being migrated and, later on, a new instance is created based on version 2. After version 3 is published, both running instances are migrated to the latest definition.

As previously mentioned, this pattern is the standard for most BPM systems but it can be different in notation and also in the number of extra features that are offered for the process of migration.

2.3.2 Versioning in Appin

The solution from Appian, which was mentioned in section 1.3.3, offers versioning by creating multiple versions of the same process model [24]. Every single version has a different, unique, number. This unique number is saved to the property pm!version for each process that starts on it. The version

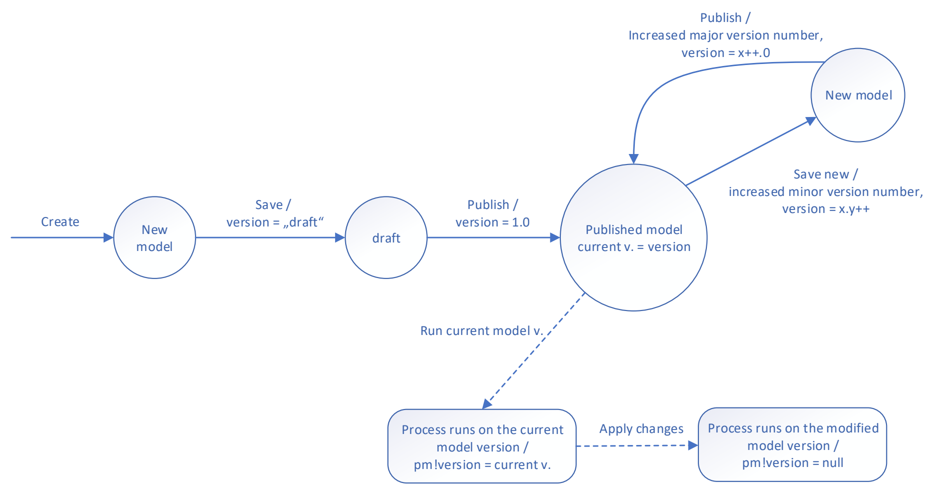


Figure 2.6: Process of versioning models and using them for new business processes in Appian

number consists of a major version number and a minor version number. The versioning is based on the process of increasing these version numbers.

When a new process model is saved, it is represented as a draft (version number is also 'draft'). After publishing the process model, its version is set to 1.0 (major version = 1, minor version = 0). Whenever a process started with this model, its property pm!version is set to 1.0. Modifying the model of the running process will cause the process to be run on the modified model and the process's pm!version is set to null. The process model 1.0 is modified and, after saving changes, a new process model with version 1.1 is created (but new processes will run on 1.0). After publishing model 1.1, process model 2.0 is created and new processes will run on the latest model. The visual description of the versioning process is in picture 2.6.

2.3.2.1 Sub-Process Model

It is important to cover process models that can also be run as a sub-process of another process model. The element is called Sub-Process activity and it allows one to launch a sub-process from a parent process. The Appian system defines two sub-process types:

Asynchronous The running parent process launches an asynchronous type of sub-process and the parent will continue once the sub-process starts. The variables can be sent to the child but not back to the parent.

Synchronous When the parent process launches the synchronous type of sub-process, it waits until the child completes. The variables can be

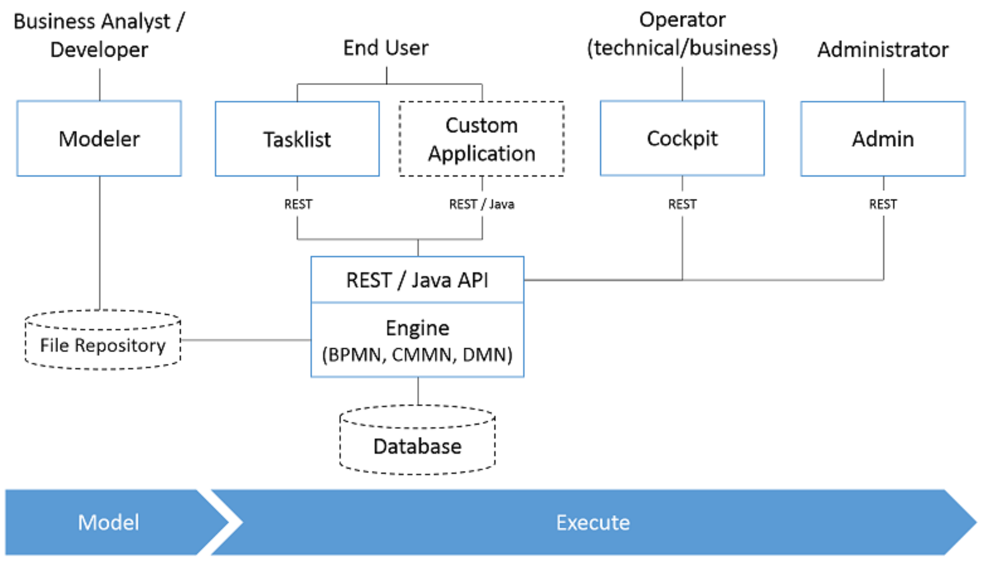


Figure 2.7: Camunda architecture overview [3]

sent back and forth between parent and child processes.

The rules for creating and running a sub-process are similar to the basic ones that were formerly mentioned. A sub-process which started via the Sub-Process Activity is created based on the latest published version. The latest version is also used if the model was republished after the parent process started. This also applies when the process model is deleted and the latest published version that existed before removing is used.

2.4 Camunda Analysis

As previously stated, Camunda is a Java-based solution with an open-source idea. Thanks to that, the engine is more transparent than closed systems and it is possible to analyze how the system architecture is done.

The picture 2.7 shows the essential parts and user roles that manage these parts.

Modeler A desktop modeling tool for BPMN 2.0 (also CMMN 1.1 and DMN 1.1), where the functionality can be improved with plugins.

Tasklist A web application (image 2.8a) that allows working on User Tasks without any coding. It supports Embedded Forms for customization as well. It enables users to start a process, create a task, or claim another user.

Cockpit A web application for monitoring and operations (image 2.8b). Users can access deployed BPMN processes and check their running instances. In the enterprise bundle, it provides options for deployment or access to statistics of performance. All of that, and more, can be added to Cockpit via plugins.

Admin A web application that provides a typical admin site to manage users, groups, their authorizations, and so on.

Engine The core of the entire Camunda solution is a java library, which is responsible for executing processes based on BPMN, CMMN, and DMN.

Java API Camunda module that provides programming model integration CDI for Java dependency injection. This module allows programmers to add Java code behind tasks in BPMN.

REST API The process engine provides access to the entire relevant functionality through REST API calls, which can be used for remote applications.

2.4.1 Process Analysis and Bottlenecks

Another reason for why changes happen to process models is to improve the efficiency of the process. With a growing amount of running instances, the risk of a bottleneck may occur. The bottleneck term is derived from the real world neck from a bottle, where water can not flow fast enough. The bottleneck in the business process world is used the same way, where running instances of the process can slow down at some spot of the definition.

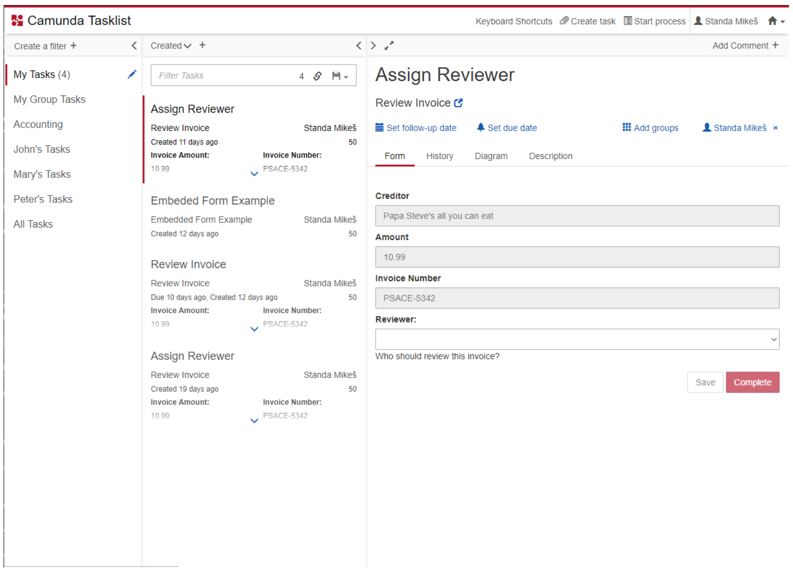
The bottlenecks can have many causes. In the corporate bottlenecks research [25] was found the most common reasons for bottlenecks and their negative effects.

Most common reasons:

- confusing processes
- colleagues fail to meet deadlines
- cumbersome approval process
- stuck on one high level decision maker

Negative effects:

- low morale
- missed deadlines
- unsatisfied customers
- lost revenue
- regulatory problems



Camunda Tasklist

Keyboard Shortcuts | Create task | Start process | Standa Mikeš

Create a filter + | Created + | Add Comment +

My Tasks (4)
My Group Tasks
Accounting
John's Tasks
Mary's Tasks
Peter's Tasks
All Tasks

Filter Tasks 4

Assign Reviewer
Review Invoice — Standa Mikeš
Created 11 days ago — 50
Invoice Amount: 10.99 — Invoice Number: PSACE-5342

Embeded Form Example
Embedded Form Example — Standa Mikeš
Created 12 days ago — 50

Review Invoice
Review Invoice — Standa Mikeš
Due 10 days ago, Created 12 days ago — 50
Invoice Amount: 10.99 — Invoice Number: PSACE-5342

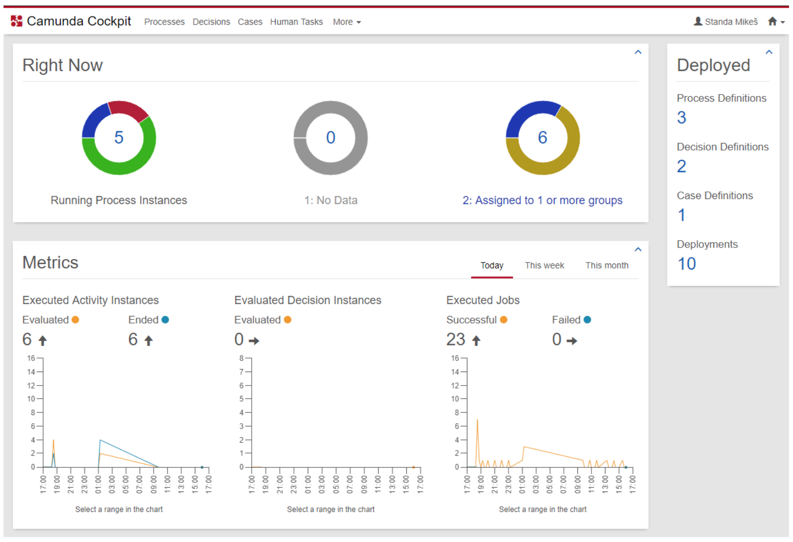
Assign Reviewer
Review Invoice — Standa Mikeš
Created 19 days ago — 50
Invoice Amount: 10.99 — Invoice Number: PSACE-5342

Assign Reviewer
Review Invoice
Set follow-up date | Set due date | Add groups | Standa Mikeš
Form | History | Diagram | Description

Creditor: Papa Steve's all you can eat
Amount: 10.99
Invoice Number: PSACE-5342
Reviewer:
Who should review this invoice?

Save | Complete

(a) Tasklist screen



(b) Cockpit main screen

Figure 2.8: Camunda administration screens

2.4.1.1 History and Heat Maps

One efficient way to discover these issues is using tools as a historical view and a heat map in BPM. Camunda offers a tool for exploring the history of instances. The heat map can be overlayed over the BPM historical view 2.9. The heat map displays elements with different colors based on a number of nodes that flow through the process model.

As seen in picture 2.9, the user tasks are the most overloaded parts of the

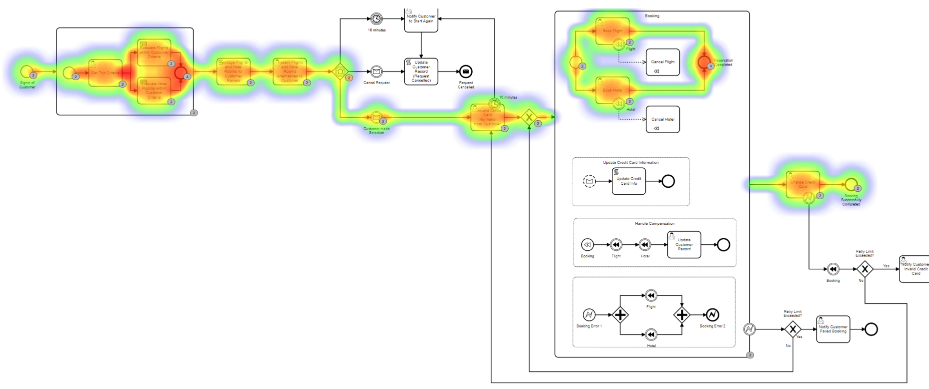


Figure 2.9: Camunda heat map tool

process. For example, that is the spot where an analyst should go deeper into the problem and a newer version of the definition will be released.

User and external task instances can be evaluated by their parameters like the assignee, owner, creation date, completion date, the duration, due date, follow up date, amount of retries, or priority.

2.4.2 Process Versioning

Versioning of process models is considerably challenging. A business process can run for a longer period of time. Processes, by nature, may last for weeks or months, depending on internal rules and laws. Furthermore, the duration depends on the number of stakeholders, who need to pass documents or only to get their approval to continue.

However, the process model must reflect the changes of the rules, laws, or the idea behind the process. There can be running instances that have to be adequately handled during deployment of a new version.

Camunda process engine stores a version and ID of process models into its database. The process engine handles process versioning by three simple rules.

- When deployment of a new model of the same process is done, a new record is added to the database with a higher version number.
- Already running instances, which were created with the older version, will continue without any changes.
- New process instances will be created based on the new model version - unless explicitly specified. The determination of which version will be used depends on two properties.
  - Creating a model by the **key** starts an instance from the latest deployed model version, which is defined by the key.

