UNIVERSITY OF MACEDONIA
DEPARTMENT OF APPLIED INFORMATICS
GRADUATE PROGRAM

Business Process and Decision automation:
End-to-end deployment with a BPMN and DMN-based workflow engine

M.Sc. THESIS

of

Nikolaos Nousias

Thessaloniki, February 2021

Business Process and Decision automation:
End-to-end deployment with a BPMN and DMN-based workflow engine

Nikolaos Nousias

B.Sc. in Accounting and Finance, Department of Accounting and Finance
University of Thessaloniki, 2018

M.Sc. Thesis

submitted as a partial fulfillment of the requirements for

THE DEGREE OF MASTER OF SCIENCE IN APPLIED INFORMATICS

Supervisor: Dr Konstantinos Vergidis

Approved by examining board on 26 February 2021

Prof Alexander Chatzigeorgiou   Prof Efthimios Tambouris   Dr Konstantinos Vergidis

Abstract

Today's organizations deliver a broad range of products and services. Contemplating the perpetually increasing number of their business processes, an organizational culture, conducive to process thinking and responsive to process changes, is a determining factor for the effectiveness and competitiveness of contemporary enterprises. Given the fact that currently everything has to be performed faster, with greater flexibility and agility, business process and decision automation have become an integral part of every digital transformation initiative. In this regard, workflow automation, has come in frontline for automating business process and decisions, relying on a predefined process model.

The present research aims to investigate practically the workflow automation paradigm, on the basis of executable notations, namely the BPMN and DMN notations, by introducing an end-to-end loan application-to-approval process and automating it by the means of a workflow engine. For this purpose, the BPMN notation is thoroughly presented, before shedding light on the DMN notation and its capabilities to be well integrated with BPMN, rendering business processes as decision-aware and decision-intelligent. With the ultimate aim to automate the introduced process, this research highlights how business-oriented models can be executable, paving the way for their automated enactment by the means of a workflow engine of a contemporary BPMS.

Keywords

BPMN, DMN, separation of concerns, workflow, workflow engine, business process automation, decision automation, workflow automation, Camunda BPM

Conference Papers

George Tsakalidis, **Nikolaos Nousias**, Kostas Vergidis (2020), "An inclusive representation approach to assess the redesign capacity of BPMN models", *in Proceedings of the XIV Balkan Conference on Operational Research (virtual BALCOR2020)*, Thessaloniki, Greece, 30 September - 3 October 2020

Acknowledgements

Writing a thesis is a hard and strenuous journey, that it would be impossible without the kind support and help from many individuals.

Foremost, I would like to thank God for the wisdom he bestowed upon me, the strength, peace of my mind and good health in order to finish this research.

From the bottom of my heart, I would like to say a sincere thank you to my supervisor, my mentor, Kostas Vergidis, for his relentless advice and guidance throughout this journey. I am extremely grateful for all of our conversations and his personal support in my academic and business endeavors.

I would like to express my special gratitude towards my family for supporting me spiritually throughout my life. To my parents, my heroes, who have always been on my side, encouraging me on every aspect of my life, and to my sister, my angel, who has unconditionally supported me and always being there for me as a friend. I am forever indebted to them for giving me the opportunities and experiences that have made me who I am.

Last, but not least, my thanks and appreciations also go to all my professors during my master studies, who have imparting me their knowledge and assisting me to delve into the fascinating world of Applied Informatics.

Without your help, support and wise guidance, this thesis would have not been the same!

Table of contents

List of figures

List of tables

Abbreviations

API - Application Programming Interface

BPD - Business Process Diagram

BPM - Business Process Management

BPMI - Business Process Model Initiative

BPMN - Business Process Model and Notation

BPMS - Business Process Management System

BPR - Business Process Reengineering

CRM - Customer Relationship Management

DaaS - Decision as a Service

DBMS - Database Management System

DMN - Decision Model and Notation

DRD - Decision Requirements Diagram

DT - Digital Transformation

EAD - Exposure At Default

EL - Expected Losses

EPC - Event-driven Process Chain

ERP - Enterprise Resource Planning

FEEL - Friendly Enough Expression Language

IT - Information Technology

KPI - Key Performance Indicator

LGD - Loss Given Default

MDE - Model-Driven Engineering

ML - Machine Learning

OMG - Object Management Group

PD - Probability of Default

PO - Process Orientation

REST - Representational State Transfer

RPA - Robotic Process Automation

SCM - Supply Chain Management

SE - Software Engineering

SOA - Service-Oriented Architecture

SoC - Separation of Concerns

UoB – Unit of Behavior

war - Web Application Resource

WfMC - Workflow Management Coalition

WfMS - Workflow Management System

CHAPTER 1: Introduction

Today’s business enterprises need to deliver a broad range of products and services. As a result, the number of businesses processes inside organizations has exponentially increased. Depending upon the power of Business Process Management (BPM), business processes are considered to be the crown jewels of successful organizations [1]. In this sense, a plethora of process modeling techniques has been proposed in literature, while the Business Process Model and Notation (BPMN) has been consolidated as the most prevalent one for rendering business operations in a graphical, yet executable way.

Even if decisions are deemed to be an integral part of business processes, process modeling techniques do not explicitly model the decision logic of business operations. Thus, decision modeling has gained a significant uptake in literature, adhering to the Separation of Concerns (SoC) paradigm and treating decisions as separate concerns. Considering that operational decisions are frequently convoluted with the process logic of business processes, decisions are typically hidden in process model constructs, escalating their complexity and decreasing the maintainability and understandability of both processes and decisions. To this extent, Decision Model and Notation (DMN) has emerged as a standard notation for supplementing the BPMN notation, by externalizing the decision logic of business processes to separate models. As a result, the BPMN and DMN integration has received special attention in recent literature, with the purpose to render business processes as decision aware and decision intelligent.

Traditionally, business processes were enacted manually, while business decisions were dependent on human decision-making. However, in today’s rapidly changing environment, where everything needs to be executed faster, business process and decision automation have emerged as key success factors for the digital transformation and the innovation empowerment of today’s organizations. In this regard, workflow automation constitutes an on the rise trend for the automatic enactment of business processes and decisions, where a workflow engine interprets executable models, as these models were the source code of a software solution [2].

1.1 Motivation

Considering that change is the only constant in the business environment, business agility is deemed to be indispensable to perpetual growth and value creation. Admittedly, as business processes become more advanced, there is a shift from pre-defined process models to intelligent systems that support and automate processes. Maintaining a competitive advantage, thus, is deemed to be more crucial than ever before [3], given the global competition and the force to organizations to get digitalized.

At the outset of BPM, a full automated process was a wishful thinking rather than a reality. Currently, there is an undergoing trend towards process and decision automation, where the human intervention can be eliminated. Repetitive and tedious non-valued-added tasks, that would otherwise be executed manually, can now be automatically enacted, changing the way of doing business.

In the previous years, the most common approach for executing processes and decisions in an automatic way, was in a classical programming language, where the logic was hidden inside thousands of lines of code [2]. However, with the advent and the significant adoption of executable notations, workflow engines are established as modern approaches that execute automatically the process and decision models of business processes. Refining the traceability, readability and transparency of both processes and decisions, workflow automation in the premise of executable notations, such as BPMN and DMN notations, is deemed to prevail in the years to come, establishing at the same time the motivational factor behind the current thesis.

1.2 Aim and objectives

The main aim of this thesis is to practically assess the workflow automation paradigm on the basis of executable notations, such as the BPMN and DMN notations. For this purpose, an end-to-end development of a loan application-to-approval process is introduced, aiming to fulfill the practical scope of this thesis.

In particular, the aim of this thesis is to render how a textual description of a process can be translated to a conceptual business process model, before specifying its execution semantics, executing it with a means of a workflow engine and ideally automate it. In this sense, special attention is given on the way that BPMN notation can render business processes in a graphical, let alone intuitive way. However, considering that BPMN is not tailored to the modeling of the decision logic of business processes, the main objective is to externalize the decision logic of BPMN models to separate DMN models, before their integration serves as a foundation for a robust process and decision automation.

In this regard, the main objectives of this research, include:

1. Presenting the BPMN notation as a powerful instrument to modeling advanced concepts of business processes
2. Displaying how the process logic of BPMN models can be externalized to DMN models
3. Rendering how BPMN and DMN notations can be integrated in end-to-end business processes, rendering business processes as decision-aware and decision-intelligent

4. Developing an end-to-end loan application-to-approval process, starting from its textual description and ending with its automatic enactment
5. Automating a process with a workflow engine of a BPMS

1.3 Thesis layout

A brief description of the chapters of this thesis is provided below, while the thesis layout is presented visually in Figure 1.

Chapter 1 presents an introduction to the current thesis, briefly explaining the motivation behind it, its aim and objectives, as well as rendering the overall thesis structure.

Chapter 2 presents the theoretical background of this thesis, establishing the foundation behind its practical scope. In essence, it renders the process orientation paradigm, defines business processes, illustrates process modeling techniques, emphasizes the separation of concerns paradigm in process and decision modeling, while gives prominence to executable models and to workflow automation.

Chapter 3 introduces the Business Process Model and Notation (BPMN) as the standard notation for rendering business processes in a graphical and executable way. Presenting its powerful symbol armory, it aims to render the correct utilization of its elements, as well as to highlight advanced modeling concepts, that are frequently omitted in BPMN modeling initiatives.

Chapter 4 introduces the Decision Model and Notation (DMN), as the standard notation for rendering the decision logic of business processes. Considering that BPMN is not tailored to the modeling of decision logic of business operations, this chapter presents the DMN notation as a supplementary notation to BPMN, where the decision logic can be externalized to separate models. Thus, introducing its fundamental constructs, special focus is given on the way that DMN can be integrated with BPMN, rendering business processes as decision aware and decision intelligent.

Chapter 5 presents the concept of Business Process Management Systems (BPMSs), while introduces a renowned BPMS, namely the Camunda BPM platform. Rendering the notion of Process Aware Information Systems (PAISs), special attention is given to the rise of Workflow Management Systems (WfMSs) and their enhanced variants, namely the BPMSs. In addition, it introduces the Camunda BPM platform, shedding light on its components and the way that they interact with each other, before Camunda’s ecosystem being utilized in the following chapter for an end-to-end process deployment.

Chapter 6 fulfills the practical scope of this thesis, presenting an end-to-end deployment of a loan application-to-approval process. Initially, it renders the challenges for automation in the banking industry, before introducing a fictitious loan application-to-approval process with the ultimate aim to automate it. Presenting, thereafter, a BPMN representation of the process, special attention is given on the identification and externalization of its decision logic, before specifying the execution semantics of the process model. Providing an overview of the deployment architecture, the chapter concludes with an execution scenario and the expected benefits of workflow automation initiatives.

Chapter 7 concludes this thesis with a discussion on the practicality of workflow automation in business process management initiatives. Providing the thesis overview and the research contribution, the chapter highlights research limitations, as well as establishes directions for future work, stemming from this research.

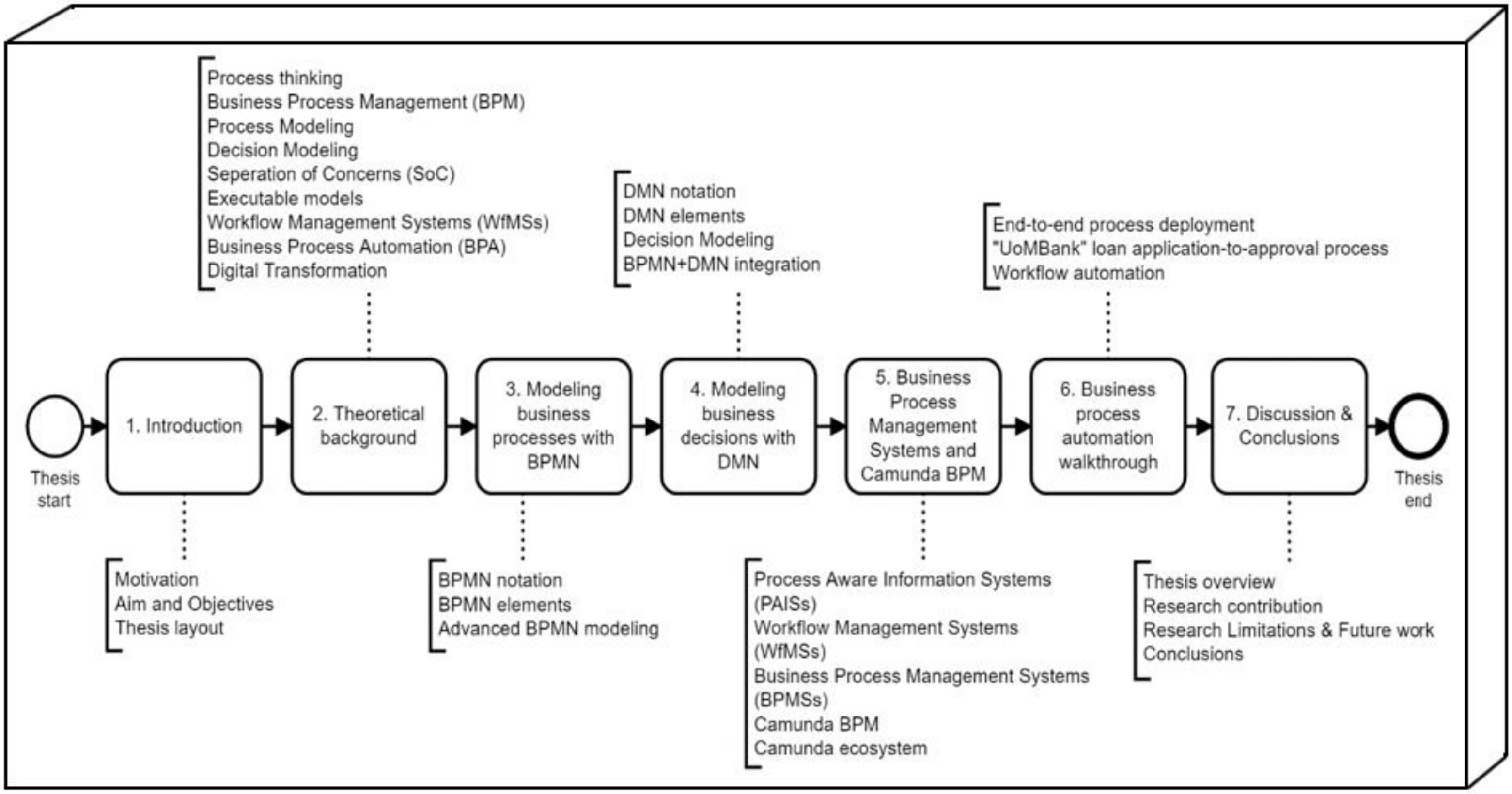

Process thinking
Business Process Management (BPM)
Process Modeling
Decision Modeling
Seperation of Concerns (SoC)
Executable models
Workflow Management Systems (WfMSs)
Business Process Automation (BPA)
Digital Transformation
DMN notation
DMN elements
Decision Modeling
BPMN+DMN integration
End-to-end process deployment
"UoMBank" loan application-to-approval process
Workflow automation
Thesis start
1. Introduction
2. Theoretical background
3. Modeling business processes with BPMN
4. Modeling business decisions with DMN
5. Business Process Management Systems and Camunda BPM
6. Business process automation walkthrough
7. Discussion & Conclusions
Thesis end
Motivation
Aim and Objectives
Thesis layout
BPMN notation
BPMN elements
Advanced BPMN modeling
Process Aware Information Systems (PAISs)
Workflow Management Systems (WfMSs)
Business Process Management Systems (BPMSs)
Camunda BPM
Camunda ecosystem
Thesis overview
Research contribution
Research Limitations & Future work
Conclusions

Figure 1 - Thesis layout

CHAPTER 2: Theoretical Background

This chapter delves into the theoretical background of the main concepts to be discussed in the premise of this thesis. The aim is to establish a robust background that will facilitate the smooth transition from theoretical terms to the practical scope of this thesis. To achieve that, it discusses valuable findings from academia and industry around the business processes' spectrum, as well as highlights how the attention has switched from traditional process management to business process automation in today's digital era. In this regard, special focus is given in the way that organizations' process orientation, the ubiquitous penetration of Information Technology (IT) in business processes and the underlying Digital Transformation (DT) of contemporary organizations, have established the foundations of business process automation. As such, a relationship map, rendered in Figure 2, is introduced in order to precociously communicate how fundamental theoretical concepts are interrelated, before they form the foundations of the thesis's practical scope.

2.1 Process orientation: The emergence of process thinking notion

From the late 1970s to the early 1990s, the focus of Information Technology (IT) was principally on data. Primarily, the storing, retrieval and presentation of data was at the core of every run time infrastructure, leading to the development of data-centric information systems, such as the Database Management Systems (DBMSs). At that time, the concept of business processes was significantly neglected, while their logic was proliferated across multiple applications, thereby thwarting their refinement and optimal adaptation to changes [4].

Thereafter, during the 1990s, management trends, such as Business Process Reengineering (BPR), have brought the emergence of process thinking notion, as an approach to look at entire end-to-end processes instead of focusing on particular tasks and functions [5].Embracing a business process view of a corporation, special attention is implied on what is done and how it is done within its boundaries [6]. Admittedly, contemplating a corporation as a wholeness, business operations as well as their interrelationships are well identified [7].

In this sense, process-oriented organizations are primarily concerned with the optimal management of their business processes [8]. Process Orientation (PO), thus, entails focusing on business processes rather than on functional and hierarchical structures [7], [8]. PO facilitates the process transparency, let alone establishes an environment for streamlined operations through reengineering initiatives [8]. Currently, the value of process thinking is widely

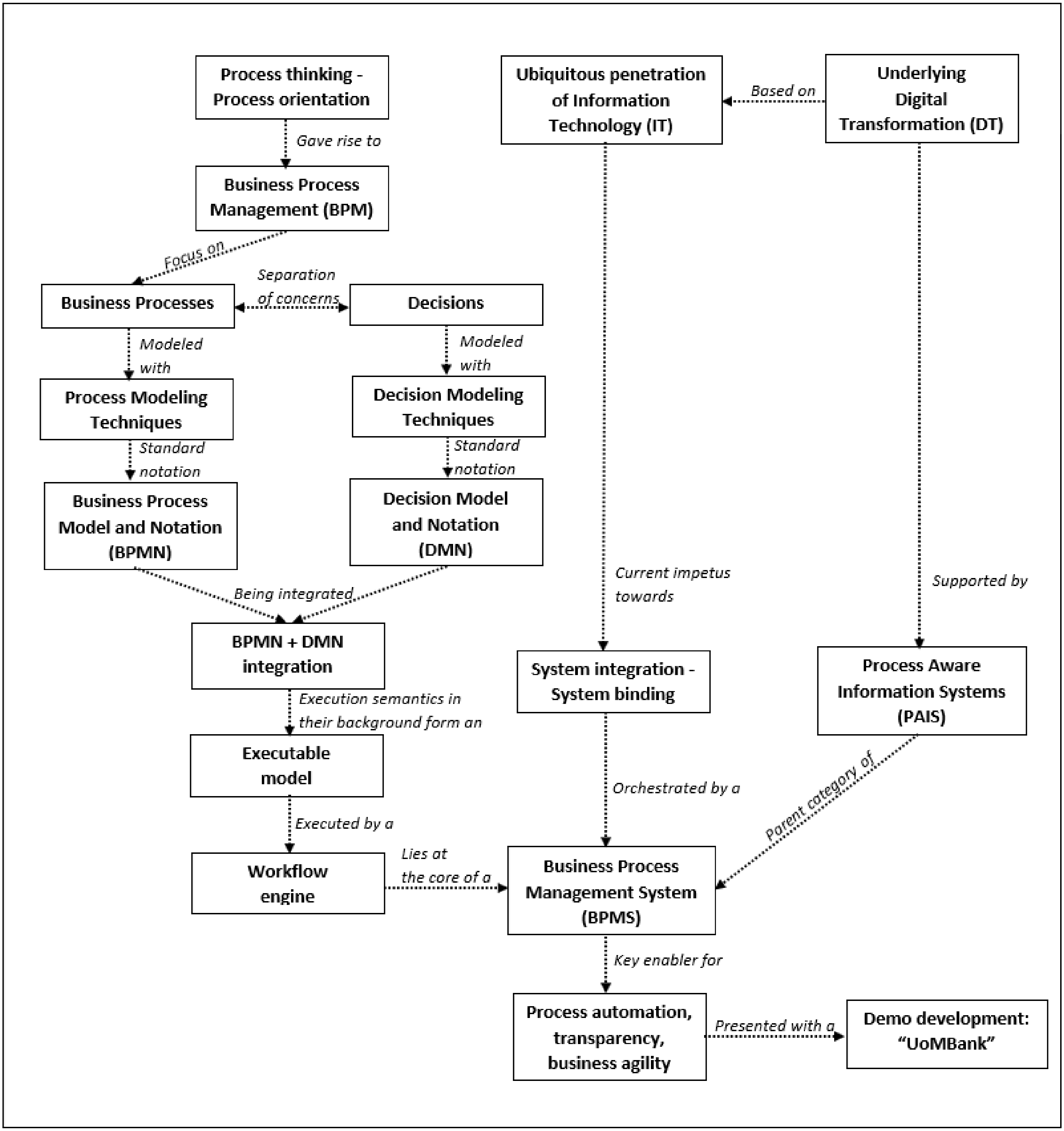

Process thinking - Process orientation
Gave rise to
Business Process Management (BPM)
Focus on
Business Processes
Separation of concerns
Decisions
Modeled with
Process Modeling Techniques
Modeled with
Decision Modeling Techniques
Standard notation
Business Process Model and Notation (BPMN)
Standard notation
Decision Model and Notation (DMN)
Being integrated
BPMN + DMN integration
Execution semantics in their background form an
Executable model
Executed by a
Workflow engine
Lies at the core of a
Ubiquitous penetration of Information Technology (IT)
Based on
Underlying Digital Transformation (DT)
Current impetus towards
Supported by
System integration - System binding
Process Aware Information Systems (PAIS)
Orchestrated by a
Parent category of
Business Process Management System (BPMS)
Key enabler for
Process automation, transparency, business agility
Presented with a
Demo development: "UoMBank"

Figure 2 - Relationship map: From theoretical background to thesis's practical scope with a demo development

acknowledged from most organizations [9], while it lies in the core of several development strategies and methods that are utilized today, like the Business Process Management (BPM) and the Business Process Reengineering (BPR) [6].

2.2 Business Process Management (BPM)

Based on early work in organization and management, as well as the process orientation trend of the 1990s, Business Process Management (BPM) has come in forefront as a new approach of organizing enterprises on the basis of business processes [10]. In this regard, combining knowledge from information technology, management sciences and industrial engineering, the main objective is the refinement of business operations [10], [11]. Several definitions of BPM are available in literature, while for the premise of this thesis three definitions are well adopted. Weske [10], highlights BPM as an approach that "includes concepts, methods, and techniques to support the design, administration, configuration, enactment, and analysis of business processes". Dumas et al. in their work "Fundamentals of Business Process Management" [5], define BPM as "the art and science of overseeing how work is performed in an organization to ensure consistent outcomes and to take advantage of improvement opportunities". Additionally, with a more sophisticated perspective, Kluza et al. [12], define BPM as a "modern approach to improve organization's workflow, which focuses on reengineering of processes in order to optimize procedures, increase efficiency and effectiveness by constant process improvement". Therefore, the BPM approach is tailored to the management and refinement of entire chains of events, activities and decisions, rather than the improvement of individual activities [5].

Admittedly, a process-oriented organization excessively depends upon the power of BPM [5], [7]. Being a disciplined approach, thus, BPM consists of a series of phases (Figure 3), emerging as the BPM lifecycle unfolds [5]. As such, BPM supports organizations from modeling and analysis, to execution and evaluation of redesigned processes [13]. More specifically, lying at its foundation, process identification entails the recognition of a business problem and its relation with one or more business processes [5]. In turn, process discovery renders the process in an explicit model representation, known as an as-is model, facilitating its better understanding [5], [10]. Following qualitative and quantitative techniques, process analysis phase underlines process weaknesses and their impact on crucial performance metrics. The issues identified, are well mitigated with proposed changes and a plethora of redesign heuristics, during the following process redesign phase. The main output is deemed to be a redesigned process model, typically encountered as a to-be process model, where the proposed changes are fundamentally aligned with organization's performance objectives. Subsequently, the process implementation phase facilitates the enactment of business processes with the leverage of BPM technology [5]. In this sense, the continuous process monitoring of running processes, intents to identify how well the process is performing. In respect to predefined performance objectives, thus, valuable insights emerge, before serving as an input to the process discovery phase and trigger a new enactment of the BPM lifecycle with process adaptations and new analysis questions [5], [11].

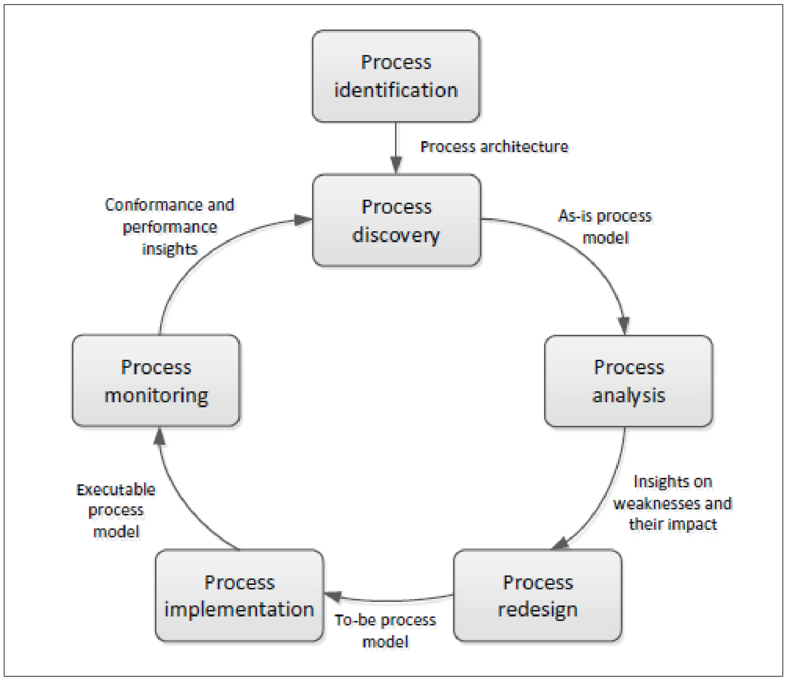


Figure 3 - BPM lifecycle [5]

Interestingly, the BPM lifecycle must be continuously seen as a circular approach [5]. As Hammer [14] highlights, “Every good process eventually becomes a bad process”, since technology, customer needs, competition and the surrounding environment constantly change, causing the process obsolescence. Altogether, flexibility of BPM is an important asset for aligning with the market changes in an effective manner [10]. Currently, enterprises implement changes to the traditional BPM paradigm, fueled by technological developments and the increasing automation of business processes [15]. Due to the dynamic, customer-facing processes of today’s digital organizations, BPM has been undergoing a transformation [16]. In this regard, the lion’s share of attention is given to the innovation, agility and flexibility, rather than the traditional objective of continuous process improvement and standardization [17].

Ultimately, engaging in BPM initiatives, organizations are deemed to streamline their business performance and eliminate prior vulnerabilities [5]. According to Aalst et al. [9], the ultimate objective of BPM should be the process refinement rather than the improving of process models. Being correlated with performance management, the success can be measured with a plethora of Key Performance Indicators (KPIs), that align business performance with organization’s objectives [18]. When properly implemented, customer satisfaction, increasing productivity, reducing cost of doing business, establishment of new products and services, as well as improvements in any other metric, are well attainable [11], [18].

2.3 Business Process Definition

Having introduced the BPM paradigm, it is apparent that the concept of business processes constitutes the focal pillar of every BPM initiative [11]. Since the 1990s and the first emergence of the business process term in literature, several authors have proposed their own definitions, highlighting specific aspects of business processes [19]. As a result, several definitions are available in literature, reflecting the variety of interpretations that emerge around the business process spectrum. For the premise of this thesis, the definition proposed by Hammer and Champy [20] is well adopted, while further definitions are presented in Table 1. In their work "Reengineering the Corporation: A Manifesto for Business Revolution", Hammer and Champy [20] define a business process as "a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer."

Apparently, most definitions are similar, giving prominence to the concepts of activities, inputs and outputs [19]. Typically, activities are deemed to constitute the central components of business processes, while are contextually related to each other by transforming inputs to outputs [6], [19]. In this sense, business processes are the means of coordinating a number of activities and facilitating the understanding of their interrelationships [10]. In turn, inputs constitute the resources indispensable to activities execution, while outputs render the transformed resources with the ultimate goal of meeting customers' and organizational needs in the most efficient way [19], [21], [22].

Today, since products and services, provided to the market, are the outcome of a chain of activities, performed in the context of a business process, organizations focus on refining their business operations as a response to the perpetually increasing competition and the more demanding customer needs [8], [10]. Along with changes in the business, the technological environment, the legal context and the fluctuation of socio-economic factors, organizations need to be able to standardize, streamline and adapt their operations to foreseen and unforeseen changes [11], [23]. As a result, process flexibility is required to accommodate changes in the environment and the need for evolving business processes [11], [24]. Such process evolution might be incremental, in terms of performing gradual changes, or revolutionary, in the context of implementing radical changes, with process innovation or process reengineering approaches [24].

Author(s)	Business Process Definition
Hammer and Champy [20]	“A business process is a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer.”
Davenport [25]	“A process is simply a structured, measured set of activities designed to produce a specified output for a particular customer or market.”
Jacobson [26]	A business process is “the set of internal activities performed to serve a customer.”
Stohr and Zhao [27]	“A business process consists of a sequence of activities. It has distinct inputs and outputs and serves a meaningful purpose within an organization or between organizations.”
Weske [10]	“A business process consists of a set of activities that are performed in coordination in an organizational and technical environment.”
Vergidis [19]	“A business process is perceived as a collective set of tasks that when properly connected and sequenced perform a business operation. The aim of a business process is to perform a business operation, i.e., any service-related operation that produces value to the organization”.
Chinosi and Trombetta [28]	“A business process (BP) is a set of one or more linked procedures or activities executed following a predefined order which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles or relationships”.

Table 1 - Business Process Definitions

Nevertheless, according to Mendling et al. [29], organizations are doomed to fail when they are unaware of the business processes that they need to support. Therefore, they operate inefficient processes that thwart them from fulfilling the demands of today's changing world, while they continuously fail to adopt the rapidly changing technology and follow the global competition [27]. Altogether, in [30], the authors conclude that organizations which endeavor to survive in the long-term, need to operate responsive and adaptable processes, denoting in that way the importance of process flexibility and the overall focus on business processes concept.

2.4 Process Modeling and the emergence of Business Process Model and Notation (BPMN)

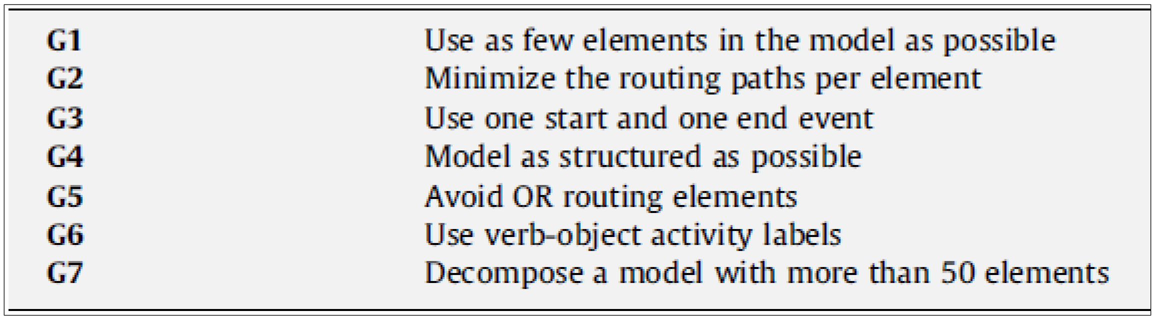
2.4.1 Process Modeling

With the prevalence and the acknowledgement of the importance of business processes both in academia and industry, process modeling has been in forefront for many years [27]. As business processes become more convoluted and extend over multiple organizations, process modeling is deemed to be powerful for documenting processes and eliciting valuable insights [11]. As a result, a plethora of modeling techniques have been proposed in literature, illustrating the heterogeneity in business process modeling [11]. However, their essence is quite similar, since they primarily focus on the ordering of activities. Their main objective is to transform ideas and observations around business operations, in a formal design and a correct representation [10], [12]. According to Kluza et al. [12], business process models assist an organization to visualize its processes with graphical representations. At the same time, Vergidis [19] articulates that "a business process design is the representation of a business process depicting the participating tasks and their connectivity patterns that determine the flow of the process. The aim of the design is to capture, visualize and communicate a business process". Constituting the means for a process design, visual notations, comprised of graphical symbols (visual vocabulary), a set of compositional rules (visual grammar) and the symbol semantics (visual semantics), are perceived to formalize every interaction within a business process [23], [31].

Undoubtedly, process modeling plays a vital role in communicating the process logic of business operations. Considering the powerful and highly parallel human visual system, visual representations can convey information more effectively and accurately than the ordinary language [31]. Human-oriented in nature, they communicate information that is more likely to be remembered by the human mind due to their picture superiority effect [31]. In this direction, Weske [10] explicitly underlines that graphical representations of business processes prevail over written organizational procedures or business policies, buried inside thousands of lines of code. Furthermore, due to the flexibility provided by explicit representations, changes to current

processes can effortlessly be translated to the process model level and be implemented to actual business process instances. In comparison with ambiguities arising from freeform textual descriptions, graphical representations do not leave room for misinterpretations [5]. In this sense, according to Mendling et al. [29], not only should the model of a business process be intuitive, but also unambiguously perceived. For this reason, modeling notations have to be cognitively effective, facilitating the information communication and the problem solving [31].
Making a reference to the work of Larkin and Simon [32], Moody [31] interprets cognitive effectiveness of visual notations as "the speed, ease, and accuracy which a representation can be processed by human mind".

In parallel with cognitive effectiveness, modeling competence, irrelevant from the selected modeling notation, is purported to be another critical aspect of process modeling [29]. Considering that ineffective diagrams can even be less effective than textual descriptions, correct process modeling becomes of utmost importance for communicating the desired process logic [31]. Towards this direction, Mendling et al. [29] propose seven modeling guidelines that can be utilized in the context of creating understanding diagrams. Summarized in Figure 4, they are mainly focused on the structuredness and complexity of visual representations in a way that extraneous cognitive load can be decreased [29], [33].



G1	Use as few elements in the model as possible
G2	Minimize the routing paths per element
G3	Use one start and one end event
G4	Model as structured as possible
G5	Avoid OR routing elements
G6	Use verb-object activity labels
G7	Decompose a model with more than 50 elements

Figure 4 - Process modeling guidelines [29]

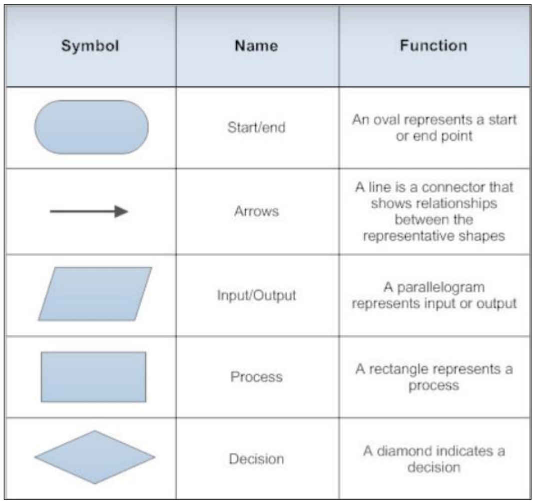
However, on the basis of process nature, process modeling is primarily tailored to the rendering of structured processes [5], [10], [24]. Conversely, complex knowledge-intensive processes, that are rather unpredictable and inclined to unfold during process execution, cannot be fully specified in terms of a process model. Mainly characterized by non-repeatability, unpredictability and emergence, the exact execution of a process instance is not known a priori and might alter during the process execution, posing in such way a challenge and difficulty in their modeling [24].

2.4.2 Process Modeling Techniques

Business process models are fundamental artifacts in today's organizations, providing the means for understanding, analyzing and improving business processes [21], [34]. Numerous notations and frameworks have been emerged as blueprints for rendering and documenting entire chains of activities, events and decisions [10], [23]. As a result, activity, event and control nodes constitute the main elements of modeling notations [5]. At their core, activity nodes are considered to be the cornerstone of every graphical notation, illustrating a work unit that is performed by a human or a software actor. In turn, event nodes represent milestones of process execution that are either triggered by the process itself or from the surrounding environment. Subsequently, control nodes form another integral component of modeling notations, capturing the execution flow decisions between notation elements [5].

Flowchart

Widely accepted, the first attempts to visually represent business processes were simple flowcharts [5], [19], [27]. In their basic format, flowchart diagrams are comprised of oval elements, which represent starting and ending points, rectangles, which denote activities, diamonds, which capture decision points, and arrows, documenting the sequence of the process flow [5]. The basic elements of flowchart modeling technique are rendered in Figure 5, before being utilized in a flowchart example (Figure 6), capturing a hiring business process. Nonetheless, considering their inefficiency to capture more complex processes, more standard approaches emerged [19].



Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point
→	Arrows	A line is a connector that shows relationships between the representative shapes
▱	Input/Output	A parallelogram represents input or output
▭	Process	A rectangle represents a process
◇	Decision	A diamond indicates a decision

Figure 5 - Basic Flowchart symbols [35]

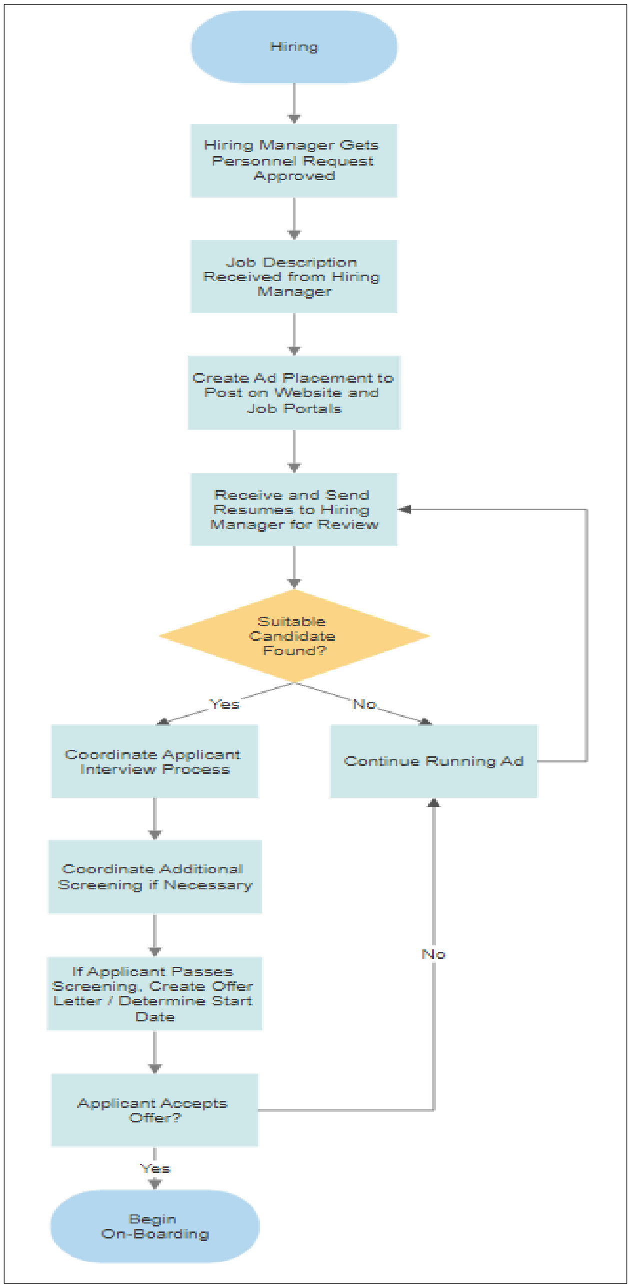

Hiring
Hiring Manager Gets Personnel Request Approved
Job Description Received from Hiring Manager
Create Ad Placement to Post on Website and Job Portals
Receive and Send Resumes to Hiring Manager for Review
Suitable Candidate Found?
Yes
No
Coordinate Applicant Interview Process
Continue Running Ad
Coordinate Additional Screening if Necessary
If Applicant Passes Screening, Create Offer Letter / Determine Start Date
No
Applicant Accepts Offer?
Yes
Begin On-Boarding

Figure 6 - Flowchart example [35]

IDEF0

One such, IDEF0 emerged as the first proposed method of IDEF family for functional modeling. Sponsored by the US Air Force Integrated Computer-Aided Manufacturing (ICAM) program, IDEF family of methods appeared in the mid-seventies for enterprising modeling and analysis, aiming to increase manufacturing productivity with the embracement of computer technology [36]. An IDEF0 diagram consists of activities, namely functions, and arrows, illustrating the data flow between activities. Interestingly, supporting the hierarchical modeling approach by representing functions at increasing levels of detail, functions can either be collapsed or expanded into lower-level graphs. In this way, functions are modeled in a higher-level of detail as “boxes”, displaying their inputs, outputs, triggers and mechanisms that are needed for their execution (Figure 7). Subsequently, expanded into a diagram at a lower level of detail (Figure 8), information that was not displayed at a prior higher level is well communicated [36].