  – Creating a model by the **ID** starts an instance from the model that is stored in the database under this ID. This allows the process to start with a specific version.

The entire process of versioning and creating instances is captured in picture 2.10.

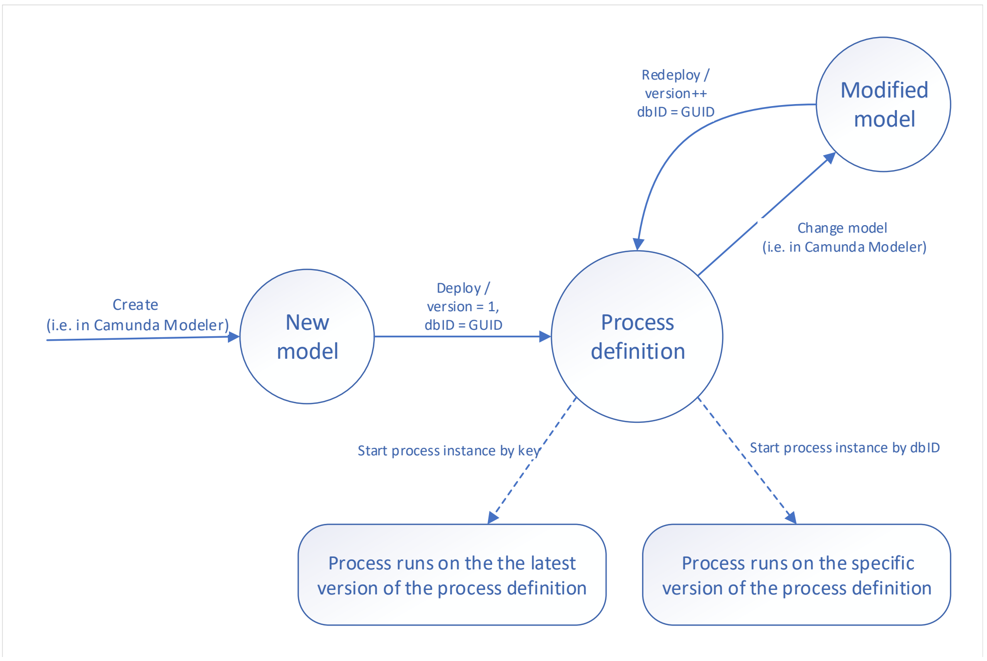

Redeploy /
version++
dbID = GUID
Modified model
Change model
(i.e. in Camunda Modeler)
Create
(i.e. in Camunda Modeler)
New model
Deploy /
version = 1,
dbID = GUID
Process definition
Start process instance by key
Start process instance by dbID
Process runs on the the latest version of the process definition
Process runs on the specific version of the process definition

Figure 2.10: Process of versioning models and using them to creat new process instances in Camunda

The rules, which were mentioned above, raised a question about migrating already running processes to the latest model versions. Camunda, compared to Appian, supports this scenario with its Process Instance Migration tool (section 2.4.4), which is available in the Enterprise version. However, thanks to the open REST API, it is possible to create ones own tool for migration with only open-sourced Camunda core bundle.

2.4.2.1 Subprocess Versioning

BPMN 2.0 defines two elements which can be used for calling a process from a parent process. The child process, from a conceptual point of view, is called a subprocess. A Call Activity runs an external process definition. However, the subprocess is embedded within the original process definition. The main idea of having two different types of subprocesses is to be able to reuse a process definition, which can be called from multiple processes. When a parent process executes the call activity, a new process instance is created. The parent process instance waits until the subprocess is completed and continues the original process afterward.

The regular embedded subprocess will be versioned with the parent process. However, there must be particular properties defined within the Call Activity, so that the engine can recognize which version is supposed to be used. The properties are:

latest uses the latest version of the process definition.

deployment uses the process definitions that were deployed together, allowing versioning of both (parent and children) processes.

version uses the specific version, which is hard coded into the parent process definition.

versionTag uses the specific version with the defined versionTag, which hard coded.

2.4.3 Process Instance Migration

As it was already mentioned in section 2.4.2 and captured in picture 2.10, the process instances are created based on the latest version of the process definition (unless otherwise specified).

When some process instances have been running for awhile and a new process definition is deployed, none are affected. All the instances based on previously deployed versions will be finished.

However, it is likely that already running instances will have to be completed according to the new definition. In this case, the migration of running instances has to be done. The migration is supported by a migration plan, which describes how to migrate instances from one process definition to another. The migration does not have to be done only from the previous version to the latest but also the other way or else it is possible to downgrade the process definition too.

A migration plan consists of a set of migration instructions that describes a mapping of activities between the two process definitions. An instruction manages that an instance of the source activity is migrated into an instance of the target activity. The main condition for having the migration plan valid is when all the instructions cover all active source activities.

It is mandatory to map activities that are semantically equal or it will not be possible to migrate the activity. There is a way to manage cases where the migration has to be done, even if the activities are not semantically equivalent. The process instance can be modified before the migration so that the particular activity instance is canceled and started again after the migration. This case can also be applied when the activity changes its variables, that have to be known for the rest of the process, so a user or some external process has to go through the task again after migration to have a valid running process instance.

2.4.4 Enterprise UI Approach

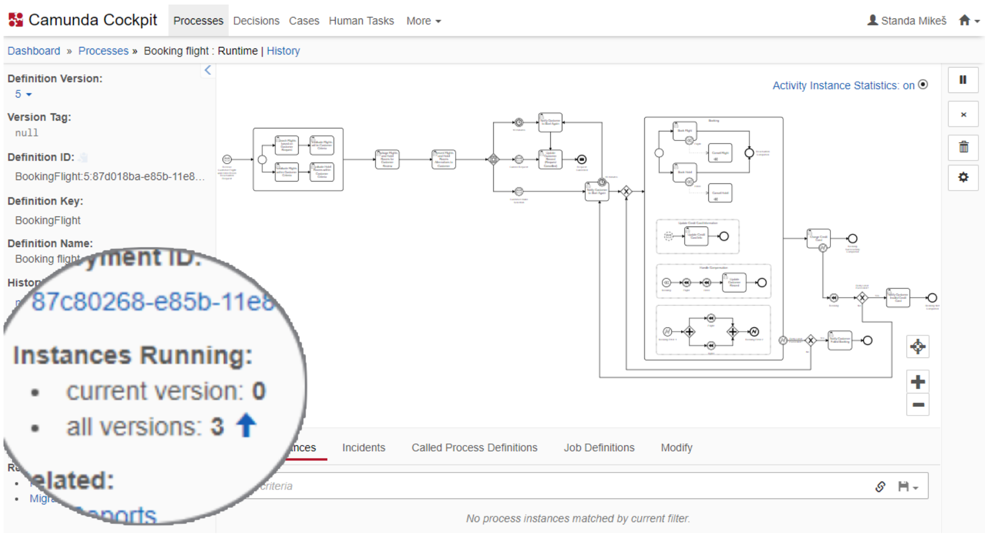
The tool from the Camunda team covers the entire instance migration process. Although it is Camunda opensource and free solution, this tool is delivered as an enterprise feature in the bundle called Enterprise Edition (with many other useful features but they are not related to this thesis). The tool is provided as an extension of the user interface in the Camunda Cockpit.

This tool can help with a better presentation of how the process of migration is being done.

The following four steps are taken to complete the migration based on the Camunda documentation [26]:

1. Create a mapping of the source activities to the target activities
2. Select running instance to migrate
3. Confirm the migration
4. Check the results and eventual errors

The environment can recognize that the newer version was deployed and it it will give an option of migration to the user, as seen in picture 2.11.


Camunda Cockpit
Processes
Decisions
Cases
Human Tasks
More
Standa Mikeš
Dashboard » Processes » Booking flight : Runtime | History
Activity Instance Statistics: on
Incidents
Called Process Definitions
Job Definitions
Modify
criteria

Definition Version:
5

Version Tag:
null

Definition ID:
BookingFlight:5:87d018ba-e85b-11e8...

Definition Key:
BookingFlight

Definition Name:
Booking flight

Deployment ID:
87c80268-e85b-11e8

Instances Running:
- current version: 0
- all versions: 3

Related:
- Migration
- Reports

No process instances matched by current filter.

Figure 2.11: Option for process instance migration in Camunda Cockpit

On the next page, the tool will leave up to choose the source process definition version and the target one, as seen in picture 2.12.

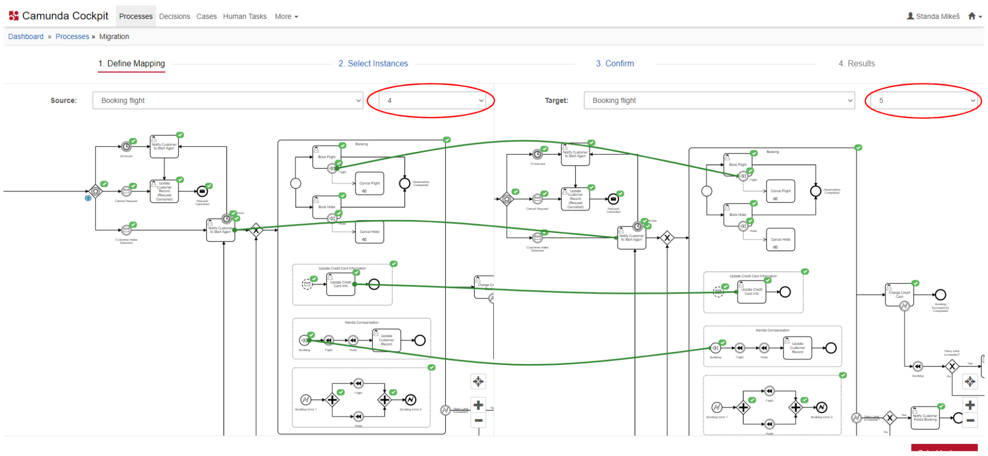


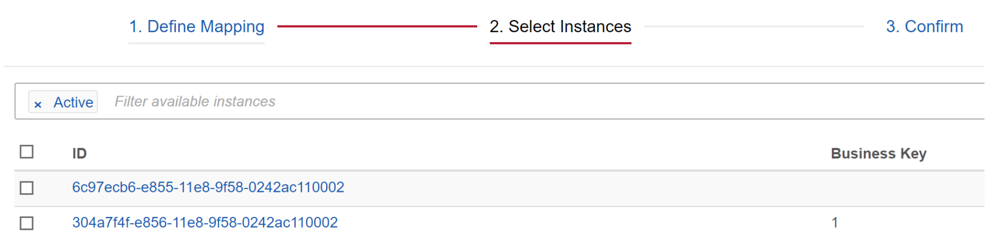
Figure 2.12: Choise of the source and target versions and highlighted ways of activity mapping in Camunda Chockpit

After the selection, the source and target diagrams are displayed with a generated migration plan for the user (picture 2.12). However, this plan does not have to reflect the plan of the user, so it is possible to change the mapping there. The mapping has some limitations that are described in chapter 3.4. During this step, the plan is being validated whenever the mapping gets changed.

Occasionally, if the running activity instances cannot be mapped to any target activity, they will be marked, and the process of migration will fail. The users can map the activities by themselves with the drag and drop tool. The tool will help by highlighting the semantically similar activities.

The window with displayed diagrams can be synchronized for movement so that it can help with orientation with large models. It also provides a tooltip for showing the user an error or the particular mapping issue.

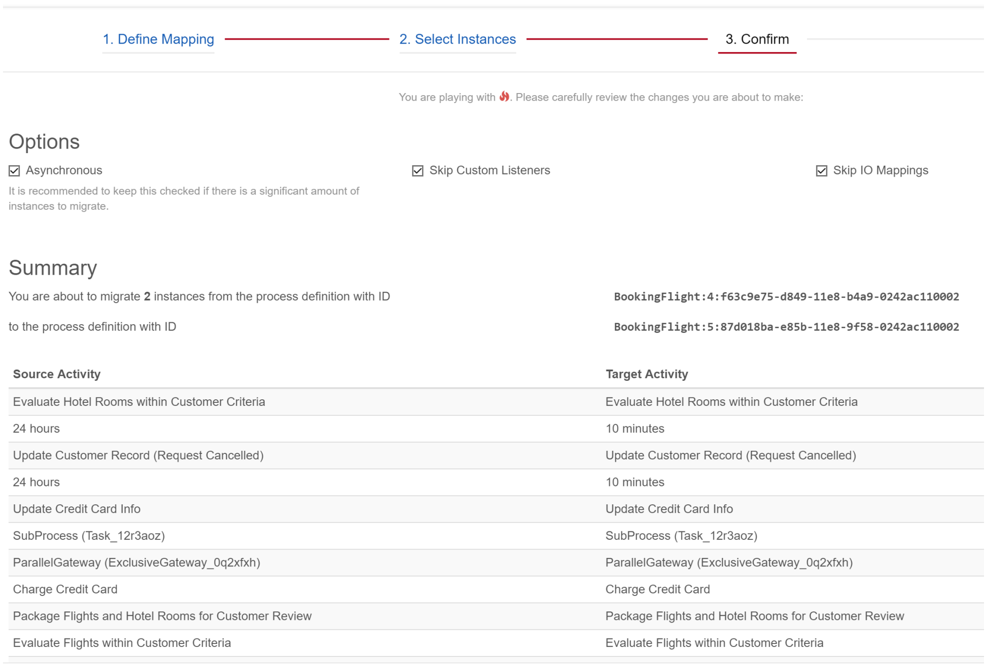
After creating the migration plan, there is another step to select the wanted process instances (or all), which will be the plan applied (in picture 2.13). These instances can be filtered by their properties (i.e., business key).