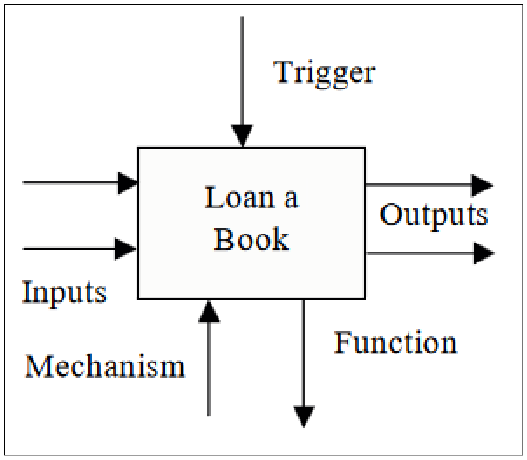


Figure 7 - IDEF0 function [36]

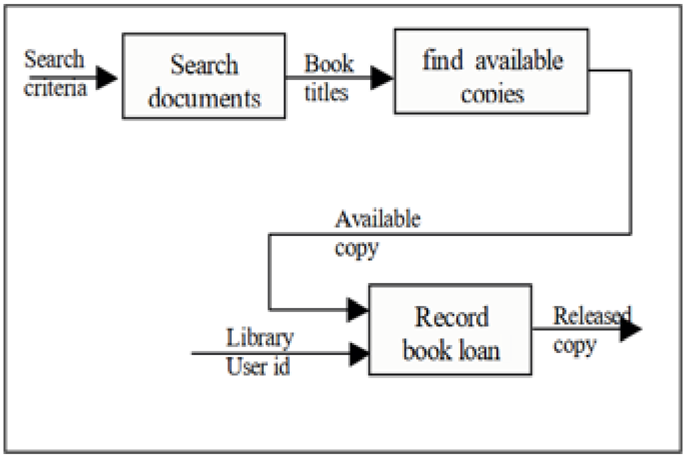


Figure 8 - IDEF0 diagram [36]

IDEF3

Since the first launch of IDEF0, the IDEF family was updated with new modeling methods. Supporting two fundamental aspects of behavioral modeling, namely the process execution and the state changes, IDEF3 constitutes one of the cornerstones of IDEF family, illustrating the behavioral view and the dynamic aspects of business processes [36].

In the context of IDEF3, models are deemed to be scenario-driven, where each scenario is depicted in two dimensions, with process-centered models, highlighting the sequencing of the process execution, and object-centered models, denoting the state changes [37]. In Figure 9, process and object schematic symbols, supported by IDEF3, are depicted. More specifically, process schematics are considered to be the most broadly utilized IDEF3 components. At their core, Unit of Behavior (UoB) boxes represent activities as numbered boxes. Additionally, links define the connection of UoB boxes, while junctions delineate the logic of process branching [37]. An example of process-centered IDEF3 model is depicted in Figure 10.

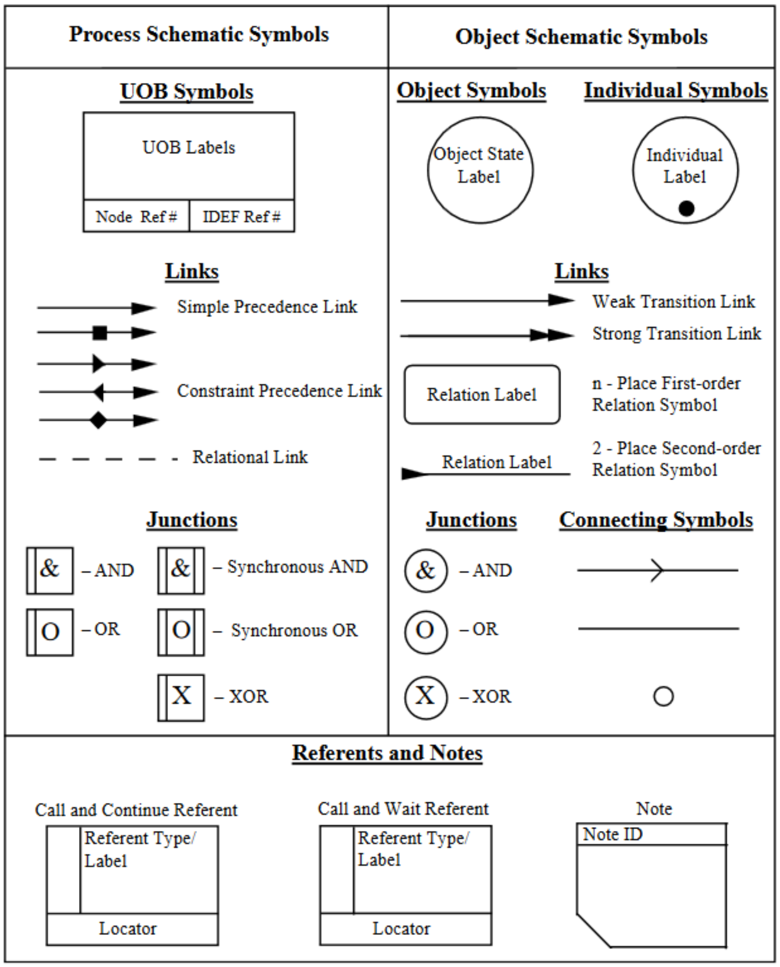


Figure 9 - IDEF3 elements [37]

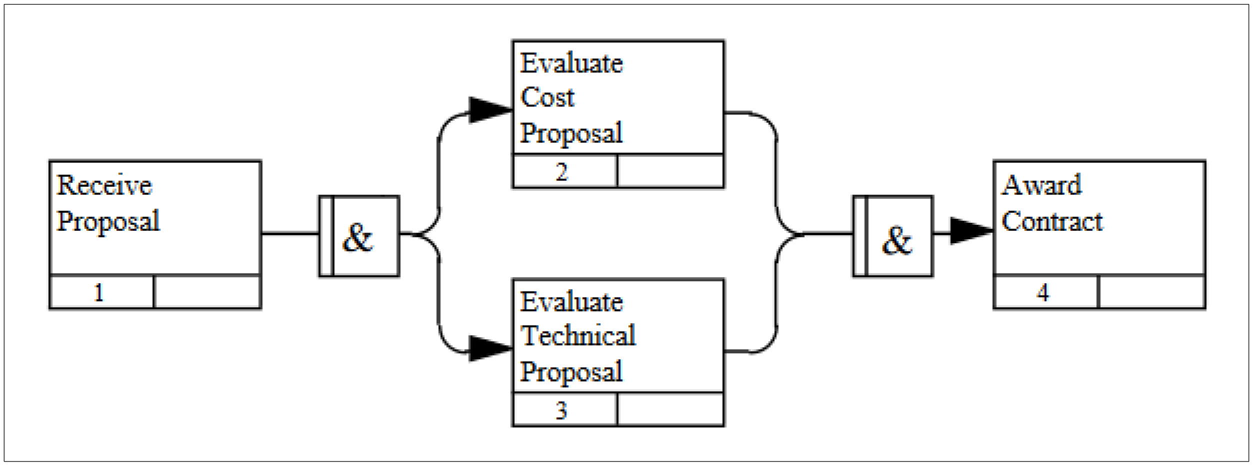


Figure 10 - IDEF3 process-centered example [37]

In parallel, except for process-centered models, IDEF3 is complemented by object-centered representations, where the attention is given to the state changes of various objects [37]. In this regard, a certain kind of object is represented by a circle, encompassing a label and a state. As process unfolds, objects move to new state, following transition schematics (i.e., links). Importantly, Referents (e.g., UoB boxes) can be attached to transition schematics, indicating that one a transition occurs an activity is involved, while transition junctions are utilized for delineating the state transition behavior [37]. In Figure 11, an example of object-centered IDEF3 model is depicted, illustrating a purchase request (PR) process example.

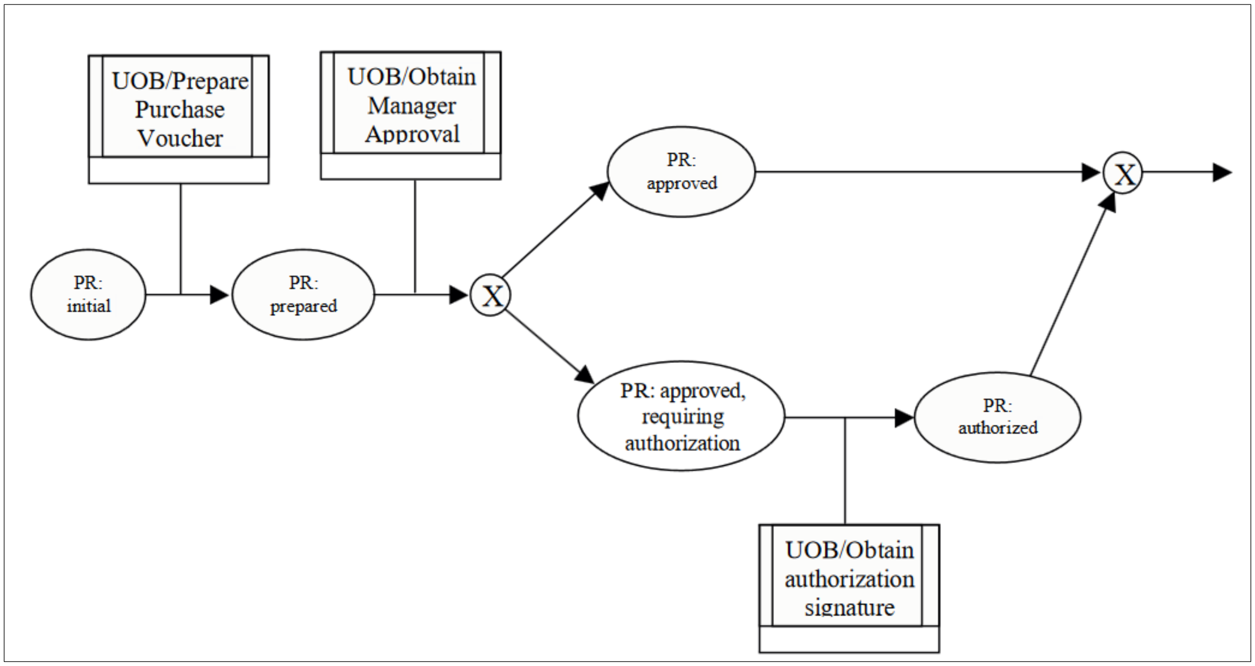

UOB/Prepare Purchase Voucher
UOB/Obtain Manager Approval
PR: initial
PR: prepared
X
PR: approved
X
PR: approved, requiring authorization
PR: authorized
UOB/Obtain authorization signature

Figure 11 - IDEF3 object-centered example [36]

Petri nets

Developed by Carl Adam Petri, in the context of his doctoral dissertation, in 1962, Petri nets constitute another fundamental process modeling notation [10], [11], [37]. As a modeling approach with a graphical representation and an equivalent mathematical formalization, Petri nets have been the basis for numerous modeling notations [10], [11]. More specifically, they consist of a small number of constructs, namely places, represented by circles, transitions, defined by rectangles, and directed arcs that connect places and transitions [10], [11]. Primarily, their power resides to the fact that they can model concurrency in business processes, denoting that a plethora of activities might happen in parallel [11].

A model represents a Petri net, which is a static structure of a process or system, while the dynamic behavior is rendered by tokens that reside on places [10]. On the basis of firing rules, tokens are transited from positions to positions when a transition is fired and a token exists in each of its preceding input places. In such scenario, a token is removed from each input place and one token is transited to each output place [10]. In Figure 12, a fired transition is displayed, while Figure 13 depicts a not enabled transition due to the lack of a token in one of the two preceding input places. Altogether, an order processing example is illustrated in Figure 14.

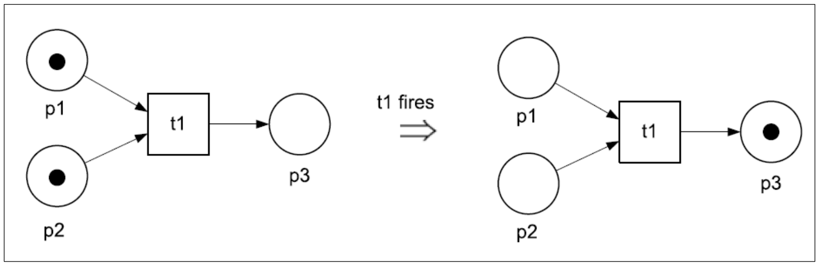


Figure 12 - Petri net: Fired transition [10]

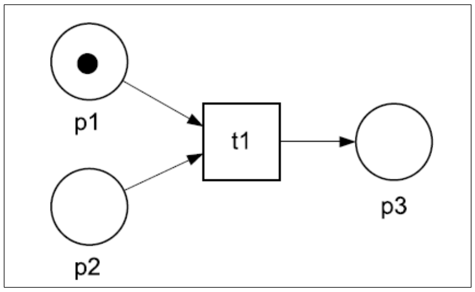


Figure 13 - Petri net: not enabled transition [10]

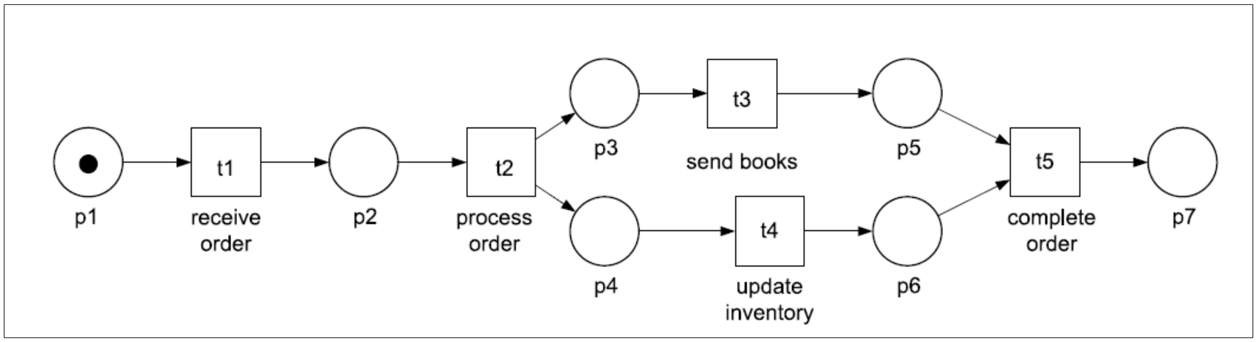


Figure 14 - Petri net: order processing example [10]

Event-driven Process Chains (EPCs)

Event-driven Process Chains (EPCs) constitute another wide-spread approach for modeling business processes [38]. In their basic format, they support three basic elements, namely events, functions and connectors [10], [37], [38]. More specifically, events, rendered by hexagons, represent state changes in an event-driven process chain [10]. In turn, functions, illustrated by rounded rectangles, determine unit of works, that receive one or more inputs and transform them to a series of outputs. Importantly, each function is triggered by one or more preceding events, while their completion leads to a number of event occurrences [10]. Subsequently, connector elements are utilized in order to represent the process logic and the logical relationships between process elements. In this regard, three main connectors are widely adopted, namely the AND (Λ), OR (V) and XOR (X) constructs [10]. In Figure 15, the basic EPC elements are rendered, before being utilized in an event-driven process chain, displaying an order processing example (Figure 16).

Altogether, EPCs bear some similarities with flowcharts, yet, they highlight events as fundamental components of business processes [5]. However, abstract in nature, they are primarily focus on representing domain concepts and processes, rather than formal aspects of technical implementation [10]. Due to the alternation of events and functions, the level of models' complexity might dramatically escalate, leading to cumbersome process representations [10], [37].

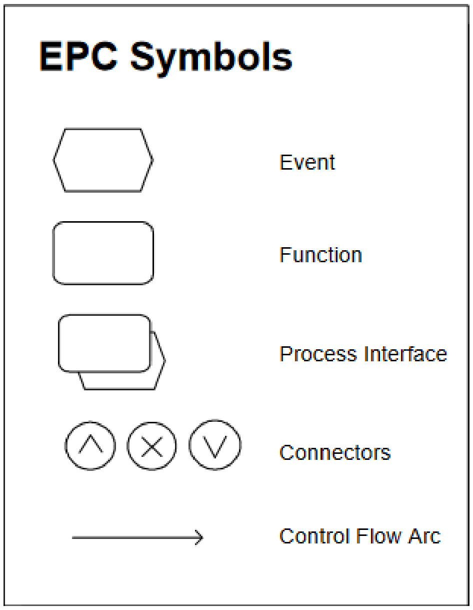


Figure 15 - Event-driven Process Chain (EPCs) basic symbols [38]

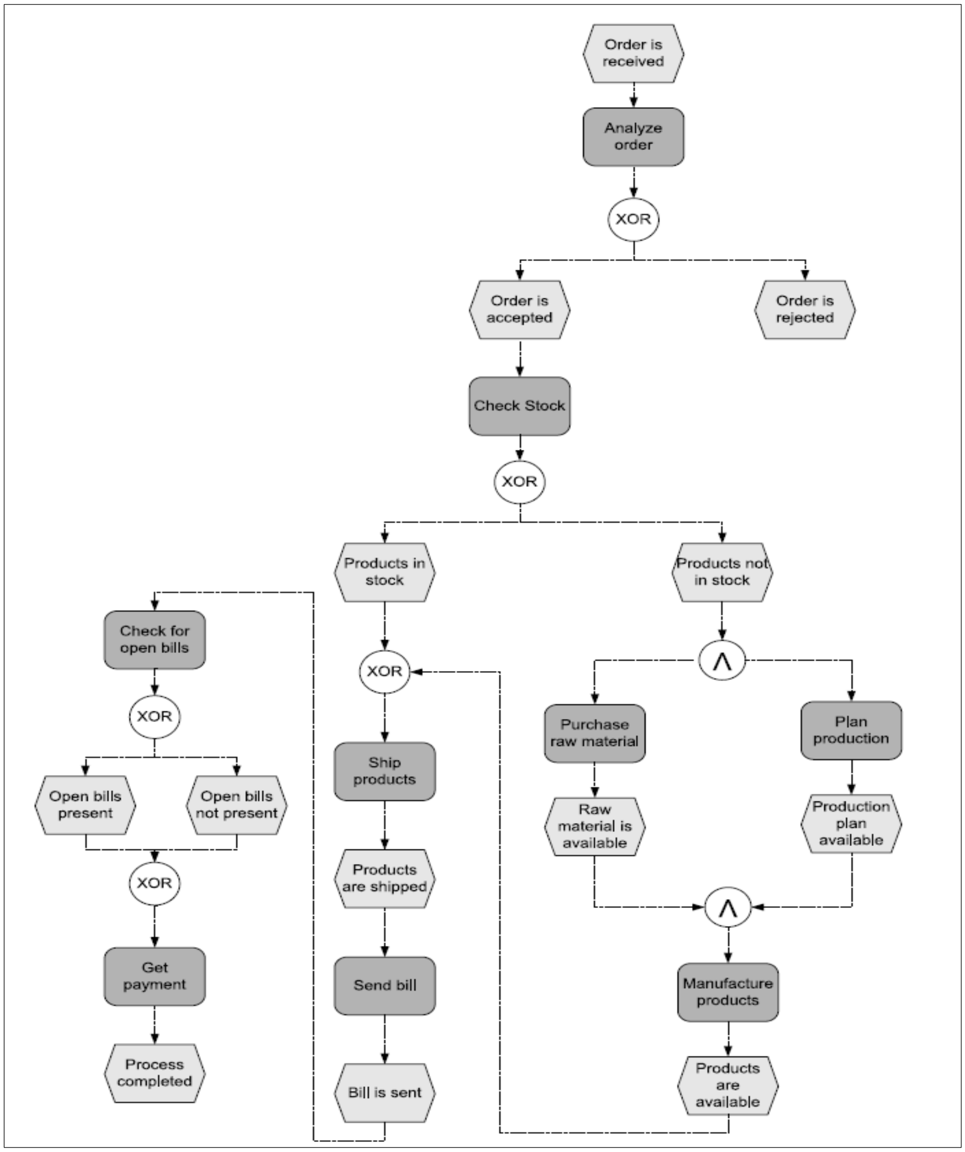


Figure 16 - EPCs order processing example [10]

2.4.3 Business Process Model and Notation (BPMN)

In the last years a plethora of modeling notations for business processes have emerged [23], [27], [28], [39]. The underlying fragmentation that was witnessed between existing tools and notations has engendered a need for a standard modeling language [28]. Business process models have been historically separated from their technical implementation, leading to numerous errors at the manual translation of visual representations to executable models [28], [39]. As a result, Business Process Model and Notation (BPMN) has come in forefront as an expressive, formal, understandable by various stakeholders, notation, that is semantically powerful to translate graphical diagrams to execution information [28], [40].

Inspired by a plethora of previous notations and methodologies [10], [39] BPMN was originally published in 2004 by the Business Process Modeling Initiative (BPMI) as a graphical notation to delineate the visual layout of business processes [28]. On the basis of the growing interest in academia and industry upon this notation, BPMN was adopted as an OMG standard in 2006 [28]. The initial versions (BPMN 1.X versions), aiming primarily at the visual representations of business processes, did not have a serialization format nor distinctly defined semantics [28]. With the advent of the BPMN 2.0 version, new features were added, resolving inconsistencies of previous versions [28]. In fact, BPMN 2.0 introduced a standardized serialization and interchange format, as well as standardized execution semantics for all BPMN elements. Hence, BPMN process models were able to be exchanged between various vendors, let alone be executed by different execution engines [28], [39], [41], [42]. With the portability and interoperability [42], introduced by its major second version, BPMN has been widely adopted as the de facto process modeling notation [39], while has been accepted as an ISO (ISO/IEC 19510:2013) standard [43] [44].

The overall goal of BPMN 2.0 was to encapsulate a modeling notation, a meta-model and an interchange format within one language [45]. This goal is also reflected in the official specification document [46], which states that "The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation".

Interestingly, BPMN is tailored to the modeling of entire end-to-end processes [39] in a Business Process Diagram (BPD) [28], [47]. Highlighting the control-flow of business processes [47], BPMN elements are divided into four main categories [10], [28], [39], namely Flow Objects, Connecting Objects, Swimlanes and Artifacts, as illustrated in Figure 17. Concretely, Flow Objects, comprised of activities, events and gateways, delineate the actions that can occur during a process execution and determine its behavior [28]. In turn, Connecting Objects, namely sequence flows, message flows and associations, define the connection between various objects [10], [28]. Subsequently, Swimlanes, namely pools and lanes, associate a set of actions with a specific resource, giving the capability of grouping modeling elements on the basis of each process participant's perspective [28], [48]. Last, Artifacts, comprised of data objects, groups and annotations, provide additional information to the process diagram without affecting the flow of the process [10], [28]. The aforementioned elements are displayed in an order processing diagram, depicted in Figure 18, while the expressive power [10] of BPMN is illustrated in Figure 19, by introducing a complete set of elements.

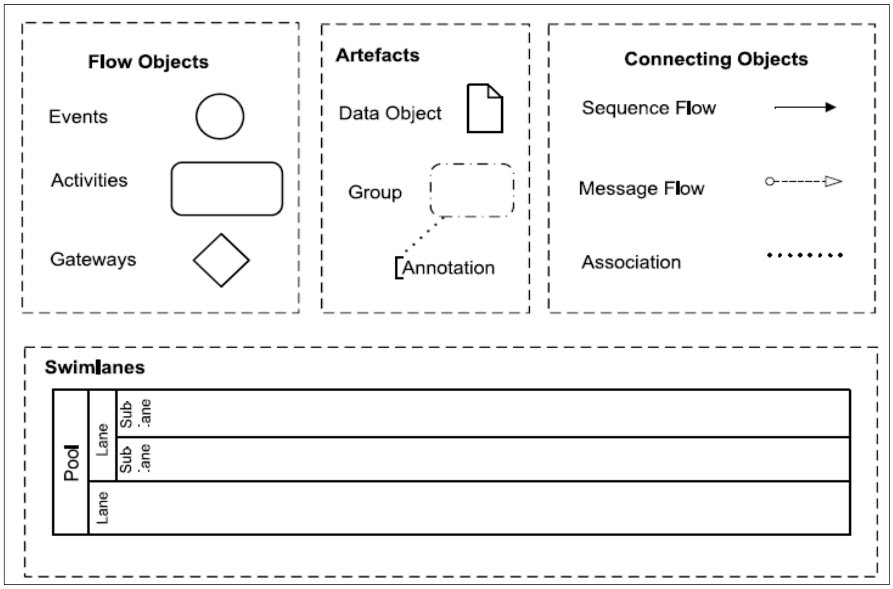


Figure 17 - BPMN basic elements [10]

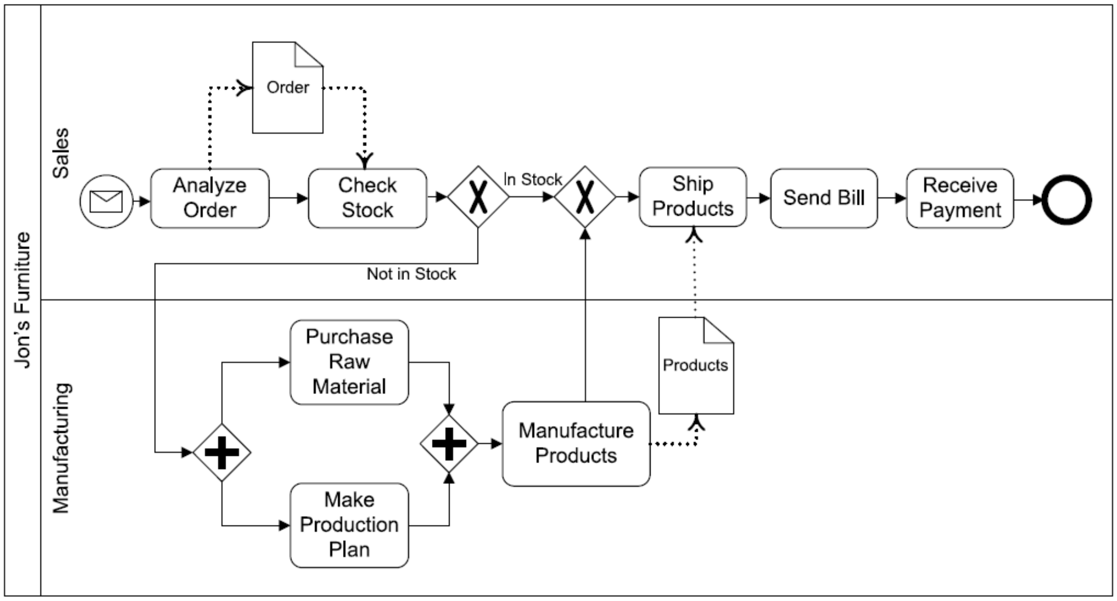


Figure 18 - BPMN order processing example [10]

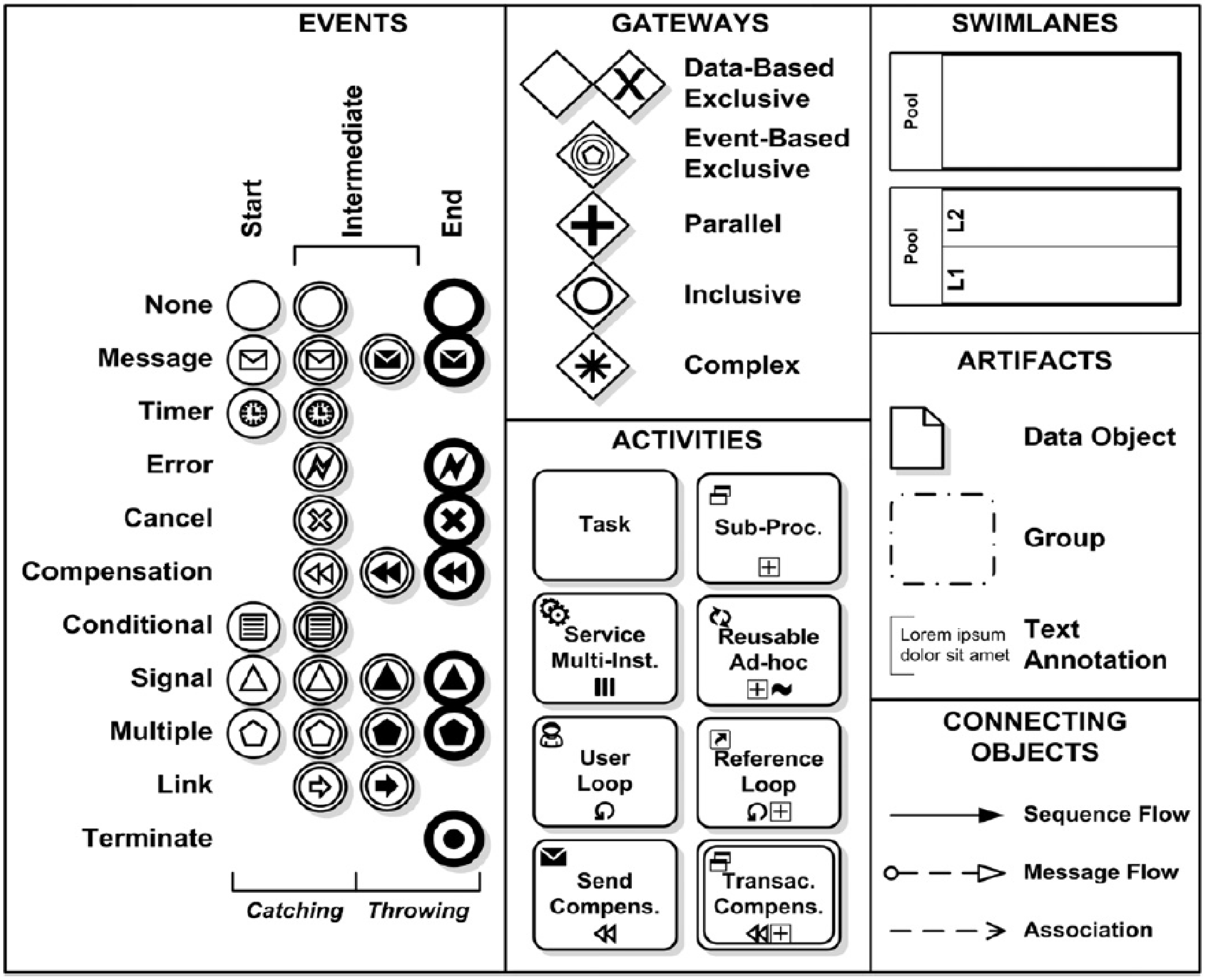


Figure 19 - BPMN complete set of elements [28]

Ultimately, since the standardization of BPMN and its prevalence as the de facto business process modeling language, a plethora of academic works denote its significant adoption both in academia and industry. However, due to the introduction of a plethora of modeling notations, the evaluation of available methods for process modeling is of utmost importance [45]. In this way, evaluation results can guide the users to select the optimal method on the basis of their needs [45]. For this reason, numerous academic works have evaluated the suitability and effectiveness of BPMN in different ways. To exemplify some, in [48], the suitability of BPMN for business process modelling is examined, using the Workflow Patterns [49] as an evaluation framework, while in [39] the cognitive effectiveness of BPMN is analyzed in the context of the Physics of Notations [31], a collection of evidence-based principles that together form a theory of notation design.

2.5 Separation of Concerns (SoC): Decision modeling and the emergence of Decision Model and Notation (DMN)

2.5.1 Separation of Concerns (SoC)

Even if decision-making is an integral part of BPM, BPMN notation does not support explicitly the modeling of decision logic in business processes [50]. Typically, as process execution unfolds, multiple decisions are taken [34]. Due to the intertwining nature of processes and decisions [23], [2], a convolution of decision and process logic is frequently encountered [51]. Considering that operational decision-making is embedded within the process models and encoded through control flow structures [52], [53], the complexity of BPMN models dramatically escalates, while posing a threat in maintainability, scalability and flexibility of both processes and decisions [34], [54].

Intuitively, decisions might span over multiple activities or even the entire process [54]. Utilizing a set of cascading gateways and a plethora of control flow elements [50], decision trees are generated within the process models [23]. More specifically, striving for emulating the decision logic of decision artifacts [51], [54], such as decision tables, complex routing structures are created [34], leading to the modeling of labyrinths [55] or spaghetti-like processes [34], [54], [56]. In such approach, maintainability of both processes and decisions is endangered, while a change in the decision logic has a paramount effect on the structuredness of the process model [50]. In Figure 20, a decision tree within a process model is illustrated, while a spaghetti-like BPMN model, due to the excessive usage of exclusive gateways, is depicted in Figure 21.

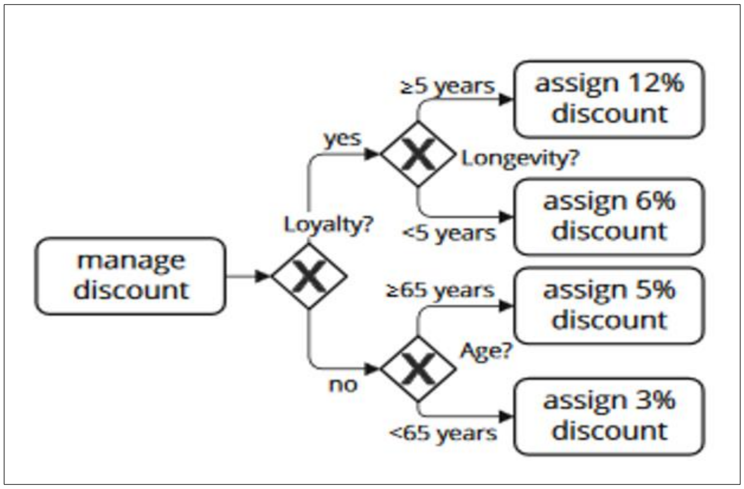


Figure 20 - Decision tree within a BPMN model [34]

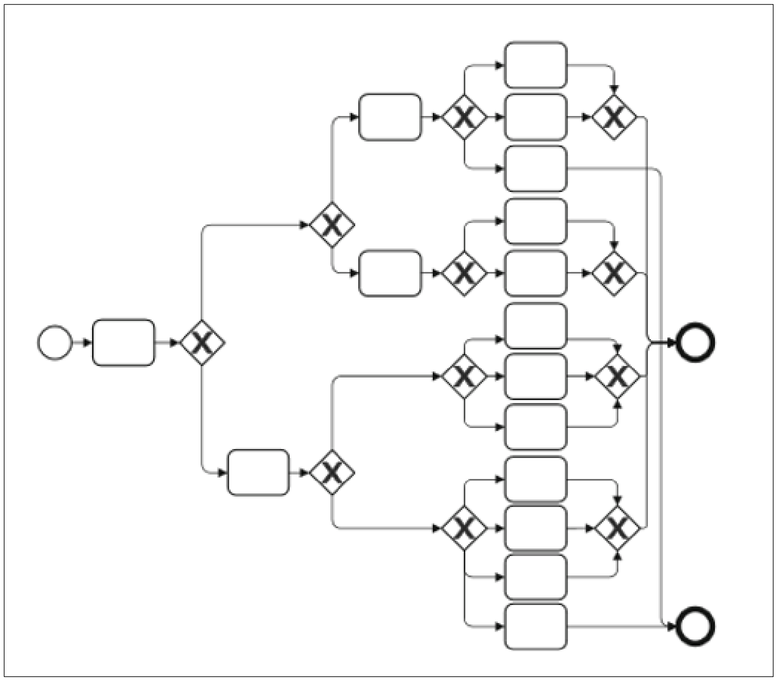


Figure 21 - Spaghetti-like BPMN model [34]

Recently, special attention is given to the decoupling of decision and process logic, advocating the Separation of Concerns (SoC) paradigm. In this sense, there is a vast amount of literature on Separation of Concerns approach in business process models [23], [34], [50], [52], [54], [56], [57]. To exemplify some, in [34], a three-step approach for process and decision logic separation was proposed with the ultimate goal of identifying decision patterns in process fragments and transfer them to decision models (Figure 22). Additionally, in [57], the authors proposed an approach of extracting decision models from process models, on the basis of utilized split gateways and event logs. Altogether, literature concludes that separating the concerns of decision and process logic, higher operational flexibility is well attainable [23], while readability, traceability and maintainability of both process and decision models are well refined [23], [24].

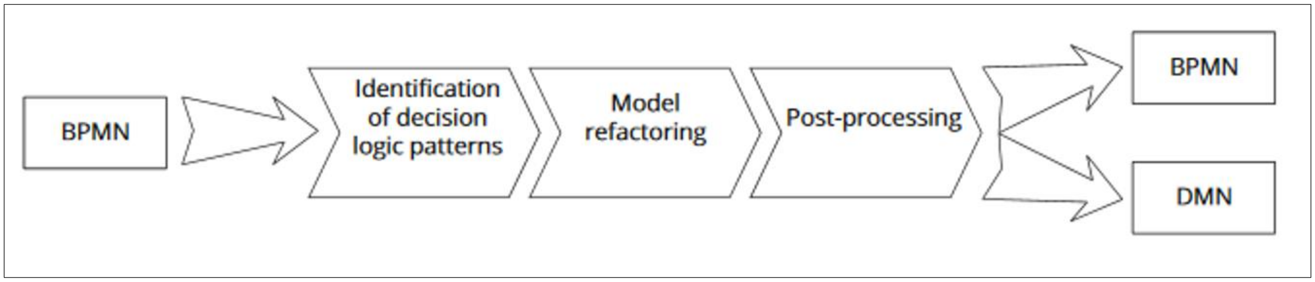


Figure 22 - Three-step approach for process and decision logic separation [34]

2.5.2 Decision Model and Notation (DMN)

Considering the booming interest in documenting, capturing and analyzing decision aspects of processes as a separate concern [54], Business Decision Management (BDM) has emerged as an elementary component of BPM, aiming at the systematic management of business decisions [2]. Since decisions constitute an integral part of every business operation, they are considered to be a first-class citizen in every BPM initiative [23], [33]. Concretely, a decision is defined upon a number of inputs that determine an output on the basis of the underlying decision logic [23], [52]. Typically, creditworthiness decisions, as well as acceptance and eligibility decisions, are frequently encountered within business processes [23].

For years, the modeling of business decisions was not considered to be a separate concern [52]. However, due to the growing interest in separating process and decision logic, decision modeling has consolidated as an approach to explicitly represent decision rules and their interrelationships [52]. In this regard, Decision Model and Notation (DMN), currently DMN 1.3, emerged as an OMG standard in 2015. According to its official specification [58], “The primary goal of DMN is to provide a common notation that is readily understandable by all business users, from the business analysts needing to create initial decision requirements and then more detailed decision models, to the technical developers responsible for automating the decisions in processes, and finally, to the business people who will manage and monitor those decisions. DMN creates a standardized bridge for the gap between the business decision design and decision implementation. DMN notation is designed to be usable alongside the standard BPMN business process notation.”

Admittedly, DMN finds its origin in decision table modeling, where the decision logic was represented with a means of tables [33]. Tailored to the modeling of repetitive operational decisions, characterized by high frequency and process orientation, operational decision-making comes in forefront [33], [55]. Currently, DMN constitutes the standard notation for modeling decisions within organizations [50], playing a crucial role in knowledge-intensive and decision-intensive processes [51]. DMN has been successfully adopted both in academia and industry [33], [51], receiving increasing attention as a means of externalizing decisions from process flow and automating decision enactments for processes [54], [59].

According to DMN, a decision determines an output on the basis of the evaluation of predefined inputs [54]. The DMN standard defines two levels of decision modeling, namely the decision requirement level, represented by the Decision Requirement Diagram (DRD) and the decision logic level, captured by decision tables [34], [51], [54], [56]. More specifically, a DRD constitutes a visual representation of business decisions, emphasizing the input details and the decision interrelationships, while facilitating the hierarchical structuring of decisions [33], [54]. Comprised mainly of decision nodes, represented by rectangles, input data, depicted by ovals, as well as information requirements, rendered by arrows, DRD representations capture the decision logic in an intuitive way, bridging the gap between business and IT. Delving into the second level of decision modeling with DMN, decisions are captured in a tabular form, namely in decision tables, where decision rules are defined, consisting of conjunctions of input and output parameters [33], [34]. Considering that such decision tables might be evaluated further by decision engines, DMN provides the declarative FEEL (Friendly Enough Expression Language) expression language for the notation and the syntax expressions of the decision logic [50], [51]. In this regard, DMN constitutes an executable notation, where decisions are captured at a logical level and the semantic constructs of DMN are defined by a formal XML meta-model [2], [33], [52]. As a result, decision engines are able to interpret DMN models by reading XML files and execute the models directly, paving the way for the automation of decision-making processes [2].

In Figure 23, a DRD is rendered, illustrating decision requirements and interrelationships for a loan eligibility assessment, while in Figure 24, the decision logic level is displayed on terms of predefined inputs (i.e., Risk Category) and outputs (i.e., Credit Contingency Factor) of a DMN decision table.