	ID	Business Key
☐	6c97ecb6-e855-11e8-9f58-0242ac110002	
☐	304a7f4f-e856-11e8-9f58-0242ac110002	1

Figure 2.13: Selecting instances which will be migrated in Camunda Cockpit

At the last step, the user is asked for permission to execute the migration. There are also options for if the migration will be executed asynchronously in a batch, or whether custom listeners and IO mappings should be skipped (picture 2.14). The asynchronous migration is done one by one process instance and it is useful for a larger amount of these instances. The result of the migration will be displayed with related errors.



You are playing with 🔥. Please carefully review the changes you are about to make:

Options

☑ Asynchronous

It is recommended to keep this checked if there is a significant amount of instances to migrate.

☑ Skip Custom Listeners

☑ Skip IO Mappings

Summary

You are about to migrate **2** instances from the process definition with ID `BookingFlight:4:f63c9e75-d849-11e8-b4a9-0242ac110002`

to the process definition with ID `BookingFlight:5:87d018ba-e85b-11e8-9f58-0242ac110002`

Source Activity	Target Activity
Evaluate Hotel Rooms within Customer Criteria	Evaluate Hotel Rooms within Customer Criteria
24 hours	10 minutes
Update Customer Record (Request Cancelled)	Update Customer Record (Request Cancelled)
24 hours	10 minutes
Update Credit Card Info	Update Credit Card Info
SubProcess (Task_12r3aoz)	SubProcess (Task_12r3aoz)
ParallelGateway (ExclusiveGateway_0q2xfxh)	ParallelGateway (ExclusiveGateway_0q2xfxh)
Charge Credit Card	Charge Credit Card
Package Flights and Hotel Rooms for Customer Review	Package Flights and Hotel Rooms for Customer Review
Evaluate Flights within Customer Criteria	Evaluate Flights within Customer Criteria

Figure 2.14: Confirmation window with options and summary of the migration in Camunda Cockpit

2.4.5 RestAPI Approach / SDK

Since the previous section 2.4.4 is describing the UI approach, which Camunda supports only in Enterprise edition, Camunda offers Rest API (and their Java SDK) that covers the all relevant interfaces of the engine. It is possible to use a different engine than the Camunda's default one. Typical usage of the query for tasks is */engine/task*, but for another engine, it is */engine/name_of_the_engine/task*.

The SDK way uses the same interface but it only wraps the API calling by java methods, so it is easier to use in any already running application. The Camunda is Java-based, which means that all the Service tasks in process models can be implemented as a java class. The entire project with these classes will be deployed together as a .war file.

The migration API consists of three methods which cover the process of migration.

2.4.5.1 Initialization

As indicated previously, the migration is initialized by generating a plan with migration instructions. The instructions describe a mapping of process model elements between two versions of process definitions.

REST API method
: POST `/migration/generate`

Parameters
: The REST request has to be parametrized by a JSON body. The parameters refer to the source definition ID and the target definition ID. This method also sets if event triggers in the process instances will be updated or not during the migration.

Result
: The basic mapping will be applied, and the response body (again, in JSON format) will return generated instructions of equal elements. The instructions include a note about updating event triggers, which depends on the particular type of the element. The instruction is not a 1×1 mapping, but it can be $N \times M$ by nature; however, that is an unwanted behavior.

2.4.5.2 Validation

To make sure that the migration plan is correct and will not affect running instances, it has to be validated. Validation is done two times during the migration process.

Creation time validation validates a migration plan before the migration for static aspects without executing the plan.

REST API method
: POST `/migration/validate`

Request
: The result body of the previous step of initialization is used as the request body for this method

Result
: The result body contains information about correct mapping and occurred errors if any instruction is not valid.

After the plan is validated, an execution time validation will occur before the plan is applied to every instance. All leaf activities must have a migration instruction.

2.4.5.3 Execution

Whenever the migration plan is valid, the execution can start. This process applies the migration plan to selected instances with the execution time validation.

REST API method
: POST `/migration/execute`

Request
: The body contains the entire migration plan (mentioned above), selected process instance (their IDs) and other additional parameters. These parameters define if execution listeners and if I/O mappings should be invoked during the migration.

This method is described with more details in the section about the enterprise approach 2.4.4.

Chapter 3

Process Model Evolvability

The meaning of the term evolvability is the ability to undergo adaptation, development, or evolutionary change [27].

For running process instances that do not require a change in their definition and therefore can continue without interruption, there is no need to address the issue of evolvability. Conversely, for running instances that require a change of definition, evolvability is essential and it is necessary to go through the migration process.

This need for evolvability is due to a legal regulation that is crucial to this process and there is no exception for already running cases. These instances can also be affected by a business change. Another reason for changing the definition can be the analysis of a process that contains errors or bottlenecks and consumes an unnecessarily large amount of resources (financial or human). Also, there may be processes that are carried out within B2B where processes could be changed on the other side, or this party no longer exists and must be replaced by another supplier/customer and corresponding processes.

It is also important to decide which instances to migrate. It is not always necessary to migrate all instances because some may already be in a stage that is not affected by the change of the process definition and migration only means creating potential problems and errors. Another reason to consider migrating a given instance is to have initialized variables that were created by the system or user that could be lost and, for example, the process would behave differently from an external perspective than was previously designed.

BPM migration also involves migrating and validating other components in which the model is dependent on and using them. These include DMN, scripts, external tasks, simple expressions, or subprocesses. In some cases, a collision of transferable BPM elements may also occur. These issues are dealt with in this chapter, which seeks to provide solutions for them. This chapter uses Camunda solution for creating examples.

3.1 BPMN Elements

BPMN is the leading notation for defining models of business processes. The standard of BPMN which will be used in this chapter is based on the official normative documents [28] created by OMG (Object Management Group).

The basic categories of elements are:

- Flow Objects
- Data
- Connecting Objects
- Swimlanes
- Artifacts

The main graphical elements that define the behavior of a process and will be affected by the changes in the model are Flow Objects.

These Flow Objects consists of:

- Events
- Activities
- Gateways

Since this thesis uses Camunda as a representant of a BPM system, it is necessary to check the elements that are supported and implemented by the engine. The system supports most of the BPM elements [21] except for Parallel Event-Based gateway, Multiple and Multiple Parallel events, Undefined and Receive (instantiated) Tasks. The Data elements (Data Objects, Data Inputs, Data Outputs, Data Stores) are also not supported by the engine.

Migration depends on the type of activities a process model contains, causing the migration to have different effects.

3.1.1 Scopes

For evolvability, it is important to explain what BPMN scopes means. A scope describes the context in which the execution of an Activity happens [28]. The scope consists of:

- Events ready for catching or throwing triggers
- Data Objects available (input/output)

A scope contains exactly one main flow of activities. Scopes are used to define the visibility of data objects, event resolution, and start/stop of token execution.

3.2 Basic Definition

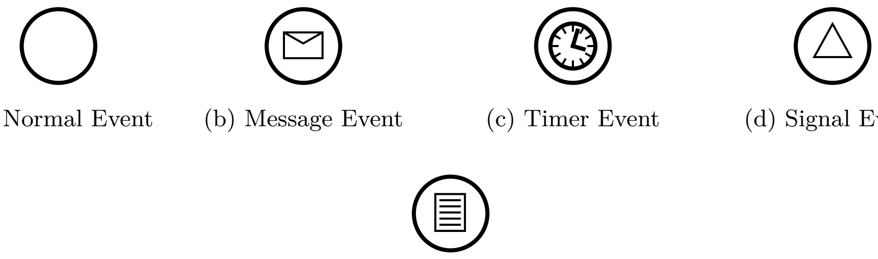
Before starting an exploration of evolvability of business process models, the basic information about BPMN elements is described in this section to continue with whole business process models. The main source of information of migration basis is from Camunda documentation [26] and OMG BPMN standard [28].

3.2.1 Events

In BPMN, there are three types of events: start, intermediate and, end events. These types can be either catching or throwing events. Intermediate events can be used as boundary events pinned on activities, where they can be interrupting or non-interrupting.

3.2.1.1 Start Event

Start event (picture 3.1) starts the entire process or subprocess. It can be migrated with persisting event trigger if the source start event is not mapped it will be removed and the new start event is initialized.



(a) Normal Event (b) Message Event (c) Timer Event (d) Signal Event

(e) Conditional Event

Figure 3.1: Graphical representations of Start Event types

3.2.1.2 End Event

End event acts the same way as the Start Event during migration. Unlike the Start Event, the End Event can be represented as Escalation, Error, or Compensation End Event. The End Event cannot have a Timer event trigger. There does not have to be a focus on updating the timer.

3.2.1.3 Boundary Event

Boundary Event (picture 3.2) caches a throwing event from the inside of the Activity in which the event is attached. Whenever the boundary event is attached to a multi-instance, all the instances are canceled. When a boundary event is triggered the execution continues through the attached sequence flow.

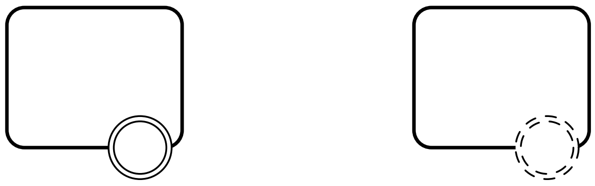

(a) Interrupting (b) Non-interrupting

Figure 3.2: Graphical representations of general Boundary Event types

3.2.1.4 Intermediate Catch Event

Intermediate Events (picture 3.3) handle triggering Events. Waiting for the occurs of the Events starts when the Intermediate Event is reached. Once it catches the Event, the token leaves the Intermediate Event by a Sequence Flow.

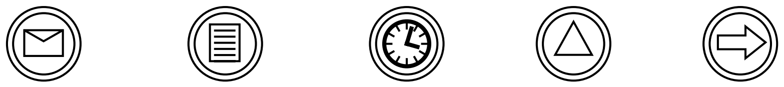

(a) Message (b) Conditional (c) Timer (d) Signal (e) Link

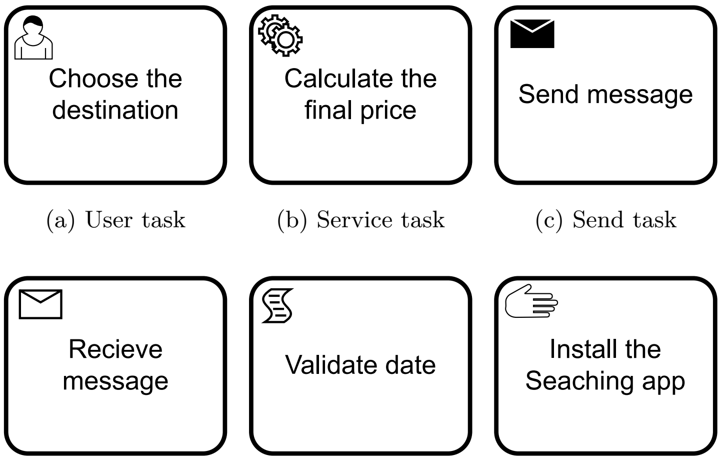
Figure 3.3: Graphical representations of Intermediate Catch Event types

3.2.1.5 Compensation Event

Compensation is used for undoing steps that were already done. These steps can be represented as activities that could cause some unwanted effects and should be reversed. If the task is running, it cannot be compensated and the cancel event has to be applied. In the case of a subprocess, the cancel event can result in compensation of the already finished tasks and the entire subprocess. Compensation is mostly performed because of some error inside the particular activity.

3.2.2 Activities

An Activity is a single work item in a process definition - it is the executable element of a BPMN process. Activity can be atomic or non-atomic (compound). Activities are divided into three types (Task, Subprocess and Call Activity). Task graphical elements are in the figure 3.4.



(a) User task (b) Service task (c) Send task

(d) Receive task (e) Script task (f) Manual task

Figure 3.4: Graphical representations of Task types

A task is an atomic Activity, which means that it cannot be broken down to a finer level of detail.

The main rule for migrating tasks is that the type of target and source activity has to match.

3.2.2.1 User Task

A User Task (picture 3.4a) is a typical task which is performed by a human performer. There can be a software interface that assists with the task.

When a user task is migrated, all properties of the task instance are preserved except the process definition ID and task definition key. The task is not reinitialized: Attributes like assignee or name do not change.

3.2.2.2 Service Task

Service Task (picture 3.4b) uses some sort of service, which can be a web service or an execution of a method. This task is used for orchestration of external tasks and it can delegate work through different implementation, which depends on the architecture of the process engine. For example for Camunda - it can be implemented as a java class or external code which is accessible via REST API; it also supports expressions or connectors.

As previously noted, a service task can be defined as an external task which means that is implemented by an application running out of the execution engine. When the service task is implemented as external, it can be mapped to other tasks which have an option of external implementation. Other tasks that have this option is Send Task (picture 3.4c). This allows mapping these tasks with each other.

3.2.2.3 Receive Task

Receive Task (picture 3.4d) is a simple task that waits for a message from an external source. The task is completed whenever it receives a message.

Since this task is the simplest one, there is a rule about specifying a message and setting if a persistent event trigger can be updated or preserved during migration. More information about the event trigger is in section about events 3.2.1.

3.2.2.4 Script Task

Script Task is defined by a script in language which is supported by the execution language (mostly different types of JavaScript). Once the task is ready to start, the engine executes the script and when the script is completed the task is completed as well.

The script task can only be mapped to the same type of task. The script task is atomic, so it will not preserve any initialized variables during the migration.

3.2.2.5 Manual Task

Manual Task (picture 3.4f) is performed without any help of a business process execution engine or any application. This type of task is used for manual work, which is considered as a unmanaged task (e.g., a technician installing a telephone at a customer location).

There is nothing to preserve during migration.

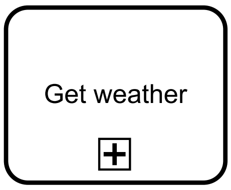
3.2.2.6 Subprocesses

Subprocess is another type of Activity. Unlike tasks, subprocesses are non-atomic. They can consist of Activities, Events, Gateways, and sequence Flows. A subprocess defines a contextual scope (including attribute visibility, transactional scope, handling of throwing events, or compensations).