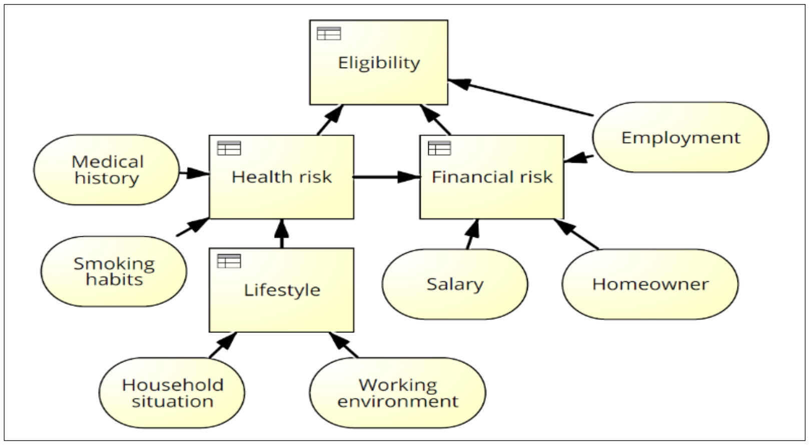
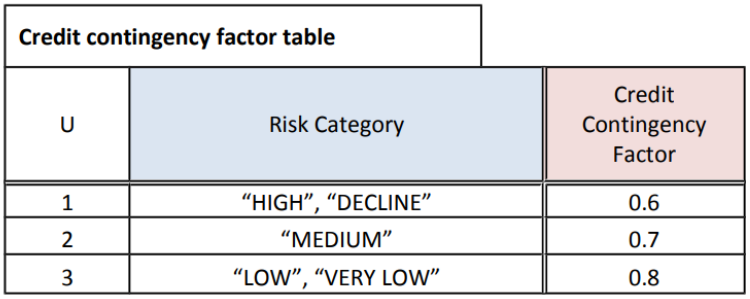


Figure 23 - DRD: loan eligibility assessment [50]



Credit contingency factor table

U	Risk Category	Credit Contingency Factor
1	“HIGH”, “DECLINE”	0.6
2	“MEDIUM”	0.7
3	“LOW”, “VERY LOW”	0.8

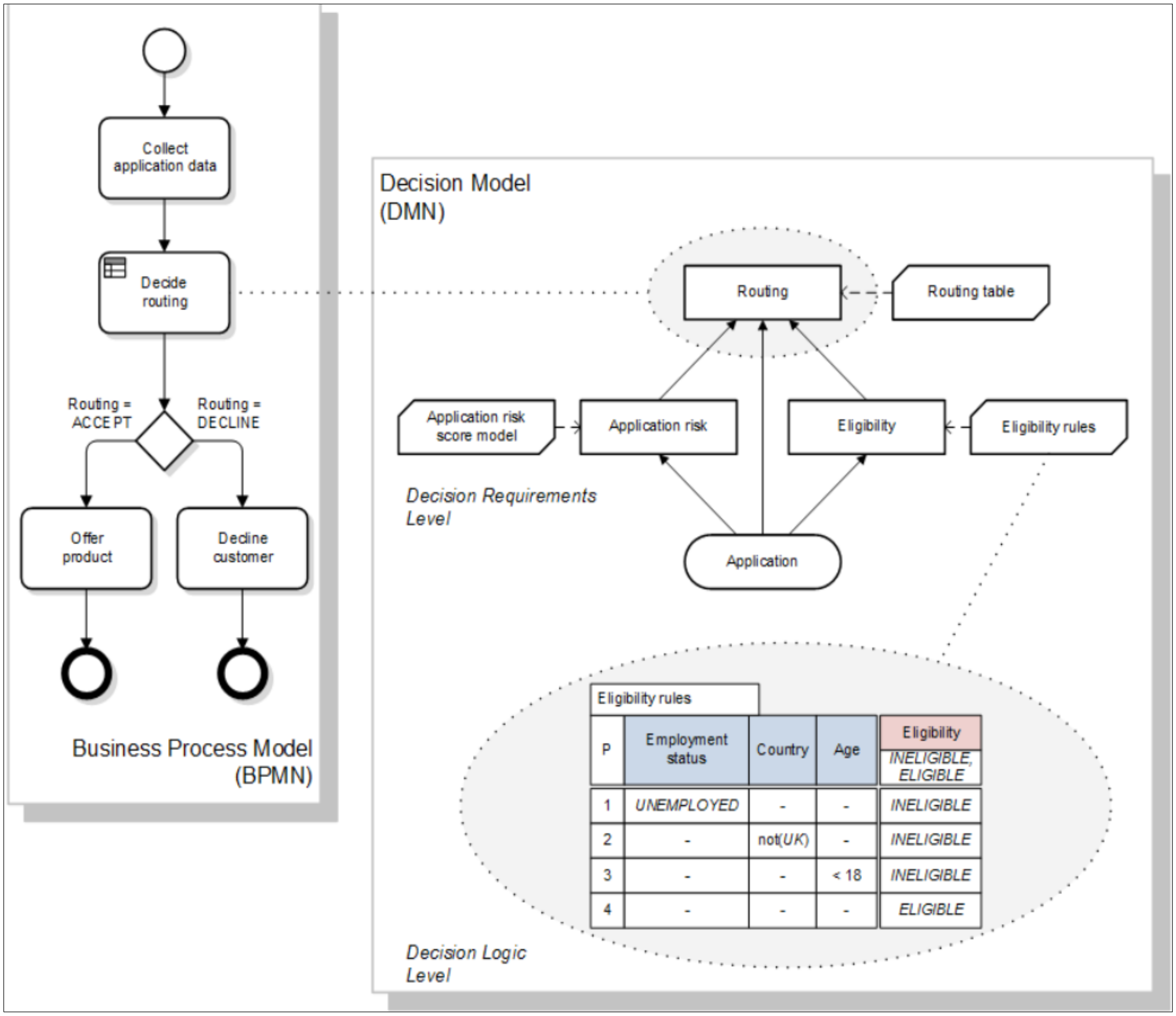
Figure 24 - DMN Decision Table [58]

2.5.3 BPMN and DMN integration

With the introduction of DMN, decisions can be separated from business processes, applying the principle of Separation of Concerns (SoC) [55]. However, with its excessive adoption both in academia and industry, research challenges arise on whether decisions and processes can be separated, yet, consistently integrated [33], [54]. Considering that DMN is designated to be a complementary standard to the BPMN notation [58], emphasis is placed on process and decision management integration [50]. In this regard, decisions can be externalized and encapsulated in separate DMN models and at the same time linked to the invoking context of business processes that are rendered in BPMN models [51]. As a result, decision outcomes can be utilized and returned as inputs to the process control flow, facilitating the integration of process and decision management [50].

A considerable amount of literature has investigated the combination of BPMN and DMN standards. The BPMN notation is considered to highlight the business logic of processes, determining which activities can be executed and by what order, while the DMN notation is associated with the decision logic of business operations, delineating the decision results [34]. In [55], the authors propose the business rule tasks of BPMN standard, as an instrument of invoking the decision logic of DMN models and replacing the cascading gateways of BPMN models. Hasić et al. [54], introduce five principles (5PDM) for integrated process and decision modeling, while in [51], the aforementioned principles are utilized in order to render processes decision-aware and decision-intelligent. Interestingly, in [59], adhering to the paradigm of Service-Oriented Architecture (SOA), Decision as a Service (DaaS) paradigm is introduced, presenting decisions as externalized services that processes need to invoke in order to receive their decision result. In [56], the authors investigate on IoT processes the advantages of BPMN+DMN paradigm, over the BPMN approach, concluding that aggregation of context information, as well as scalability and flexibility are well refined.

Overall, the integration of business processes and decisions is expected to be a cornerstone in business process management in the years to come [33]. Importantly, process and decision models can have their own consistency [55], yet ideally be integrated, considering each other as a black box [55] and defining merely the required input and output parameters [50]. Process model complexity, based on model-size based metrics, is deemed to be decreased [56], while traceability and transparency of decision-making processes are significantly refined. However, new challenges arise in their modeling consistency, when modifications on decision models have impact on the process models and vice versa [50], [55], while additional complexity is introduced due to the introduction of additional decision models [56]. In Figure 25, the BPMN and DMN integration comes in frontline, by the means of invoking a DMN decision from a BPMN business rule task.



Eligibility rules

P	Employment status	Country	Age	Eligibility INELIGIBLE, ELIGIBLE
1	UNEMPLOYED	-	-	INELIGIBLE
2	-	not(UK)	-	INELIGIBLE
3	-	-	< 18	INELIGIBLE
4	-	-	-	ELIGIBLE

Figure 25 - BPMN and DMN integration [58]

2.6 Executable process models

Graphical representations of business processes highlight the visual structure of models, rather than shedding light on their technical aspects [10]. Primarily, they are tailored to describing and documenting business processes in a human-readable way [3]. On the other hand, executable models are precisely designated in a way that a computer is able to interpret and execute it [3]. Lying at the foundation of Model-Driven Engineering (MDE), executable business processes constitute the intersection between BPM and Software Engineering (SE) [3].

Traditionally, general-purpose programming languages that interpret the flow of process models into programming code [3] have driven the implementation and execution of business processes. In this sense, their actual execution is expected to deviate from the desired behavior, posing a threat on their optimal enactment [42]. Additionally, due to the need to administer and implement differentiating and personalized business processes, organizations need to hard code their needs, that cannot be covered by default processes, delivered by standard software (e.g., SAP) [3]. As a result, they typically resort to code their unique needs, utilizing environment's programming languages or proprietary development environment, in order to implement their personalized operations [3].

However, with the advent of BPMN 2.0 and its introduction to token-based execution semantics and an XML serialization, a new way of implementing processes emerged [3], giving prominence to execution capabilities of a modeling language [28]. In this regard, BPMN constitutes a notation where visual details are combined with technical specifications within the same process model [41], [54], [60]. As a result, BPMN models remain effortlessly readable for business users, while all technical terms are hidden in the background as XML code [2]. In fact, a clear mapping of descriptive business processes to executable ones is facilitated [3], while the distance between the desired and the actual behavior of process execution is minimized [42].

Along with BPMN, DMN notation constitutes another modeling language, where all execution properties are stored in the background of decision models as XML code [58]. With the process and decision enactment, all activities and decisions are performed, following strictly the execution constraints that are specified in an XML format [10]. However, converting a conceptual model to executable one, details that were omitted for the sake of simplicity and understandability of the visual representation, should be specified [11], [3]. Prior to the execution of process models, bridging the diverging level of granularity between conceptual and executable models, is of utmost importance [5]. Additionally, all ambiguities, encompassed in the conceptual model need to be detected, preventing problems at the runtime [11], [42]. Overall, Dumas et al. [5], propose a five-step method to incrementally transform a conceptual process model into an executable one, which are summarized to: 1) Identify the automation boundaries, 2) Review manual tasks, 3) Complete the process model, 4) Bring the process model to an adequate level of granularity, and 5) Specify execution properties. In such an approach, conceptual process models are gradually transformed to IT-oriented, laying the foundation of process execution.

2.7 Workflow Management Systems (WfMSs) and Business Process Management Systems (BPMSs)

With the rising interest in executing business processes, Workflow Management Systems (WfMSs) have come in frontline for the effective coordination and faster execution of business processes [61]. The concept of workflow has been around for years, while the first commercial workflow management systems were introduced in the early nineties [10], [61], [62]. According to the Workflow Management Coalition and its Workflow Reference Model, introduced in 1995 [63], a workflow is perceived to be "The computerized facilitation or automation of a business process, in whole or part", while a WfMS is defined as "A system that completely defines, manages and executes "workflows" through the execution of software whose order of execution is driven by a computer representation of the workflow logic". In this regard, a WfMS is able to perform a series of coordination activities, according to procedural rules that are predefined in an executable process model [27].

At its core, a workflow engine, lying at the workflow runtime enactment service, takes care of the coordination, control and execution of workflows [11]. Acting as a centralized agent that controls process orchestrations [10], an engine interprets executable models, as these models were the source code of software solutions [2]. While, the terms of workflow engine, process engine and execution engine have been utilized interchangeably in literature, for the premise of this thesis the workflow engine term is primarily adopted.

Interestingly, in most enterprise application systems, such as ERPs, workflow components are embedded as integral parts of their software (Figure 26). Running a set of executable workflow models, they are able to get customized and provide processes to a higher architecture level, namely the graphical user interface [10]. However, instead of focusing on workflow components of single applications, a WfMS can be additionally perceived as a "middleware" system that orchestrates the integration of diverse applications, such as mainframe legacy systems and ERP applications [27] (Figure 27). Importantly, high degree of flexibility is engendered, considering that modifications of process models change the behavior of a WfMS. In such a model-driven approach, business processes are enacted directly, without hard-coding any changes [10].

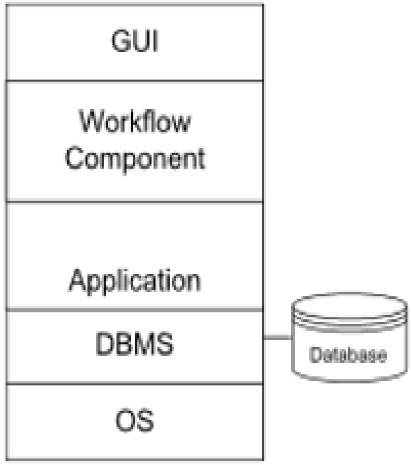


Figure 26 - Workflow components in enterprise application systems [10]

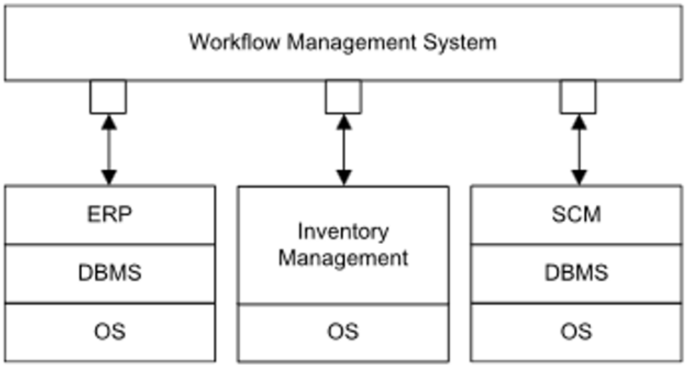


Figure 27 - WfMS as middleware system [10]

Currently, with the rising interest in executable BPMN 2.0 business processes, a new generation of proprietary and open-source workflow engines has emerged, supporting and driving process execution according to BPMN 2.0 specification [3], [39]. At their core, a workflow engine interprets and executes the process logic of a predefined BPMN 2.0 model, ensuring that the right information reaches the right person or computer application at the right time [5]. In such a scenario, the workflow engine is aware of the process control flow, knowing at any time what has to be performed next [2]. Hence, the WfMS is able to transfer work items to user participants, execute software tasks, as well as invoke third-systems by utilizing web-services and different messaging protocols [3], as specified in the process model [10] (Figure 28).

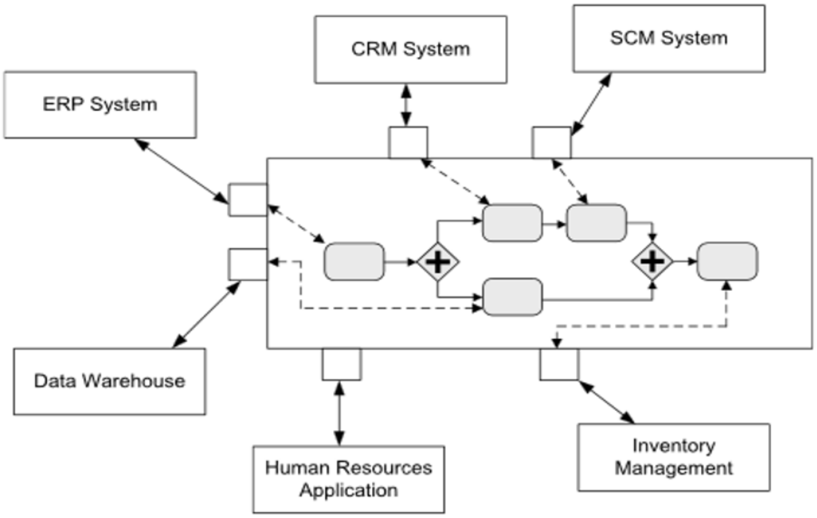


Figure 28 - System integration with a BPMN 2.0 workflow engine [10]

With the evolution of business processes and the need to monitor, administer and optimize them, contemporary Business Process Management Systems (BPMSs) have gradually replaced WfMSs. A BPMS constitutes an enhanced variant of a WfMS, focusing not only on process execution, but also on process design, process administration, instance monitoring and process optimization [13], [18]. Weske [10], emphasizing on a BPMS's capability in process enactment, explicitly defines that "A business process management system is a generic software system that is driven by explicit process representations to coordinate the enactment of business processes". With a more comprehensible perspective of a BPMS, Reijers [8] articulates that "A BPMS is typically described as a piece of generic software that supports activities such as the modeling, analysis and enactment of business processes". Mainly comprised of an execution engine, a process modeling tool, a worklist handler and an administrative and process monitoring tool, a BPMS lays the foundation for process transparency, traceability, automation and reduced lead times and hand-off errors [5], [8].

Ultimately, contemporary organizations have a current demand and impetus to incorporate BPMS platforms in their daily operations [13]. Today, there is a plethora of BPMSs, either proprietary or distributed as open-source BPM platforms [12], [60], that extend the functionalities of earliest generations of WfMSs and provide more sophisticated build-time and runtime capabilities, as well as enhanced features for system integration [8]. Notwithstanding a plethora of advantages introduced by BPMSs, the low level of process orientation within organizations, is characterized in literature as an influential and inhibitory factor for the success of every BPMS initiative [8].

2.8 Workflow and Business Process Automation (BPA)

While benefits can be derived from every BPMS initiative, there are potentially more to be acquired if such systems are utilized as driver forces for the automation of key business processes [62]. Process automation has been around for years, from the advent of the industry revolution and the introduction of factory machines [64], where activities conducive to automation were transposed to machine production [30]. In the context of organizations, traditional information technology (IT) has been historically the key enabler for work automation [64]. From the early 1970s and the introduction of computing technology in the office work, to the office automation prototype in the 1990s, the aim was to eliminate paperwork and facilitate the automation of a large dimension of office load [15]. With the excessive adoption of information systems in the workplace, processes became partially or totally automated, focusing on production efficiency and cost reduction [22].

In today's changing market, everything has to be performed faster with greater flexibility and agility [65]. In this context, Business Process Automation (BPA) is perceived to refine organizational efficiency by eliminating redundant procedures, restructuring human capital and implementing software solutions with the ubiquitous adoption of Information Technology (IT) [53], [66], [67]. As the art and science of executing automatically tedious and repetitive tasks, special attention is given on eliminating the superfluous and non-value adding activities, while focusing on operations that truly create value [66], [68]. Additionally, automation of business processes is closely-related with workflow automation. Leading to minimized error rates, user satisfaction and reliability, workflow automation provides unprecedented opportunities to orchestrate work and increase performance [27]. The main objective of workflow automation is to automate business processes and decisions with the leverage of a workflow engine [27]. In such an approach, computerized information flows execute faster routine tasks and decisions without the need for human intervention [66]. As a result, work units are automatically assigned to human participants and software applications, progress can be monitored and events are automatically logged, serving for analysis and redesign purposes [62].

Taking into account that the number of processes inside organizations dramatically increases, a need has engendered to support business processes by means of automation techniques [69]. Process automation, thus, is a digital transformation imperative, driving growth and efficiency in contemporary organizations. With rapid changes in how business operations are performed and the need to deliver service more efficiently [70], process automation is purported to speed operations and facilitate automated process execution and decision-making [66].

2.9 Ubiquitous Penetration of Information Technology (IT) and the Digital Transformation (DT) of today's organizations

Having highlighted the BPA concept, Information Technology (IT) and Digital Transformation (DT) of today's organizations are deemed to be the key enablers and driving forces for such initiatives. Since information systems are rendered to be the focal pillar of every business operation, IT has ubiquitously penetrated all concepts of BPM [10]. Considering their evolving role, from their supporting nature in office automation to today's excessive prevalence, IT has changed the way of doing business [71]. With the perpetual emergence of new software applications and innovative technologies, organizations strive to enhance the user experience, rationalize their business operations and launch innovative products and services [72]. Interestingly, while IT supports individual tasks in isolation, organizations are more efficient if systems are integrated in every part of their operations [3]. As a result, IT is a key instrument to refine business operations and reinvent process automation [5]. However, due to the extensive prevalence of IT systems in the organization's ecosystem, such as the Enterprise Resource Planning (ERP) systems, the Customer Relationship Management (CRM) systems and the Supply Chain Management

(SCM) systems, to exemplify some, there is a current impetus towards an optimal system integration, in a way that information exchange and system consistency is facilitated [5].

Given the fact that organizations operate in a dynamic environment, the ability to adapt core business operations to the changes in the environment, is a key factor for competency and satisfying the ongoing, dynamic needs of customers [16], [65]. Maintaining that competitive advantage in today's era of global digitalization is more crucial than ever before [3]. However, the mere embracement of IT, without harmonizing the organization's strategies with the modern disruptive and innovative technologies, can lead enterprises to be out of business very soon [15]. Therefore, Digital Transformation (DT) has come in forefront as an approach of creating new business models that entail radical changes in the way of doing business, driven highly by the adaptation to the digital age [72]. For the premise of this thesis, the following definition of DT is well adopted, since it highlights process automation as an integral part of digital transformation initiatives. According to Clohessy et al. [73], "DT is concerned with the changes digital technologies can bring about in a company's business model, which result in changed products or organizational structures or automation of processes".

Today, with the outbreak of COVID-19 pandemic, digital transformation is deemed to be more imperative than ever before [74]. Considering the remote work, the changing working conditions, as well as the force to traditional companies to go online, the COVID-19 crisis has come to accelerate the digital transformation of today's organizations and consolidate the process automation initiatives, assisting them to stay competitive and avoid short-term economic collapse [70], [74].

2.10 Summary

This chapter presented the theoretical background of this thesis, laying a robust foundation for its practical scope. Process thinking notion is deemed to constitute a focal pillar of every process-oriented organization, seeking to remain competitive and refine its key operations. With the emergence of BPM paradigm and the focus on business processes concept, a plethora of modeling techniques came in frontline, serving as the means for rendering business interactions in a more perspicuous way. One such, BPMN has been established as the de facto process modeling technique. However, considering that decisions constitute a separate concern in process modeling, DMN emerged as the standard notation for modeling decision-making processes and decoupling the decision logic out from the process logic of BPMN models. Executable both in nature, their integration can serve as the source code for every BPMN and DMN based workflow engine, lying at the core of WfMSs and their enhanced variants, namely BPMSs. Altogether, process and decision enactment by a workflow engine, enhances the automatic execution of operational processes and decisions, while process automation is well attainable, facilitating, thus, the digital transformation imperatives of today’s organizations.

CHAPTER 3: Modeling business processes with BPMN

This chapter delves into the Business Process Model and Notation (BPMN) and its expressive power for modeling business processes. The aim is to establish a robust understanding of its fundamental construct elements, to elucidate more advanced BPMN elements and modeling concepts, as well as to explicitly denote the modeling power that process modelers have at their disposal. The chapter starts with an introduction to BPMN notation, as a standard modeling technique for modeling business processes. Thereafter, presenting a set of BPMN elements, special focus is given on their correct utilization inside BPMN models. Throughout this chapter, advanced BPMN modeling comes in forefront, leveraging the full BPMN element armory. In this regard, advanced process concepts, such as interrupting and non-interrupting boundary events, exception handling and transaction compensation, frequently omitted in most modeling initiatives, are meticulously rendered. Finally, the chapter concludes with the inability of BPMN notation to render the decision logic of business processes and its subsequent complexity escalation. Overall, for the premise of this chapter, the main source of information is educed from the BPMN 2.0 specification [46], from Dumas et al. and their work “Fundamentals of Business Process Management” [5], as well as from Camunda’s CEOs and their work “Real-Life BPMN” [2], imparting both an academic point of view and an industrial perspective around the BPMN spectrum.

3.1 Introduction to BPMN

Business Process Model and Notation, commonly abbreviated as BPMN, was originally introduced in 2004 by the Business Process Modeling Initiative (BPMI) as a graphical notation for modeling business processes [28]. In 2006, BPMN was adopted by Object Management Group (OMG) and since then it is considered to be the de facto modeling notation for rendering business interactions [39]. With the advent of BPMN 2.0 in 2011, execution semantics of BPMN elements were introduced in a standardized format [46], giving rise to its higher adoption both in academia and industry for modeling and executing business operations. As such, BPMN was accepted as an ISO (ISO/IEC 19510:2013) standard [44], [49], establishing itself as the most prevalent notation in the BPM spectrum.

Being a method of unambiguous communication between systems, IT and non-technical business analysts [12], BPMN is deemed to bridge the gap between business and IT [2], [5]. In this regard, both strategical and operational details are able to be captured in a single BPMN model [2], [46], facilitating the common understanding of various stakeholders, ranging from process participants, who perform the process, to process analysts, who graphically render the business process, and to process engineers, who will ultimately implement the process, leveraging technology [2].

Establishing a strategic BPMN process model, an immediate understanding of the process logic is communicated to all process stakeholders. In the course of BPM lifecycle, the operational BPMN process models come in frontline, as human and technical interactions are rendered. In this way, leveraging the powerful BPMN symbol armory, process analysts are able to meticulously capture the process logic in their models, while any technical details, necessary for process execution and automation, are hidden in the underlying BPMN model as XML code. Thus, BPMN models remain effortlessly readable for the business users, as any technical terms, indispensable to their execution, are hidden in their background [2].

Ultimately, upon rendering a process model in BPMN, process analysts might contemplate the power that have at their disposal. A full understanding of BPMN semantics, thus, is an imperative foundation so as to reap its powerful potentials [2]. Ranging from fundamental to advanced BPMN modeling, notation's elements are divided in main categories, which highlight a different aspect of business operations (Figure 29). At first, initial focus is given on process participants, utilizing the BPMN pools and lanes elements. Subsequently, shedding light on process behavior, BPMN activities, gateways and events, emerge as fundamental components of every BPMN model. Moreover, denoting the flow of the process and participants' interactions, BPMN sequence flows, message flows and association flows, come in forefront as integral parts of every BPMN model. Additionally, rendering the data flow of business processes, BPMN data objects and data stores, are widely adopted, while BPMN artifacts, like text annotations and groups, increase the comprehensibility of process models. Altogether, with the correct BPMN modeling of business operations, misinterpretations and ambiguities, arising from freeform textual descriptions can be well mitigated, while paving the way for a more straightforward communication within an organization [5].

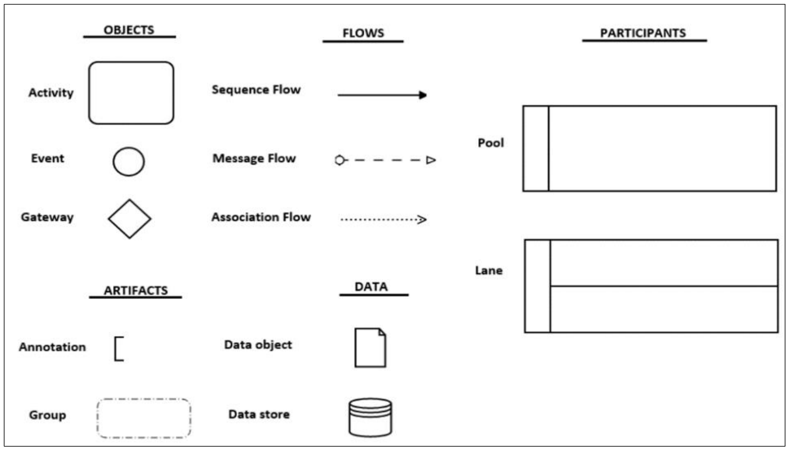


Figure 29 - BPMN elements [2]

3.2 Rendering participants in process models: BPMN pools and lanes

As business processes inside organizations become more complex, in the vast majority of cases, more than one entity, namely process participant, is involved during their operation. Utilizing the BPMN notation, a process participant is explicitly captured by a pool element, illustrated as a rectangle. Each pool, representing the highest-level of control in business processes, symbolize the process boundaries for a specific participant. Depending on the desired level of granularity, the process flow for process participants might be hidden, by transforming their expanded pools as "black boxes" and collapsed elements.

Interestingly, shedding light on the way that various participants can interact with each other, as the process unfolds, BPMN pools can communicate and interact, leveraging the BPMN message flows [5]. Being illustrated as dashed lines that are strictly utilized outside of BPMN pools, they primarily denote the direction of messages between different process participants. In this regard, an extremely important aspect of BPMN arises, considering that even if participants interact within the same end-to-end process, every participant has a different understanding and perspective of the process [2]. In this way, participants are merely interested in their process flow, while they are able to interact with other participants by exchanging messages. To this extent, collaboration diagrams come in frontline, highlighting the interactions between process participants (Figure 30).

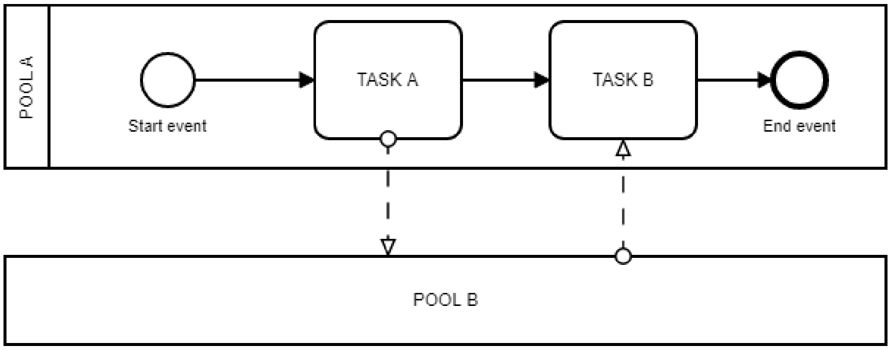


Figure 30 - Collaboration diagram with message flows

In turn, visualizing a participant's process with a higher level of precision, BPMN lanes arise as intrinsic parts of pools. In fact, BPMN pools can be divided in more than one BPMN lanes, where each of them constitutes a graphical sub-division of a BPMN pool, representing responsibilities that are assigned to a specific person, department, role, application, etc. [2]. As a result, the flow of the process remains strictly inside a participant's pool, while it can move between different lanes (Figure 31).

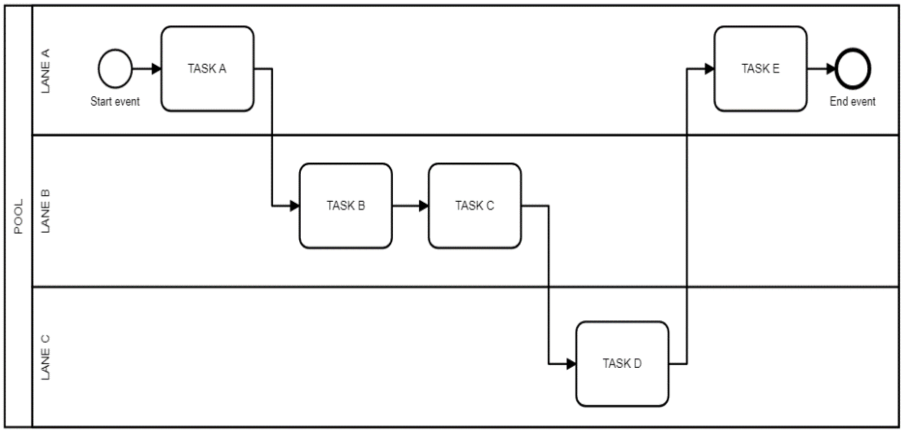
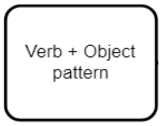
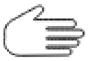
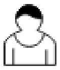
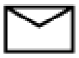
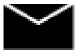
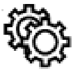
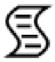
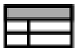


Figure 31 - BPMN pool and lanes

3.3 BPMN activities

Business processes are comprised of a plethora of activities that are necessary to be executed so as to deliver the desired outcome [10]. Frequently encountered as BPMN tasks, they lay on the core of BPMN notation as they render an action that is required to happen [5]. Due to the powerful BPMN armory, there is a variety of task types that business modelers are able to resort to, during their process modeling initiatives. Each of these tasks should have a unique label, epitomizing the activity that is required to be executed [2]. According to Mendling et al. [29], the comprehension of a process model is negatively affected by vagueness in activity labels. Among BPMN modelers prevail a vast majority of modeling conventions. As such, a naming convention for BPMN tasks, namely a verb and object pattern, facilitates not only the correct naming of tasks, but also the understandability of the process logic and the communication between business and IT [2], [5].

Embarking on a BPMN journey, a concrete understanding of the BPMN task types might seem imperative for the day-to-day engagement with BPMN and the understanding of the different task type functionalities. Admittedly, BPMN 2.0 tasks serve both human and non-human intervention. They are represented as rectangles, while they are annotated on their top left corner with different symbols. Task elements in BPMN are connected with each other leveraging the sequence flows, that render the time-logic sequence of the BPMN objects [2]. In addition, they are supplemented with task markers, which are annotated in the bottom area of task types and indicate an additional process logic to task elements. Both task types and task markers, supported by BPMN 2.0, are presented in Table 2.

BPMN Tasks		
Task		BPMN tasks are utilized for rendering a specific activity that needs to be executed.
Task Type	**Symbol**	**Functionality**
Manual Task		An activity that is executed by human actors without the usage of IT systems. This type of task is not automatically assigned by a workflow engine to a user's task list.
User Task		An activity that is executed by human actors with the assistance of an application or an IT system. It might be assigned to them by a workflow engine, since this type of task is placed automatically to a user's tasks list.
Receive Task		An activity that waits for the arrival of an external message. The execution of the process cannot be continued unless a specific message is received.
Send Task		An activity that sends a specific message to an external recipient. Send tasks can be automatically executed by engines.
Service Task		An activity that uses some sort of service, such as a Web Service or an automated application. This type of task is performed automatically without human intervention.
Script Task		An activity that is automatically executed by a workflow engine. This type of tasks is written in a language that the workflow engine is able to interpret.
Business Rule Task		An activity utilized for the implementation of business rules in a process model. This type of task might additionally be linked to DMN notation, as a means of referencing DMN models from BPMN models.
BPMN Task Markers		
Marker Type	**Symbol**	**Functionality**

Compensation Marker	⏪	Represents a compensation handler task in case of a compensation event is triggered in a process. In fact, it is utilized for compensations only and indicates a task that is outside the normal process flow.
Loop Marker	↻	Indicates a repetition of a task until a defined condition either applies or ceases to apply.
Multiple instance Marker	Parallel Multi Instance ⦀	Indicates a task that needs to be executed multiple times, either in parallel or sequentially.
	Sequential Multi Instance ≡	

Table 2 - BPMN tasks (Adaptation from [2] and [46])

3.4 BPMN gateways

3.4.1 Modeling with BPMN gateways

Each time that a process execution is triggered, a new process instance is generated. For the sake of understandability, a process instance can be cognitively associated with a token, which flows inside a process until it gets consumed. Each process instance, however, might follow different routing paths depending on certain circumstances, conditions and the embedded process logic. In this direction, BPMN provides a specific type of elements, namely the gateway element, illustrated by a diamond, that serves towards the modeling of routing points in business processes. Depending on the process data (i.e., process variables), once the process instance (i.e., token) arrives at a gateway element, one or more paths can be activated. In this regard, a cardinality of process tokens is generated, based not only on the type of the utilized gateway element, but also on the evaluation of the gateway's conditions.

Interestingly, irrespective of the utilized gateway type, gateway elements can be classified into two fundamental categories, namely the split and join gateways (Figure 32). Admittedly, a split gateway receives an incoming sequence flow, while it has two or more outgoing sequence flows, indicating the division of a process flow after the evaluation of a condition. Thus, once the process instance arrives at a split gateway, one, more or all outgoing sequence flows can be activated, causing the generation of a corresponding number of tokens. Conversely, a join gateway has two or more incoming sequence flows and exactly one outgoing flow, denoting primarily the intersection of flow paths. As such, a join gateway, depending on the gateway type, indicates the required number of tokens that need to arrive, before the process instance can be further executed.

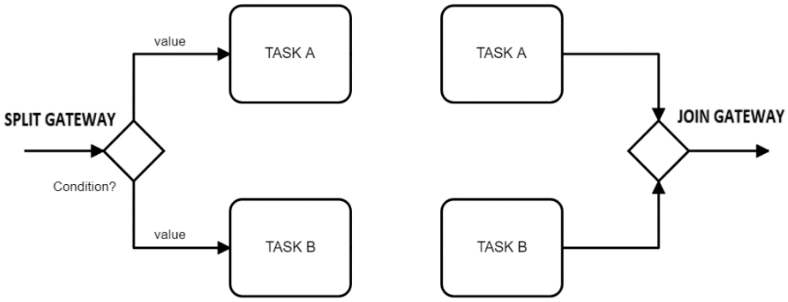
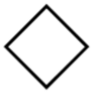
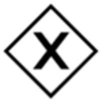
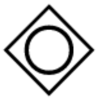


Figure 32 - Split and join gateways

Ranging from fundamental BPMN gateways, namely the Exclusive (XOR), Inclusive (OR) and Parallel (AND) gateways, to more advanced gateway elements, such as the Complex and the Event-Based gateways, they are undoubtedly an integral part of every BPMN modeling initiative. The aforementioned gateway types, along with their functionality, are rendered in Table 3.

<table>
<tr><th colspan="6">BPMN Gateways</th></tr>
<tr><td colspan="2">Gateway</td><td colspan="3"></td><td>BPMN gateways are utilized for modeling diverging and converging points in business processes.</td></tr>
<tr><th colspan="2">Gateway Type</th><th>Symbol</th><th>Split Gateway</th><th>Join Gateway</th><th>Functionality</th></tr>
<tr><td rowspan="2">Data-based</td><td>Exclusive Gateway (XOR)</td><td></td><td>The exclusive split gateway is utilized to generate mutually exclusive alternative paths, that only one of them can be taken on the basis of a condition.</td><td>The exclusive join gateway is utilized to merge alternative paths, where there is no any synchronization.</td><td>The exclusive gateway indicates mutually exclusive alternative paths, where a routing decision is based on process data.</td></tr>
<tr><td>Inclusive Gateway (OR)</td><td></td><td>The inclusive split gateway is utilized to generate independent alternative paths, where one, more or all of them can be followed on the basis of a condition.</td><td>The inclusive join gateway is utilized to merge independent alternative paths, where there is a synchronization of activated flow paths, before the process being further executed.</td><td>The inclusive gateway denotes independent alternative paths, where a routing decision is based on process data.</td></tr>
</table>

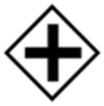
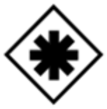
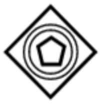
	Parallel Gateway (AND)		The parallel split gateway is utilized to generate parallel flows that can be executed concurrently, without evaluating any condition.	The parallel join gateway is utilized to synchronize parallel flows, before the process is further executed.	The parallel gateway denotes the parallelism of paths, where no condition is evaluated before.
	Complex Gateway		In practical models, the complex split gateway is rarely utilized.	The complex join gateway is utilized for the realization of m out of n merges before the process is further executed.	The complex gateway, mainly utilized as join node, denotes a complex synchronization behavior, where the desired outcome is the realization of m out of n merges.
Event-based	Event-based Gateway		The event-based split gateway is utilized to generate mutually exclusive alternative paths, where exactly one path is taken based on the occurrence of a subsequent event, rather than the evaluation of process data.	In practical models, the event-based join gateway is rarely utilized. In most of the cases, an exclusive join gateway is used instead.	The event-based gateway is utilized when routing decisions are based on the occurrence of a subsequent event, rather than the evaluation of process data.

Table 3 - BPMN gateways (Adaptation from [2] and [46])

Having presented the full-set of BPMN gateways, the usage of an exclusive and parallel gateway is displayed in Figure 33. Interestingly, parallel gateways are utilized as a classic process optimization technique, namely a redesign heuristic, due to the fact that the total running time of a process can be significantly decreased. In fact, between a split and a merge parallel gateway, there might be more than one branches, comprised of one or more activities. The branch which takes more processing time to be completed, however, is the only one taken into account, determining, thus, the total runtime of the process.

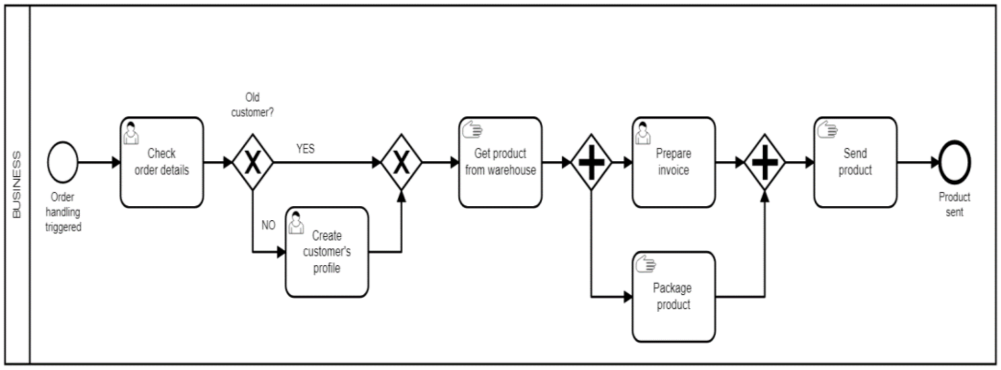


Figure 33 - Usage of exclusive and parallel gateways in a BPMN model

3.4.2 Advanced modeling with BPMN gateways

The expressive power of BPMN undoubtedly emerges when delving into more advanced BPMN elements. In this sense, complex and event-based gateways, frequently omitted in most modeling initiatives, can be characterized as enhanced variants of gateway elements. More specifically, a complex gateway, mainly utilized as a join node, renders intersection points, where m out of n merges need to be synchronized, before the process can be further executed. Interestingly, in case of a realization of m merges, a single token traverses the outgoing sequence flow, while if more tokens keep reaching the complex merge node, they are ignored and automatically consumed. In the following example (Figure 34), in case two out of the three incoming merges are executed, the complex gateway is activated and the process continues to Task E.