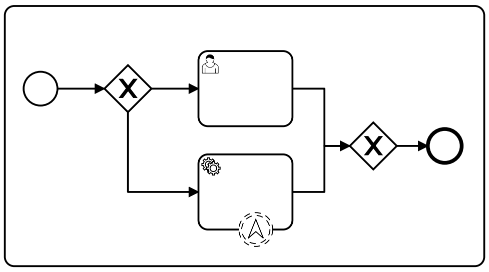
Subprocesses can be collapsed (picture 3.5a), which hides the model inside, or expanded (picture 3.5b), which reveals the model.

There can be different styles of subprocesses, some of which can be seen in picture 3.6 and described in the section below.

Migrating a subprocess (except Call Activity) applies the same rules as to the standard task - it will preserve the subprocess state such as variables. If there is no migration instruction for the subprocess, the instance is canceled. The regular subprocesses (embedded/event/transaction) can sbe mapped between each other.s

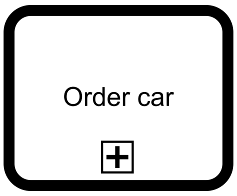


(a) Collapsed

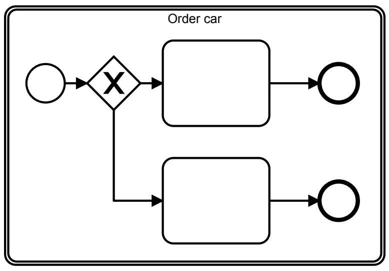


(b) Expanded

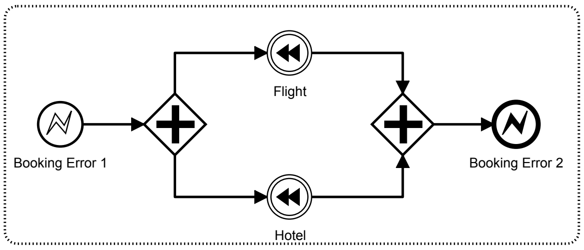
Figure 3.5: Views of subprocesses



(a) Call Activity



(b) Transaction



(c) Event Subprocess

Figure 3.6: Graphical representations of Subprocess types

Call Activity

Call Activity (picture 3.6a) is defined as an external subprocess. Call activity is a reusable process definition that can be called from many other processes. When the process arrives at the call activity element, a new instance is created out of the call activity definition. The parent process waits until the subprocess is completed and continues afterwards.

Migration of Call Activity is done separately since the definition is not a part of the parent definition.

Transaction
Transaction groups sets of activities, events, sequence flows, and gateways. The main sign of a transaction is that the entire process will fail or succeed collectively.

BPMN transactions should not be confused with a transaction known in the technical world with ACID properties. The BPMN transaction is not atomic and it can last four hours.

A transaction can end up in three different states:

- **Successful completion** - normal sequence flow leaves the subprocess.
- **Failed completion (Cancel)** - a transaction is canceled if any activity triggers the Cancel End Event. A compensation is triggered and after it is completed the token leaves the transaction through the Cancel Boundary Event.
- **Hazard** - something went wrong in the subprocess, and the cancel is not possible to execute. When a hazard happens, the activity is interrupted without any compensation and a sequence flow will leave from the Error Intermediate Event.

Event Subprocess
Event subprocess is not part of a regular flow of the parent process. There are no incoming and outgoing flows. This subprocess does not have to occur every time in the process but it is possible that it will occur multiple times. The event subprocess will start by the Start Event with a trigger. The parent process can continue when the subprocess was started but it can be interrupted, which is defined by the type of the start event in the subprocess.

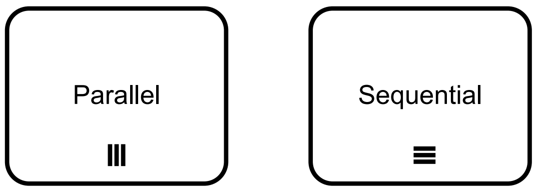
3.2.2.7 Multi-Instance

Multi-instance task offers to create multiple instances of one activity. These activities can be executed in parallel (picture 3.7a) or sequentially (picture 3.7b). Both of these types have an attribute Loop Cardinatily, which indicates how many instances will be created and executed. The multi-instance activities can be represented by a normal Task, Subprocess or Call Activity.

There are two cases when a multi-instance activity can be migrated:

- The target and source types have to be the same - parallel to parallel and sequential to sequential.
- Multi-instance activity can be mapped on any no multi-instance activity.

The previous two types of migration have different behavior.



(a) Parallel Task (b) Sequential Task

Figure 3.7: Graphical representations of Multi-instance Tasks

- Whenever a migration of the same target and source type happens then the inner state is preserved. The number of active instances are the same after the migration.
- If the target is a normal activity, the multi-instance variables are deleted (number of total instances, number of completed instances and number of active instances). However, the number of inner activity instances are preserved.

3.2.3 Gateways

Gateways are used to control token flow in a business process. If the flow has to be controlled, there is a gating mechanism that decides which way the token will go. Tokens can be merged at the gate or split apart on output, known as forking and joining. A single gateway can have multiple inputs and outputs. This behavior has to be managed during a modeling phase to decided which option will be performed.

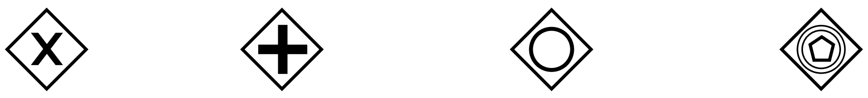
There are five types of gateways: inclusive, exclusive, parallel, event-based (figure 3.8) and complex.

The complex gateway is not supported by the representative system - Camunda BPM.

The following four types of gateways are possible to migrate. However, the common restriction is that only the target type has to be the same as the source type.

The other rules can vary whether the gateway is event-based or data-based. The event-based restriction is described in section. However, for data-based gateways are the restriction similar.

- The migration plan for the target gateway must have at least the same amount of incoming paths as the source gateway.
- A maximum of one source gateway can be mapped every same type gateway in the target definition.
- The migration has to be valid for the scope of the gateway.



(a) Exclusive Gateway (b) Parallel Gateway (c) Inclusive Gateway (d) Event-based Gateway

Figure 3.8: Graphical representations of Gateway types

3.2.3.1 Exclusive Gateway

Exclusive Gateway, also called XOR gateway (picture 3.8a), is used to model a data-based decision. The gateway is waiting for a token that will be sent through the sequence flow that meets the condition (if more sequence flows meet a condition then the first defined one is evaluated).

3.2.3.2 Parallel Gateway

Parallel gateway (picture 3.8b) is used to model concurrency in business processes. It simply offers to fork into multiple paths that are executed simultaneously and offers to join multiple incoming paths.

Forking is a straightforward process of splitting the sequence flow; the joining process requires more caution because all the previous executions have to wait for each other before continuing.

There can be multiple incoming tokens in this gateway. These tokens will join first and fork after the gateway. A parallel gateway is the only gateway which does not evaluate conditions.

3.2.3.3 Inclusive Gateway

A combination of exclusive and parallel is an inclusive gateway (picture 3.8c). This gateway allows defining conditions for outgoing tokens. Since it combines parallel as well, it can receive more than one sequence flow.

3.2.3.4 Event-based Gateway

Compared to the rest of the gateways, the event-based gateway (picture 3.8d) branches the sequence flow where the alternative paths that follow the gateways are based on events.

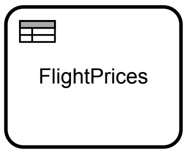
The gateway does not evaluate a data-based expression, but the event (mostly receiving a message) determines the path where the token will continue. The decision is not made by the gateway but by some other participant which has access to the data.

The event-based gateway has different rules about outgoing sequence flows. There must be two or more outgoing ones. This type of gateway is followed by Intermediate Catch Events or Receive Tasks.

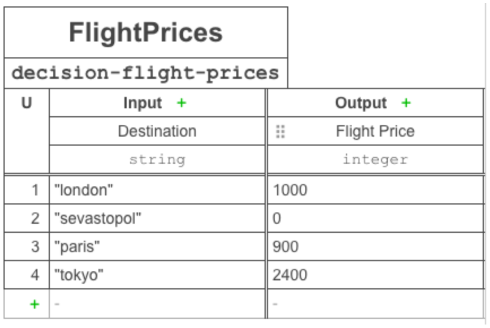
3.3 DMN

As BPMN, Decision model and notation (DMN) was declared by OMG [29] to provide the constructs that are needed to model decisions.

In picture 3.9, an example of a simple decision table can be seen. Even though the table is simple; it will not get more complicated. The main parts are inputs, outputs, and corresponding rules. A decision table also supports a declaration of the Hit Policy. If rules overlap, multiple rules can match, and a hit policy indicates how to handle the multiple matches. The Hit Policy can acquire values from a unique item to a collection.



(a) Representation in BPMN



FlightPrices

decision-flight-prices

U	Input +	Output +
	Destination	Flight Price
	string	integer
1	"london"	1000
2	"sevastopol"	0
3	"paris"	900
4	"tokyo"	2400
+	-	-

(b) Decision table

Figure 3.9: Example of a simple business rule

As it was mentioned, a decision table is not a complex element. However, for making a complex domain of decisions, Decision Requirements Graph (DRG) is used (picture 3.10). This graph shows the most important elements involved in the process and their dependencies between them. The graph consists of decisions (represented by a decision table), input data, knowledge sources, and business knowledge models (except the decisions tables Camunda DMN engine does not implement the rest of the DRG). DRG is visually represented by Decision Requirements Diagram (DRD) [29].

3.3.1 DMN Versioning

A decision model, as a part of a process model, can depend on the changes in the process model, or more, it can be the reason for the change and subsequent migration of process instances.

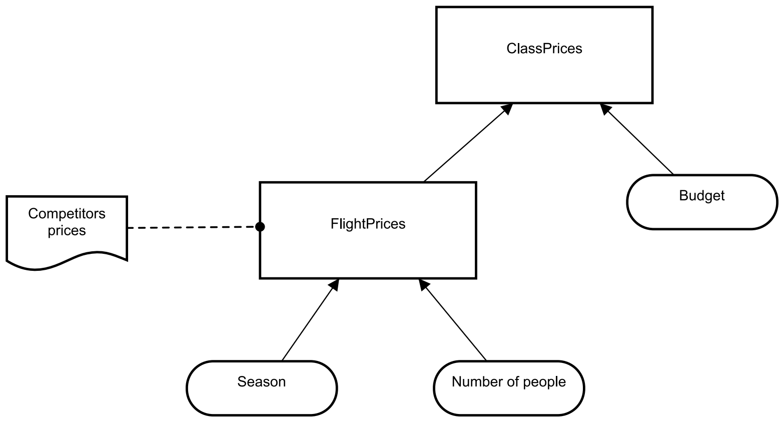


Figure 3.10: Example of a Decision Requirements Diagram

There is no possible validation of the migration plan for a decision table. A person who is responsible for versioning of a business process has to check the input and output variables of the decision table. The properties of the variables are data types, names, and hit policy. These properties have to be validated against the parent model.

The version of decision table which will be used in future instances, can be defined in the BPMN model. There are four types of bindings - deployment (a version which was deployed along with the calling case definition), latest (the latest decision definition version), version (the exact version) and versionTag (a version with the given tag - used for patching a particular definition version).

3.4 Rules of Migration

The migration of business process models has to follow rules of the individual BPMN elements, which are described in the previous sections and rules of the entire process scope, which consists of more elements that can depend on each other in different ways.

The complexity of migration may vary based on the complexity of models, scripts, linked external systems, or decision tables.

Once some running instances pass the spot of the change, it can be complicated to decide whether to migrate particular instances or not. Questions such as initializing some needed variables or checking a document may appear.

The following examples were tested with Camunda BPM Enterprise and their implementation of migration proces which is described in the section 2.4.4.

3.4.1 Minor Changes

This section deals with applying minor changes to the model where no special constructs are needed for the migration process. It covers the situations where simple elements are migrated.

The basis of minor changes is renaming elements. In this case, the migration is not challenging and, if there are no higher needs to migrate, then the running instances can remain the same.

3.4.1.1 Logical Elements

Logical elements are meant to be BPMN elements that have only descriptive features but no functional features, and they are not executable. For demonstration is used a process of payment wihtin a bank (picture 3.11).

A Text Annotation element is not mentioned in the following sections because it is not connected with conceptual modeling of the process. It is only annotation for better understanding by a human.

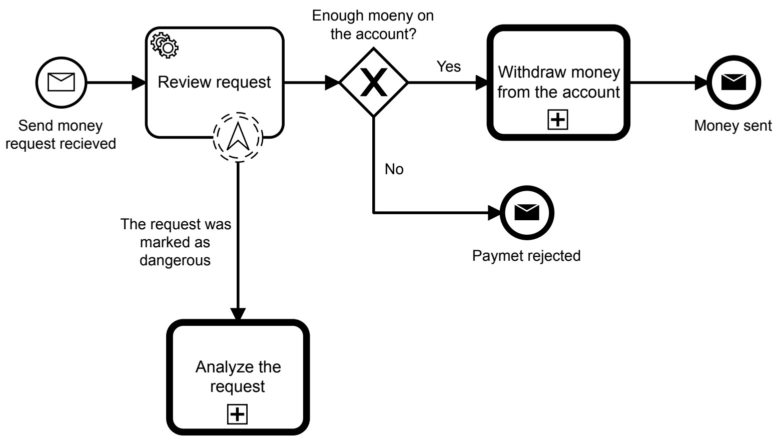

Enough moeny on the account?
Review request
Yes
Withdraw money from the account
Send money request recieved
Money sent
No
The request was marked as dangerous
Paymet rejected
Analyze the request

Figure 3.11: Payment process model without pools

Collaboration
Collaboration is required for BPMN Process Modeling Conformance, but it is not required for BPMN Process Execution Conformance [28].

Collaboration package includes Pools and Message Flows. A pool represents a participant in the process collaboration, and a message flow represents connections between pools (connects pools or objects inside of pools). A pool is not required to have process inside and can be represented as a black box (example can be seen in the picture 3.12).

Every pool is a different process and has to be initialized (with Start Event) separately. However, lanes represent only participants of one process.

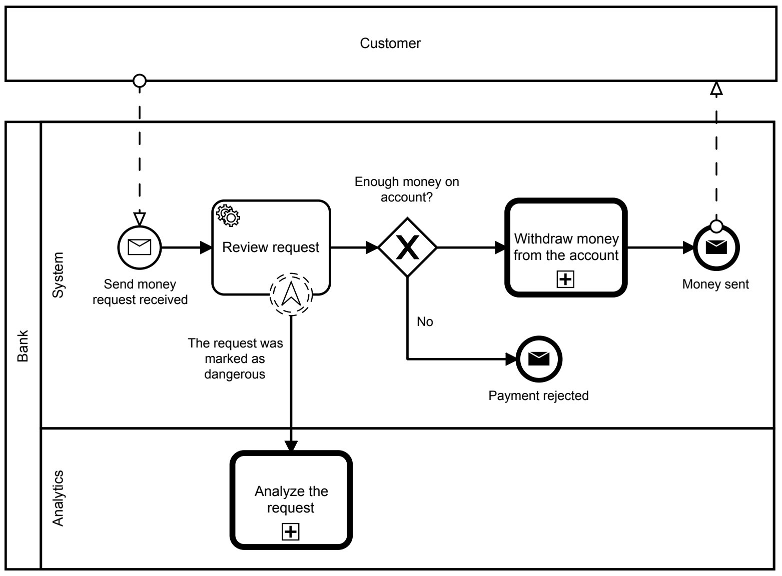


Figure 3.12: Payment process model with pools

Pools are used for modeling business parties and lanes are used for modeling departments or internal roles within the company.

By adding lanes or pools, the migration process does not have to be handled with any special structures. There is no need for migrating the running instances.

Data Object

A Data Object represents information flowing through the process, such as business documents, e-mails, or letters. The Data Object in the extended payment process model (picture 3.13).

In this case, the Data Objects acts as a descriptive element and does not affect the process.

Data Store

Data Store is an element which represents a place where data is persisted and the process can read or write information from. It can be a database, a file, or a cabinet. It keeps data beyond the lifetime of the process instance.

The Data Store representing a database of transactions is used in a new version of the payment process model in the picture 3.14.

Data Store acts like the other logical elements and it does not change the behavior of the process. The migration of running instances does not have to be performed.

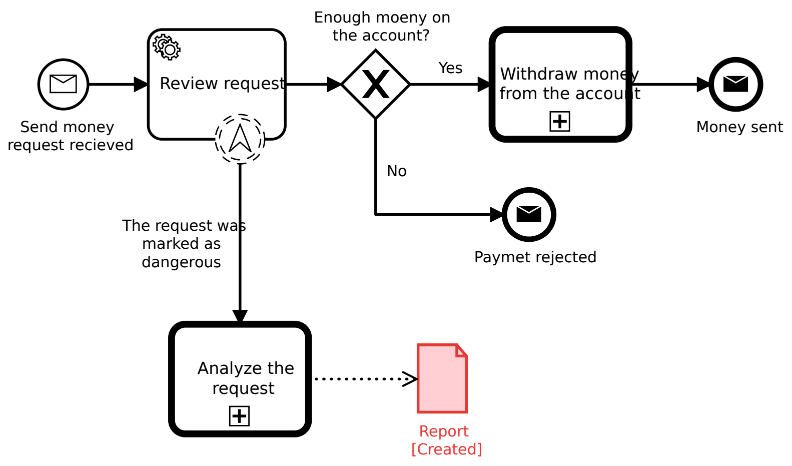

Enough moeny on the account?
Review request
Send money request recieved
Yes
Withdraw money from the account
Money sent
No
Paymet rejected
The request was marked as dangerous
Analyze the request
Report [Created]

Figure 3.13: Payment process model with Data Object

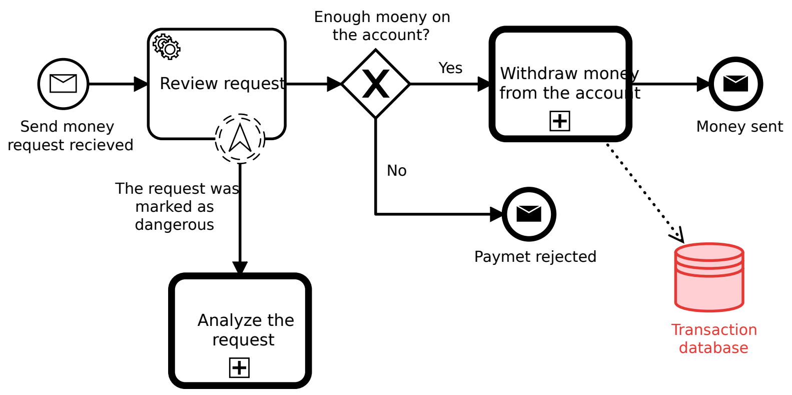

Enough moeny on the account?
Review request
Send money request recieved
Yes
Withdraw money from the account
Money sent
No
Paymet rejected
The request was marked as dangerous
Analyze the request
Transaction database

Figure 3.14: Payment process model with Data Store

3.4.1.2 Tasks

Tasks were described in the section 3.2.2. Migration of tasks have one significant limitation, and that is one type of task can be mapped only on the same type.

User Task

A user task migration has some properties to look out for.

User Tasks are supported by forms that a human can interact with. Implementation of these forms can depend on a vendor. Camunda engine uses three types of forms - generated by the engine, embedded (written in HTML + JS) or external (user is directed to another application). Attention must be focused on differencing these form types.

- Generated forms have fields that were defined already in the modeling process (id, type, and default value). Validation is also offered.

- Embedded forms are defined in HTML and behavior in JavaScript. The Camunda engine uses its meta-attributes in HTML tags to recognize a type and id of the forms fields.

- External forms act the same way as the embedded ones, but they are not part of the application.

The migration between embedded and external forms is not complicated, and a developer has to take care of right names of variables. Awareness of variables is the same in the case of a generated form as well. Once the instance's token is before the execution of the patched spot, there is a danger of using variables that were not used before migration and the previous definition does not count with it. Alternatively, two variables with the same name can be defined in different forms and have different variable types - this will cause an error.

Once the token is at the task which it is being migrated, there are properties which can not be changed. By specification, an assignee (a person how is assigned to a task) will not change during migration, and the task preserves it (a value of assignee has to be changed manually). Properties Due Date and Follow Up Date will not be updated either and have to be changed manually.

When the token is behind the migrated task, there is no conflict. There is also no need for migration.

Service Task
Service Task is described in the section 3.2.2.2. Service Task can be implemented in several ways which will cause different behavior during migration and subsequent execution of the rest of the process instance. Handeling a proper programming technique is not in the scope of this thesis.

The first approach is invoking Java code (specifying a class with predefined methods or a method explicitly). Most of the responsibility is toward Java developers who manage the code. Java code can access to instance variables and can set new variables. A service task offers an option to inject fields straight to the underlining implementation. BPMN engine and the implementation of a service task are two separate systems and must be defined with respect to the fact that names of variables have to match, or even be initialized properly.

Similar rules can be applied to External implementation. This approach is handled as a RESTful API service, which is called by other applications. The awareness of the consistency of variables is the same as with Java code. An external task is executed by the process of fetching and locking of the particular task instance and then completing by the same worker that locked the task. If migration is performed, while the task is locked by a worker, the lock will persist and the worker can complete the task.

Migration between these two implementations is not allowed because of incompatible activity behaviors.

Send Task
A Send Task acts the same way as a Service Task; there is only a difference in the conceptual view. The mapping is possible between Send Task and Service Task (only the type implementations have to match).

Script Task
Script Task is described 3.2.2.4. A script task executes a script, which is defined in a body of the task or is deployed as an individual script file. A kind of script depends on the BPM engine.

Migration of a script task means versioning of the underlying script. A script can be changed and packed with a deployment of a new process definition.

It is necessary to control variables that are used by the source and the target script task instance.

Business Rule Task
Business Rule Task provides a mechanism for a process to provide the output based on the input and business rules. Business Rule Task also refers to DMN, which is described in the section 3.3.

The core of the element is similar to Service and Send tasks - it can be implemented with Java code or as a web service (external task). This means that the migration relates to the rules of these tasks too and can be migrated between each other (the types of implementation have to match).

The only difference is in the usage of DMN, which is described in the section 3.3.1.

3.4.1.3 Gateways

Gateways are described in section 3.2.3. Gateways can be mapped between the same types only.

Exclusive Gateway
Exclusive Gateway is described in section 3.2.3.1.

The second part of migrating an exclusive gateway is changing conditions of following sequence flows. Only the variables have to exist and be initialized from the previous pass of the process instance.

Inclusive Gateway
Inclusive Gateway is described in the section 3.2.3.3. The migration plan for

the target gateway must have at least the same amount of incoming paths as the source gateway.

The other rule for an inclusive gateway is the common rule for most of the other elements that access instance variables. There have to be variables valid for all the conditions within the following sequence flows.

Parallel Gateway
Parallel Gateway is described in the section 3.2.3.2. The migration plan for the target gateway must have at least the same amount of incoming paths as the source gateway.

The migration of a parallel gateway is more complicated than the other gateways because of splitting a token into multiple tokens at a time. After a token is multiplied and the number of sequence flows increase, there is no token in the newly added flows, and the entire sequence is not executed. Whenever the number of sequence flows decrease the token will be removed, and the rest of the definition have to count with that and adjust using of variables.

Event-based Gateway
An Event-based Gateway is described in section 3.2.3.2. The event-based gateway can be migrated to the same type only, and in order to migrate the gateway's event trigger, the following events may be migrated as well. The migration of events is described in the following section 3.4.1.4.

3.4.1.4 Events

Events are described in the section 3.2.1. The events which define a persistent event trigger (message, conditional, timer, and signal events) can be migrated.

There are two scenarios for event triggers during instance migrations.

- The event trigger is not updated event the target one is a different type of trigger. The migrated event instance will follow the definition that was defined previously in the source model.
- The event trigger is updated and follows the new target definition. It may have different behavior base on the type of the event trigger.

Start Event
Start Event is described in 3.2.1.1. Start Events are mutually interchangeable. A start event can be exchanged between its subtypes (Message, Conditional, Timer, and a regular Start Event).

The migration is not needed in case of patching only a start event, the instance already started and it would not cause any breaking changes. Just in case of a subprocess, it is possible to migrate a start event and will follow the same rules as any other event have (updating trigger).

End Event
End Event is described in 3.2.1.2. End Events are not mutually interchangeable.

Even if the throw message end event is used and implements an external service or Java code, it can not be mapped on Send Task or Service Task.

Intermediate Event
An Intermediate Event is described in 3.2.1.4. If the process instance is waiting for the particular event during the migration, it has to be mapped on the target definition.

Most of the Intermediate events can be mapped between each other. However, there is an exception of migrating throw and catch actions (the source and target type has to match their actions).

Boundary Event
A Boundary Event is described in 3.2.1.3. Boundary Events are not mutually interchangeable. However, it is possible to migrate the same type but with a changed property of interruption. The non-interrupting and interrupting events are mutually interchangeable.

Migration of a boundary event has to be done by mapping the event along with an activity which is its bearer.

The boundary event can be migrated if it is mapped, however if it is not mapped to any target it will be removed during the migration.

3.4.2 Major Changes

This section deals with applying major changes to the model where might be some special constructs, or precautions have to be applied. The major changes can be patching of one element but it can cause other troubles or lead to a subsequent extension of the target model. The major changes topic can target for example subprocess instances or new events.

The new process definition must be designed to provide backward compatibility to instances that are in scope of migration.

For the demonstrations of extensive changes in business process models as an example was used a booking process.

There are three different changes made in this process to see how significant the changes can be before migration is not feasible.

3.4.2.1 The Basic Flight Booking

The full model diagram can be find in appendix B.1.

The user starts the process by entering the full name. It shows a destination offer, a length of stay, and a type of the trip D.1. Additionally, the process determines the pricing for a given destination based on the business rule task that is implemented with a decision table (DMN) C.1. The system shows the recapitulation of the order D.3. At the event-based gateway, the process will decide in which way to continue based on the sent message. If the user waits for more than 10 minutes or instructs the system to terminate, the process is terminated. If the user continues, they are asked to enter the credit card information D.4 (also must be done within 10 minutes otherwise the process will be terminated). Then the flight booking is performed but in case of error and exhaustion of tries, the process is restored and the customer is notified of the error. In case of success, the system will try to charge the credit card. If any error occurs, the whole process is compensated, and the user has to start again. In the positive case, the user is notified of the success of the booking.