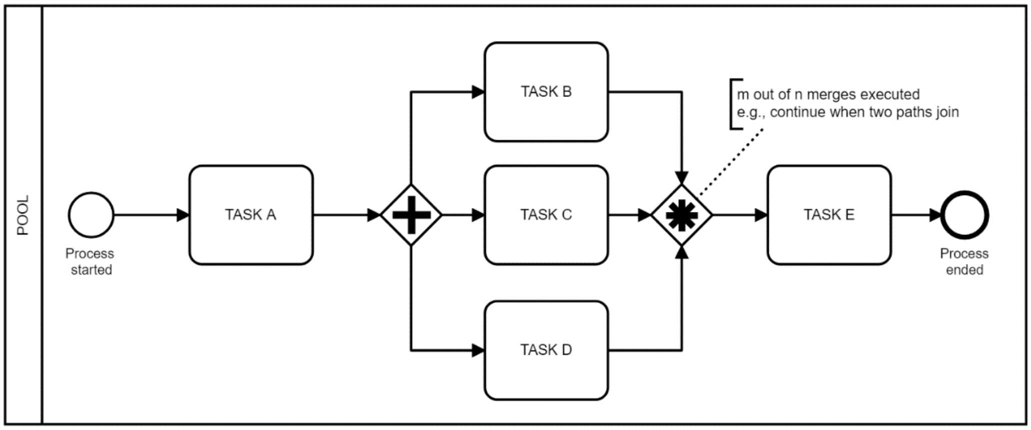


Figure 34 - Complex gateway

Another advanced gateway element, namely the event-based gateway, is an alternative way of designing process paths in BPMN models. Notably, routing paths are followed based on the occurrence of subsequent events, rather than evaluating process data (i.e., process variables). According to the BPMN specification [46], an event-based gateway can merely be followed by intermediate message, signal, timer, conditional and multiple catching events (introduced in Section 3.5), as well as by a receive task, indicating that the routing decision is based on information that the process is not already aware of. Considering the token approach, once a token arrives at an event-based split gateway, it is required to wait until one of the subsequent events occurs. In this case, the process token follows the path where the triggered event is located, ignoring all the other events that might occur. In practical models, the event-based join gateway is rarely used and instead an exclusive (XOR) join gateway is utilized. In the following example (Figure 35), once a customer pays a requested order, they need to wait to receive the product or three days to be elapsed, before contacting the vendor. In this regard, the routing path is triggered on the basis of an event occurrence, without needing to evaluate any process data.

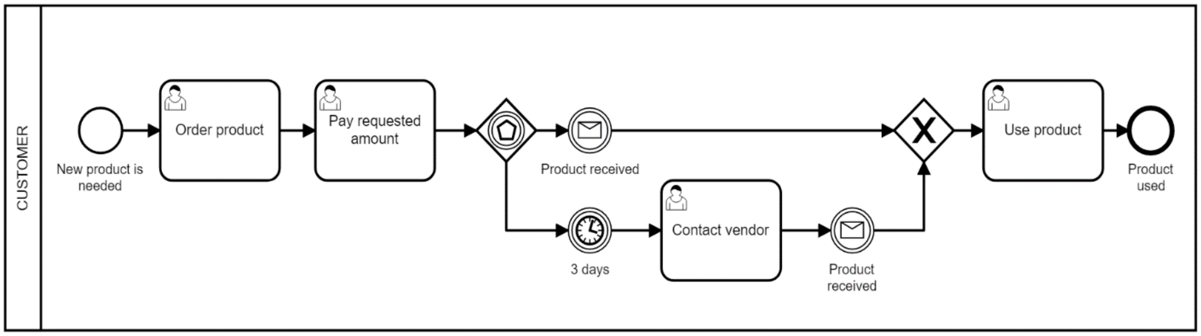


Figure 35 - Event-based gateway (Adaptation from [75])

3.5 BPMN events

3.5.1 Modeling with BPMN events

Event elements constitute another fundamental concept of BPMN notation. Playing a crucial role in process modeling, they are deemed to bridge real situation events with processes, which react to these events or trigger them [10]. Based on their position in a process model, they are classified into three fundamental categories, namely, the start, intermediate and end events. Intuitively, start events indicate the trigger and the outset of a process execution, intermediate events stand for milestones during a process lifecycle, while end events render a process completion.

Irrespective of the event type (introduced in Section 3.5.3), BPMN events can be further classified to catching and throwing event elements. Interestingly, business processes interact with events when either something independent of the process fires an event (i.e., the process “catches” the event) or when the process itself triggers an event execution (i.e., the process “throws” an event). As a result, catching events, illustrated by an unfilled event marker, are triggered by external factors, where a trigger condition is satisfied (e.g., receive a message from an external participant), before the process instance being further executed. On the other hand, throwing events, depicted by a fill event marker, are triggered by the process itself, where an event is instantly activated, instead of waiting to react to a trigger condition (e.g., send a message to an external participant). In Figure 36, the concepts of start, intermediate, end, catching and throwing events, are exemplified in a simple BPMN model.

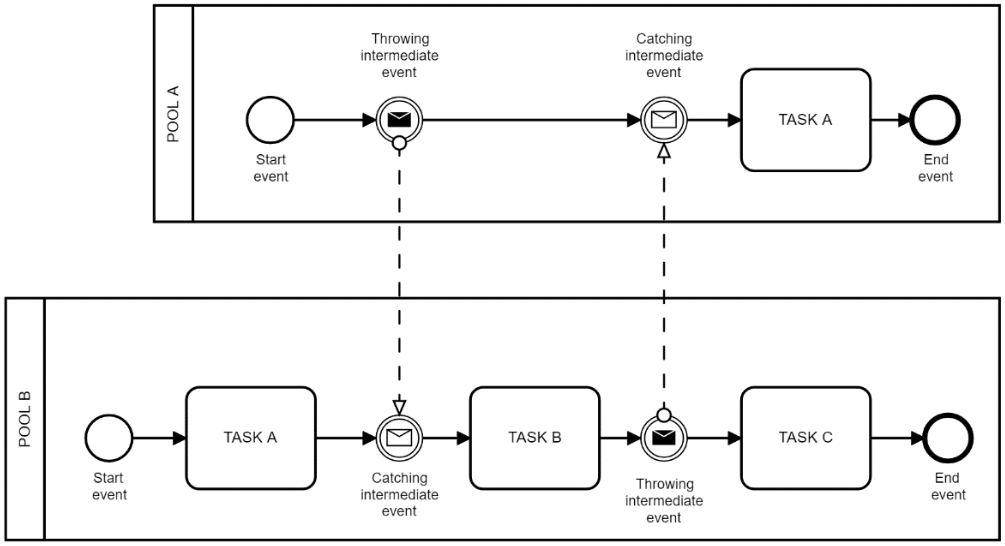

POOL A
Throwing intermediate event
Catching intermediate event
TASK A
Start event
End event
POOL B
Start event
TASK A
Catching intermediate event
TASK B
Throwing intermediate event
TASK C
End event

Figure 36 - BPMN events

3.5.2 Advanced modeling with BPMN events

Even if event elements are fundamentally utilized in practical BPMN modeling initiatives, advanced modeling concepts around BPMN events can highly increase the expressiveness of BPMN models. In this regard, boundary intermediate events, attached to the boundary of tasks, emerge as sophisticated BPMN elements that enhance the cognitive power of BPMN modeling.

Interestingly, while tasks are being processed, several events might occur, causing their interruption or their exceptional execution [2]. Depending on if the event occurrence interrupts the enclosing task execution, they are further being separated into interrupting and non-interrupting boundary events. Depicted as attached events with a double continuous outline, boundary interrupting events completely cancel the task execution, while an exception flow, emanating from the boundary interrupting event, is further activated. In this regard, in the pool A of Figure 37, while the task A is being processed, an interrupting event occurrence immediately cancels the task A and the process instance triggers the task C. On the other hand, boundary non-interrupting events, rendered as attached events with a double dashed outline, activate an exception flow path without cancelling the enclosing activity. In this sense, in the pool B of Figure 37, while the task A is being processed, a non-interrupting event occurrence triggers an exception flow directing to task C, while the task A continues to be processed. With respect to token approach, the process instance, thus, cannot be finished unless all the generated tokens are consumed. However, the occurrence of a boundary intermediate event is ignored, once the task that is attached to, ceases to be active [2].

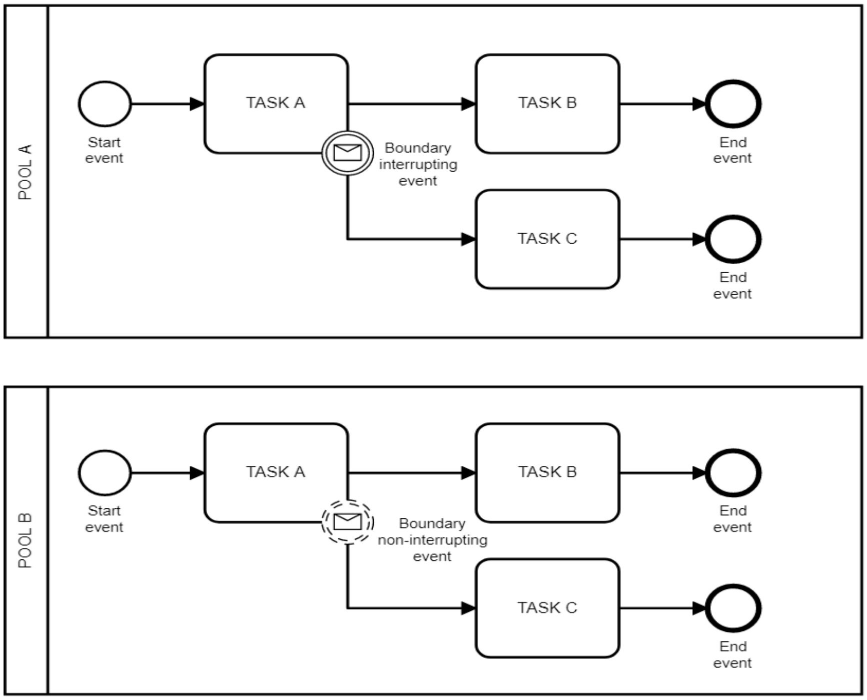
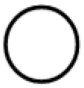
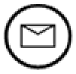
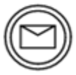
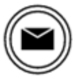
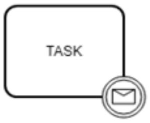
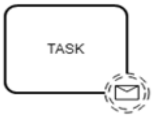
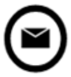
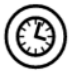
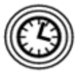
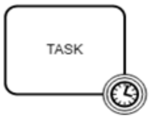
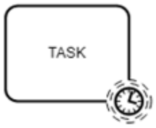
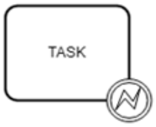
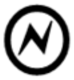
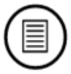
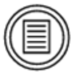
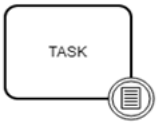
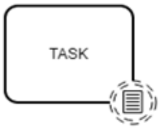


Figure 37 - Boundary interrupting and non-interrupting events

3.5.3 Event types

BPMN 2.0 supports a plethora of event types as it does for task and gateway types. Depending on the process logic, a variety of event types might be leveraged so as to appropriately adapt the BPMN model to the desired process logic. The full-set of BPMN event types, along with their functionality, is presented in Table 4, before delving into the error, terminate and compensation events in the following subsections.

BPMN Events							
Event							BPMN events are utilized for capturing important milestones of a process execution.
Event type	**Start event**	**Intermediate catching event**	**Intermediate throwing event**	**Boundary interrupting event**	**Boundary non-interrupting event**	**End event**	**Functionality**
Message event							The message event is utilized when a message is forwarded to a specific recipient or when the process instance receives an external message.
Timer event			-			-	The timer event is utilized so as to denote the concept of time in a process model.
Error event	-	-	-		-		The error event is utilized so as to denote potential errors during a process execution.
Conditional event			-			-	The conditional event is utilized so as to denote a condition that needs to be satisfied during the process execution.

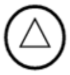
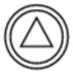
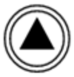
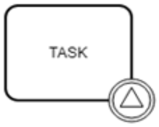
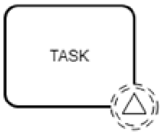
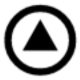
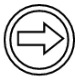
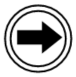
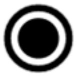
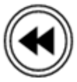
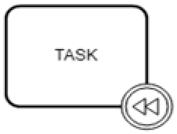
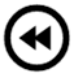
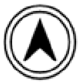
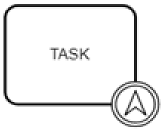
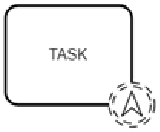
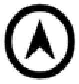
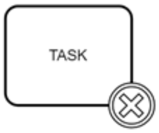
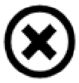
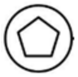
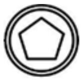
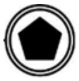
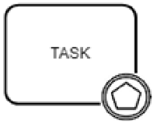
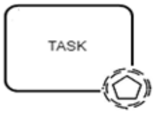
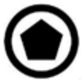
Signal event							The signal event is utilized when messages (i.e., signals) are forwarded to a global audience, rather than a specific recipient, or when the process instance receives an external signal.
Link event	-			-	-	-	The link event is utilized so as to connect two sections of a process, avoiding long sequence flow lines.
Terminate event	-	-	-	-	-		The terminate event is utilized so as to precociously terminate process instances.
Compensation event	-	-			-		The compensation event is utilized in the context of triggering or handling compensations in a process execution.
Escalation event	-	-					The escalation event is utilized so as to communicate events from subprocesses to parent processes.
Cancel event	-	-	-		-		The cancel event is particularly utilized in the context of transaction subprocesses so as to denote the cancellation of an entire transaction.
Multiple event							The multiple event is utilized in a way that several event types can be encapsulated to a single symbol.

Table 4 - BPMN events (Adaptation from [2] and [46])

(a) Error events and error handling in BPMN models

Error-prone processes constitute a fundamental road block to the efficiency and effectiveness of organizations. Admittedly, the immediate proper handling of process errors is an imperative requirement of today's business administrations. In this regard, BPMN treats errors as first-class citizens inside process modeling, leveraging the error event elements for their proper handling. Interestingly, being utilized not only for technical errors, but also for business errors such as the unavailability of a product, they primarily indicate an error occurrence during the execution of a specific activity, while they establish a recovery procedure so as to handle this error.

A simple order handling process is introduced (Figure 38) so as to delve into the concept of error events in BPMN notation and their capability for error handling. More specifically, once an order is received, the customer details are checked, before the product availability being estimated. In case of a product's unavailability, the process ends with an error occurrence, indicating a failure during its execution. On the other hand, the product is received from warehouse and subsequently the total cost is automatically calculated, before the product being delivered. Nonetheless, in case an error occurrence during the automatic calculation of the order's cost, a recovery procedure is established by calculating manually the total cost. For this purpose, an interrupting error event is attached to the boundary of the service task, indicating that in case of an error occurrence a recovery procedure is initialized and the error is handled by calculating manually the order's cost. As such, not only is a potential error handled, but also the process instance continues being executed without the need of terminating the whole process [5].

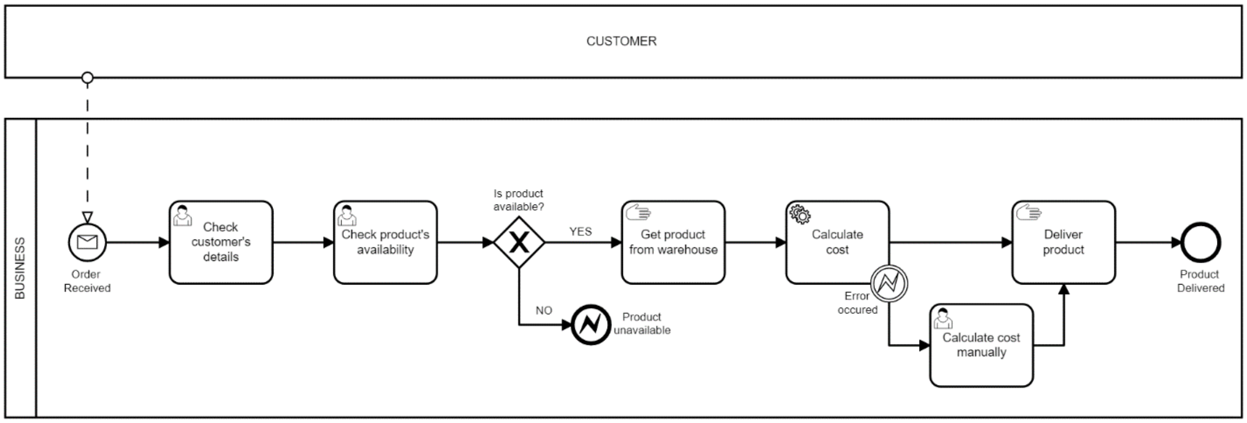


Figure 38 - Error handling in BPMN models

(b) Terminate events and knock-out parts in BPMN processes

Recognized as a fundamental redesign heuristic, knock-out parts in business processes are frequently encountered in today's organizations so as to precociously terminate process executions that are doomed to fail. By adhering to the heuristics of knock-out parts in businesses processes, organizations are able to redesign their processes, let alone to refine substantially their key performance indicators (KPIs). According to Dumas et. al [5], a knock-out part is defined as a part of a business process where the negative evaluation of a condition causes the termination of the whole process. Thus, a process instance has to satisfy a sequence of various conditions, namely a plethora of knock-out parts, before being able to deliver a desired outcome.

In this sense, terminate events are extensively utilized in BPMN models in order to establish knock-out parts in business processes and terminate them under certain circumstances. In Figure 39, a sequence of terminate events is deployed into the process logic of a fictitious loan application process. From a bank's perspective, a number of knock-out parts is established in order to precociously terminate defective process instances which lack of needed requirements. As a result, once a process instance reaches a terminate event, all the alive process tokens are immediately consumed, leading to the whole process termination. The process, thus, is immediately terminated, before upcoming activities have come, saving in this way valuable resources, let alone time and cost.

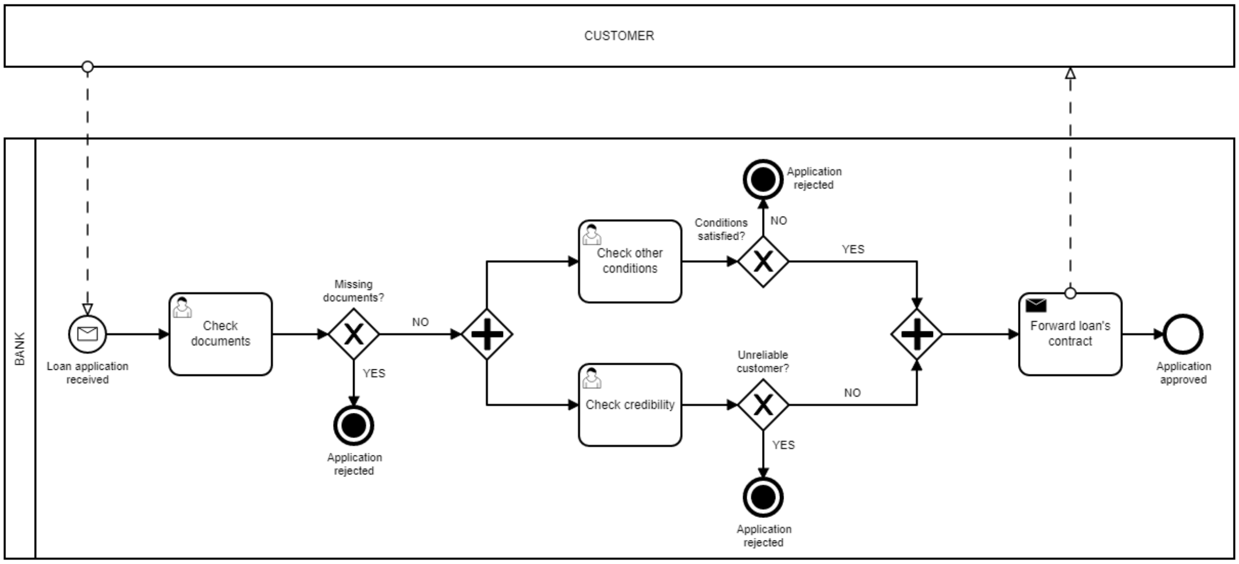


Figure 39 - Terminate events and knock-out parts in BPMN models

(c) Compensation events and compensation handling in BPMN models

Considering the complex processes of modern organizations, a notorious challenge for organizations and business analysts is to successfully handle instances that require compensations. In this regard, compensation events are widely utilized in BPMN models so as to compensate tasks that have been previously completed [2]. For this reason, a compensation handler is established in order to perform steps, indispensable to reverse the effects of a previous completed activity [46]. According to BPMN specification [46], a compensation handler starts with a catch boundary compensation event, which directs to a compensation activity through an association flow, indicating in this way that the compensation is outside the normal flow of the process (Figure 40).

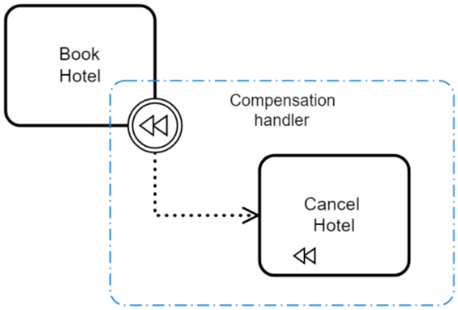


Figure 40 - Compensation handler (Adaptation from [46])

In Figure 41, a simple process model is illustrated rendering the compensation handling concept in BPMN models. More specifically, a compensation is triggered using a compensation throw event, which can either be an intermediate or an end event. In the underlying example, in case of an error occurrence during the “Charge credit card” service task, an intermediate throwing compensation event, namely the “Cancel booking” event, triggers a compensation which is propagated to all the relative compensation handlers within the process scope. In turn, compensation handlers are activated, directing the process instance to a compensation activity, and indicating, thus, the execution of an exception path rather than the regular process sequence [2].

Importantly, according to BPMN 2.0 specification [46], boundary compensation events are triggered only if the activities to which they are attached, have been successfully completed and are no longer active. In any other case, nonetheless, if activities to which compensation events are attached, are still active, they can only be cancelled rather than being compensated [46].

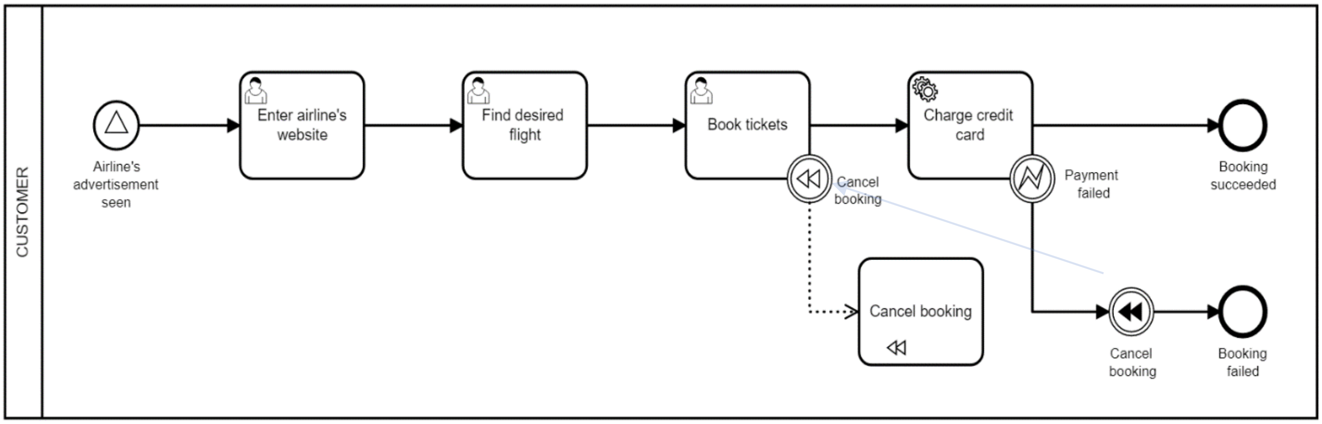


Figure 41 - Compensation events and compensation handling in BPMN models

3.6 BPMN subprocesses

3.6.1 Modeling with BPMN subprocesses

A subprocess constitutes another fundamental modeling concept in BPMN notation. Regarded as an alternative activity type, a subprocess has its own process logic, containing other BPMN elements, such as activities, events and gateways. Encompassing a lower-level process logic, a subprocess, thus, is a distinct graphical element within a process model [46]. Depending on the desired level of granularity, business analysts are able to utilize collapsed or expanded subprocesses in order to hide or show their details. Collapsed subprocesses, hence, are able to hide the complexity of detailed sequences, while expanded ones might thoroughly represent their process flow, within the view of the process in which they are contained [46].

Considering a token approach, once a token arrives at the subprocess, it needs to wait until the subprocess is completely executed. In the meantime, a new token is generated within the subprocess boundaries, before it is finally consumed by an end event. Thereafter, the parent token traverses further the process paths until it gets consumed by its own end event. In Figures 42 and 43, the concepts of collapsed and expanded subprocesses are exemplified within BPMN process models.

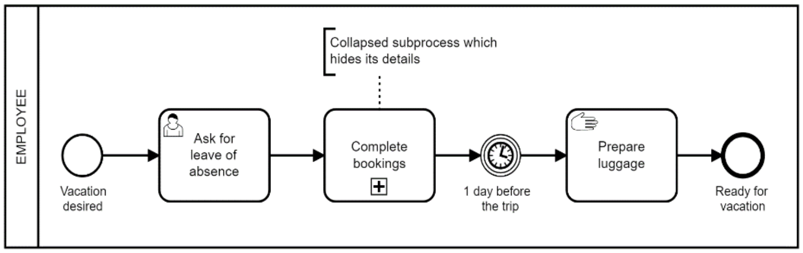


Figure 42 - Collapsed subprocess

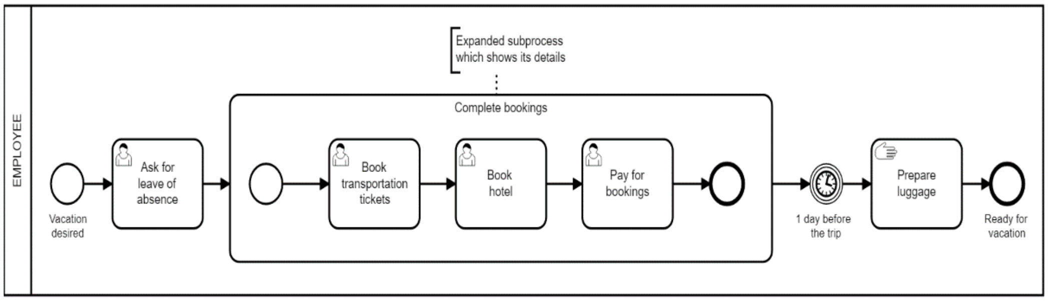


Figure 43 - Expanded subprocess

3.6.2 Embedded and global subprocesses

According to BPMN 2.0 specification [46], subprocesses can further be classified into embedded and global ones depending on the desired level of modularity. More specifically, embedded subprocesses, either as collapsed or expanded elements, are located within a pool or a lane of a parent process. Being not allowed to contain their own pools or lanes, their main functionality is to hide or reveal process details depending on the desired level of granularity. On the other hand, a global subprocess might be triggered from different parent processes by a means of a call activity, namely an activity with noticeably thicker borders [2], [46]. Global subprocesses, thus, constitute modular, let alone reusable parts of BPMN models, where they can be invoked by different processes utilizing their unique identifier [10]. In this regard, process flows are able to be moduled and reused from many parent processes without needing to change neither their structure nor their process logic.

However, technically speaking, data transfer between parent processes and subprocesses differentiates in the context of embedded and global subprocesses. Importantly, embedded subprocesses are able to interpret directly all the available process data (i.e., process variables) of a parent process. On the other hand, a data mapping is indispensable to data transfer between parent processes and global subprocesses, considering their different scope during their integration [2].

The aforementioned concepts are illustrated in Figures 44 and 45, where embedded and global subprocesses are rendered respectively. In Figure 44, a loan application process is illustrated from a bank’s perspective, utilizing an embedded, let alone expanded subprocess, in order to shed light on the process logic behind the customer’s credibility check. The same process, as well as an opening bank account process, are depicted on the upper and middle part of Figure 45, utilizing the paradigm of call activities so as to invoke the global “Check customer’s credibility” subprocess. In this regard, the same subprocess in invoked and reused from more than one parent processes, without modifying its process logic.

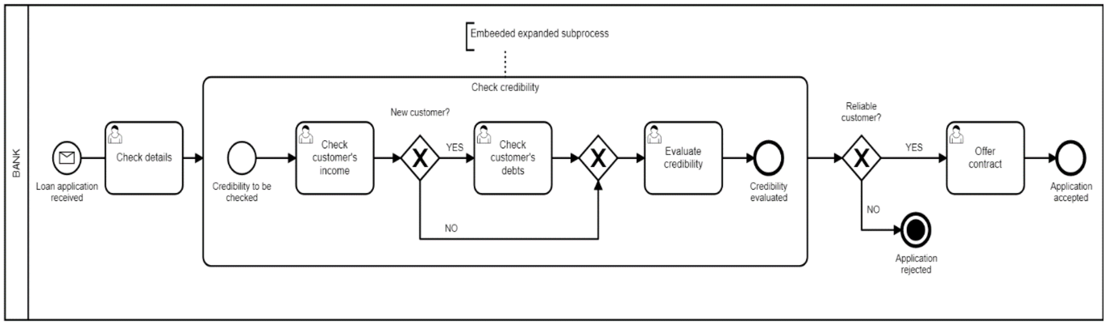


Figure 44 - Embedded subprocess

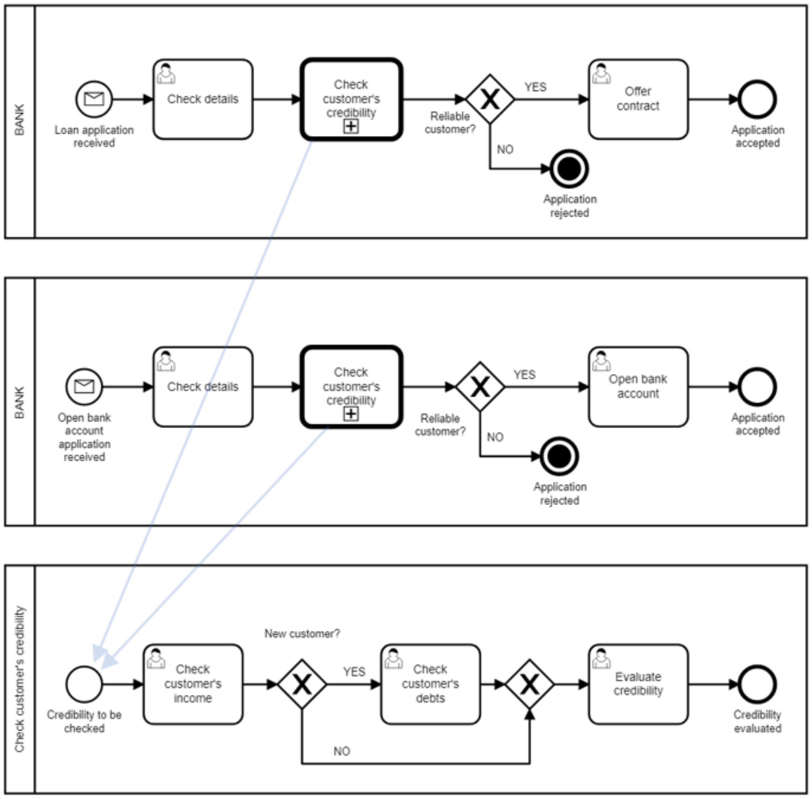


Figure 45 - Call activities and global subprocesses

3.6.3 Advanced modeling with BPMN subprocesses

(a) Boundary events on subprocesses

The aforementioned boundary interrupting and non-interrupting events, except for merely being attached to tasks, might be additionally attached to subprocess boundaries. Interestingly, while subprocesses are active, several events might occur, causing their interruption or their exceptional execution [2]. In this regard, as long as the subprocess is executed, boundary intermediate events are utilized as listeners to an event occurrence. Technically, the boundary events are triggered if the subprocess remains active, while in any other case, their occurrence is ignored. Considering the process logic of a BPMN model illustrated in Figure 46, once a cancellation request message is received during the execution of the "Process order" subprocess, a boundary interrupting message event is triggered, causing the interruption of the whole subprocess. Conversely, once an inquiry message is received while the "Process order" subprocess remains active, a new user task is generated in order to handle an inquiry without interrupting the execution of the subprocess.

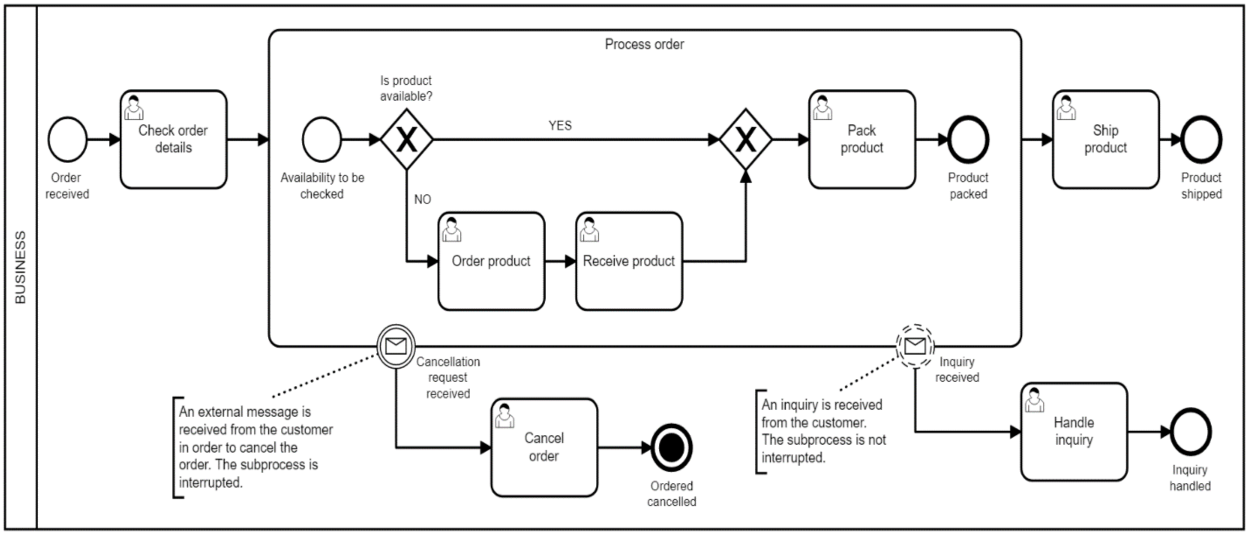

BUSINESS
Process order
Order received
Check order details
Availability to be checked
Is product available?
YES
NO
Order product
Receive product
Pack product
Product packed
Ship product
Product shipped
Cancellation request received
An external message is received from the customer in order to cancel the order. The subprocess is interrupted.
Cancel order
Ordered cancelled
Inquiry received
An inquiry is received from the customer. The subprocess is not interrupted.
Handle inquiry
Inquiry handled

Figure 46 - Boundary events on subprocesses

Moreover, instead of catching events, mainly triggered from external circumstances, boundary intermediate events on subprocesses are additionally utilized in order to handle event occurrences inside them. A subprocess, thus, is able to report an event occurrence to its parent process, where the event is handled based on the desired process logic. In Figure 47, once a customer is evaluated as unreliable in the "Check customer's credibility" subprocess, an error event is thrown, before being communicated to its parent process. In this sense, by utilizing an interrupting error event, attached to the subprocess, in case of an error occurrence during the subprocess execution, the interruption of the normal process flow is indicated. On the other hand, if more documents are needed for the customer's credibility evaluation during the subprocess execution, an escalation event is thrown, before being reported to the parent scope. As a result, the parent process gets aware of an event occurrence inside the subprocess, before handling it in a specific way, on the basis of the desired process logic.

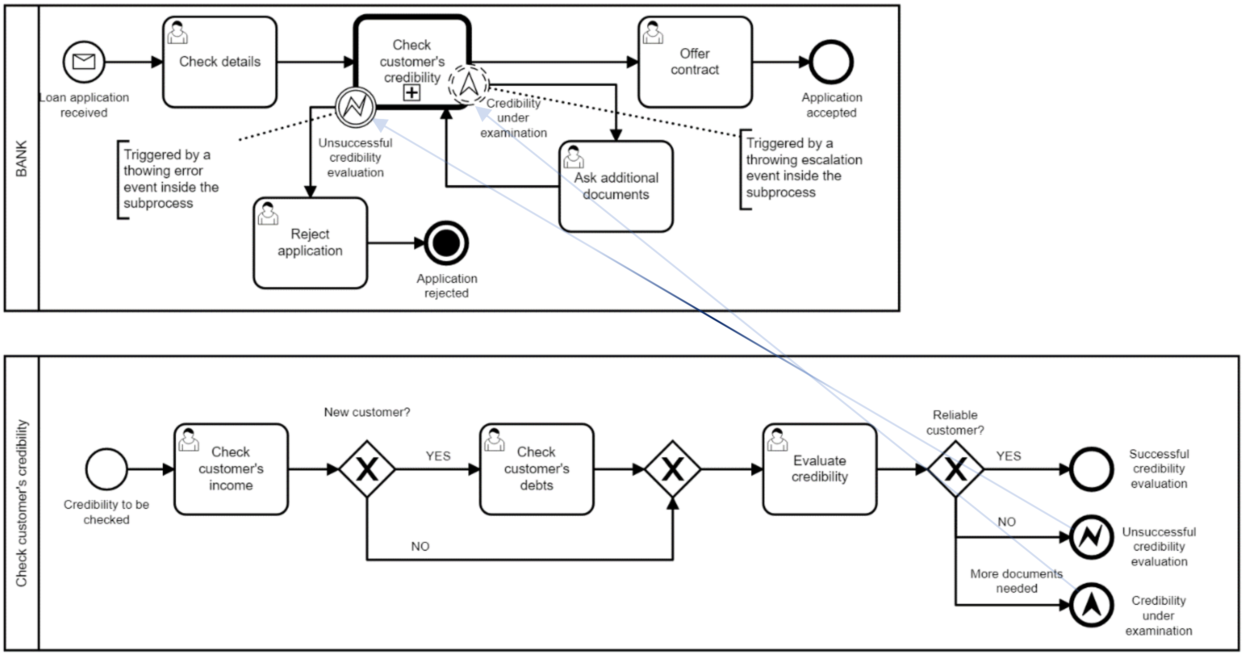

BANK
Loan application received
Check details
Check customer's credibility
Offer contract
Application accepted
Credibility under examination
Unsuccessful credibility evaluation
Triggered by a thowing error event inside the subprocess
Triggered by a throwing escalation event inside the subprocess
Ask additional documents
Reject application
Application rejected
Check customer's credibility
Credibility to be checked
Check customer's income
New customer?
YES
NO
Check customer's debts
Evaluate credibility
Reliable customer?
YES
NO
More documents needed
Successful credibility evaluation
Unsuccessful credibility evaluation
Credibility under examination

Figure 47 - Boundary events on subprocesses - Communication of subprocesses with parent processes

(b) Transaction subprocesses

Transaction subprocesses constitute a distinct type of embedded subprocesses, where once utilized, additional functionalities emerge in process modeling initiatives with BPMN notation. Illustrated with a double border, they are exclusively being utilized for grouping activities in a transaction in the context of a business process [75]. According to BPMN specification [46], a transaction subprocess can successfully be completed or canceled, or interrupted by a hazard, that prevents it to normally be succeeded or get cancelled.

Intuitively, a transaction which is neither canceled nor interrupted by a hazard, is successfully completed and the process instance leaves the transaction by following its outgoing sequence flow. On the other hand, in case a process instance reaches a cancel end event within the transaction boundaries, the transaction gets cancelled and all compensation handlers of previously executed activities are triggered. Once all compensation has been completed, the process instance leaves the transaction by following the outgoing sequence flow, emanating from the attached boundary cancel event. However, in case the transaction is neither succeeded nor canceled, a hazard is detected, which in turn interrupts the transaction subprocess and the flow continues from the boundary error event without any compensation [76].

The usage of a transaction subprocess is exemplified in Figure 48, where a transaction is rendered within a product ordering example. In this sense, a customer is prompted to complete an order in the context of a transaction, where activities succeed or fail collectively. More specifically, in case of an error occurrence during the execution of the "Charge credit card" service task, a cancel end event is triggered, causing the cancellation of the entire transaction. As a result, all compensation handlers of previously executed activities are triggered, activating, thus, their compensation tasks (i.e., "Cancel order", "Cancel payment"). Interestingly, a cancel intermediate event, attached to the boundary of the transaction (i.e., Order cancelled), will direct the flow after the transaction has been rolled back and all compensation has been completed [46]. Alternatively, in case of an error during the transaction execution, the transaction is interrupted and the flow of the process follows the outgoing sequence flow, emanating from its boundary error event (i.e., "System failure"). In any other case, the transaction is successfully completed and the product is ordered.

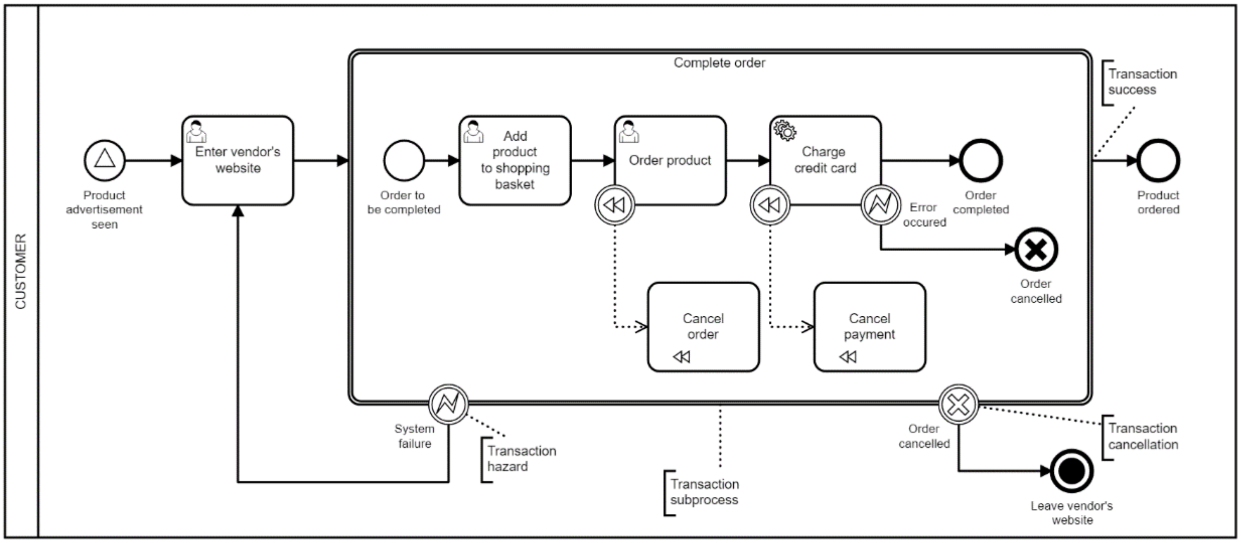


Figure 48 - Transaction subprocess

(c) Event subprocesses

Event subprocesses constitute another advanced modeling concept in BPMN notation. Recognized by their dotted-line frames, they are triggered by their interrupting or non-interrupting start events, as long as their enclosing process, or subprocess, remains active. Importantly, considering that an event subprocess is not a part of the normal process flow, there are no incoming or outgoing sequence flows that connect it with its parent process [46]. In fact, each time that its start event is triggered, the event subprocess is activated, causing the interruption or the parallel execution of its enclosing process.

Considering the process logic of a fictitious order handling process in Figure 49, the usage of an event subprocess is well exemplified. In this regard, as long as the process instance remains active, either an inquiry or an error-handling event-subprocess might be triggered. More specifically, once an inquiry message is received, the inquiry handling event-subprocess is triggered without interrupting its parent process, considering its non-interrupting start event. Conversely, in case of an error occurrence throughout the process execution, an error handling event-subprocess is triggered, causing the interruption of its enclosing process due to its interrupting start event.

Ultimately, event-subprocesses are deemed to constitute a powerful modeling concept in BPMN notation. Reacting to events that might occur throughout the whole process instance execution, rather than during specific tasks, event-subprocesses are deemed to increase the expressive power of BPMN models, reflecting real case scenarios, frequently encountered in today’s business processes.

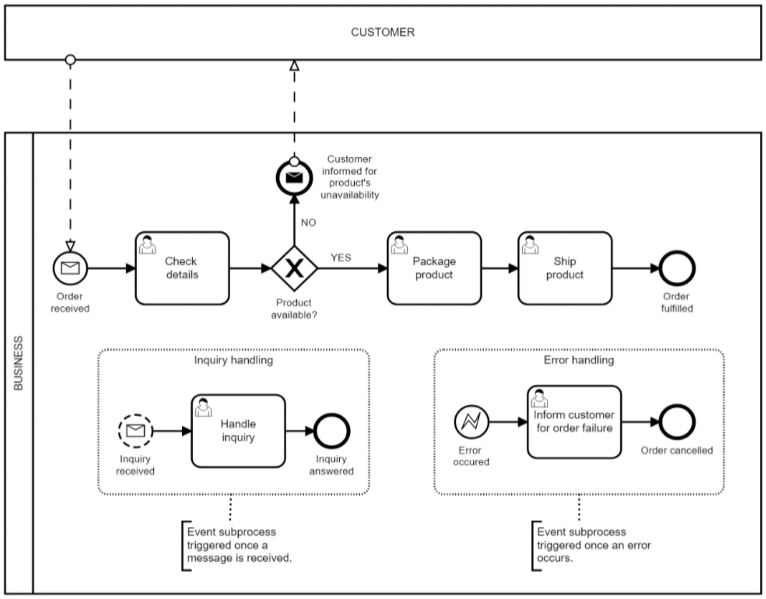


Figure 49 - Event subprocess

3.7 Data modeling in BPMN models

BPMN notation except for modeling the sequence and routing logic of business processes, is able to graphically illustrate the data flow inside a process. Considering that activities frequently need data in order to be executed, or alternatively produce data as a result of their execution [2], data modeling excessively enhances the expressiveness of BPMN models.

In this regard, Data Object and Data Store elements, constitute the primary constructs for modeling data flows inside BPMN process representations. More specifically, Data Objects are utilized in order to render various kinds of information, irrespective of their physical nature, including paper documents, abstract information or electronic data [2]. By means of association flows, they are able to be connected with flow objects and sequence flows, indicating their origin and their direction inside the process flow. Moreover, Data Stores render the way that process activities retrieve or update data on the basis of the underlying process logic. As a result, BPMN tasks are able to refer to Data Stores, illustrating such interactions by means of association flows. The BPMN data modeling elements are presented in Table 5, before being utilized in a process example in Figure 50.

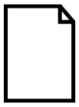
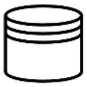
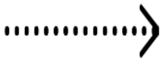
BPMN data modeling elements		
Element	**Symbol**	**Functionality**
Data object		The data object element is utilized in order to graphically illustrate every piece of information (e.g., paper documents, electronic data, etc.), which flows inside a process.
Data store		The data store element is utilized in order to represent a software, a database or any other place where information is stored (e.g., journal), indicating how data is retrieved and stored within a process.
Data association flow		The data association flow is utilized in order to connect flow objects and sequence flows with data object and data stores.

Table 5 - BPMN data modeling elements (Adaptation from [2] and [46])

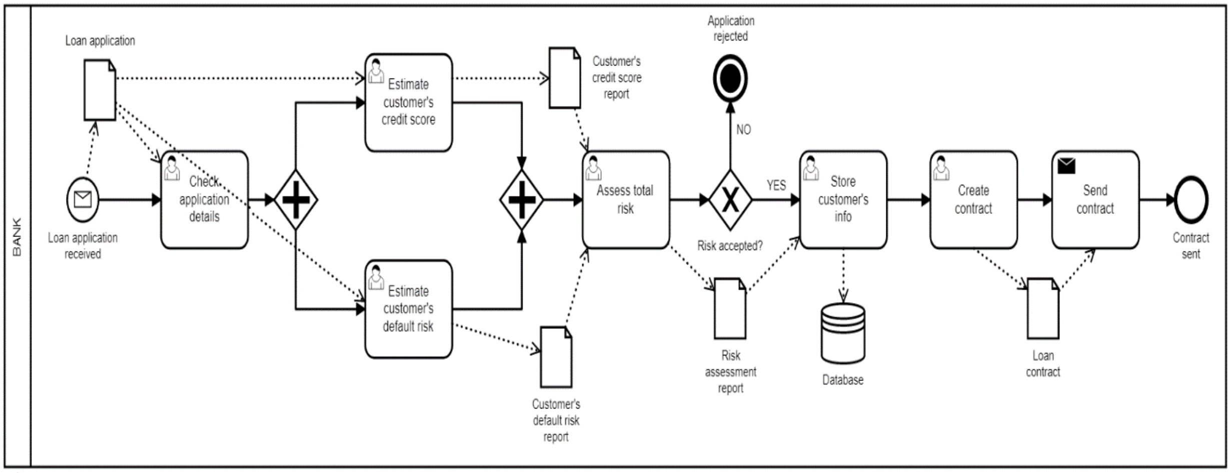


Figure 50 - Data modeling in BPMN models

3.8 Rendering decision logic in BPMN models: From BPMN to DMN models

Undoubtedly, within a business process context, operational decisions are frequently convoluted with activities, while they dominate the way in which the process unfolds [50]. Even if BPMN notation constitutes the standard technique for rendering business processes, it is not regarded to be tailored to the modeling of decision logic of business operations. Trying to render repetitive operational decisions with BPMN constructs, the complexity of BPMN models dramatically escalates, posing a threat to their understanding and their maintainability [34], [54]. Considering that there are various ways of incorporating decisions within the process logic of BPMN models, their identification is of utmost importance before treating them as separate concerns [50].

Typically, decisions in BPMN models are frequently emulated with intricate process control flows, like BPMN gateways, leading to complex routing structures and the creation of spaghetti-like processes [34], [54], [56]. Since multiple decisions are arranged in sequence, cascading gateways are utilized in order to preserve their decision dependencies and the desired decision logic [34] (Figure 51). Contemplating, thus, that structural characteristics of process models are negatively correlated with their understanding [29], the increasing number of gateways and their control flow paths result in process models which are difficult to be maintained, implemented, let alone redesigned [34], [50].

Moreover, repetitive operational decisions in BPMN models can be encapsulated and hidden in script tasks, where the decision logic is embedded within a script code [56] (Figure 52). Utilizing a programming language that is able to be interpreted by a workflow engine, decisions are hard-coded in the background of a script task. However, this makes hard to evaluate the outcomes and the factors for invoking the decisions, while it is more about coding rather than modeling

[56]. Apparently, when further conditions are added or old ones need to be removed, this entails changes in the script code, resulting in increasing changing costs, let alone time.

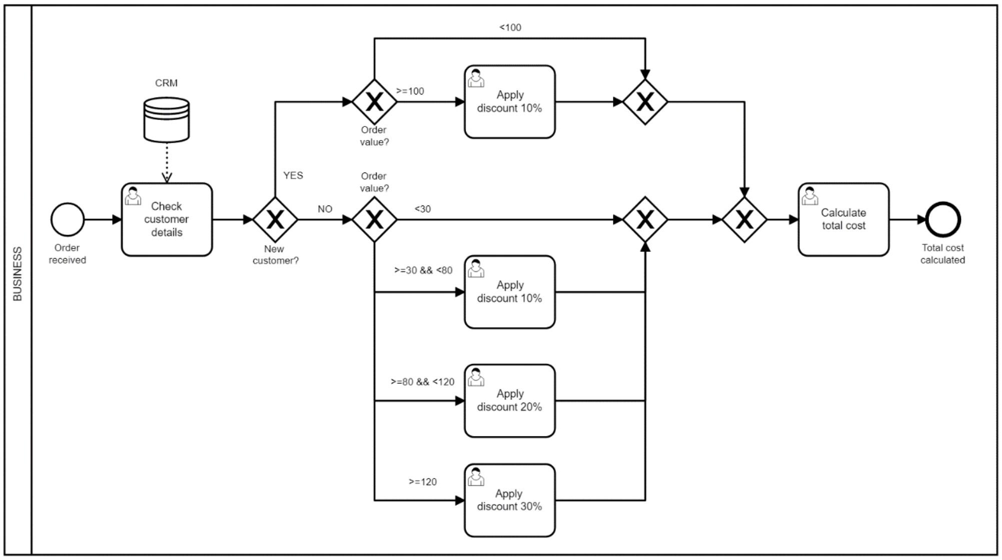
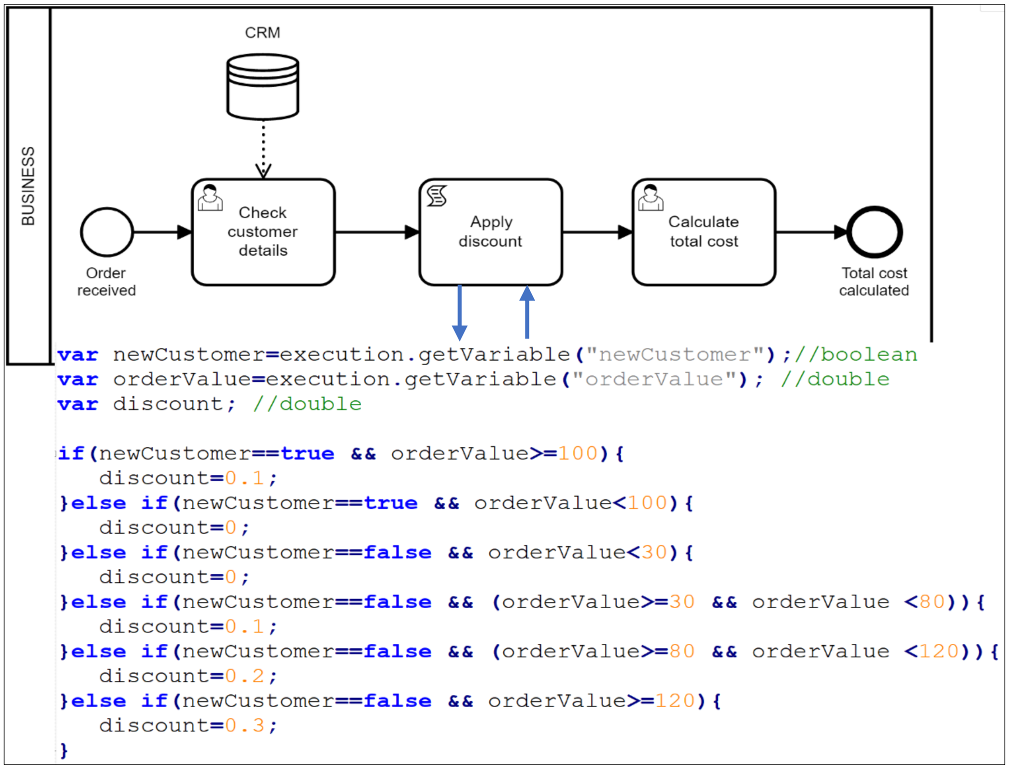


Figure 51 - Rendering decision logic with cascading gateways



```
var newCustomer=execution.getVariable("newCustomer");//boolean
var orderValue=execution.getVariable("orderValue"); //double
var discount; //double

if(newCustomer==true && orderValue>=100){
   discount=0.1;
}else if(newCustomer==true && orderValue<100){
   discount=0;
}else if(newCustomer==false && orderValue<30){
   discount=0;
}else if(newCustomer==false && (orderValue>=30 && orderValue <80)){
   discount=0.1;
}else if(newCustomer==false && (orderValue>=80 && orderValue <120)){
   discount=0.2;
}else if(newCustomer==false && orderValue>=120){
   discount=0.3;
}
```

Figure 52 - Rendering decision logic with script tasks

Thus, a notorious challenge for process modelers is to maintain the complex business logic of repetitive daily decisions outside of the process logic of their models. Embracing the Separation of Concerns (SoC) paradigm, already proposed in computer science, decision modeling can be regarded as a separate concern, where decisions are managed externally in separate decision models. In this regard, identifying decisions hidden in BPMN models and externalizing them, higher maintainability, understandability and agility of both processes and decisions are well attainable. For this purpose, Decision Model and Notation (DMN) is considered to be the standard method for achieving the separation of concerns paradigm, as well as to complement BPMN for modeling decisions related to process models [77]. The ultimate objective is to create integrated, yet separated, models that enhance the consistency and the optimal interaction between the process and decision logic of business operations.

3.9 Summary

This chapter presented the BPMN notation, as well as its fundamental and advanced capabilities in modeling business processes. BPMN is considered to be the de facto modeling language for rendering business operations in a graphical and executable interchange format. Providing a powerful symbol armory, its expressive value is undoubtedly compelling when advanced construct elements come in frontline. In this regard, sophisticated process concepts, frequently neglected in fundamental modeling initiatives, such as the error handling, the compensation handling and the transaction procedures, enhance not only the cognitive power of BPMN representations, but also reflect real complex situations in a straightforward way. However, given the fact that activities and decisions are closely entwined in business processes, striving to render end-to-end processes in BPMN models, their complexity dramatically escalates when business decisions are captured excessively with BPMN constructs. As a result, capturing the decision logic of business processes, is considered to be a separate concern in process modeling. In this regard, Decision Model and Notation (DMN) emerges as the standard notation to model repetitive operational decisions, while it is interestingly well integrated with the BPMN notation.

CHAPTER 4: Modeling business decisions with DMN

This chapter presents the Decision Model and Notation (DMN) and its capabilities to be integrated with the BPMN notation in end-to-end business processes. The main objective is to render how DMN treats the decision modeling as a separate concern, as well as to highlight how DMN decision models can be integrated with BPMN process models. The chapter starts with an introduction to DMN, as the prevalent standard for modeling repetitive operational decisions in a graphical and executable way. Presenting its two levels of decision modeling, namely the decision requirements and the decision logic level, special focus is given on the optimal utilization of its fundamental constructs, such as the Decision Requirement Diagrams (DRDs) and the Decision Tables. Subsequently, the chapter presents how DMN can be integrated with BPMN models by presenting the Decision as a Service (DaaS) paradigm, as well as the five principles for integrated Process and Decision Modeling (5PDM), as proposed in recent literature. Ultimately, the chapter concludes with the expected benefits of a decision and process model integration.

4.1 Introduction to DMN

Considering that BPM is moving towards the separation of concerns paradigm by externalizing decision logic from process flows [51], [54], the organizations' ability to manage centrally their decisions is undisputedly vital to the success of every BPM initiative [2]. Even-if the concept of modeling decisions is not new, since rule-based systems have been leveraged by organizations over years, the need to model decisions and manage them as a separate concern, revived the interest in decision modeling techniques. In this regard, Decision Model and Notation (DMN) has lately come in forefront as a standard notation for modeling and automating decisions. Introduced in 2015 by Object Management Group (OMG), DMN (currently DMN 1.3) has been successfully adopted since then, both in academia and industry, filling the void of decision modeling in the BPM spectrum [51]. Bridging the gap between business and IT, DMN notation is deemed to constitute an intuitive standard for both business users and technical developers [2], [33], where a decision output is generated in a modeled, visible and executable way.

Undoubtedly, decisions are an integral part of every business operation. According to the DMN specification [58], "a decision is the act of determining an output value (the chosen option), from a number of input values, using logic defining how the output is determined from the inputs". However, even if decisions can be distinguished to operational and strategical ones, DMN exclusively focuses on recurrent operational decisions [58], that depend on an established decision logic. Considering their high frequency and process orientation [55], routine decisions that need to be made repeatedly every day [2] formalize the DMN's scope, which intends to define, depict and optionally automate them [58]. Conversely, strategical decisions that are primarily unique in nature, they are not worthwhile to be modeled with DMN notation, since they follow specific set rules without being integrated properly with operational processes [2].

Interestingly, the DMN standard allows to model decisions on two levels, namely the decision requirements and the decision logic level [50], [54]. Upon the former, utilizing a Decision Requirements Diagram (DRD), decisions are modeled in a graphical and hierarchical way, where special attention is given to their interrelationships. However, even if this level of description is sufficient for business analysis, greater detail emerges upon delving into the decision logic level. In this way, the decision logic is encapsulated into decision tables, where its expressions are defined with the Friendly Enough Expression Language (FEEL) [58].

Ultimately, given the fact that DMN was introduced as a standard to complement BPMN [58], which does not capture the decision logic in detail [33], research challenges arise in the context of integrating DMN with BPMN notation. In this sense, the main objective is to consistently integrate the aforementioned notations in end-to-end business processes in a way that business decisions are externally managed, before their outputs are utilized for task executions and routing decisions in BPMN models [2]. As a result, literature has proposed a plethora of approaches in order to integrate DMN decision models with BPMN process models, while for the premise of this thesis, the Decision as a Service (DaaS) paradigm [59] and the five principles for integrated Process and Decision Modeling (5PDM) [54], are well adopted and presented in the following lines.

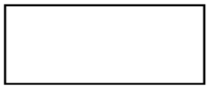
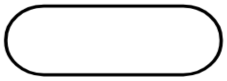
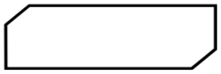
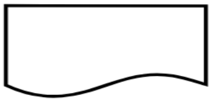
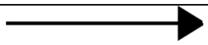
4.2 DMN Constructs

4.2.1 Decision Requirements Diagram (DRD)

Delving into the notation and embarking on decision modeling initiatives with DMN, the decision requirements level come in frontline as the first stage of capturing a decision-making process. Utilizing a Decision Requirements Diagram (DRD) for this purpose, decisions are modeled in a graphical way, where special attention is given to their requirements and their interrelationships. According to the DMN specification [58], “A DRD shows how a set of decisions depend on each other, on input data, and on business knowledge models”. In this regard, considering today’s complex decisions that highly depend on a vast number of inputs, as well as on the outcomes of previously enacted decisions, DRDs can be considered powerful instruments for rendering such complex decision-making in an intuitive and graphical way.

In its simpler format, a DRD is mainly comprised of decisions, inputs and business knowledge models, that collectively define the fundamental notation of the decision requirements level. Admittedly, a decision node, illustrated as a rectangle, is considered to be the focal component of every DRD. Depending on the outcomes of lower-level decisions and a plethora of input data, illustrated as oval elements, decisions are associated with their input parameters, utilizing the DRD's information requirement arrows. Upon evaluating the received input data, decisions invoke one or more business knowledge models, which encapsulate the business "know-how" and the decision logic behind every decision [58]. In this sense, DRD defines the knowledge requirement arrows, tailored to the modeling of the invocation of a business knowledge model by one or more decisions. Additionally, knowledge sources, associated with decision nodes or business knowledge models, by the means of authority requirement arrows, denote an authority or a source of knowledge, like a source document from which the desired knowledge is derived or a domain expert who maintains and updates the desired decision logic [58].

Ultimately, decision models, captured in the context of a DRD, facilitate the hierarchical decision modeling [24], where the upper elements are deemed to "require" information from the lower ones [58]. As a result, DMN enhances the information aggregation and the hierarchy of decision information in different levels of granularity, by aggregating low-level information into higher-level one [56]. In Table 6, the DRD elements are presented, before being utilized in a simple DRD representation model in Figure 53.

	DRD elements	
Element	**Symbol**	**Functionality**
Decision		A decision node illustrates an output from a number of inputs, after invoking the decision logic of one or more business knowledge models.
Input Data		An input data element illustrates the information that is required as input by one or more decisions.
Business Knowledge Model		A business knowledge model renders the decision logic and the business "know-how" behind each decision.
Knowledge Source		A knowledge source illustrates an authority for a business knowledge model or decision, from which the desired logic is derived (e.g., source document, domain expert, etc.).
Information requirement		An information requirement arrow renders input data, as well as decision

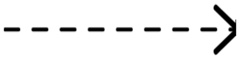
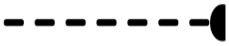
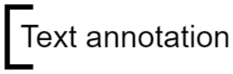
		outputs, that are utilized as inputs to a decision.
Knowledge requirement		A knowledge requirement arrow illustrates the invocation of a business knowledge model by a decision.
Authority requirement		An authority requirement arrow illustrates the dependency of a DRD element on another DRD element that acts as an authority and a source of knowledge.
Text annotation	 Text annotation	A text annotation artifact illustrates an explanatory text or comment, associated with a DRD element.

Table 6 - DRD elements (Adaptation from [58])

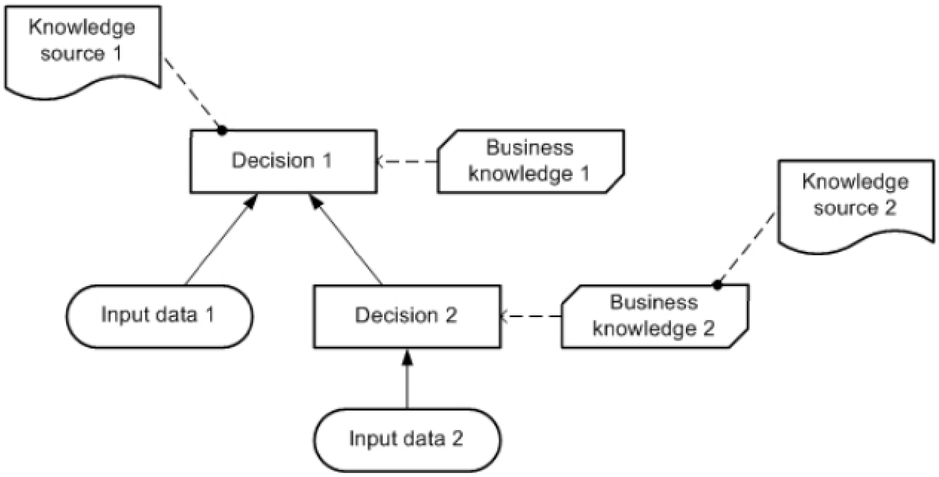


Figure 53 - A simple DRD [58]

Having presented the DRD notation and illustrated a simple DRD, a loan eligibility decision-making process is rendered in the context of a DRD representation in Figure 54. Primarily, the decision-making process is determined by two decision nodes (i.e., “Credit Score”, “Eligibility”), where a dependency between them is established. Following a hierarchical decision modeling approach, a loan eligibility decision is considered to “require” the output of a lower-level decision, where the applicant’s credit score is established. Additionally, invoking the decision logic of the associated business knowledge models (i.e., “Credit Score Calculation”, “Eligibility assessment”), a decision output is determined from the evaluation of a number of input values (i.e., “Previous debts”, “Monthly income”, “Occupation”, “Age”). As a result, in order to decide if an applicant is eligible to get a loan granted, a chain of decisions is made. At first, the applicant’s credit score is determined after evaluating its previous debts, the monthly income and the occupation type. Such outcome serves as an input to a higher-level decision, where along with the applicant's age, the loan eligibility decision is made. Altogether, the decision logic for these decisions is defined by the associated business knowledge models, which in turn educe the needed information from their knowledge sources (i.e., "Bank's policy", "Legislation").

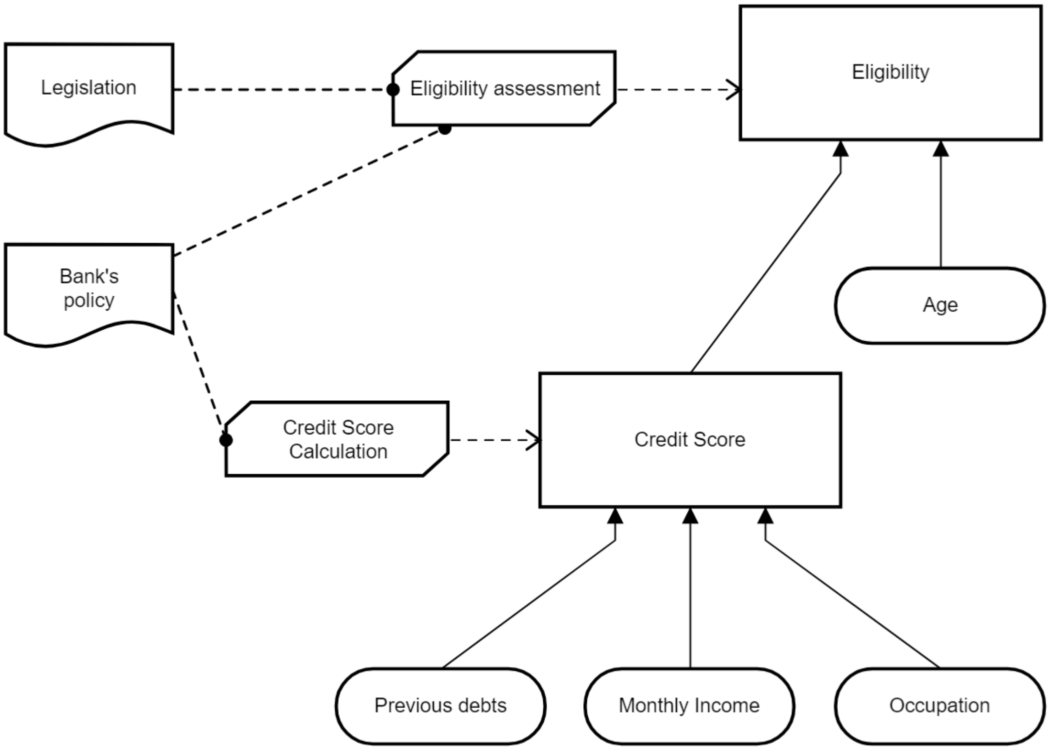
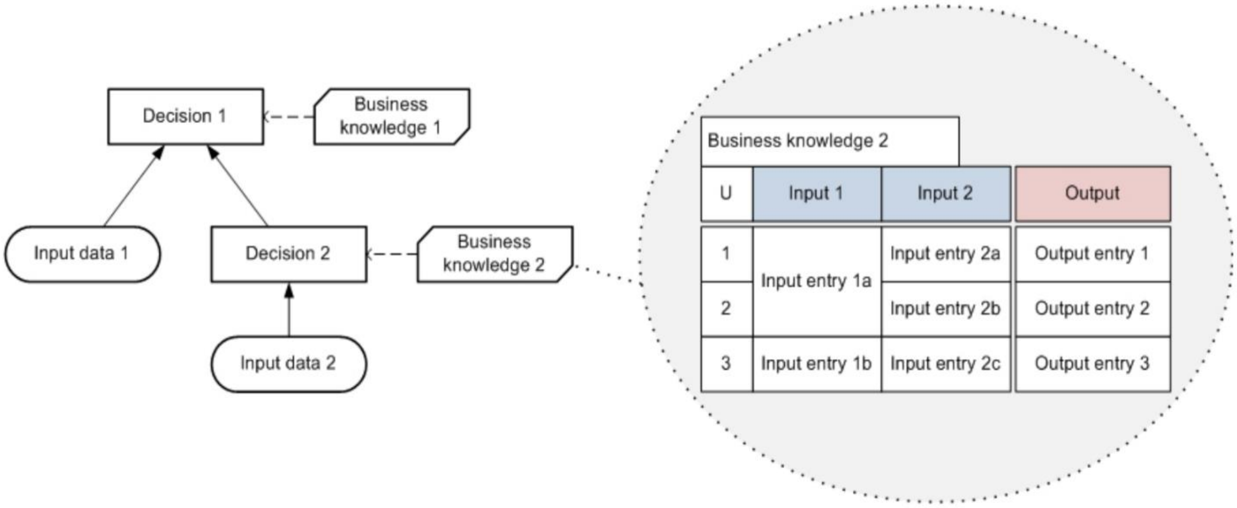


Figure 54 - DRD loan eligibility decision-making process

4.2.2 Decision Tables

Even if the fist level of decision modeling with DMN is sufficient in order for someone to identify decisions, their requirements, as well as the sources of their business knowledge, greater detail emerges at the second level of DMN decision modeling, namely the decision logic level. In this sense, specific representations of the applied decision logic, such as the DMN decision tables, come in frontline, rendering the way that decision's outcomes are derived from its inputs [58].

In most of the cases, the logic of a decision at the DRD level is encapsulated into a business knowledge model, before being expressed by the means of a decision table at the decision logic level. According to the DMN specification [58], a decision table is defined as "a tabular representation of a set of related input and output expressions, organized into rules indicating which output entry applies to a specific set of input entries". Serving as the means for achieving business-IT alignment, decision tables, established in the background of a DRD's business knowledge model, constitute the focal pillar of presenting the decision logic of a decision-making process [2]. In Figure 55, the correspondence of a DRD element, namely a business knowledge model, with the decision logic of a decision table is graphically rendered. However, the grey ellipse and its dotted line do not form part of the notation of DMN and they only indicate a visual correspondence between different levels of decision modeling with DMN.

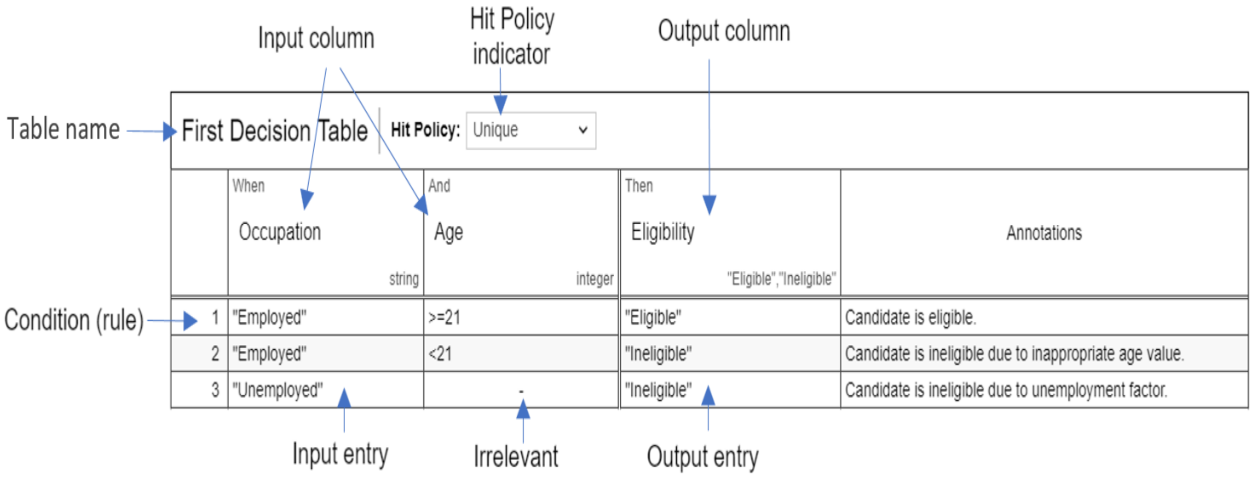


Business knowledge 2

U	Input 1	Input 2	Output
1	Input entry 1a	Input entry 2a	Output entry 1
2		Input entry 2b	Output entry 2
3	Input entry 1b	Input entry 2c	Output entry 3

Figure 55 - Business knowledge model and corresponding decision table [2]

Intuitive in nature, decision tables (Figure 56) are deemed to constitute predefined sets of rows and columns, which encapsulate the desired decision logic. Each table's row corresponds to a single rule or condition, while its columns denote input and output parameters, involved at the evaluation of its conditions. Considering the input and output entries that populate the table's cells, conditions are deemed to apply once a set of values match their input entries, as defined in the decision table.



First Decision Table **Hit Policy:** Unique

	When Occupation string	And Age integer	Then Eligibility "Eligible","Ineligible"	Annotations
1	"Employed"	>=21	"Eligible"	Candidate is eligible.
2	"Employed"	<21	"Ineligible"	Candidate is ineligible due to inappropriate age value.
3	"Unemployed"	-	"Ineligible"	Candidate is ineligible due to unemployment factor.

Figure 56 - Decision Table

Due to the fact that multiple inputs can be involved in the evaluation of a condition, the correct readability of rules is of utmost importance, before evaluating a decision table's result. In this sense, multiple input columns are linked by a logical AND operator, while compacted input entries, separated by a comma within a single cell, are linked with a logical OR operator. During the evaluation of a condition, input entries marked with a dash ('-'), are ignored and subsequently regarded as irrelevant values for the evaluation of the containing rule. The aforementioned concepts are presented in Figure 57, highlighting the readability of decision table rules.

	Input parameter 1	**Input parameter 2**	**Output**
1	Input entry a, input entry b	Input entry c	Output entry d
2	Input entry e	-	Output entry f

Rule 1 is elaborated as:

If the value of input parameter 1 satisfies the *input entry a* **OR** the *input entry b*

AND the value of input parameter 2 satisfies the *input entry c*

Then the rule 1 is activated and the result of the decision table is the *output entry d*

Rule 2 is elaborated as:

If the value of input parameter 1 satisfies the *input entry e*

Then the rule 2 is activated and the result of the decision table is the *output entry f*

Figure 57 - Readability of decision table rules (Adaptation from [58])

Ultimately, a plethora of rules might apply at the same time, matching a given set of input values [58]. However, in order to avoid anomalies, arising from ambiguities between overlapping matching rules [33], DMN decision tables are ruled by hit policies, which elucidate the applied decision logic and determine the cardinality of rules that can apply simultaneously. Considering, additionally, that decisions can be ideally deployed and automatically enacted by decision engines, DMN standard proposes an expression language, namely the Free Enough Expression Language (FEEL), which expresses the decision table's conditions in an executable, yet intuitive, way. In this regard, the supported hit policies and the FEEL expression language are presented in the following lines.

(a) Hit policies

A hit policy is an integral part of every decision table, indicating the cardinality of rules that might apply and the way of determining the table's result when several rules overlap at once [2]. Represented by its first-letter indicator, hit policies are distinguished between single and multiple ones. According to the DMN specification [58], even if a single hit policy might contain overlapping rules, a single result is returned as the decision table's output. In such a case, the hit policy indicates which of the rules to select, determining a single result out of them. Conversely, a multiple hit policy might return output entries from multiple matching rules, where the result is defined as a list of outputs or a simple function of them [58]. The full-set of hit policies, supported by DMN 1.3, is presented in Table 7, before shedding light on the unique and collect sum hit policies, utilized in a decision-making process, which is further simulated in Figures 58 and 59.

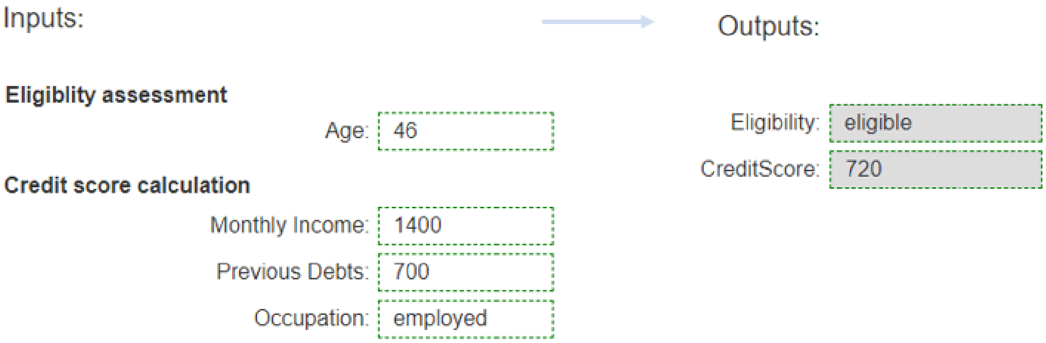
Hit Policies			
Hit policy		Indicator	Functionality
Single hit	Unique	U	The Unique hit policy determines that exactly one rule has to apply and no overlapping rules are allowed.
	First	F	The First hit policy denotes that the result is determined by the first matching rule, while all the other matching rules are ignored.
	Any	A	The Any hit policy denotes that all matching rules must

			generate the same output entry for each output.
	Priority	P	The Priority hit policy assigns a priority to each output entry. The result with the highest priority of all matching rules is returned.
Multiple hit	Collect	C	The Collect hit policy determines the decision result as a list of all output entries of all matching rules.
	Collect sum	C+	The Collect (sum) hit policy returns the aggregation of all output entries of all matching rules.
	Collect max	C>	The Collect (max) hit policy returns the maximum output entry of all matching rules.
	Collect min	C<	The Collect (min) hit policy returns the minimum output entry of all matching rules.
	Collect count	C#	The Collect (count) hit policy returns the number of all matching rules.
	Rule order	R	The Rule order hit policy determines a list of output entries, ordered based on the matching rules order.
	Output order	O	The Output order hit policy determines a list of outputs entries, ordered in a decreasing order of output priority.

Table 7 - Hit policies (Adaptation from [58])

Considering the loan eligibility decision-making process, illustrated in Figure 2 in the context of a DRD representation, in the following lines the aforementioned process is enriched with decision logic behind each decision node. In this regard, decision tables are introduced in the background of business knowledge models, indicating the way that decisions' outcomes derive from their inputs. More specifically, in order to decide if an applicant is eligible to get a loan granted, a credit score is firstly calculated. Conforming to the table's collect sum hit policy, the applicant's credit score is determined after aggregating the scores of all matching rules. As an input to a higher-level decision and adhering to the table's unique hit policy, the loan eligibility is defined by exactly one rule, which satisfies the given set of input values.

This particular example is further examined in Figures 58 and 59, where an applicant's loan eligibility is determined by simulating the underlying decision-making process. Thus, considering the input parameters of Figure 58 and delving into the decision logic of Figure 59, the applicant is characterized as eligible for getting a loan granted. In fact, evaluating at first the monthly income, the previous debts and the occupation type, a credit score is decided (i.e., 720), after aggregating the output entries of all matching rules (i.e., rule 5, 7, 11). Subsequently, utilizing such score and the applicant's age in a higher-level decision, the unique hit policy comes in practice, defining exactly one rule (i.e., rule 1) that determines the final outcome.