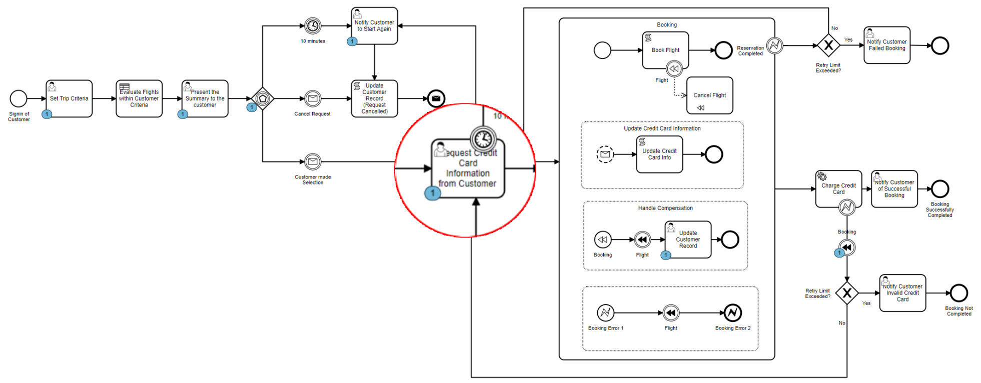

Signin of Customer
Set Trip Criteria
Evaluate Flights within Customer Criteria
Present the Summary to the customer
10 minutes
Notify Customer to Start Again
Cancel Request
Update Customer Record (Request Cancelled)
Customer made Selection
Request Credit Card Information from Customer
Booking
Book Flight
Reservation Completed
Flight
Cancel Flight
Update Credit Card Information
Update Credit Card Info
Handle Compensation
Booking
Flight
Update Customer Record
Booking Error 1
Flight
Booking Error 2
No
Yes
Retry Limit Exceeded?
Notify Customer Failed Booking
Charge Credit Card
Notify Customer of Successful Booking
Booking Successfully Completed
Booking
Retry Limit Exceeded?
Yes
No
Notify Customer Invalid Credit Card
Booking Not Completed

Figure 3.15: Model of the booking process with highlighted running instances

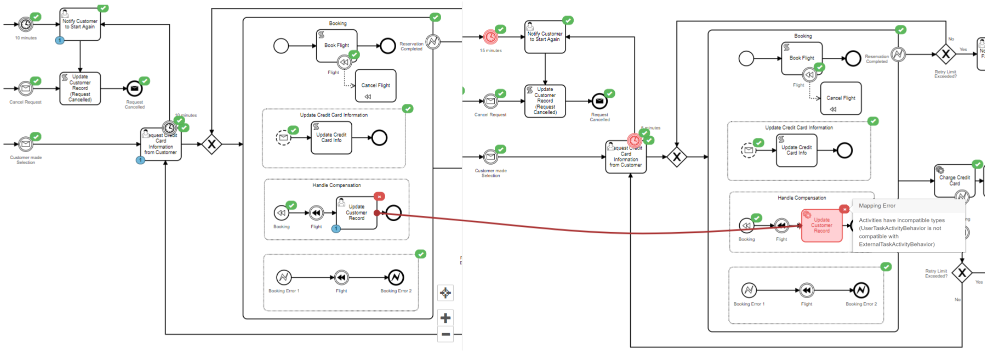
In the picture 3.15, the blue marks show spots (e.g., tasks, gateways) in which the individual instances of the process are located.

3.4.2.2 Patching Process

Example of patching in picture 3.16 illustrates the migration of some changes in the model. Simple changes were made to the Intermediate Timer Event, which does not indicate problems for mapping, but can have two different consequences.

Based on the detailed rules in the section 3.4.1.4, the timer event triggers will remain the same, and both will continue with previously defined 10 minutes. The other option is that the event triggers are updated during migration and the timer newly starts, and the definition is 15 and 5 minutes.

A different situation is with the Update Customer Record task which was changed from a user task to service task. It demonstrates automatization of this task the way that it does not have to be executed by a person but by a system. Referring to the section 3.4.1.2, these two tasks can not be mapped between each other. This issue means that one process instance cannot be migrated to the new process definition.


Mapping Error

Activities have incompatible types (UserTaskActivityBehavior is not compatible with ExternalTaskActivityBehavior)

Figure 3.16: Migration of instances to the patched version of the process model

3.4.2.3 Adding Accommodation Booking

In the last example of the process definition change, a new feature is added, which provides a hotel booking room with the previously modeled flight selection.

In the image of the new model 3.17, changes are highlighted by the different colors that distinguish the type of change. In the case of the red highlighted elements, this is a new element added. In the case of an orange color, it is a subprocess that wraps existing elements. The blue color highlights the element's change again the previous definition.

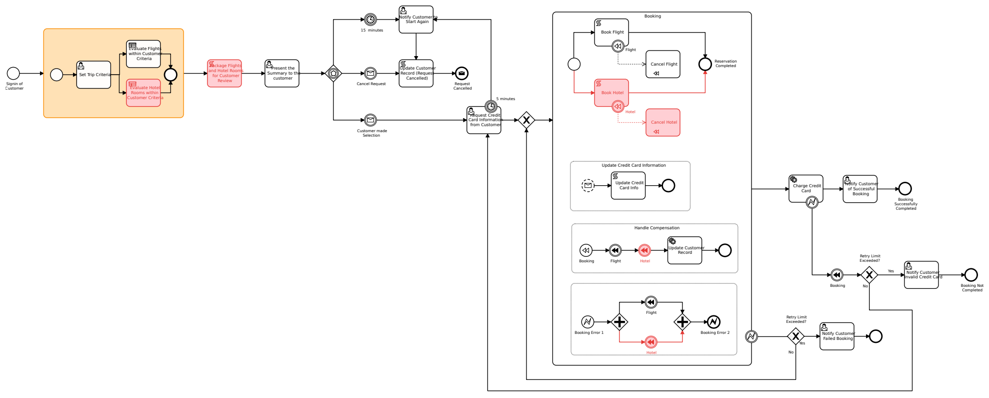

Signin of Customer
Set Trip Criteria
Evaluate Flights within Customer Criteria
Evaluate Hotel Rooms within Customer Criteria
Package Flights and Hotel Rooms for Customer Review
Present the Summary to the customer
Notify Customer to Start Again
15 minutes
Cancel Request
Update Customer Record (Request Cancelled)
Request Cancelled
Customer made Selection
5 minutes
Request Credit Card Information from Customer
Booking
Book Flight
Flight
Cancel Flight
Reservation Completed
Book Hotel
Hotel
Cancel Hotel
Update Credit Card Information
Update Credit Card Info
Handle Compensation
Update Customer Record
Booking Error 1
Booking Error 2
Charge Credit Card
Notify Customer of Successful Booking
Booking Successfully Completed
Retry Limit Exceeded?
Notify Customer Invalid Credit Card
Booking Not Completed
Notify Customer Failed Booking
Yes
No

Figure 3.17: Process model extended by a new feature of booking a hotel room

The change includes the addition of a price calculation for the hotel room as a new decision table, the calculation of the total price, which is count from the time, type, destination and the prices for the flight ticket and accommodation.

The change includes the addition of a price calculation for the hotel room as a new decision table, the calculation of the total price, which is count from the time, type, destination and the prices for the flight ticket and accommodation. Then the process counts with activities that book the hotel room and handling errors and compensations that are connected with the booking task.

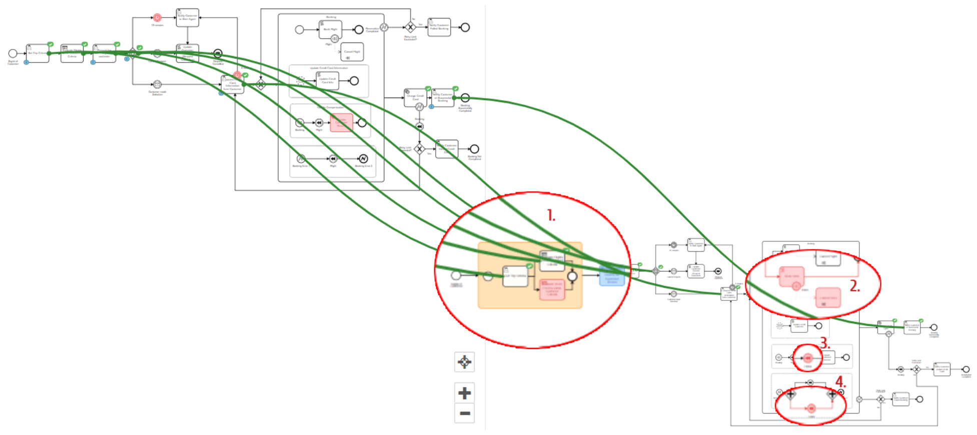


Figure 3.18: Migration of instances to the new version of the process model

In picture 3.18 of mapping is seen that migration of instance can be performed. However, the scenario which the validation tool does not count with is that there might be instances behind the decision table (which is marked as number 1.) for setting accommodation price and before processing the accommodation booking by the system with handling errors and compensations (which is marked as numbers 2., 3. and 4.).

Chapter 4

Evolvability Strategy

For the content of this chapter, the main building blocks are the findings of the previous chapter 3. The rules can be found in that chapter. Nevertheless, in this chapter, recommendations and warnings are proposed so that they can help during the designing process of a new model with respect to running instances that were created based on the old process definition.

4.1 Logical Blocks

Logical blocks that only extend the process model of conceptual elements are not a cause for migration running instance to the new process definition.

The same can be applied to strings on elements (names and annotations).

4.2 Awarness of Variables

The primary issue is the variables, whether environmental or instance variables.

- Changing types of variables can cause problems in User Tasks, Service Tasks, Script Tasks, Send Tasks or Business Rule Tasks. If these activities operate with the particular variable, they expect to be the same type.
- Adding two tasks before and behind any task that holds a token of a process instance, which is seen in picture 4.1. The antecedent task (green color) needs not to create a variable and the descendant task (green color) uses it, but the token is in the middle and during the migration is not the particular variable created. This can causes error.
- Changing variables in decision table (Business Rule Task) and migrating the running instance that our received values of the variables, however at the end of the process might be a different result than is expected.

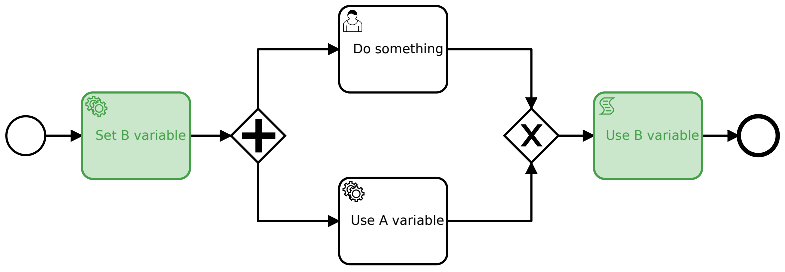


Figure 4.1: Process with two added tasks that uses previously unknown variable 'B'

If any modification to the process environment is done, variables should not be removed.

Renaming variables should not happen either. The better way is to introduce a new variable and the old one deprecated.

4.3 Compatible Elements

Whenever a token of any process instances are held on an element of the model and migration has to be executed, it can cause issues by incompatible elements.

4.3.0.1 Tasks

As an example is changing a user task that was used to performed by a person to a service task that is performed by a system. In this case, migration is not recommended, and the particular instance has to finish their job to the end.

If the migration cannot be skipped, some tools offer to move the token on different elements and start before or after the element. This can prevent problems with migrating incompatible activities.

4.3.0.2 Gateways

Migration of gateways is not actually migration of the gateways but the tokens that sent to the given sequence flows (only Event-Based Gateway holds tokens).

Data-based Gateway
Once the ancestor task is finished, the token goes through the gateway and based on the data, continues the given sequence flow.

- Exclusive gateway has simple behavior and sends a token only one way. So mapping the following tasks to the other tasks in different sequence flow is possible. Another solution is to move tokens back to the ancestor of the gateway, and the changed condition will be performed properly, according to the new process definition.

- Inclusive and Parallel gateways are more complicated because they can split the flow of a token and create multiple tokens for one process instance. As it is seen in picture 4.2, after removing one path from the parallel gateway (can be replaced by an inclusive gateway), the token held by the task "Do Task C" is orphaned and the current task with the following one will not be executed. This case cause unexpected issues, based on the implementation and defined behavior of the entire process.

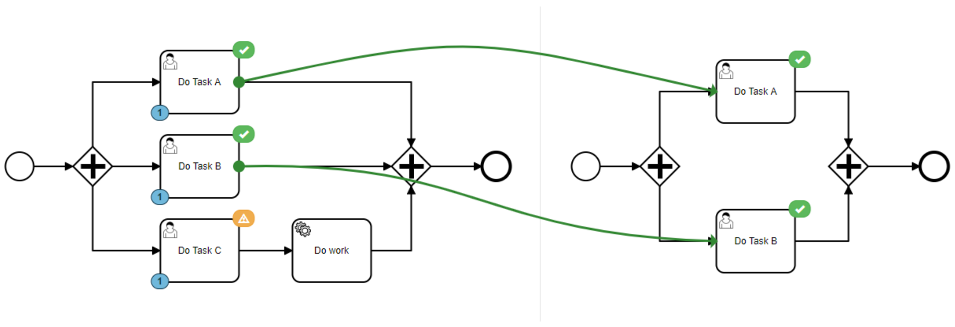


Figure 4.2: Process with multiple tokens and parallel gateway

Event-based Gateway
Event-based gateway comes with the following events (or Recieve Task). A token is held at a gateway, and the process is waiting for events to be triggered (timer, message or signal). Migrating a gateway, the following events are migrated (or updated) as well. The migration is managed according to event migration rules. Only the token is moved to the new instance definition.

4.3.0.3 Events

Event migration rules are detailed in section 3.4.1.4. The crucial aspect is Event Trigger and if that is supposed to be updated or not during the migration.

- Boundary events are mostly added later for catching errors or compensations. It is safe to add them during the migration process but the implementation of the particular task has to be ready to throw exceptions or to reverse the task.

- Intermediate Catch Event should be migrated in case the token of the process instance is waiting for the event to occur. The modeler has to be aware of the other tasks of processes if they throw the matching event.

> The catch event should not be removed without making sure that the throwing action will not occur and no catch event is not able to catch it.

Compensation Event
Compensation can be migrated, but catch events have to be mapped. This means that the task and the boundary compensation event have to be mapped together. Triggering the compensation is done from the same scope as was based in the source definition, or from the closest ancestor scope.

There can be an issue with adding compensation action to the process definition. When the compensation is triggered after migration, it will not compensate the task where the new boundary compensation event was attached.

4.4 Subprocesses

Migration refers only to a transaction, event and embedded subprocesses. Call Activity is managed by separated migration process.

Subprocesses act the same as the regular processes which can be seen in picture 4.3. The state is preserved and process variables as well. If the token is inside of the subprocess, then mapping has to be set to convert the token to the new definition, in case of no mapping is done, the process instance is canceled.