Inputs:

Eligiblity assessment

Age: 46

Credit score calculation

Monthly Income: 1400

Previous Debts: 700

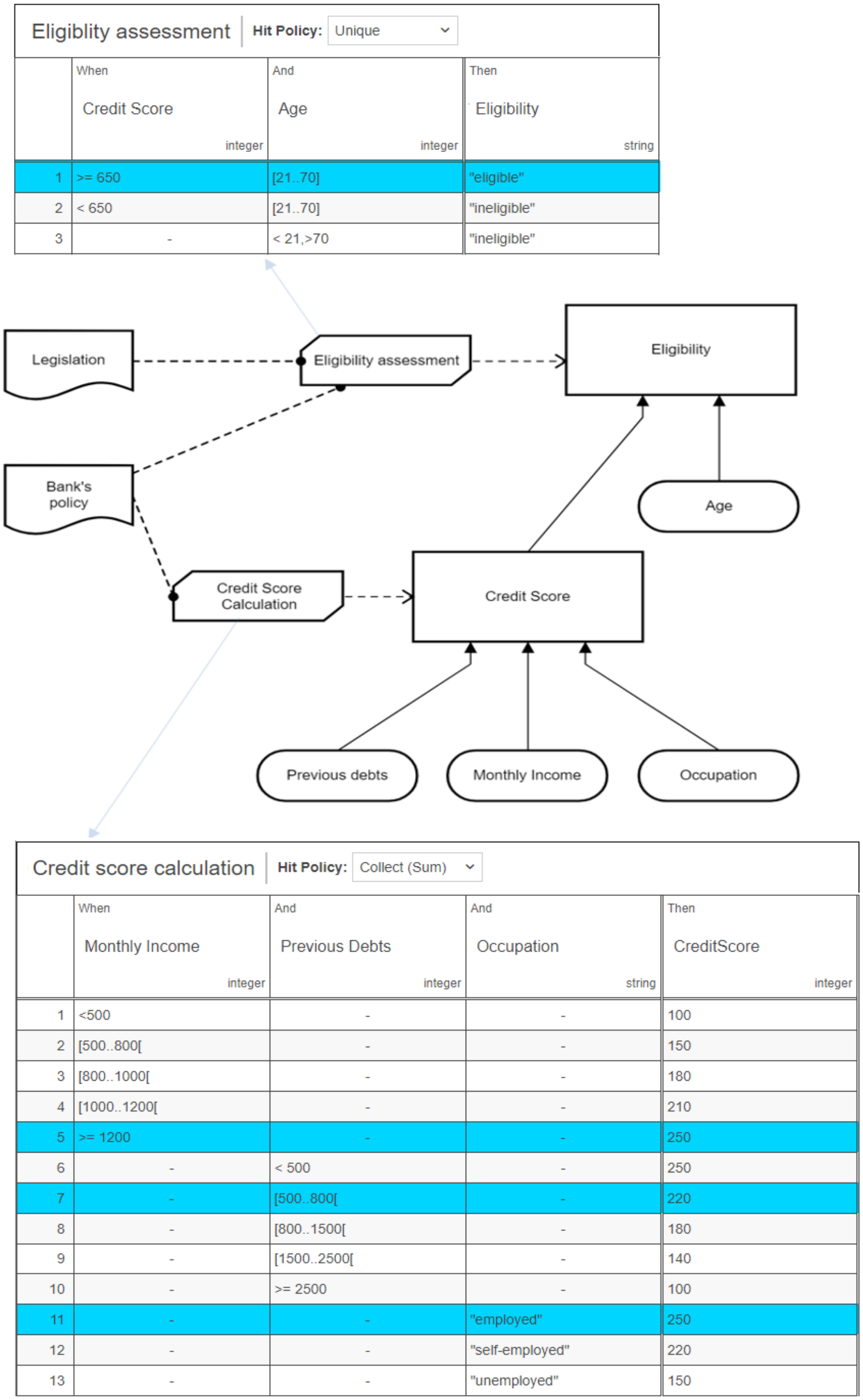
Occupation: employed

Outputs:

Eligibility: eligible

CreditScore: 720

Figure 58 - Simulating a decision-making process (1)



Eligiblity assessment **Hit Policy:** Unique

	When Credit Score integer	And Age integer	Then Eligibility string
1	>= 650	[21..70]	"eligible"
2	< 650	[21..70]	"ineligible"
3	-	< 21,>70	"ineligible"

Credit score calculation **Hit Policy:** Collect (Sum)

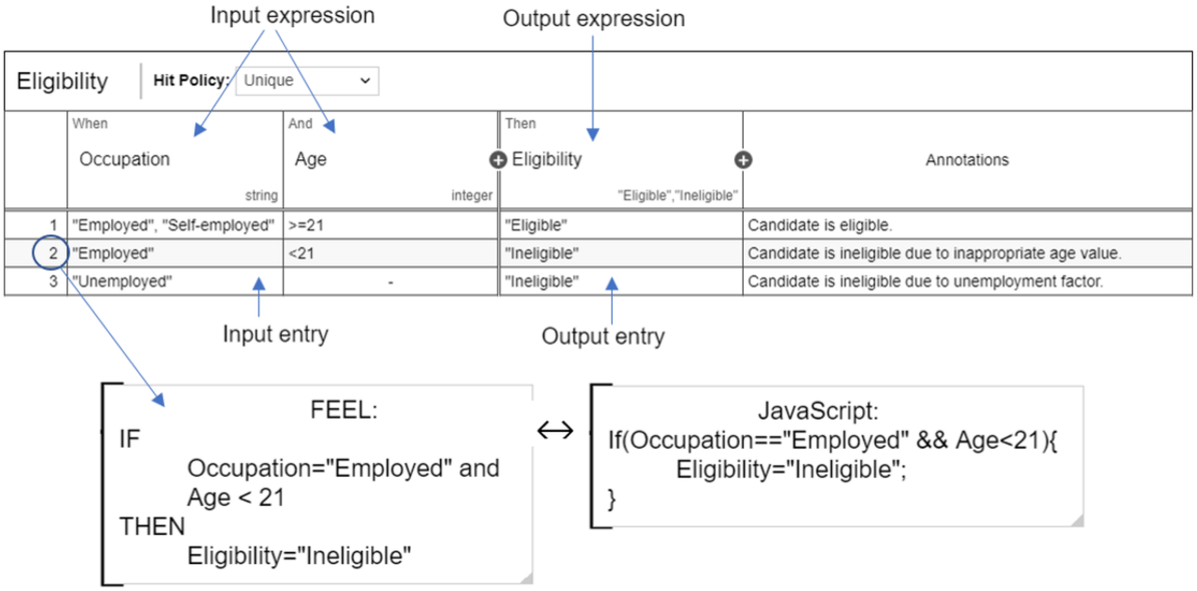
	When Monthly Income integer	And Previous Debts integer	And Occupation string	Then CreditScore integer
1	<500	-	-	100
2	[500..800[	-	-	150
3	[800..1000[	-	-	180
4	[1000..1200[	-	-	210
5	>= 1200	-	-	250
6	-	< 500	-	250
7	-	[500..800[	-	220
8	-	[800..1500[	-	180
9	-	[1500..2500[	-	140
10	-	>= 2500	-	100
11	-	-	"employed"	250
12	-	-	"self-employed"	220
13	-	-	"unemployed"	150

Figure 59 - Simulating a decision-making process (2)

(b) FEEL – Expression Language

Considering that DMN models can be optionally deployed and automatically enacted by decision engines, the input and output expressions of decision tables must be determined in a uniform and standard way. For this purpose, DMN introduces the Friendly Enough Expression Language (FEEL) as a standard language for the translation of decision tables' expressions to valid executable ones and the determination of their execution semantics [58].

Admittedly, the FEEL expression language is a formally precise language that not only is interpreted and executed by decision engines, but also remains understandable for both business and IT executives [2]. In fact, FEEL expressions are deemed to be more intuitive than scripting languages and typically selected by business users without programming knowledge [2]. However, based on the languages that a decision engine is able to interpret, as well as on the technical skills of business users, scripting languages (e.g., JavaScript, JUEL, etc.) might substitute the FEEL expression language, rendering the decision table conditions in a standardized and executable way. In Figure 60, a decision table's rule is translated to an executable condition, utilizing a FEEL syntax and a corresponding JavaScript one.



Eligibility | Hit Policy: Unique

	When Occupation string	And Age integer	Then Eligibility "Eligible","Ineligible"	Annotations
1	"Employed", "Self-employed"	>=21	"Eligible"	Candidate is eligible.
2	"Employed"	<21	"Ineligible"	Candidate is ineligible due to inappropriate age value.
3	"Unemployed"	-	"Ineligible"	Candidate is ineligible due to unemployment factor.

```
FEEL:
IF
        Occupation="Employed" and
        Age < 21
THEN
        Eligibility="Ineligible"
```

```
JavaScript:
If(Occupation=="Employed" && Age<21){
        Eligibility="Ineligible";
}
```

Figure 60 - FEEL expression language

4.3 DMN and BPMN integration

With the introduction of DMN, decisions are able to be externalized and encapsulated in a separate DMN decision model, instead of modeling decision-making processes in a BPMN process model [56]. Business decisions are treated as a separate concern, while decision modeling is regarded as a sovereign part of enterprise modeling [52]. However, research challenges arise on whether decisions and processes can be separated and consistently integrated [54], rendering business processes as decision aware and decision intelligent [51]. Considering that BPMN is an older standard than DMN, its specification does not indicate a way of invoking DMN decision models from BPMN process models [2]. Nonetheless, a plethora of research works have been introduced in recent literature, investigating the way that DMN decision models can be integrated with BPMN process models. For the premise of this thesis, the Decision as a Service (DaaS) [59] paradigm and the five principles for integrated Process and Decision Modeling (5PDM) [54], are well adopted and presented in the following lines. In both cases, the BPMN business rule tasks are utilized as an effective way of invoking DMN decision models from BPMN process models.

4.3.1 Decision as a Service (DaaS)

Taking into account that the Separation of Concerns (SoC) paradigm and the externalization of decisions have been facilitated with the introduction of DMN, the decision management has moved towards the paradigm of Service-Oriented Architecture (SOA), by treating decisions as externalized services [54], [59]. Since DMN is independent of the application and the invoking context, this makes it explicitly interesting for SOA architectures [59]. In this regard, decisions can be invoked from any process model as a service, introduced in [59] as Decision as a Service (DaaS) paradigm. In such approach, a SOA layered architecture consists of a process, a service and a decision layer, where collectively establish an interface between process and decision models [59] (Figure 61). Interestingly, a BPMN process model can invoke the service through a business rule task, providing information at runtime. In turn, the service provides to the decision layer the input data, before receiving back the decision output and sending it to the invoking process. After the service being executed, the process obtains the decision outcome and utilizes it for routing decisions and task execution as the process flow unfolds [59].

Ultimately, implementing decisions as externalized services, processes mitigate the burden of rendering decision logic with control flow structures, being responsible exclusively for ensuring a sound invocation of the decision [59]. Thus, they become undoubtedly more visible and understandable, while their complexity significantly decreases.

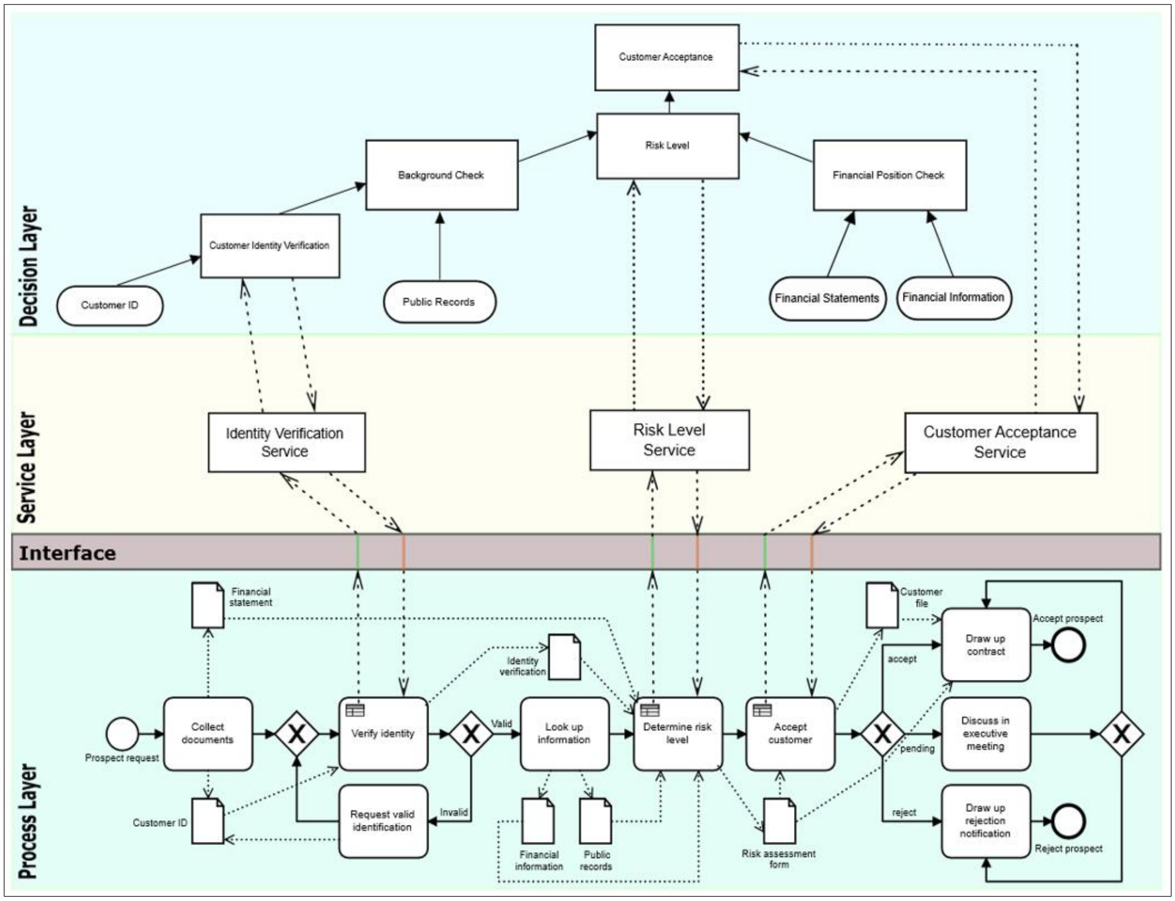
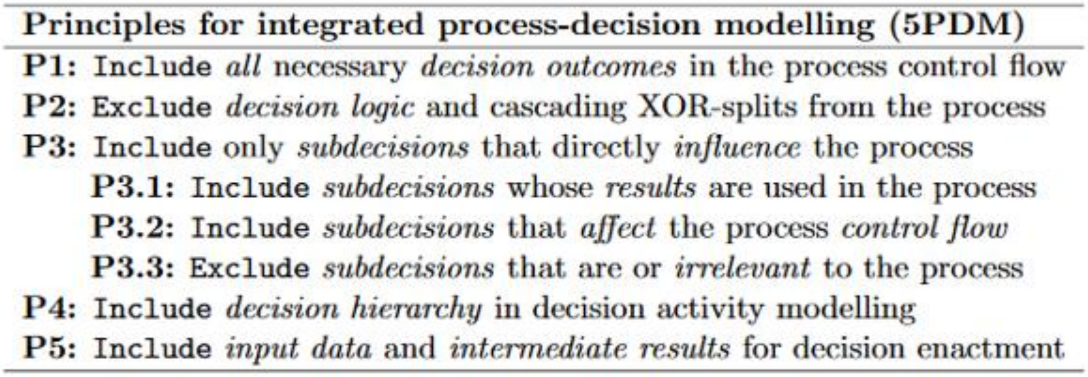


Figure 61 - Decision as a service (DaaS) [59]

4.3.2 Integrated process and decision modeling

In order to render processes decision-aware and induce their decision-intelligence, a consistent integration between process and decision modeling is of utmost importance by design. In this regard, Hasić et al. [54], introduce the 5PDM paradigm, as five principles for integrated process and decision modeling (Figure 62). Considering that an externalized decision model is necessary to remain consistent with its invoking process model, such principles can pave the way for a sound communication between the two models [51].

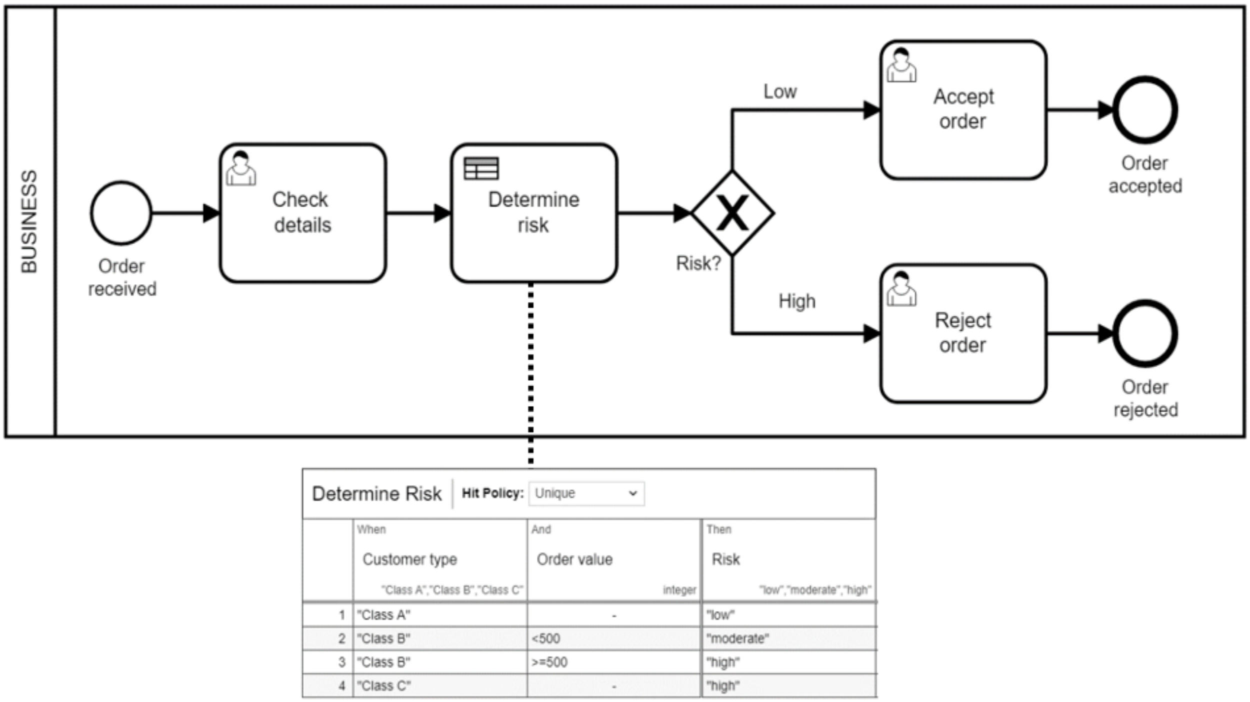
According to the first principle, all possible decision outcomes must be included in the process control flow after invoking a decision. Considering a case where no output flow is dedicated to a decision outcome, the process cannot proceed properly, proven to be inconsistent with the surrounding environment [54]. In Figure 63, an inconsistent process and decision modeling is rendered, given the fact that not all possible decision outcomes are included in the process logic flow. In this sense, in case of a moderate risk assessment, the process cannot be further executed, engendering a runtime error.



Principles for integrated process-decision modelling (5PDM)

P1: Include *all* necessary *decision outcomes* in the process control flow
P2: Exclude *decision logic* and cascading XOR-splits from the process
P3: Include only *subdecisions* that directly *influence* the process
  P3.1: Include *subdecisions* whose *results* are used in the process
  P3.2: Include *subdecisions* that *affect* the process *control flow*
  P3.3: Exclude *subdecisions* that are or *irrelevant* to the process
P4: Include *decision hierarchy* in decision activity modelling
P5: Include *input data* and *intermediate results* for decision enactment

Figure 62 - Five principles for integrated process and decision modeling [54]



Determine Risk | Hit Policy: Unique

	When Customer type "Class A","Class B","Class C"	And Order value integer	Then Risk "low","moderate","high"
1	"Class A"	-	"low"
2	"Class B"	<500	"moderate"
3	"Class B"	>=500	"high"
4	"Class C"	-	"high"

Figure 63 - Inconsistent process and decision modeling

According to the second principle, process models must exclude the decision logic, which is frequently captured with cascading exclusive gateways that lead to decision-trees and spaghetti like processes [54]. In Figure 64, a decision-tree structure is engendered, utilizing a series of cascading gateways that imitate the desired decision logic. In this regard, a business rule task that invokes a decision model can replace the hard-coded decision logic of BPMN models (Figure 65).

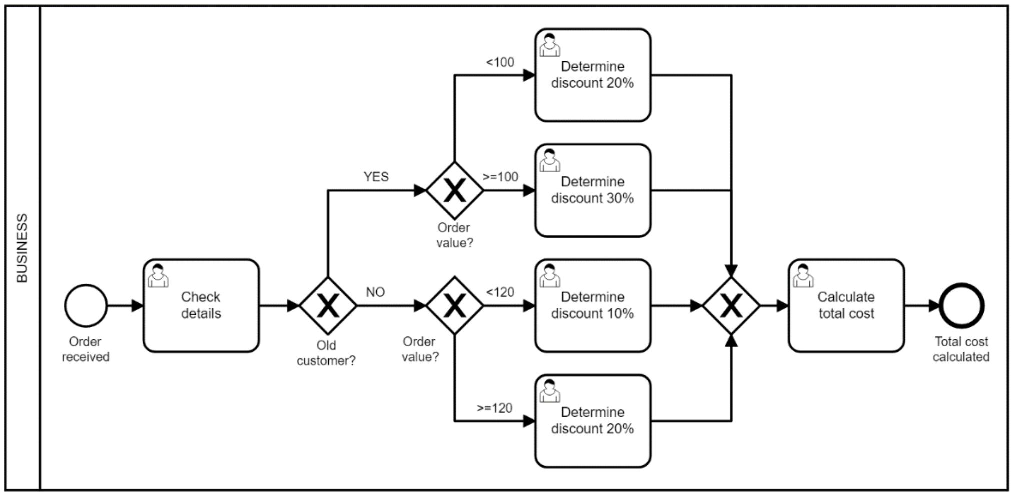
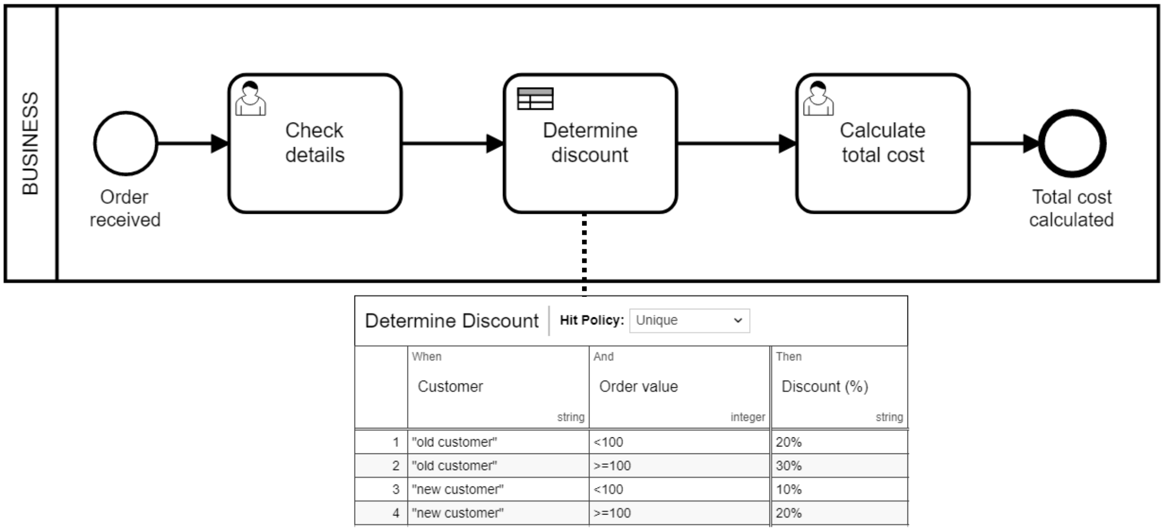


Figure 64 – Decision logic and decision-tree structures in BPMN process models



Determine Discount

Hit Policy: Unique

	When Customer string	And Order value integer	Then Discount (%) string
1	"old customer"	<100	20%
2	"old customer"	>=100	30%
3	"new customer"	<100	10%
4	"new customer"	>=100	20%

Figure 65 - BPMN business rule tasks

Adhering to the third principle, a process model should contain exclusively the subdecisions that have a direct influence on the control flow or their outcomes are utilized as the process lifecycle unfolds. That is to say, decisions that are considered to be irrelevant with the process execution, should not explicitly be modeled as decision activities within the process model, since higher level decisions are present in the remainder of the process [54], [78]. Considering a process model and a relative DRD representation in Figure 66, the process model is deemed to be inconsistent with the proposed principal. Given the fact that the “Calculate credit score” decision activity neither influences the control flow of the process nor generates a result that is utilized as the process unfolds, it can be excluded from the process model without affecting the consistency between the process and the decision model. On the other hand, the “Determine risk” decision activity should undisputedly remain in the process, taking into account that its outcome can be further utilized in the process.

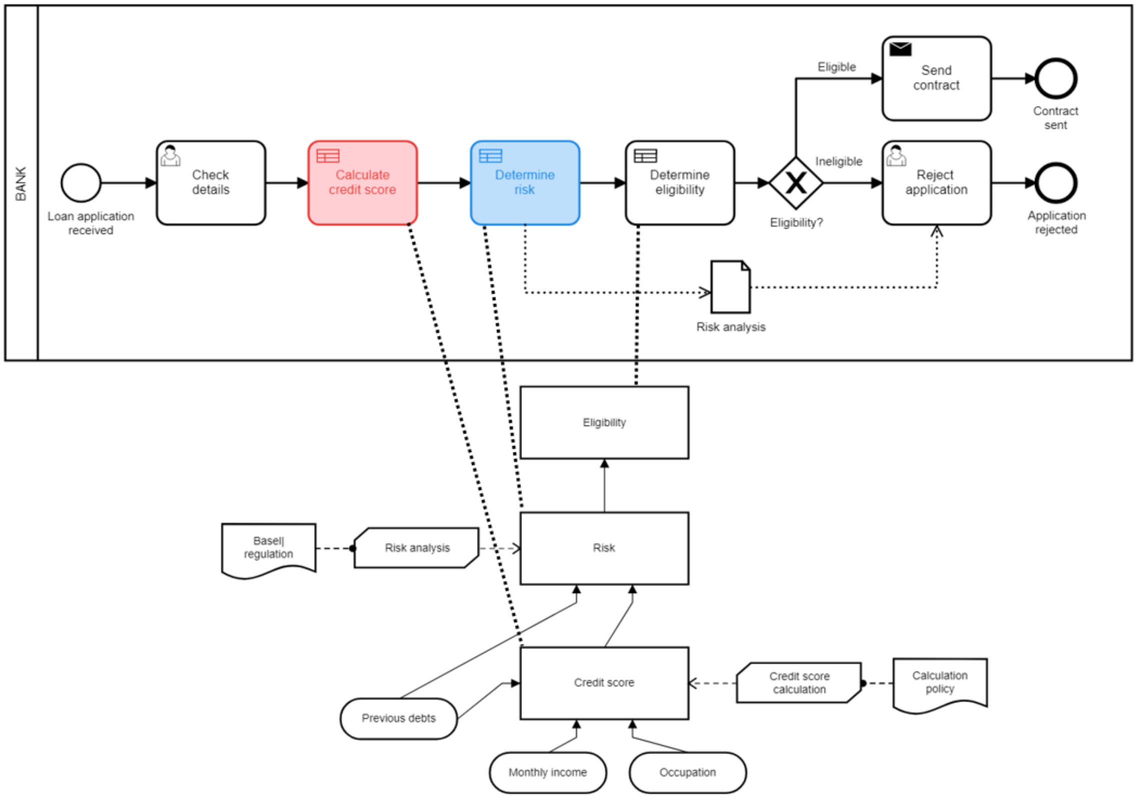


Figure 66 - Subdecisions as decision activities in a process model

Moreover, conforming to the fourth principle, the decision hierarchy present in a decision model, should explicitly be respected by the order of decision activities in a process model [54], [78]. In this regard, lower-level subdecisions that are modeled within a process model, should be located before higher-level ones, respecting the hierarchy of decision-making process of a DRD model. In Figure 67, decision activities within the process model are not ordered according to the decision hierarchy in the decision model, causing an inconsistency between the process and the decision modeling.

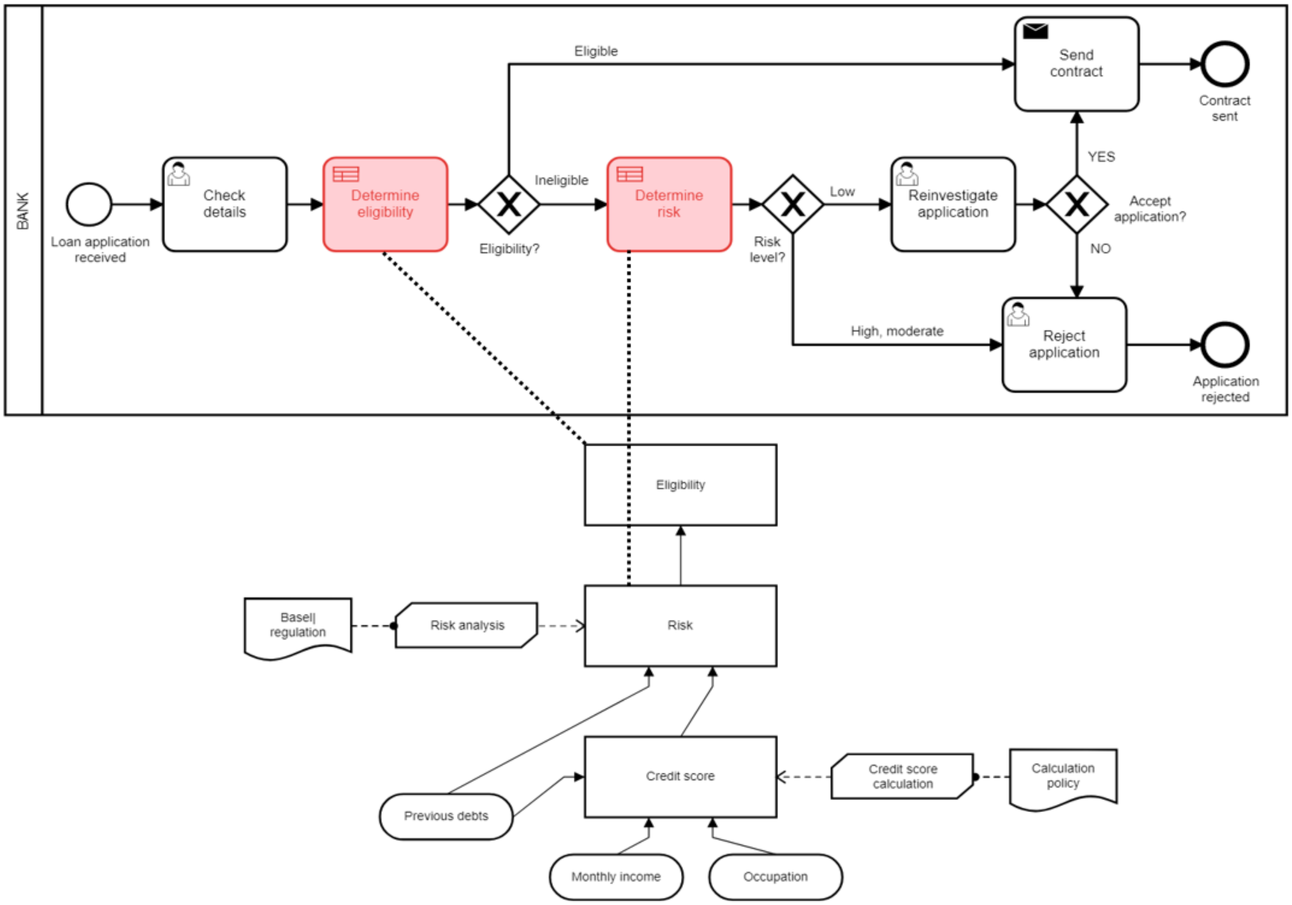
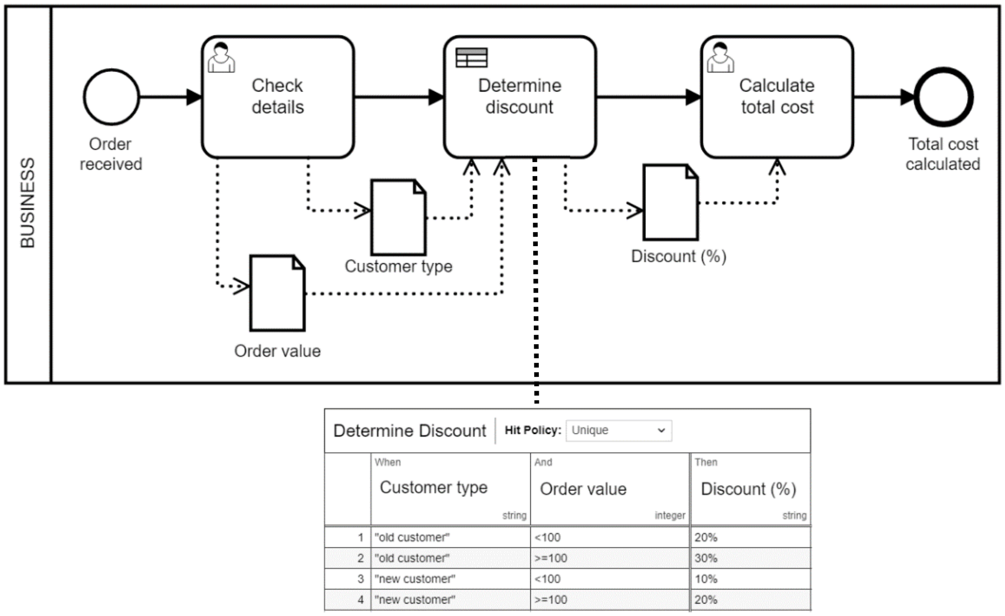


Figure 67 - Ordering decision activities in a process model

Last but not least, both data indispensable to a decision enactment and their results, should explicitly be modeled and included in the invoking process model, satisfying the fifth proposed principle. Paving the way for a sound communication between the two models, decision inputs and outputs can accurately be established [51], utilizing the BPMN data objects [54]. In Figure 68, a consistent data management between a process and a decision model is rendered, leveraging the concept of BPMN data objects. In such a way, the decision inputs and outputs are explicitly modeled, facilitating a consistent integration between the BPMN and the DMN model.



Determine Discount | Hit Policy: Unique

	When Customer type string	And Order value integer	Then Discount (%) string
1	"old customer"	<100	20%
2	"old customer"	>=100	30%
3	"new customer"	<100	10%
4	"new customer"	>=100	20%

Figure 68 - Data management for process and decision modeling integration

4.3.3 Expected benefits

Adhering to an integrated, yet separated, approach of business process and decision modeling, a plethora of benefits come in frontline. The main advantage is deemed to be the better flexibility and maintainability of both processes and decisions, treating them as separate concerns that can effortlessly interact with each other. Separating a business process into distinct sections that share a common interface, such as the process flow and the decision logic, each section can be addressed in a different model without escalating the whole process complexity. Considering that these models have their own consistency, since in order to understand one model it is not compulsory to have an insight into the other one [55], flexibility, agility and maintainability of both processes and decisions are well attainable [51], [54], [78]. In this regard, each model can consider the other one as a black box, defining exclusively their required inputs and outputs, indispensable to their sound communication [55]. Given the fact that decisions are externalized in a separate model, decisions can be invoked by any process model, considering that a sound integration between the two models exists [51]. Therefore, the modularity and reusability of the decision logic is well facilitated [52], [54], [56], comparing to the case that decisions are hard-coded and confined to a decision point in a single process model [56].

Furthermore, extracting and decoupling the decision logic into a separate DMN model that at any time can be invoked as a service, the complexity metrics of BPMN models significantly decrease [56]. Considering that such decision logic would otherwise be captured by BPMN constructs, model-size based metrics are well refined, thereby facilitating the understandability and maintainability of process models [56]. Additionally, the impact on the process model structuredness, caused by modification of the decision logic, is significantly minimized, since decisions are treated externally [23]. Comparing to the case that the decision logic is convoluted with the process logic of BPMN models, modelers will not need to rearrange elements in order to adapt their process models to the desired decision logic [2].

4.4 Summary

This chapter presented the DMN notation and its capabilities of being integrated with the BPMN notation. As a standard for modeling and executing repetitive operational decision-making processes, DMN treats decisions as separate concerns. Utilizing its fundamental constructs, namely the DRDs and the decision tables, decision requirements and the decision logic are thoroughly rendered. Given the fact that decisions can automatically be enacted by decision engines, DMN additionally introduces the FEEL expression language for transforming declarative decision conditions to executable rules. Interestingly, even if DMN captures business decisions in separate models, research challenges arise in the context of integrating DMN with BPMN in end-to-end business processes. In this regard, a plethora of academic works has enriched the recent literature, proposing modeling guidelines for their optimal integration and bringing DMN to the service-orientation paradigm. Overall, decision-aware processes can be created without escalating their complexity, while flexibility and maintainability of both processes and decisions are well refined.

CHAPTER 5: Business Process Management Systems and the Camunda BPM ecosystem

This chapter presents the evolution from traditional Process Aware Information Systems (PAISs) to contemporary Business Process Management Systems (BPMSs). The aim is to highlight various dimensions of such systems, as well as to shed light on their architectures, before a renowned BPMS product, namely the Camunda BPM platform, is utilized in Chapter 6, serving the practical scope of this thesis. The chapter begins by outlining the concept of PAISs. Next sections are dedicated to Workflow Management Systems (WfMSs), as a specific category of PAISs, tailored to the enactment of workflows. By referencing the Workflow Reference Model, as proposed by the Workflow Management Coalition (WfMC), the main characteristics of WfMSs arise, before focusing on BPMN and DMN based workflow engines. Subsequently, Business Process Management Systems (BPMSs) are presented as modern variants and descendants of traditional WfMSs. Finally, a renowned BPM platform, namely the Camunda BPM, comes in frontline, before being utilized in Chapter 6 for the orchestration and the enactment of a business process.

5.1 Process Aware Information Systems (PAISs)

During the 1990s, the focus on process-orientation and the establishment of management trends, such as the Business Process Reengineering (BPR), have set business process concept in forefront [2]. At that time, a range of information systems were utilized, however, being unaware of the processes in which they were involved [9]. Engendering a shift from data to process orientation, process-awareness was triggered in information systems [4]. As a result, Process Aware Information Systems (PAISs) came in frontline as the means for supporting BPM initiatives [79] and enacting dynamically business processes at an operational level [80]. According to Dumas et al. [4], a PAIS constitutes “a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models”. This definition is well adopted for the premise of this thesis, considering the attention that is given not only in the coordination of human participants and application softwares, but also due to its focus on process models as the means for coordinating and orchestrating processes and resources.

Notwithstanding that it is not explicitly stated in the previous definition, process models are primarily rendered in a graphical notation [10]. Decoupling, thus, the process logic out of the application code of PAISs [80], process enactment exclusively depends upon executable process models, rendering in this way the process-awareness of such systems [24]. Considering that the process logic of PAISs is driven by explicit process models, in case of perpetually changing business processes, the modifications have to be implemented in the process model level, rather than modifying the software's source code [4].

Traditionally, PAISs were tailored to the coordination of predictable and repetitive business processes, that were able to be presented explicitly in a formal process model prior to their execution [24]. However, contemplating the unstable enterprise environment, flexibility of PAISs is deemed to accommodate the need for evolving processes and support unstructured operations [24]. Such deviation from prespecified processes is well applicable on the basis of adaptive PAISs, that enable under execution to adapt the structure of process models in order to meticulously render the desired process logic [24].

Currently, sophisticated enterprise systems, such as Enterprise Resource Planning (ERP) systems and Customer Relationship Management (CRM) systems, to exemplify some, prevail in today's organization ecosystem, enhancing business agility. Such systems, being aware of the context and the processes that are involved in, are frequently characterized under the term of PAISs [5]. Although they do not compulsory coordinate processes through a generic workflow engine, the process concept is well established, since they are aware of the processes and operations that they need to support [11]. Ultimately, linking Information Technology (IT) to business process enactment, PAISs are tailored to the execution of entire business operations in a process-oriented manner, rather than executing individual tasks, that have not any knowledge and awareness of the surrounding process [5].

5.2 Workflow Management Systems (WfMSs)

5.2.1 Introduction to WfMSs

A specific category of PAISs is the Workflow Management Systems (WfMSs), embracing the concept of operating in a process-oriented manner. In the last decades, WfMSs have been primarily proposed for the execution and coordination of business processes, let alone their automation in the context of workflow automation paradigm [4].

Specifically, a WfMS constitutes a process-oriented system where its workflow engine is attributed as its focal component. Executing and interpreting the process logic of a predefined process model, WfMSs ensure that processes are executed in a preordained way without any concessions [5]. As such, process models, formerly modeled in a visual language e.g., Business Process Model and Notation (BPMN), serve as the source code of WfMSs, while facilitating the business-IT alignment [2]. Considering that processes require frequent adaptations to organizational, technical and environmental changes, the separation of the process logic and application functionality of a WfMS is one of its more distinguishable features. In this regard, modifications of the process logic can effortlessly put into effect by changing the structure of the model, as compared to the situation where the business logic is hard-coded and buried inside thousands of lines of code [5].

As the process lifecycle unfolds, a workflow consists of coordinated chain of activities, executed to fulfill a predefined objective. Hence, deploying an executable process model to the workflow engine of a WfMS, service orchestration and human workflow management are well attainable [2]. On the basis of executing the process logic of the pre-deployed process model, process participants and IT systems are automatically reached at the right time with the right information [81] (Figure 69). As such, interpreting the process logic, captured in a meta-model format, tasks and information are automatically distributed between participants, work items are added to user work lists, while applications are invoked [36].

Overall, dealing with application integration, interoperability and implementation correctness [23], a WfMS is widely accepted as a means of improving organizational performance [82]. Considering that work is routed by an automated system that reaches human participants and application services in a faster and a more straightforward way, waiting and service times can be decreased, while greater control and worker satisfaction are well attainable [82].