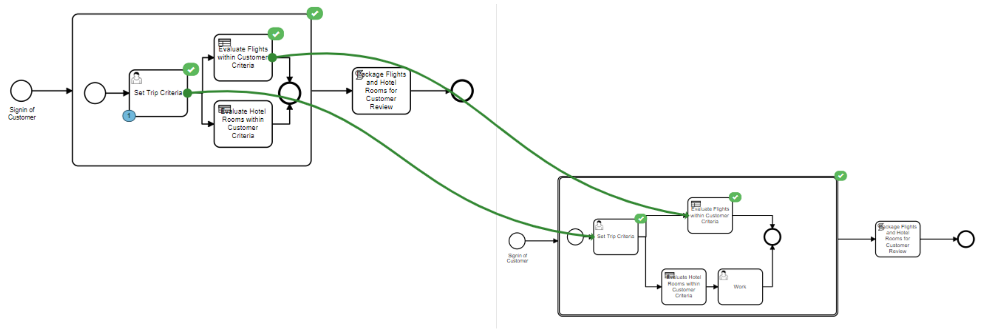

Signin of Customer
Set Trip Criteria
1
Evaluate Flights within Customer Criteria
Evaluate Hotel Rooms within Customer Criteria
Package Flights and Hotel Rooms for Customer Review
Signin of Customer
Set Trip Criteria
Evaluate Flights within Customer Criteria
Evaluate Hotel Rooms within Customer Criteria
Work
Package Flights and Hotel Rooms for Customer Review

Figure 4.3: Migration of embedded subprocess to transaction subprocess

4.5 Awarness of Tokens

If the running instances are migrated to the new model version, a problem can occur if the new version removes steps (any element) from the process definition. Whenever the target definition does not provide transition tasks (some temporary one, which is not recommended to have any useless tasks for the future instances), the process instance can fail after the migration. The advice is to move the tokens from the particular element to the ancestor element (if the element supports it). At most cases in this thesis are the tokens described.

Caution should be used with moving tokens between different types of elements (e.g., task to gateway). The problem can occur when a token is being moved from a task (which returns a nonboolean value) to gateway (which returns a boolean value), for example, from any task to an inclusive gateway with two and more outgoing paths.

4.6 Third-party Systems

This section really depends on the BPMN engine or the services that are used.

If a task implementation was used but is no longer needed, it should stay in the process application and is marked as deprecated. Later on, it can be safely removed once no instances need this implementation.

As it was earlier mentioned, variables can cause problems. Object type variables (defined in a programming language) can also cause issues. There might be different underlying systems and the classes can have a different definition, which is the spot where migration most likely fail if not handled properly. For example, a task is implemented by Java code, it uses a class variable, and the target task has different implementation, and the object is serialized and deserialized in a wrong way and process instance fails after the migration. At first is good practice to analyze previously used implementations and convert the classes to the new implementation (this can refer to an external task provisioned via web service).

Conclusion

The goal of this thesis was to investigate what types of changes are often performed and what the impact is on a BPMN system and related corporate information systems.

Once the changes have to be implemented into business process models, many problems may occur whenever already running instances also must comply these changed requirements. This process is called migration and the ability to handle this process is called evolvability.

BPMN systems, in general, can save time in further usage. Implementation of the systems might be expensive at first, but in future handling of processes either in the core business or in the customer relationship scope, the BPMN systems can help prevent many problems. Proper handling of changes in business processes and their models might save other financial resources.

This thesis provides some valuable tips on how to handle the changes and describes issues which are associated with the changes.

Meeting the Goals

The following sections report individual parts of the given requirements for this thesis and how the thesis meets these requirements.

Explore state-of-the-art BPMS'

Research of the most used platforms on the market was carried out. The chapter 1 focused on BPMN systems and low-code platforms. The low-code platforms keep gaining popularity.

From the side of evolvability and complex system requirements, BPMN platforms offer more advanced tools. For smaller businesses, low-code platforms are still relevant, but for these businesses, it may be a disadvantage if they grow too fast and they are not able to develop and manage their processes.

Explore how BPMS' handle versioning

The chapter 2 described approaches of evolvability of some popular low-code and BPMN platforms.

This chapter also includes an analysis of Camunda BPM solution, which was chosen as a representative sample. The reason for choosing this solution is that Camunda offers comprehensive system and it is open-sourced.

The chapter begins with business process reengineering, which can be used as the basic theory for change management of process models.

Create a proof-of-concept case study with minor and major changes

First of all, chapter 3 started with an exploration of BPMN specification and its related standards (DMN). Afterwards, the elements were introduced and detailed with their specific properties that can affect subsequent migration.

The chapter continues with describing rules for individual elements for the migrating process.

Based on the migration rules, examples with minor and major changes are introduces. The major changes were presented in the business process of booking flight tickets and accommodation.

Propose best practices how to handle these changes

The chapter 4 proposes recommendations on how to handle changes in business process models and their evolvability.

Based on the rules characterized in the previous chapters, there are not only recommendations but also limitations and problems that can happen during the migration process on running process instances.

Further Research

For future research, it is possible to address this issue from a practical point of view. It could be valuable to collaborate with stakeholders (BPM system suppliers, integrators, customers, end-users or business analytics).

From the theoretical point of view, it might be beneficial to build the proposed recommendations on top of a mathematical model and construct an extended methodology.

Bibliography

[1] CAMUNDA. *Workflow and Decision Automation Platform.* Available from: https://camunda.com/

[2] MICROSOFT. *PowerApps.* Available from: https://powerapps.microsoft.com/

[3] CAMUNDA. *Camunda Docs - Introduction.* Available from: https://docs.camunda.org/manual/7.9/introduction/

[4] Šmíd, V. *BPR - Business Process Reengineering.* Available from: https://www.fi.muni.cz/~smid/mis-bpr.htm

[5] Berwin Leighton Paisner LLP. The speed of business. Available from: https://www.blplaw.com/expert-legal-insights/speed-of-business

[6] Rob Dunie, V. L. B. J. W., Marc Kerremans. *Magic Quadrant for Intelligent Business Process Management Suites.* Technical report, Gartner, Inc., October 2017. Available from: https://www.gartner.com/doc/3818763

[7] Koplowitz, R. *The Forrester Wave^{TM}: Digital Process Automation Software, Q3 2017.* Technical report, Forrester research, July 2017. Available from: https://www.forrester.com/report/The+Forrester+Wave+Digital+Process+Automation+Software+Q3+2017/-/E-RES136905

[8] PEGA. *Pega Platform.* Available from: https://www1.pega.com/products/pega-platfor

[9] IBM. *IBM Business Process Management.* Available from: https://www.ibm.com/cloud/automation-software/business-process-management

[10] APPIAN. *Appian BPM Suite*. Available from: `https://www.appian.com/platform/bpm-suite`

[11] BIZAGI. *Bizagi - The Digital Business Platform*. Available from: `https://www.bizagi.com/en/products`

[12] AURORA. *Aurora Portal*. Available from: `https://www.auraportal.com/product`

[13] K2. *K2 - platform overview*. Available from: `https://www.k2.com/platform/platform-overview`

[14] ORACLE. *Oracle BPM Suite*. Available from: `http://www.oracle.com/us/technologies/bpm/suite/overview/index.html`

[15] Rymer, J. R. *The Forrester Wave^{TM}: Low-Code Development Platforms For AD&D Pros, Q4 2017*. Technical report, Forrester research, October 2017. Available from: `https://www.forrester.com/report/The+Forrester+Wave+LowCode+Development+Platforms+For+ADD+Pros+Q4+2017/-/E-RES137262`

[16] Paul Vincent, Y. V. N. K. I., Van L. Baker. *Magic Quadrant for Enterprise High-Productivity Application Platform as a Service*. Technical report, Gartner, Inc., April 2018. Available from: `https://www.gartner.com/doc/3872957/magic-quadrant-enterprise-highproductivity-application`

[17] SALESFORCE. *salesforse platform*. Available from: `https://www.salesforce.com/products/platform/overview/`

[18] OutSystems. Available from: `https://www.outsystems.com/`

[19] Mendix. Available from: `https://www.mendix.com/`

[20] ACTIVITI. *Activiti Open Source Business Automation*. Available from: `https://www.activiti.org`

[21] CAMUNDA. *BPMN 2.0 Implementation Reference*. Available from: `https://docs.camunda.org/manual/7.10/reference/bpmn20/`

[22] Justice Opara-Martins, F. T., Reza Sahandi. *Critical analysis of vendor lock-in and its impact on cloud computing migration: a business vperspective*. Available from: `https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-016-0054-z`

[23] MENDIX. *Version Control Concepts*. Available from: `https://docs.mendix.com/refguide/version-control-concepts/`

[24] APPIAN. *Process Model Versioning.* Available from: `https://docs.appian.com/suite/help/17.4/Process_Model_Versioning.html`

[25] Farrance, M. *Corporate Bottlenecks.* Available from: `https://www.bonitasoft.com/library/corporate-bottlenecks`

[26] CAMUNDA. *Camunda Docs - Process Instance Migration.* Available from: `https://docs.camunda.org/manual/7.9/webapps/cockpit/bpmn/process-instance-migration/`

[27] Oxford University Press. *British & World English.* Accessed on 02.11.2018. Available from: `https://en.oxforddictionaries.com`

[28] OMG. Business Process Model and Notation (BPMN), Version 2.0. Available from: `http://www.omg.org/spec/BPMN/2.0`

[29] OMG. Decision Model and Notation (DMN), Version 1.1. Available from: `https://www.omg.org/spec/DMN/1.1`

Appendix A

Acronyms

API Application Programming Interface

BPM Business Process Management

BPMN Business Process Model and Notation

BPMS Business Process Management System

BPR Business Process Reengineering

B2B Business-to-business

CDI Context and Dependency Injection

CMMN Case Management Model and Notation

CRM Customer Relationship Management

DMN Decision Model and Notation

DPA Digital Process Automation

DRD Decision Requirements Diagram

DRG Decision Requirements Graph

IDE Integrated Development Environment

IFTTT If This Then That

JSON JavaScript Object Notation

OMG Object Management Group

REST Representational State Transfer

ROI Return Of Investment

SaaS	System as a Service
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
SSO	Single Sign-On

Appendix B

Process models

This appendix includes full business process models which were used to demonstrate the features and limits of evolvability of complex models.

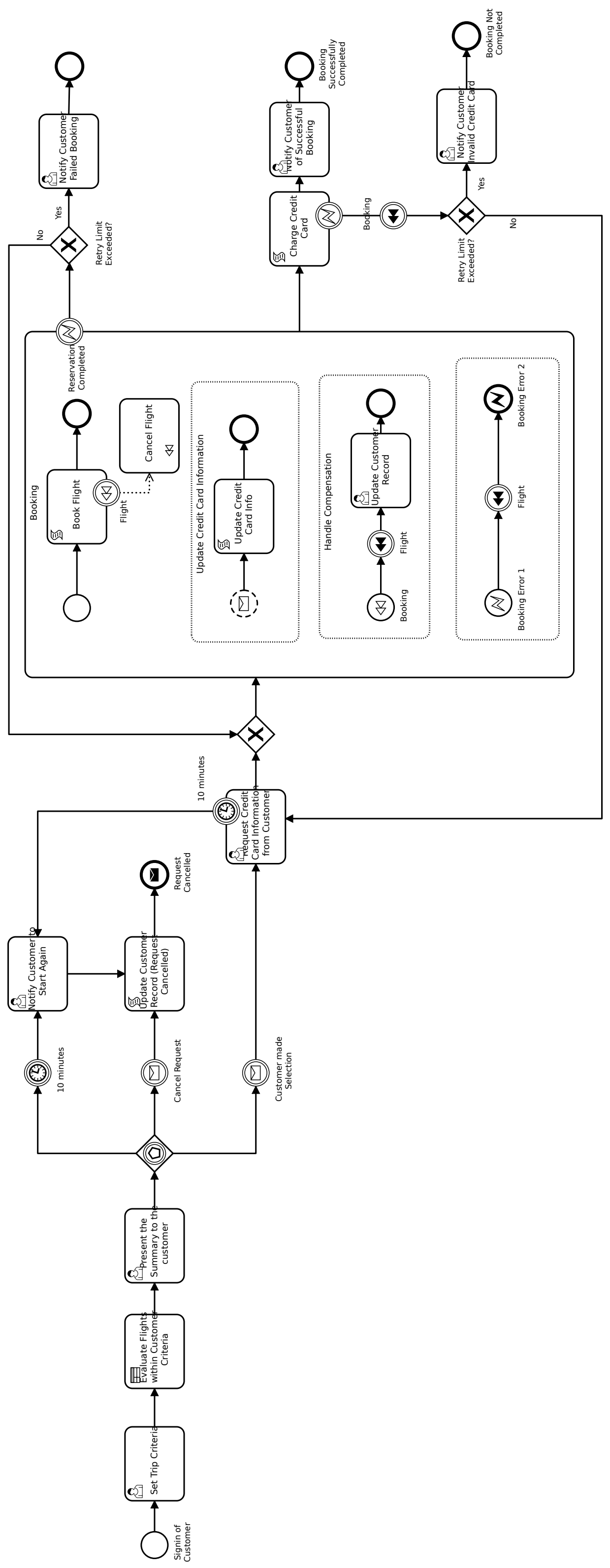
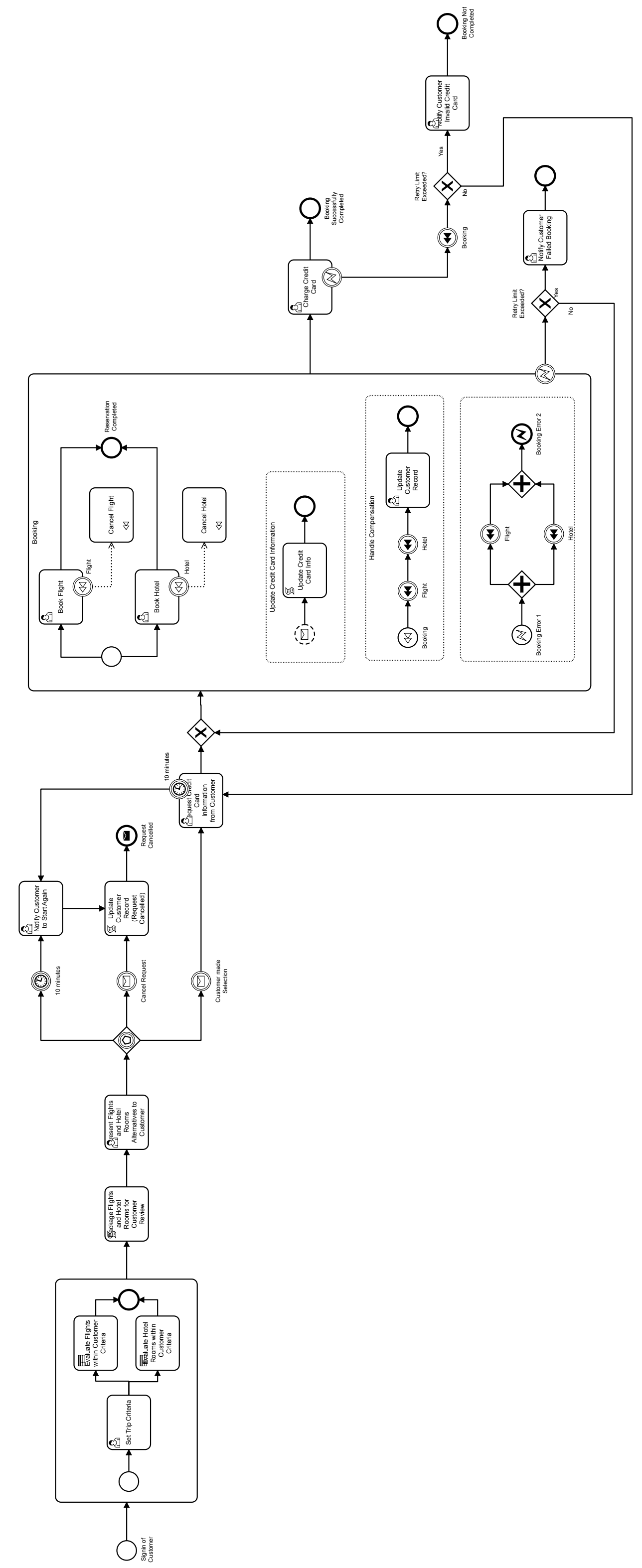