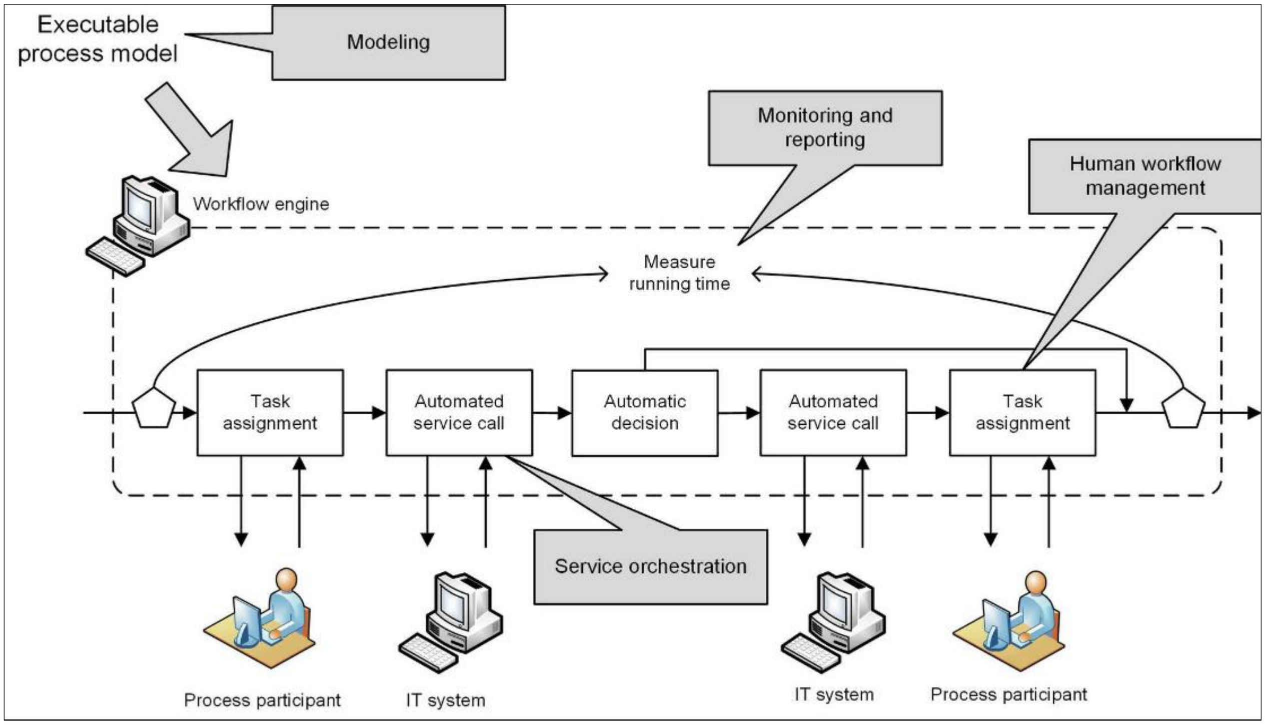


Figure 69 - Process model execution with workflow engine [2]

5.2.2 WfMS functionalities according to the Workflow Reference Model

Even twenty-five years after the foundation of the Workflow Management Coalition (WfMC) and the introduction to its reference model [63], the desired functionalities and characteristics of a typical WfMS can be adequately described according to this framework [11]. More specifically, the main functionalities of a WfMS can be distinguished in three levels. Firstly, a WfMS supports the build-time design of workflows, facilitating the translation of conceptual process representations to executable models. Secondly, runtime control functions, involving a workflow enactment service, where a workflow engine lies at their core, support the process instanciation and the management of workflow processes. Subsequently, run-time interactions facilitate the communication of a workflow with human participants, as well as the automatic invocation of application services, by passing the appropriate data on the right time [63]. The aforementioned fundamental characteristics of a WfMS are displayed in Figure 70.

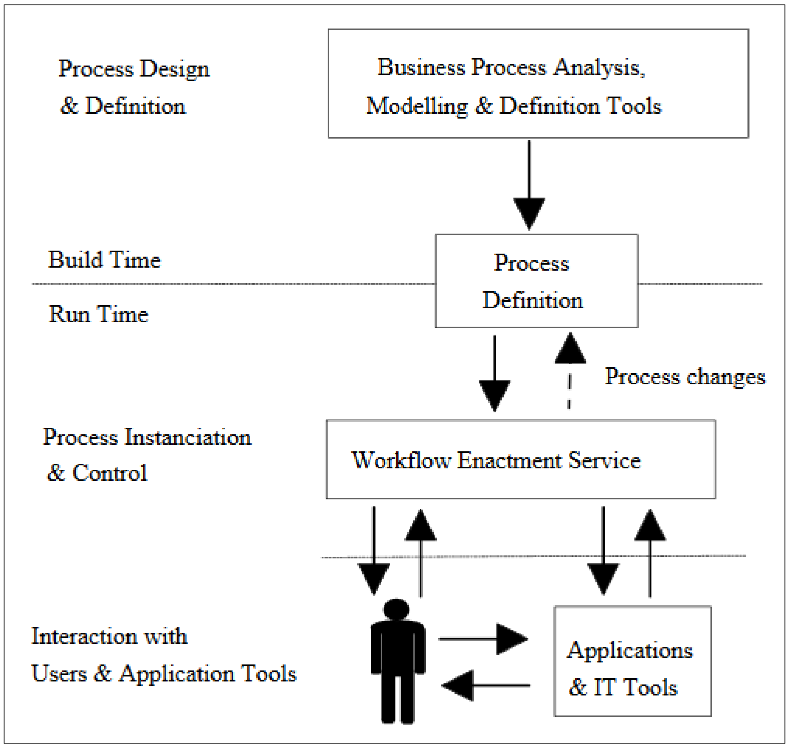


Figure 70 - Workflow Management System (WfMS) characteristics [63]

However, considering the previous abstract perspective of WfMS functionalities, the Workflow Management Coalition established the Workflow Reference Model [63] as a standardized framework that describes how the components of a typical WfMS should interact in terms of predefined interfaces (Figure 71).

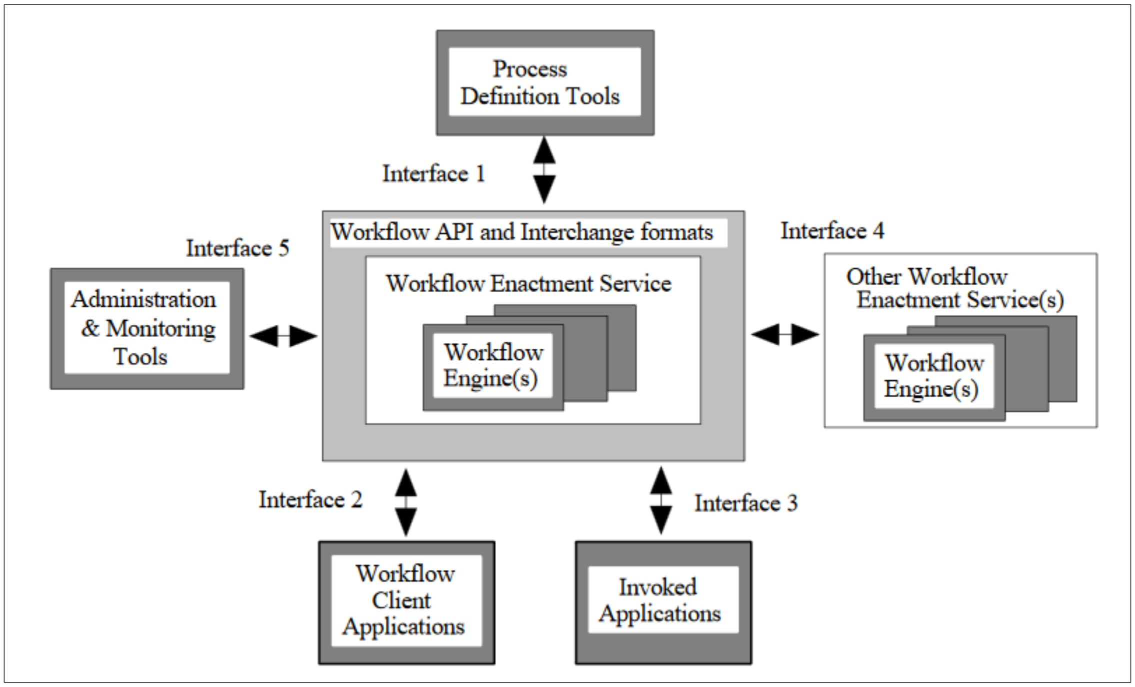

Process Definition Tools
Interface 1
Workflow API and Interchange formats
Interface 5
Interface 4
Administration & Monitoring Tools
Workflow Enactment Service
Workflow Engine(s)
Other Workflow Enactment Service(s)
Workflow Engine(s)
Interface 2
Interface 3
Workflow Client Applications
Invoked Applications

Figure 71 - Workflow Reference Model - Components and interfaces of workflow architecture [63]

Within a WfMS's architecture, the workflow engine is considered to be the focal component, orchestrating and conducting workflow enactment in a runtime environment. According to the Workflow Reference Model [63], a workflow engine is "A software service or 'engine' that provides the run time execution environment for a workflow instance." Initiating both processes and decisions, there is often an underlying integration between workflow and decision engines, in a way that they can both described under the generic term of workflow engines [2]. Importantly, the WfMS engine provides standard interfaces, by which the workflow enactment service can interact with and be accessed from other architecture components. In such interactions, Workflow Relevant Data (i.e., process variables), indicating the state transition, as well as specific process instance information, are explicitly transferred between WfMS's components, consuming system's standard interfaces [63]. Interface 1, is tailored to the exchange of complete process definitions in a way that they can be interpreted at runtime by the workflow engine. In such interface, an executable process model, in terms of a relevant meta-model, is deployed to the workflow enactment service, providing an executable description of interrelated activities, transition conditions, workflow relevant data, etc. Having being enacted, the workflow interacts with the users of the WfMS, by consuming the Interface 2. In such a way, Workflow Relevant Data are transferred to workflow client applications, such as a typical worklist handler, where process participants can interact with the workflow. Interestingly, invoking the

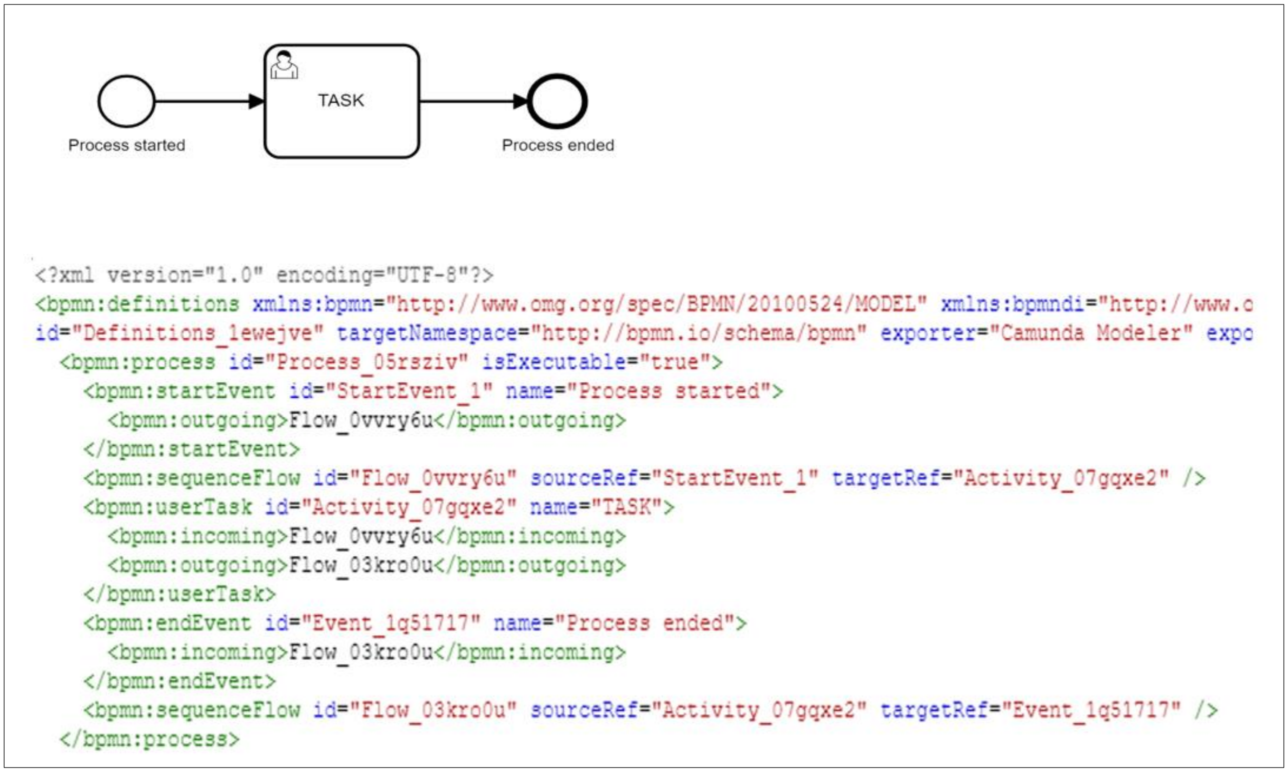
Interface 3, the WfMS can interact with external applications, such as ERPs or mainframe legacy systems, facilitating the enterprise system integration and the application binding. Moreover, leveraging the Interface 4, a workflow engine might interact with other workflow enactment services, facilitating, thus, the work allocation and the enactment of heterogeneous workflow engines. Last, Interface 5, tailored to administrating and monitoring workflow instances, transfers valuable information, gathered during runtime, from a workflow enactment service to administrative tools that support the monitoring and analysis of the overall workflow process.

Ultimately, the Workflow Reference Model is utilized in order to explicitly represent the standard components of a WfMS, as well as indicate vendors' conformance to the standard framework. However, contemporary WfMS might enhance their functionalities, by providing additional features and interfaces in order to fulfill the fast-changing customer needs.

5.2.3 BPMN and DMN-based workflow engine

Since the advent of BPMN 2.0 notation, a plethora of workflow engines came in frontline purported to conform to BPMN 2.0 specification [10]. Providing a standardized meta-model and a serialization format, BPMN models were able to be exchanged among different vendors' tools, let alone be interoperable between different engines [41], [42]. As a result, numerous vendors since then have launched their proprietary products, claiming to support partially or completely the BPMN 2.0 specification. However, since vendors deviate from the native BPMN support, migration from one engine to another one is frequently impossible, when the target engine does not support a set of required features, as captured in the model's standardized serialization format [60]. Therefore, standard conformance to the BPMN 2.0 specification is deemed to be of paramount importance for avoiding a vendor lock-in [60]. Considering that BPMN was accepted as an ISO (ISO/IEC 19510:2013) standard [43], BPMN execution conformance is explicitly defined according to the International Standard. In this regard, workflow engines "claiming BPMN Execution Conformance type MUST fully support and interpret the operational semantics" [44], in order to purport to support BPMN 2.0 notation in a native way.

Interestingly, for a BPMN-based workflow engine, an executable model constitutes the main artifact for executing the underlying process logic [2]. Defining a new level of precision, a previous conceptual and abstract model can be transformed to an executable one, encompassing all the required information needed by an engine, in order to be executed without any ambiguities and misinterpretations [3]. In this regard, upon modeling a BPMN process, each element is translated to a standardized XML format [2] (Figure 72). The engine, thereafter, reads the XML file and executes it directly, as this was the source code of a software solution [2].



```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:bpmndi="http://www.o
id="Definitions_1ewejve" targetNamespace="http://bpmn.io/schema/bpmn" exporter="Camunda Modeler" expo
  <bpmn:process id="Process_05rsziv" isExecutable="true">
    <bpmn:startEvent id="StartEvent_1" name="Process started">
      <bpmn:outgoing>Flow_0vvry6u</bpmn:outgoing>
    </bpmn:startEvent>
    <bpmn:sequenceFlow id="Flow_0vvry6u" sourceRef="StartEvent_1" targetRef="Activity_07gqxe2" />
    <bpmn:userTask id="Activity_07gqxe2" name="TASK">
      <bpmn:incoming>Flow_0vvry6u</bpmn:incoming>
      <bpmn:outgoing>Flow_03kro0u</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:endEvent id="Event_1q51717" name="Process ended">
      <bpmn:incoming>Flow_03kro0u</bpmn:incoming>
    </bpmn:endEvent>
    <bpmn:sequenceFlow id="Flow_03kro0u" sourceRef="Activity_07gqxe2" targetRef="Event_1q51717" />
  </bpmn:process>
```

Figure 72 - XML standardized serialization format

Subsequently, the workflow engine, following the process logic of a pre-deployed model, is able to control the process flow and be aware at any time of what has to be performed next [2]. As a result, utilizing the BPMN 2.0 model as a blueprint, the workflow engine is able to interpret the execution semantics of BPMN elements and enact accordingly to their specification. In this sense, process participants are reached automatically with the right information in order to perform their tasks, while enterprise applications are automatically invoked (Figure 73). In this regard, ranging from different messaging protocols and web services [3], like Representational State Transfer (REST) calls, heterogeneous, autonomous and distributed systems can be integrated with a workflow engine, facilitating, thus, the enterprise application integration [10]. Therefore, the workflow engine can be perceived as a “middleware” component [27], where the ordering of calls to the software systems adheres to the process model logic [10].

Altogether, by modifying the structure of the BPMN model, the process logic can accordingly be adapted without coding the behavior of the application system [2], [10]. In such a way, higher flexibility is well attainable [18], since process models are effortlessly adapted to fulfill additional requirements, before being immediately enacted by a workflow engine [10], [11]. In the end, a BPMN-based process engine implements religiously the semantics captured in the pre-deployed process model [3].

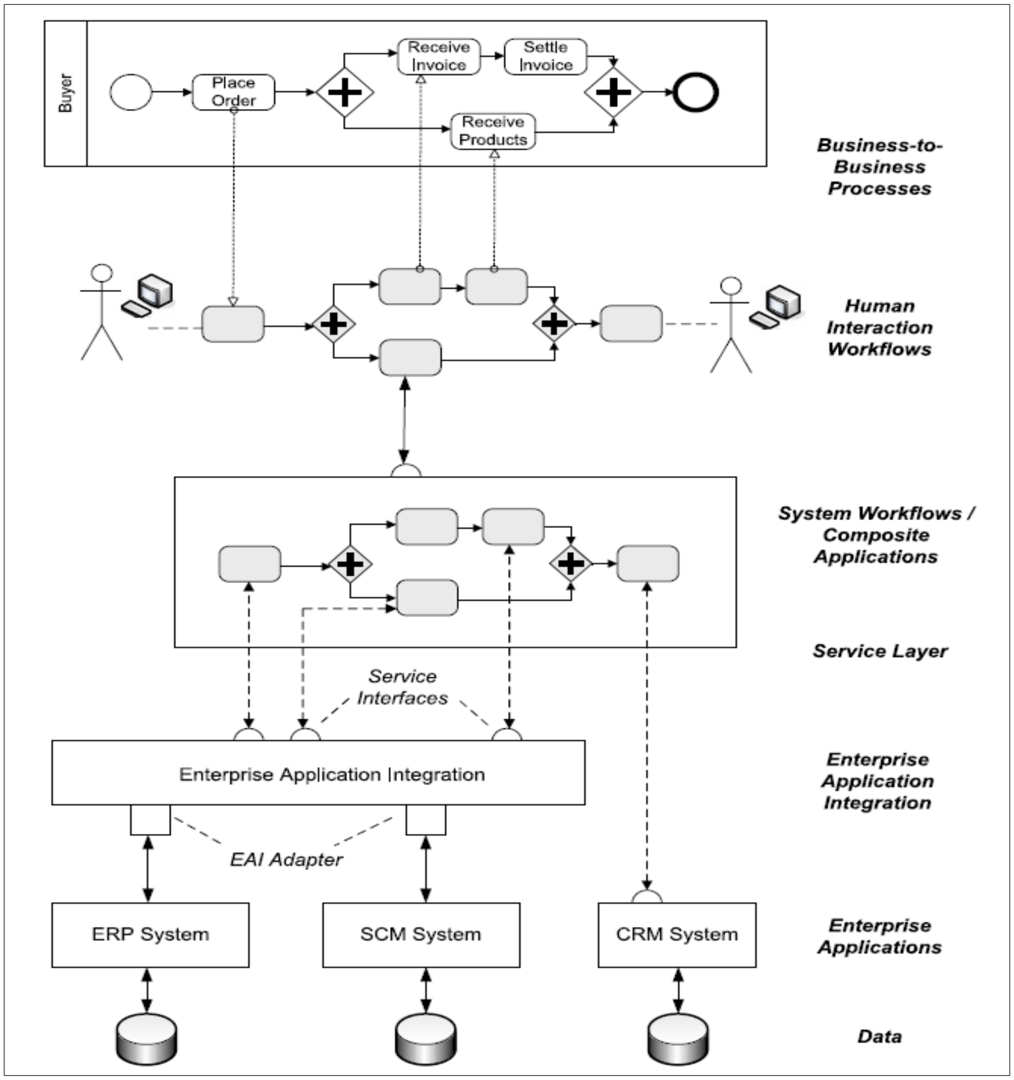

Buyer
Place Order
Receive Invoice
Settle Invoice
Receive Products
Business-to-Business Processes
Human Interaction Workflows
System Workflows / Composite Applications
Service Layer
Service Interfaces
Enterprise Application Integration
Enterprise Application Integration
EAI Adapter
ERP System
SCM System
CRM System
Enterprise Applications
Data

Figure 73 - BPMN-based workflow engine [10]

Currently, with the advent of DMN notation, decision engines, either as standalone engines or integrated with BPMN-based workflow engines, emerged as the instruments for making decisions automatically, based on a predefined business logic [2]. In the previous years, the most common approach for executing decisions was in a classical programming language [2]. However, since the emergence of DMN, decision engines were established as modern approaches that enhance the readability, traceability and transparency of decision-making processes. As a result, a new generation of BPMN and DMN-based workflow engines have entered the market, leading to consistent interaction between processes and decisions in a workflow enactment. Since both notations are executable in nature, for each notation element, there is a precise definition for how the engine should act [2]. In such approach, a decision engine is integrated within the workflow engine, and intermittently being reached in order to evaluate a decision and return the outcome back to the workflow engine (Figure 74).

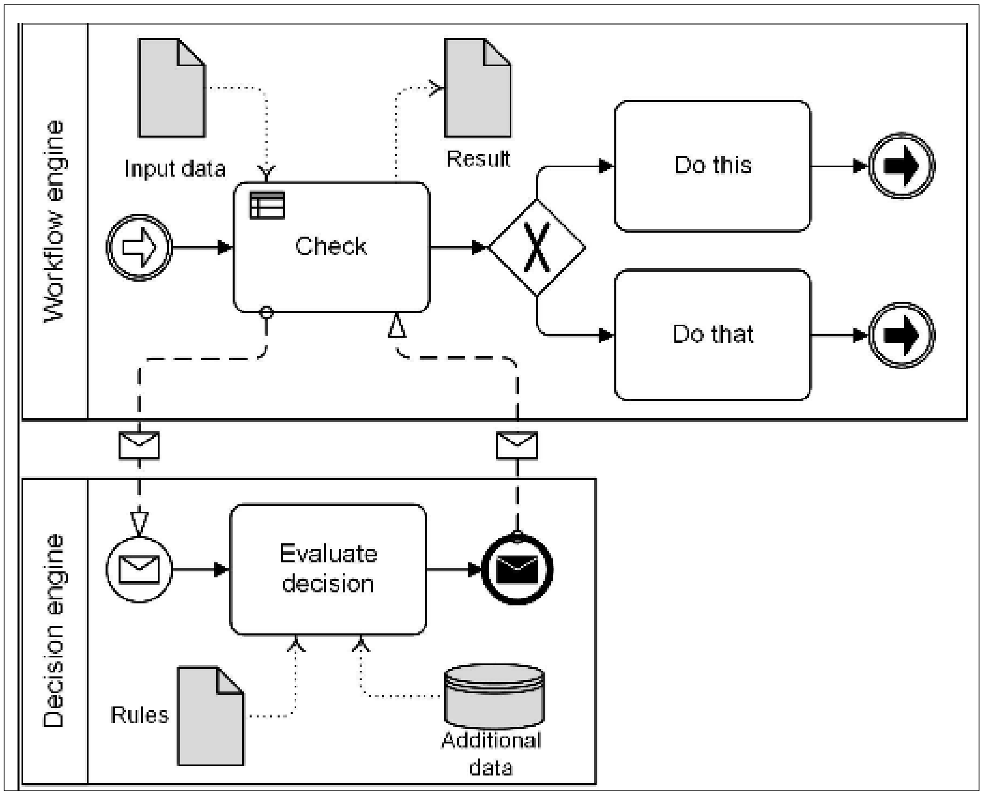


Figure 74 - Workflow and decision engine interaction [2]

5.3 Business Process Management Systems (BPMSs)

The practical merits of process and decision execution by a WfMS might seem oblivious to organizations if such initiatives are not accompanied by monitoring, analysis and optimization techniques. To this extent, Business Process Management Systems (BPMSs) have emerged as more sophisticated variants of WfMSs [8], supporting not only the execution of process models, but arising also as the technology response to support the entire BPM lifecycle [5], [13]. A Business Process Management System, often conveniently abbreviated as BPMS, supports an organization in a multidimensional manner. According to Dumas et al. [5], “The purpose of a BPMS is to coordinate an automated business process in such a way that all work is done at the right time by the right resource”. However, as the BPM lifecycle unfolds, a BPMS is not only involved in the process execution, but also in design, analysis and monitoring of business processes.

Currently, there is an abundance of BPMSs [15], since there is an increasing impetus from enterprises to incorporate BPMS platforms [13] in a way that their business operations are executed and managed by a piece of software [10]. BPMSs integrate several components, while mainly comprised of an execution engine, a process modeling component, a worklist handler and an administrative and process monitoring tool [13]. Serving as the backbone of such system, the execution engine is deemed to pull the strings of automatic process and decision execution. Generating process instances upon a process trigger and manipulating process-related data (i.e., process variables), work items are automatically delegated in runtime to process participants and software applications, facilitating the automatic process execution.

Interestingly, a modeling tool is another fundamental component of every BPMS. Being the means for rendering business processes in a visual way, misinterpretations and ambiguities, emerging from freeform textual descriptions, are well mitigated. Thus, conceptual process models can be converted to executable ones in the modeling tool's interface, before being deployed to the execution engine and determining the logic of the process [5].

Moreover, a worklist handler is another integral component of every BPMS. Forwarding work items to the worklists of process participants, a BPMS automatically delegates work units to human actors, while provides them all the required information, indispensable to their task completion. In this way, work items are rendered in corresponding electronic forms (Figure 75), where process participants are able to insert data and invoke signal completion to the process engine so as the process to be further executed [2], [24].

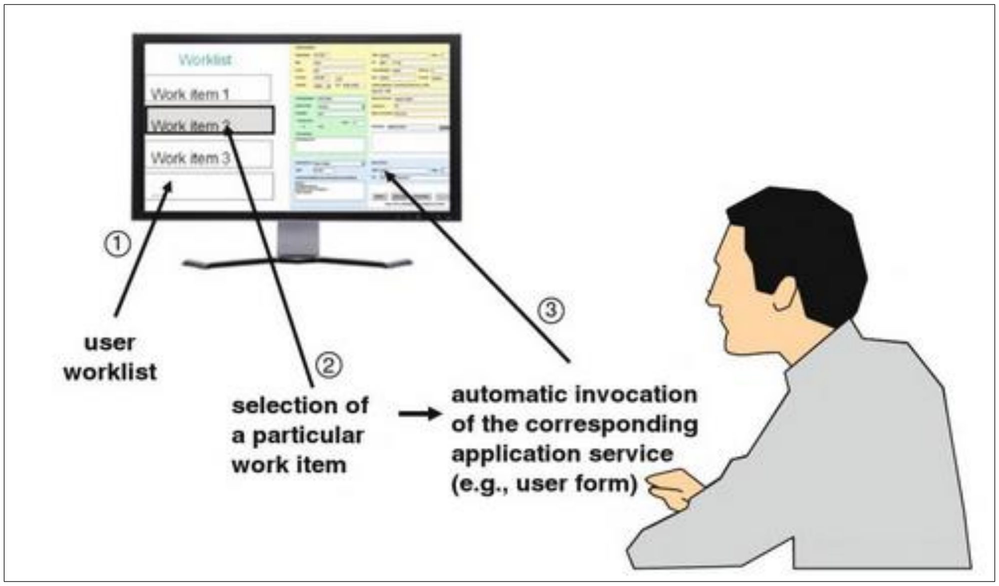

Worklist
Work item 1
Work item 2
Work item 3
①
user worklist
②
selection of a particular work item
③
automatic invocation of the corresponding application service (e.g., user form)

Figure 75 - Worklist handler [24]

Additionally, a BPMS is distributed with a default monitoring tool, mainly utilized by administrations in order to monitor process instances at a runtime and historical view. To this extent, the performance of an entire end-to-end process can be evaluated, as well as possible weak points (e.g., bottlenecks) can be identified. Allowing companies to derive much better intelligence about the performance of their key business operations [18], performance dashboards can be engendered leveraging the execution logs that have been thoroughly recorded by the execution engine upon the process elements execution [5].

Ultimately, it might be useful to involve external applications in the execution of a business process. Considering that the interest has shifted from application programming to application integration, the challenge is to interconnect and integrate different applications, rather than coding individual modules [4]. In this regard, in many cases the execution engine of a BPMS can delegate an activity to an external service, providing all the data required for the activity completion. On completion of the request, the service will return the outcome to the engine and signal that the work item is completed [5].

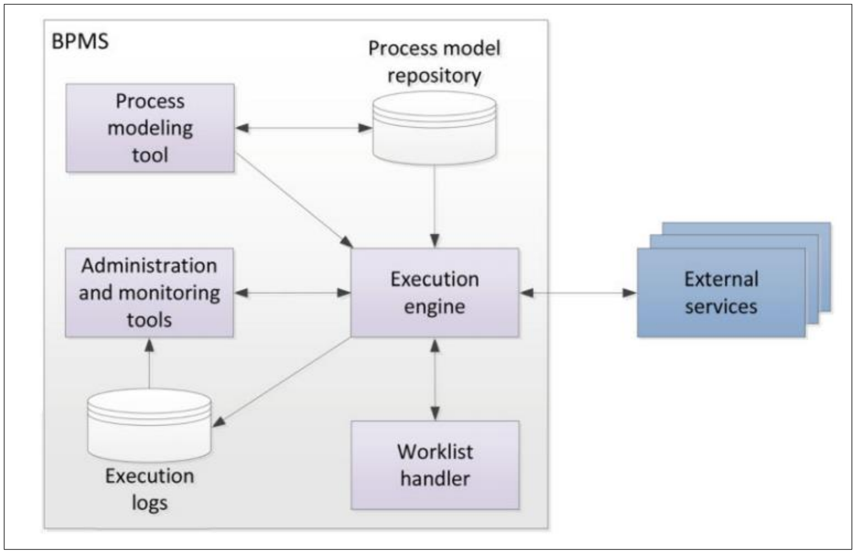


Figure 76 - BPMS Architecture [5]

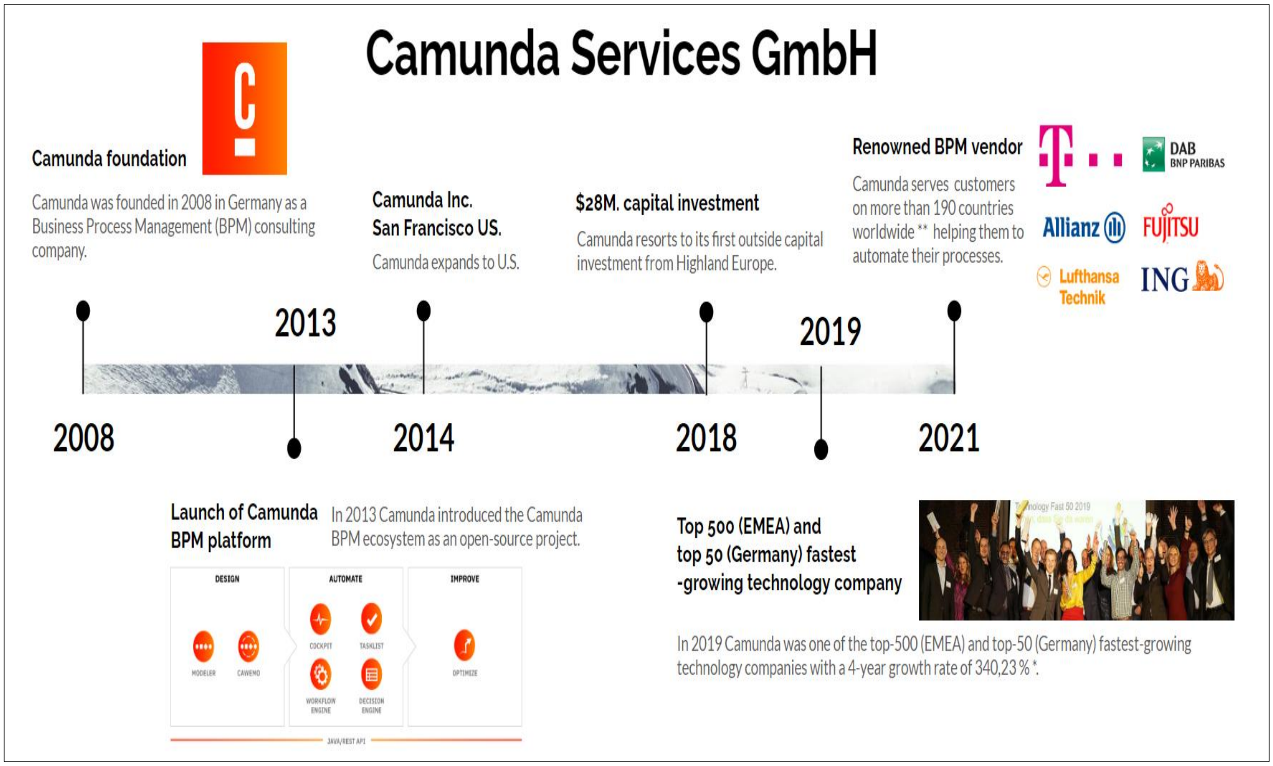
Altogether, there is a plethora of reasons, advocating the embrace of a BPMS in a contemporary organization. Considering that business processes have to be performed in a preordained way, business processes executed by the process engine of a BPMSs, leave less room for deviation from the desired process logic, which has been explicitly captured in the executable process model. Moreover, execution transparency is fostered, leveraging the power of event logs, formerly captured during the process execution by the BPMS engine. In this way, administrations and process participants can be aware of how the process is executed, as well as have an in-depth visibility into the business operations, facilitating refined decision-making and potential issue identification. Last, but not least, workload reduction is well attainable when a BPMS dispatches automatically work items, eradicating work handoffs. In such scenario, upon a task completion a signal is forwarded to the engine, prompting it to continue the model execution. Overall, process participants, working on user tasks, are provided with all the needed information for the task completion, reducing in such way their workload and the time-consuming gathering of all required information [5].

5.4 Camunda BPM

Having highlighted the concept of PAISs, the emergence of WfMSs, the evolution of contemporary BPMSs, as well as their architecture, the remaining of this chapter presents a renowned BPMS, namely the Camunda BPM platform, utilized in Chapter 6 for an end-to-end process deployment. Starting from outlining company's profile and its software's ecosystem, special attention is further given to its fundamental components and the way that they interact, as well as to the architectures that Camunda BPM platform can be utilized in.

5.4.1 Camunda Services GmbH

Camunda Services GmbH constitutes a software vendor, founded in 2008 in Berlin, Germany, by Jakob Freund and Bernd Rücker [83]. Initially established as a BPM consulting company, its main service was the consulting provision around the BPM spectrum and process-oriented initiatives. Five years later, in 2013, Camunda commenced offering its proprietary open-source BPM software product. Distributing since then both a community and a commercial platform, it switched from a consulting to a software vendor, focusing thereafter on 90% software to 10% consulting services. In 2014, Camunda expanded into U.S., establishing its offices in San Francisco. Supporting a more aggressive international expansion in 2018, it resorted to its first outside $28M. capital investment from Highland Europe [84]. In 2019, it was named by Deloitte [85], [86] as one of the top 500 fastest growing technology companies in Europe, the Middle East and Africa (EMEA), as well as one of the 50 fastest growing technology companies in Germany, with a four-year growth rate of 340.23%. Currently, Camunda constitutes a renowned BPM vendor with a customer base on more than 190 countries worldwide, assisting them to automate their business processes, reinvent their workflows and facilitate their digital transformation [83]. A timeline with its immensely important milestones until today, is rendered in Figure 77.



Camunda Services GmbH

Camunda foundation

Camunda was founded in 2008 in Germany as a Business Process Management (BPM) consulting company.

Camunda Inc. San Francisco US.

Camunda expands to U.S.

$28M. capital investment

Camunda resorts to its first outside capital investment from Highland Europe.

Renowned BPM vendor

Camunda serves customers on more than 190 countries worldwide ** helping them to automate their processes.

Launch of Camunda BPM platform

In 2013 Camunda introduced the Camunda BPM ecosystem as an open-source project.

Top 500 (EMEA) and top 50 (Germany) fastest -growing technology company

In 2019 Camunda was one of the top-500 (EMEA) and top-50 (Germany) fastest-growing technology companies with a 4-year growth rate of 340,23 % *.

Figure 77 - Camunda Services GmbH

5.4.2 Camunda BPM ecosystem

Camunda Services GmbH started offering its proprietary suite, namely the Camunda BPM platform, as an open-source project in 2013. Thereafter, distributing both an open-source and an enterprise product, Camunda BPM has been established as a renowned BPM platform. More specifically, Camunda BPM is a Java-based framework tailored to the usage of BPMN 2.0 notation for process execution and workflow automation. According to [42], Camunda BPM is a platform that exhibits BPMN 2.0 standard conformance, as a BPM suite able to execute BPMN process models with regard to its execution semantics. Along with the support of DMN notation, Camunda BPM provides a powerful ecosystem for the day-to-day engagement with business process and decision management. Its main constituent components are defined to Camunda Modeler, Cawemo, Workflow Engine, Camunda Cockpit, Camunda Tasklist, Camunda Admin, as well as Camunda Optimize, that collectively establish the Camunda BPM ecosystem as rendered in Figure 78.

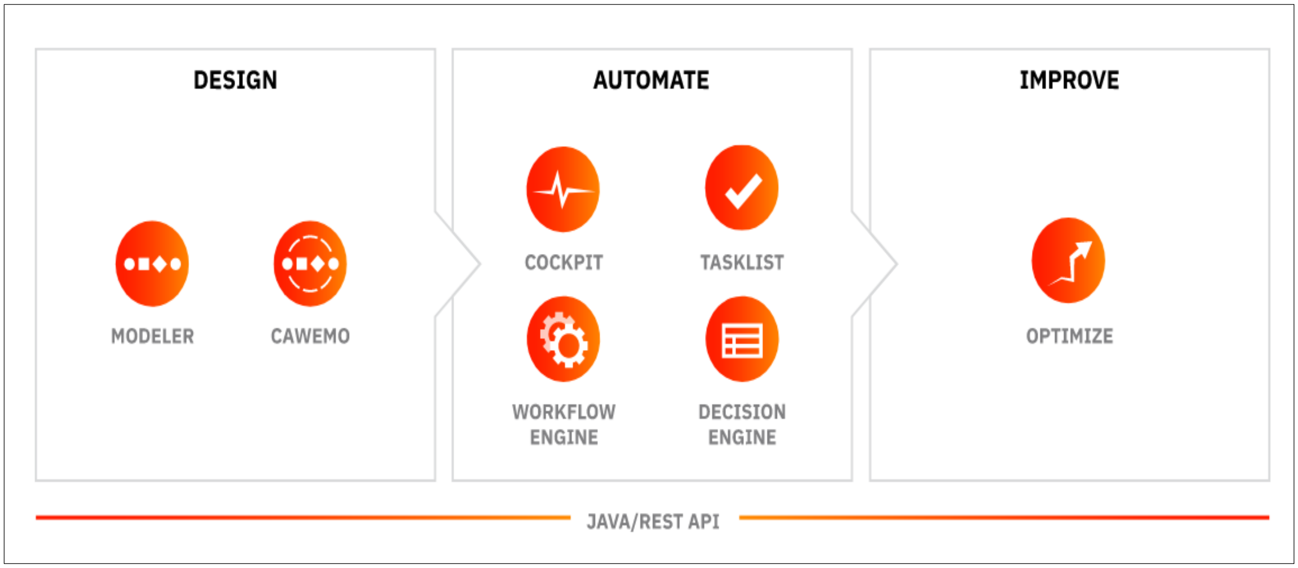
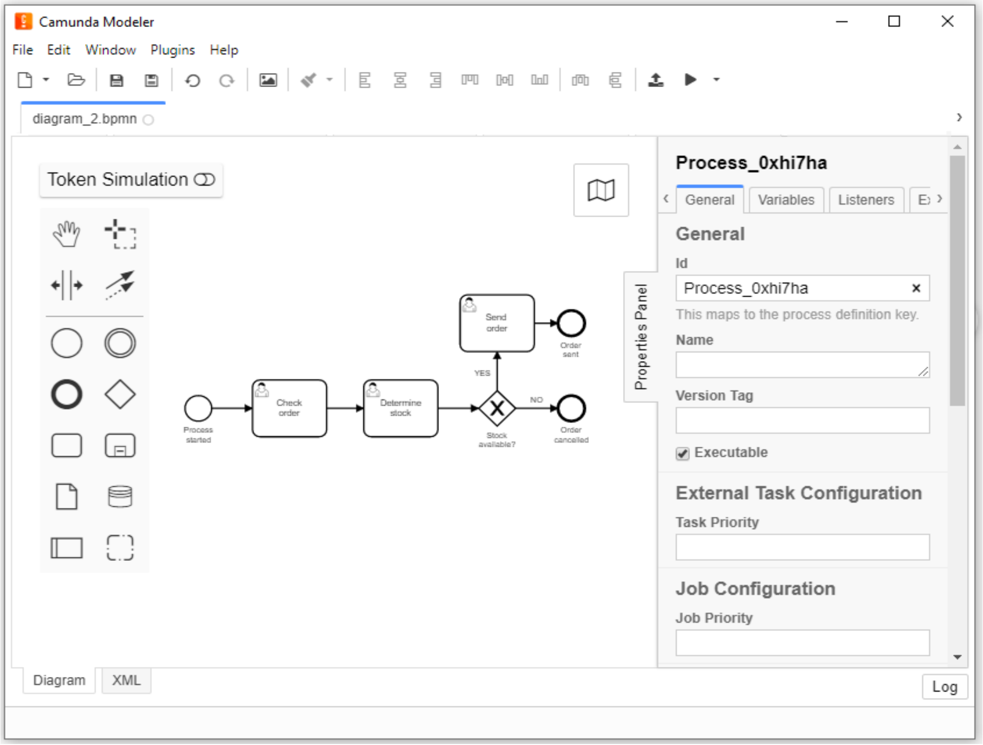


Figure 78 - Camunda BPM platform [83]

Serving as a contemporary BPMS, Camunda BPM supports the entire BPM lifecycle, rather than focusing exclusively on the process and decision enactment. Pertaining to the process designing phase, Camunda Modeler (Figure 79) and Cawemo (Figure 80) constitute the platform's modeling components, tailored to modeling processes and decisions according to the BPMN 2.0 and DMN 1.3 specifications. Interestingly, the former constitutes a desktop application, utilized not only for process and decision modeling, but also for the configuration of executable BPMN and DMN models, by enriching their elements with execution properties. The latter is a web-based application, launched either as a Service as a Software (SaaS) or on-Premise for the customers who desire to host it on their own IT infrastructure or in a private cloud. Tailored to the collaborative modeling, Cawemo allows multiple stakeholders to communicate on a real time and collaborate in modeling initiatives.