Figure B.1: Final example of the flight booking process model

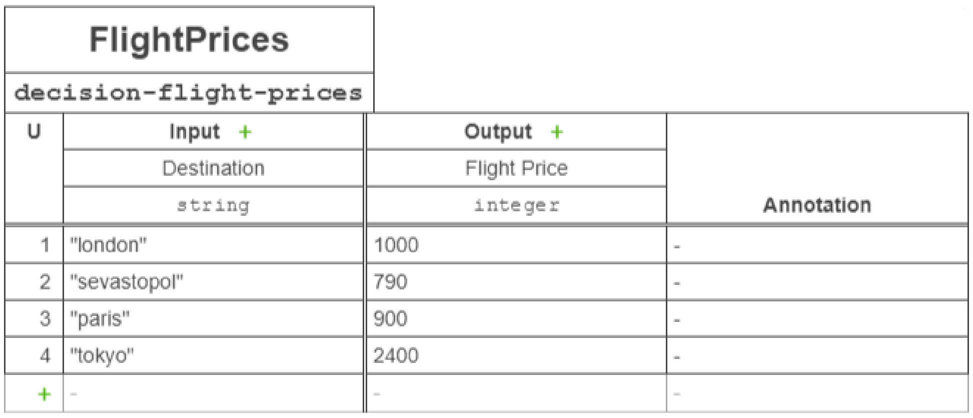


Signin of Customer
Set Trip Criteria
Evaluate Flights within Customer Criteria
Evaluate Hotel Rooms within Customer Criteria
Package Flights and Hotel Rooms for Customer Review
Present Flights and Hotel Rooms Alternatives to Customer
10 minutes
Cancel Request
Customer made Selection
Notify Customer to Start Again
Update Customer Record (Request Cancelled)
Request Cancelled
Request Credit Card Information from Customer
10 minutes
Booking
Book Flight
Flight
Book Hotel
Hotel
Cancel Flight
Cancel Hotel
Reservation Completed
Update Credit Card Information
Update Credit Card Info
Handle Compensation
Booking
Flight
Hotel
Update Customer Record
Booking Error 1
Flight
Hotel
Booking Error 2
Charge Credit Card
Booking Successfully Completed
Booking
Retry Limit Exceeded?
Yes
No
Notify Customer Invalid Credit Card
Booking Not Completed
Retry Limit Exceeded?
Yes
No
Notify Customer Failed Booking

Figure B.2: Final example of the flight and hotel booking process model

Appendix C

Decision tables

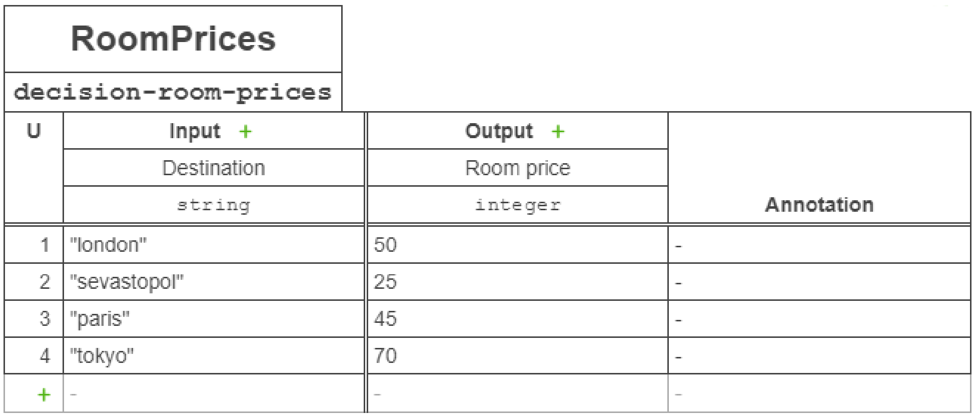


FlightPrices

decision-flight-prices

U	Input +	Output +	Annotation
	Destination	Flight Price	
	string	integer	
1	"london"	1000	-
2	"sevastopol"	790	-
3	"paris"	900	-
4	"tokyo"	2400	-
+	-	-	-

Figure C.1: Decision table for flight prices



RoomPrices

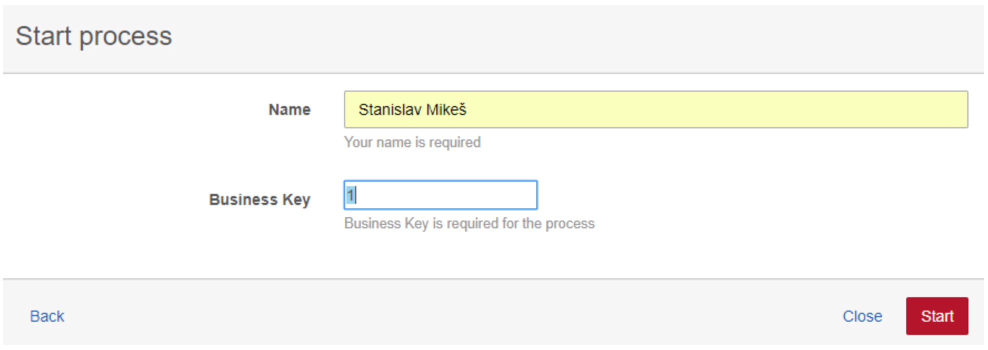
decision-room-prices

U	Input +	Output +	Annotation
	Destination	Room price	
	string	integer	
1	"london"	50	-
2	"sevastopol"	25	-
3	"paris"	45	-
4	"tokyo"	70	-
+	-	-	-

Figure C.2: Decision table for room prices

Appendix D

Process forms


Start process

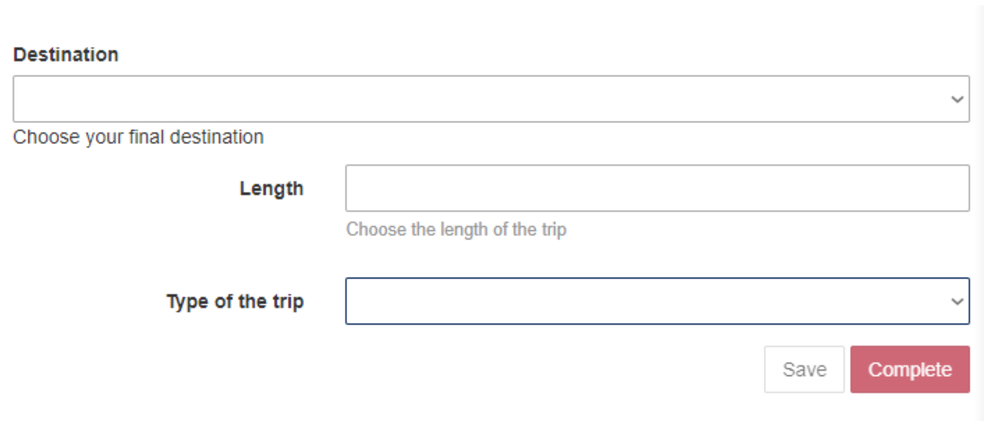
Name: Stanislav Mikeš

Your name is required

Business Key: 1

Business Key is required for the process

Figure D.1: Setting parameters for the trip by the user


Destination

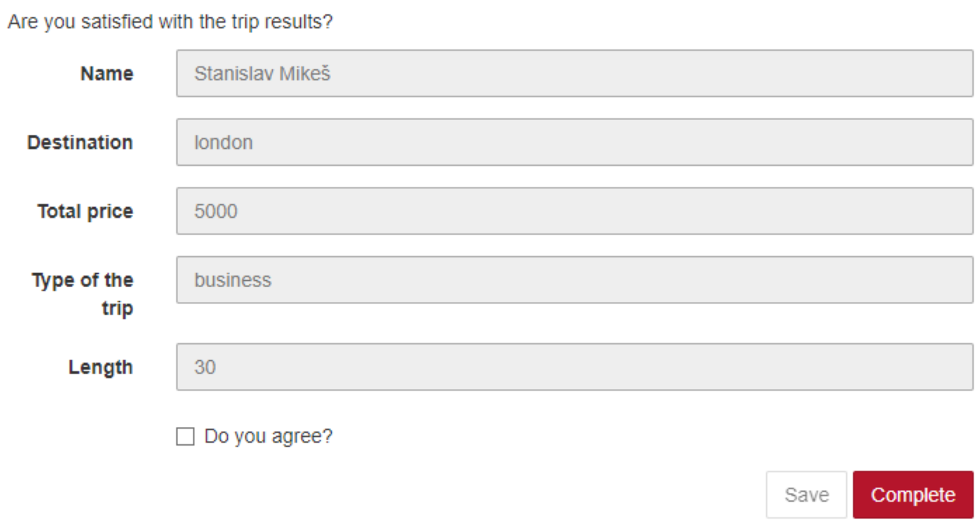
Choose your final destination

Length

Choose the length of the trip

Type of the trip

Figure D.2: Setting parameters for the trip by the user

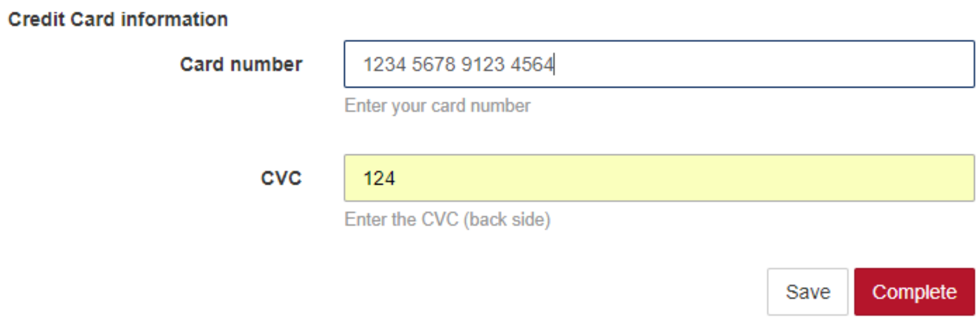


Are you satisfied with the trip results?

Name	Stanislav Mikeš
Destination	london
Total price	5000
Type of the trip	business
Length	30

☐ Do you agree?

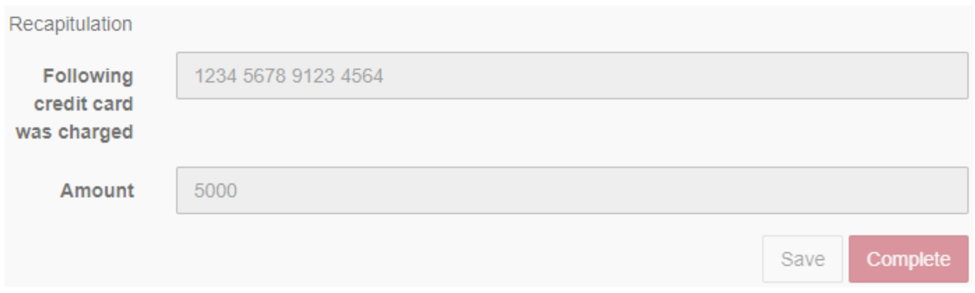
Figure D.3: Recapitulation form



Credit Card information

Card number	1234 5678 9123 4564
	Enter your card number
CVC	124
	Enter the CVC (back side)

Figure D.4: Form for input information about a credit card



Recapitulation

Following credit card was charged	1234 5678 9123 4564
Amount	5000

Figure D.5: Recapitulation of charging a creadit card

Appendix **E**

Contents of enclosed CD

```
readme.txt ....................... the file with CD contents description
src ......................................the directory of source codes
  processes........the directory for bpmn models with additional files
  thesis...the directory of LaTeX source codes and images of the thesis
text .........................................the thesis text directory
  DP_Mikes_Stanislav_2019.pdf ........ the thesis text in PDF format
```