Process_0xhi7ha

General

Id: Process_0xhi7ha

This maps to the process definition key.

Name

Version Tag

☑ Executable

External Task Configuration

Task Priority

Job Configuration

Job Priority

Figure 79 - Camunda Modeler interface

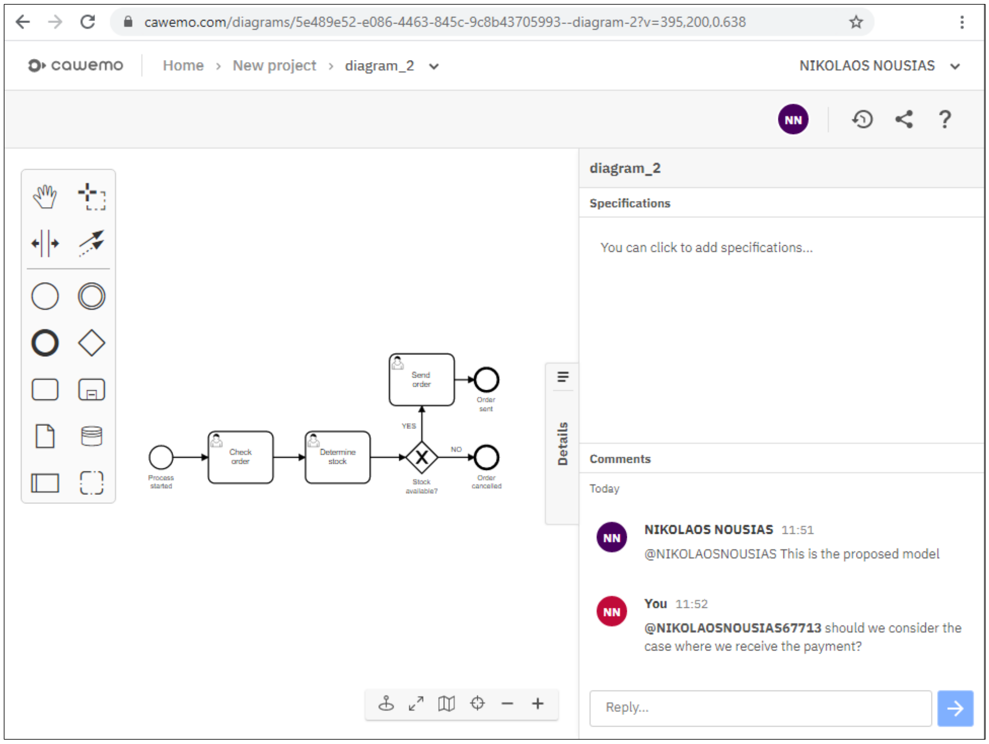


diagram_2

Specifications

You can click to add specifications...

Comments

Today

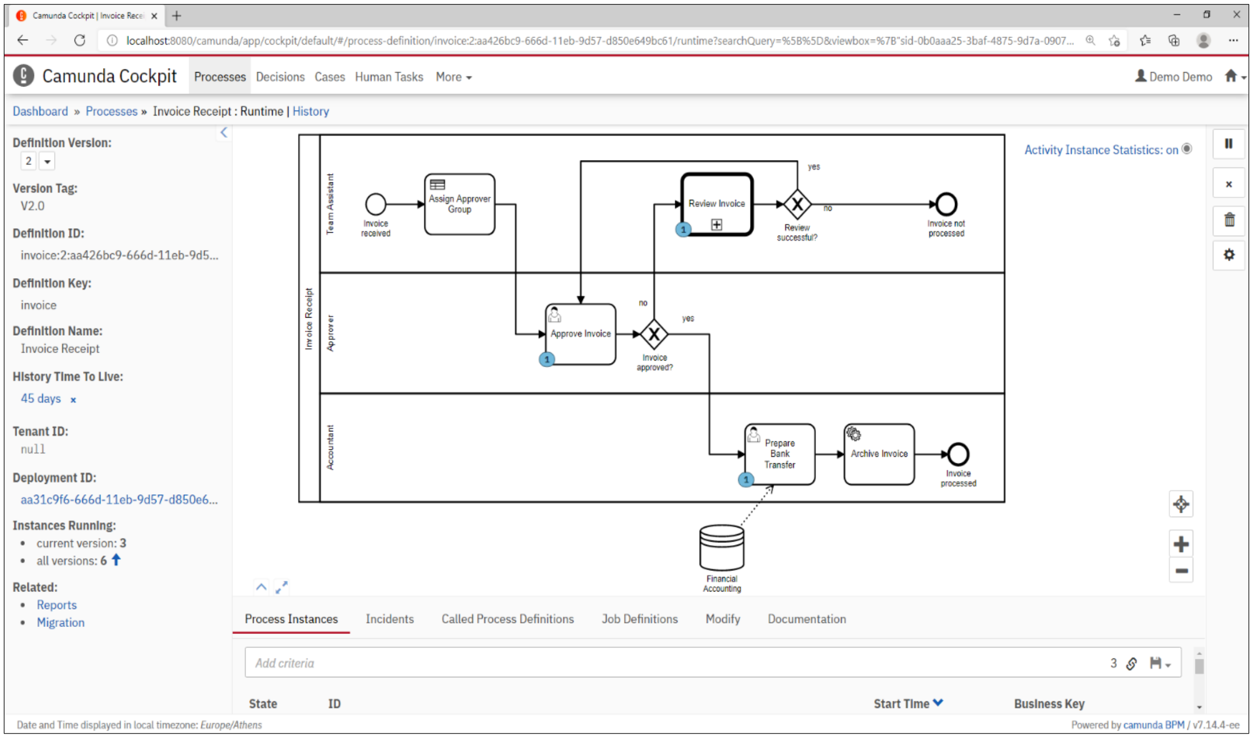
NN NIKOLAOS NOUSIAS 11:51
@NIKOLAOSNOUSIAS This is the proposed model

NN You 11:52
@NIKOLAOSNOUSIAS67713 should we consider the case where we receive the payment?

Reply...

Figure 80 - Cawemo interface

Undoubtedly, the focal component of Camunda BPM ecosystem is its workflow engine, serving as the backbone of the entire platform. Executing both BPMN 2.0 and DMN 1.3 models, Camunda's workflow engine integrates a decision engine in its background. Interestingly, the workflow engine interprets the execution semantics of BPMN and DMN models, indicating conformance to their standards and providing a run-time environment for process and decision enactment. Moreover, Camunda BPM ecosystem is distributed with a list of predefined default webapplications, which leverage the built-in REST Application Programming Interface (API) to communicate with its core engine. Concretely, Camunda Cockpit (Figure 81) constitutes a web-based application that facilitates the monitoring and the analysis of process executions in a runt-time and a historical view. Camunda Tasklist (Figure 82), serving as the product's default worklist handler, coordinates the human workflow management, by facilitating process participants to inspect and complete their workflow tasks. Additionally, Camunda Admin (Figure 83) enhances user management functionalities, while it grants permissions and authorizations to process stakeholders, who will be involved in process lifecycle and will be engaged with the Camunda BPM ecosystem as the process unfolds.



localhost:8080/camunda/app/cockpit/default/#/process-definition/invoice:2:aa426bc9-666d-11eb-9d57-d850e649bc61/runtime?searchQuery=%5B%5D&viewbox=%7B"sid-0b0aaa25-3baf-4875-9d7a-0907...

Dashboard » Processes » Invoice Receipt : Runtime | History

Definition Version:
2

Version Tag:
V2.0

Definition ID:
invoice:2:aa426bc9-666d-11eb-9d5...

Definition Key:
invoice

Definition Name:
Invoice Receipt

History Time To Live:
45 days

Tenant ID:
null

Deployment ID:
aa31c9f6-666d-11eb-9d57-d850e6...

Instances Running:
- current version: 3
- all versions: 6

Related:
- Reports
- Migration

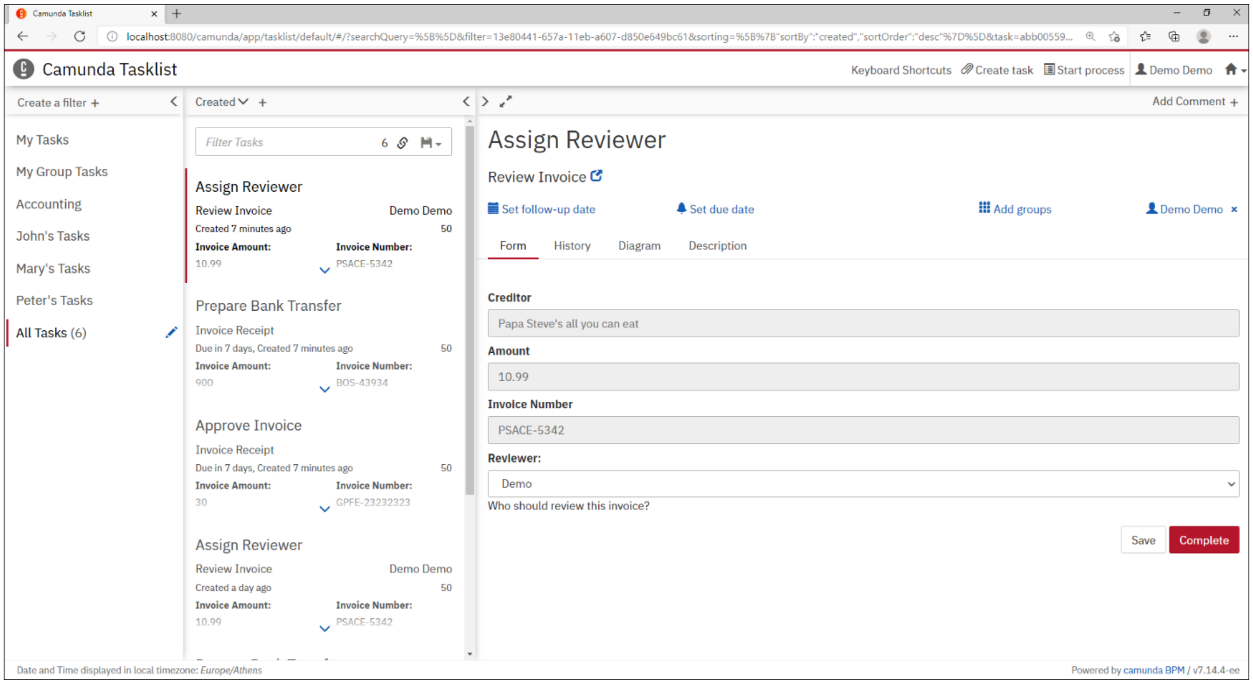
Activity Instance Statistics: on

Add criteria 3

Date and Time displayed in local timezone: Europe/Athens

Powered by camunda BPM / v7.14.4-ee

Figure 81 - Camunda Cockpit interface



Assign Reviewer
Review Invoice — Demo Demo
Created 7 minutes ago — 50
Invoice Amount: 10.99 — **Invoice Number:** PSACE-5342

Prepare Bank Transfer
Invoice Receipt
Due in 7 days, Created 7 minutes ago — 50
Invoice Amount: 900 — **Invoice Number:** BOS-43934

Approve Invoice
Invoice Receipt
Due in 7 days, Created 7 minutes ago — 50
Invoice Amount: 30 — **Invoice Number:** GPFE-23232323

Assign Reviewer
Review Invoice — Demo Demo
Created a day ago — 50
Invoice Amount: 10.99 — **Invoice Number:** PSACE-5342

Assign Reviewer

Review Invoice

Creditor
Papa Steve's all you can eat

Amount
10.99

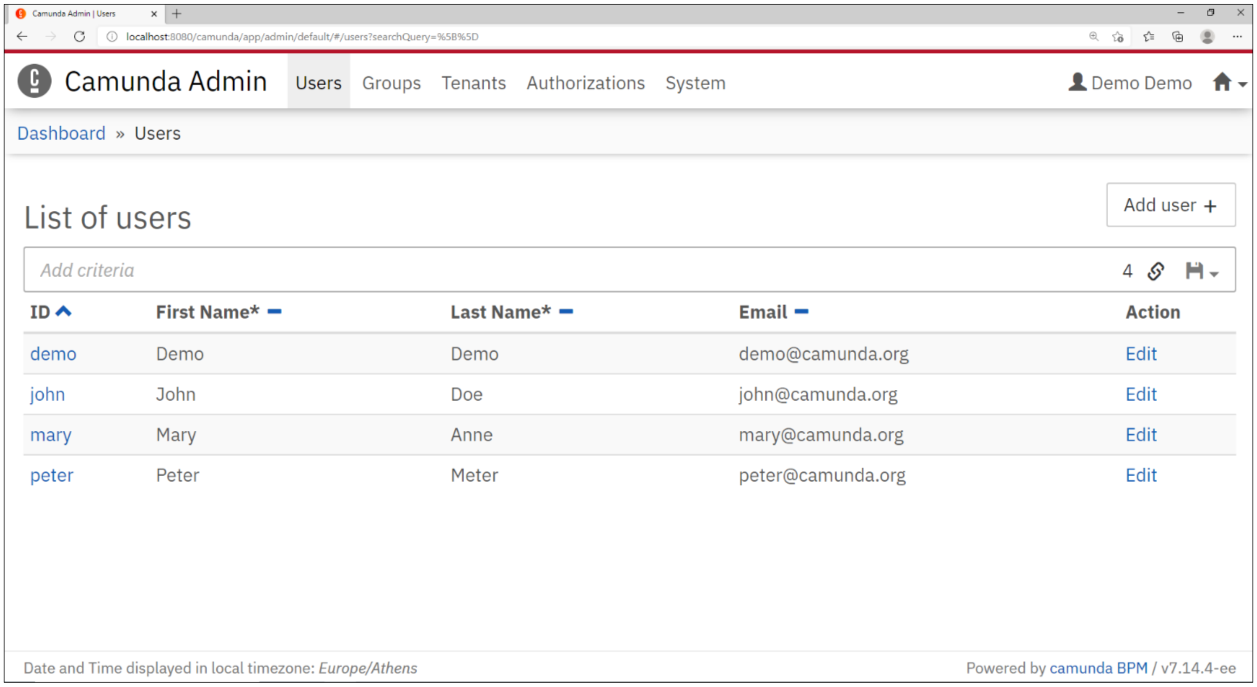
Invoice Number
PSACE-5342

Reviewer:
Demo
Who should review this invoice?

Date and Time displayed in local timezone: *Europe/Athens*

Powered by camunda BPM / v7.14.4-ee

Figure 82 - Camunda Tasklist interface



List of users

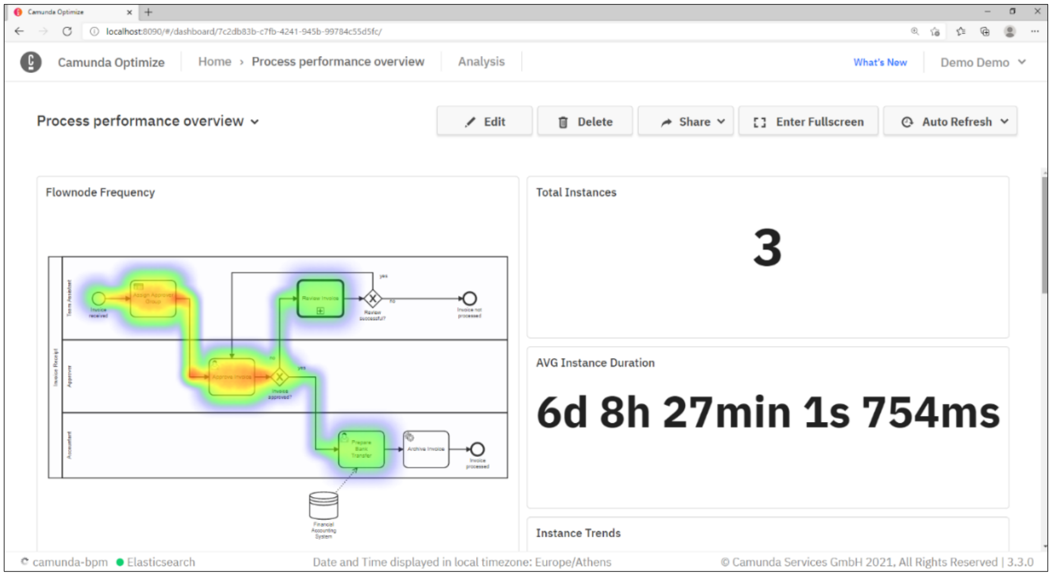
ID	First Name*	Last Name*	Email	Action
demo	Demo	Demo	demo@camunda.org	Edit
john	John	Doe	john@camunda.org	Edit
mary	Mary	Anne	mary@camunda.org	Edit
peter	Peter	Meter	peter@camunda.org	Edit

Date and Time displayed in local timezone: *Europe/Athens*

Powered by camunda BPM / v7.14.4-ee

Figure 83 - Camunda Admin interface

Distributing a commercial edition of Camunda BPM platform, enterprise customers have additional features at their disposal. One such, Camunda Optimize (Figure 84) is a webapplication, available solely for the enterprise customers, providing a continuous monitoring and explicit transparency into the deployed processes and decisions. In this regard, powerful dashboards can be generated utilizing performance metrics and data visualizations, such as heatmaps. Potential bottlenecks, thus, can be identified, as well as the means for entire end-to-end process optimization can be established.



Total Instances

3

AVG Instance Duration

6d 8h 27min 1s 754ms

Instance Trends

Figure 84 - Camunda Optimize interface

Overall, the aforementioned Camunda's BPM components collectively establish the Camunda BPM ecosystem. Facilitating and supporting all BPM phases, Camunda BPM platform refines process and decision management as the entire BPM lifecycle unfolds (Figure 85).

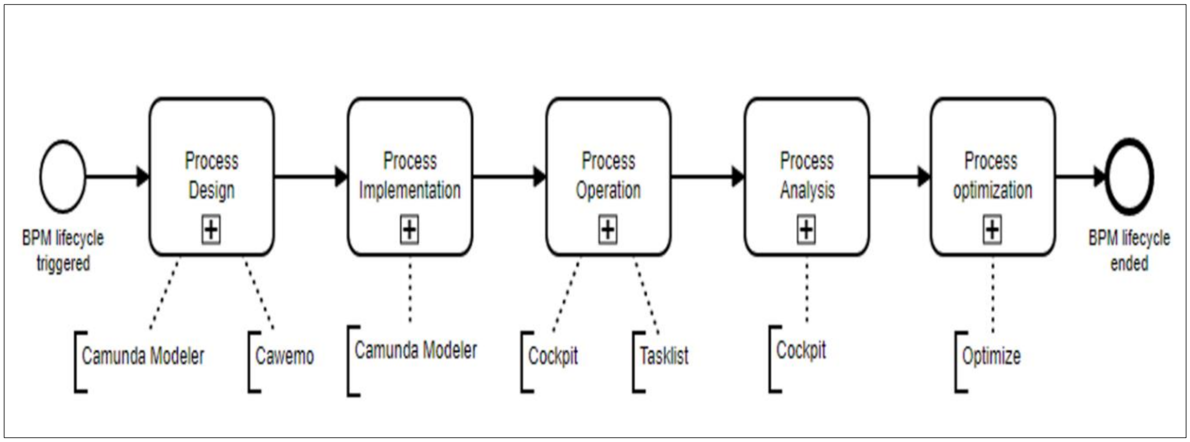


Figure 85 - BPM lifecycle with the Camunda BPM platform

5.4.3 Camunda BPM components interaction

Having highlighted the ecosystem's components, the underlying section provides a more in-depth perspective of the way that its multiple constituent parts communicate with each other in the platform's background, forming a robust ecosystem for process and decision management.

Camunda Engine, undoubtedly, comes in forefront as the foundational pillar of the entire ecosystem. Integrating both a workflow engine and a decision engine, it constitutes a Java library that facilitates the automatic execution of entire end-to-end processes on the basis of predefined process and decision models, captured extensively in a BPMN 2.0 and DMN 1.3 format. For this purpose, a BPMN 2.0 and a DMN 1.3 XML parser is running in its background, interpreting BPMN and DMN symbols and translating them to Java Objects.

Interestingly, previously created conceptual process models are able to be converted to executable ones inside the Camunda Modeler interface. Establishing a file repository with executable BPMN 2.0 and DMN 1.3 models, processes and decisions can be deployed to the Camunda engine and, ideally, be automatically executed.

Furthermore, leveraging the built-in Camunda REST API, as well as a Java API, default pre-installed webapplications, namely Camunda Cockpit, Tasklist and Admin, can communicate and interact with the core workflow engine. As such, conducting REST API calls in their background, they are able to be effortlessly integrated with the workflow engine, providing transparency and insight into the formerly deployed processes, let alone consistency in the entire platform. Moreover, consuming the Camunda REST API, custom applications can be developed and interact with the workflow engine, serving as complementary components of the entire ecosystem, that fulfill personalized needs.

Ultimately, the core process engine is connected in the background with an in-memory H2 database, facilitating platform's persistence. Hence, runtime and historical data related to process and decision execution, such as deployed process and decisions definitions, process and decisions instances, as well as process variables, to exemplify some, are concretely stored within the database tables and can be retrieved at any time by SQL statements. An overview of Camunda's BPM components interaction is rendered in Figure 86.

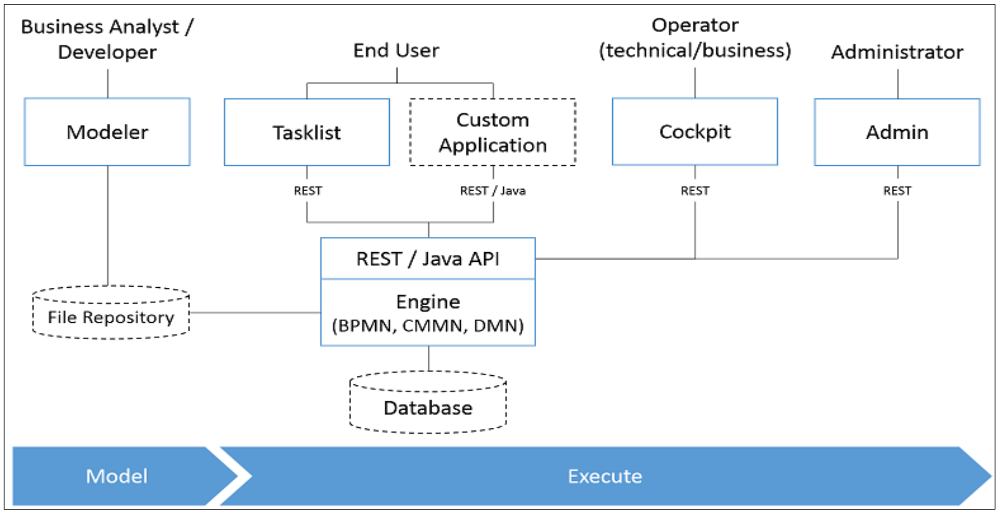


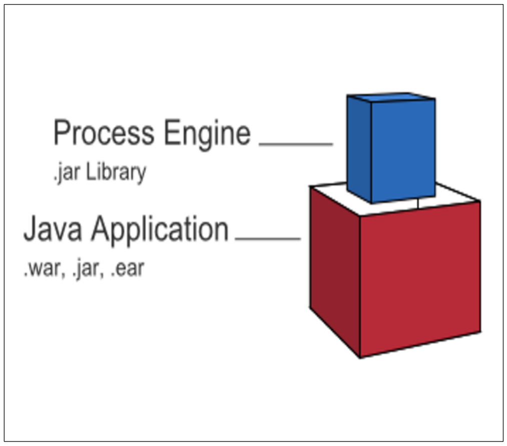
Figure 86 - Camunda BPM architecture [75]

5.4.4 Camunda BPM platform architecture

Being a lightweight Java-based framework, Camunda BPM platform can be deployed in different scenarios [75]. According to Camunda's BPM documentation [75], Camunda BPM can be implemented in an Embedded Process Engine architecture, in a Standalone (Remote) Process Engine Server architecture, as well as in a Shared, Container-Manage Process Engine architecture. The latter is selected for the premise of this thesis, as the optimal architecture for an end-to-end process deployment in Chapter 6.

First, in an Embedded Process Engine architecture (Figure 87), the process engine can be embedded as an external application library to a custom java application. In this regard, once the application lifecycle unfolds, the process engine can effortlessly be started or stopped on the basis of the desired underlying application's functionality. Alternatively, in a Standalone (Remote) Process Engine Server architecture (Figure 88), the process engine can be started remotely. Different applications can interact with the process engine, establishing a remote communication channel with its built-in Camunda REST API.

However, for the premise of this thesis, a Shared, Container-Manage Process Engine architecture (Figure 89) is well adopted. Interestingly, the core process engine is wrapped inside an application server, like Apache Tomcat or Wildfly. All applications deployed in the application server share the same process engine, which is initialized once the application server (i.e., runtime container) is started. The process engine, hence, is provided as a container service, utilized by all the deployed applications.



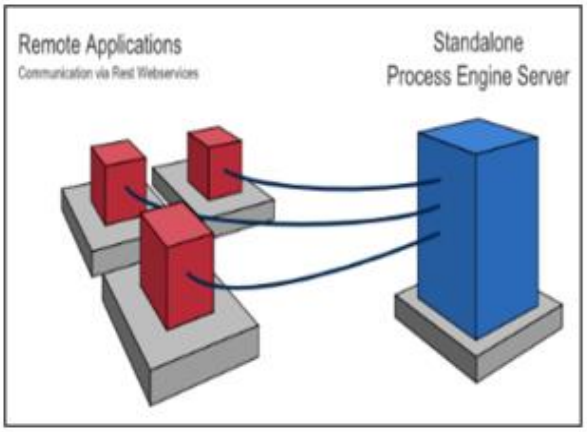


Figure 87 - Embedded Process Engine architecture [75]

Figure 88 - Standalone (Remote) Process Engine Server architecture [75]

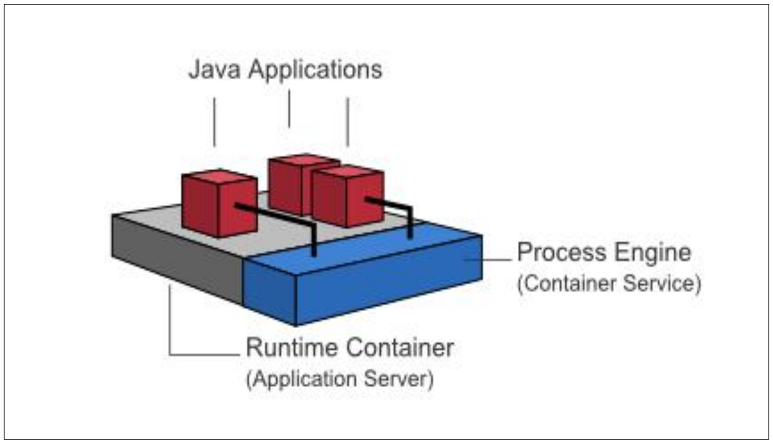


Figure 89 - Shared process engine architecture [75]

In such approach, a plethora of process applications might be deployed to the application server (Figure 90). Each application, packaged as a Web Application Resource (.war) file, encompasses all the files necessary for its deployment and execution, such as executable BPMN and DMN models, Java Delegates, Java Classes, UI Forms, etc. Interestingly, a connection link between applications might be established in the background, while each of the deployed applications shares the same process engine, which is initialized once the application server (i.e., runtime container) is started. The core process engine, hence, might execute concurrently different deployed processes, providing extensive scalability and agility in process and decision enactment.

In the spirit of process transparency and visibility provided by a workflow engine, Camunda BPM platform is enriched with a list of default webapplications that provide a thorough insight into the runtime and historical process and decision executions. Concretely, accessing custom and default webapplications, namely Cockpit, Tasklist and Admin, through a client browser and an established internet connection, information gathered by the workflow engine can be transferred and displayed in their interface in a human readable way. Leveraging the built-in Camunda REST-API, they are able to receive and send information to the workflow engine, utilizing each time the appropriate methods and messaging protocols.

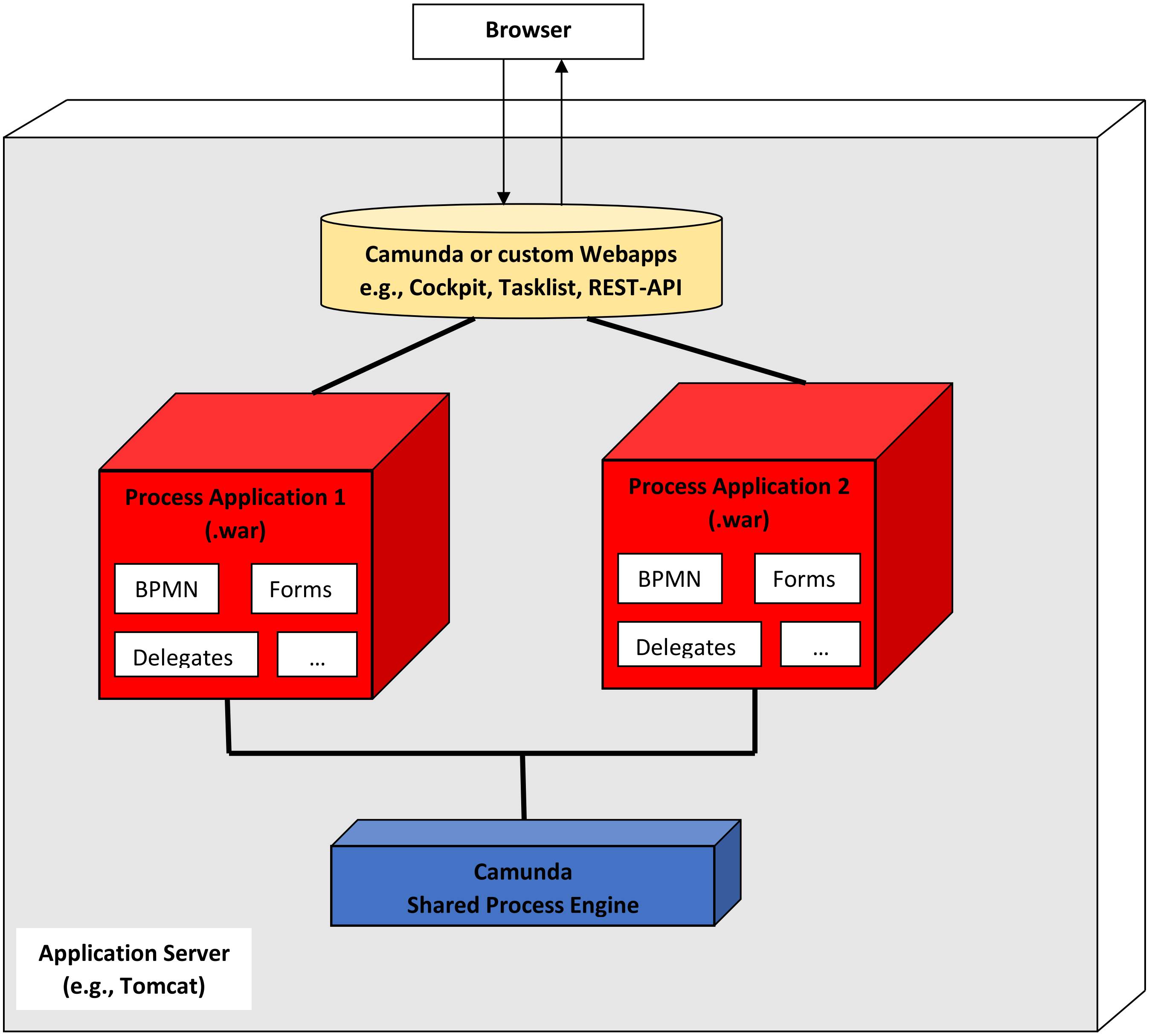


Figure 90 - Shared process engine architecture

5.5 Summary

This chapter delved into the concepts of PAISs and their specific categories of WfMSs and BPMSs. A PAIS, being aware of the context that is involved in, is considered to be a system that supports entire business processes, rather than executing individual tasks. Contemporary ERPs and CRMs, characterized under the term of PAISs, drive business operations on the basis of predefined process models. A specific category of PAISs is deemed to be the WfMSs. In such ecosystems, the workflow engine is considered to constitute their focal pillar, conducting and orchestrating process and decision enactments. With the advent of BPMN 2.0 and later with the emergence of DMN notation, workflow engines, purported to conform to BPMN and DMN specifications, came in frontline for the process and decision execution according to these standards. However, considering that organizations are focused not only on implementation initiatives, but also on modeling, analysis and redesign of processes and decisions, BPMSs emerged as enhanced variants of WfMSs, that support the entire BPM lifecycle. Concretely, a BPMS is comprised of monitoring and administrative tools, a modeling tool, a worklist handler, while its execution engine constitutes its backbone, by integrating the aforementioned tools and enacting business operations. One such, Camunda BPM is established as a renowned BPMS, that assists organizations to execute their processes and decisions, automate their operations and reinvent their workflows. In the next Chapter, Camunda BPM is utilized in a Shared, Container-Manage Process Engine architecture, serving as the fundamental instrument for fulfilling the practical scope of this thesis.

CHAPTER 6: Business process automation walkthrough: The loan application-to-approval process

This chapter presents an end-to-end deployment of a business process and its gradual transformation to an automated one, fulfilling the practical scope of this thesis. Establishing the aforementioned theoretical concepts into practice, the aim is to highlight how process and decision automation can be applicable, leveraging the execution capabilities of BPMN and DMN notations, which are directly executed by the workflow engine of a BPMS. For the premise of this thesis, a loan application-to-approval process at a fictitious bank, named “UoMBank”, is introduced, before being deployed into the Camunda BPM ecosystem and automatically executed by its workflow engine.

The chapter starts with an introduction to the nature of the application-to-approval processes and the current impetus towards their automation in the banking industry. Delving into a loan application-to-approval process at the fictitious “UoMBank”, a textual description of the end-to-end process is introduced, aiming to highlight not only its process and its decision logic, but also the resources, the systems and the strategies that are involved in throughout the entire process lifecycle. Subsequently, the chapter presents a BPMN representation of the process, rendering the entire end-to-end process in a graphical, let alone intuitive, manner. Considering that the decision logic of BPMN models is frequently convoluted with their process logic, special attention is further given on the identification of decision points within the underlying process and their externalization to separate DMN models. Next, in the context of workflow automation, the chapter presents the establishment of the execution semantics of the aforementioned conceptual models, as well as the ways of implementing the BPMN service and script tasks, as fundamental BPMN elements for enacting process activities without human intervention. Thereafter, the chapter introduces the entire demo architecture, mainly comprised of a Website portal (i.e., front end) and the Camunda BPM engine (i.e., back end), while shedding light on their communication with REST API calls. Last, an execution scenario is rendered illustrating the entire process both from an applicant’s and the “UoMBank’s” perspective, before the expected benefits of workflow automation are presented. In Figure 91, the demo development process is graphically illustrated, shedding light into the gradual transformation of the underlying process to an automated one.

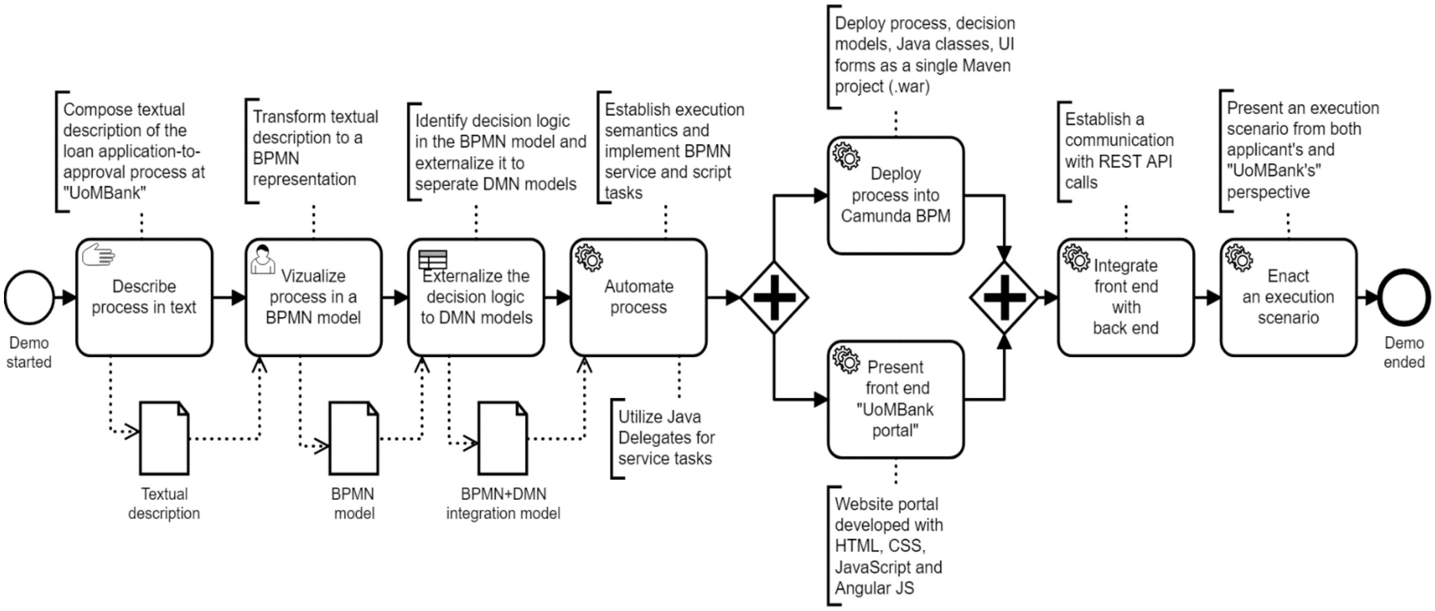

Compose textual description of the loan application-to-approval process at "UoMBank"
Transform textual description to a BPMN representation
Identify decision logic in the BPMN model and externalize it to seperate DMN models
Establish execution semantics and implement BPMN service and script tasks
Deploy process, decision models, Java classes, UI forms as a single Maven project (.war)
Establish a communication with REST API calls
Present an execution scenario from both applicant's and "UoMBank's" perspective
Demo started
Describe process in text
Vizualize process in a BPMN model
Externalize the decision logic to DMN models
Automate process
Deploy process into Camunda BPM
Present front end "UoMBank portal"
Integrate front end with back end
Enact an execution scenario
Demo ended
Textual description
BPMN model
BPMN+DMN integration model
Utilize Java Delegates for service tasks
Website portal developed with HTML, CSS, JavaScript and Angular JS

Figure 91 - Demo deployment process

6.1 Banking and financial processes: the challenge for automation

Application-to-approval processes prevail in today’s financial institutions, government agencies and administration departments typically with a bureaucratic form. Notably, an application-to-approval process is defined as a business process where a plethora of activities, events and decisions are performed, before a privilege being granted or denied [5]. Especially, in the banking industry, application-to-approval processes are highly influenced by strict regulatory compliance strategies, that delineate specific guidelines as the process lifecycle unfolds [34]. Considering that a vast majority of these activities are performed manually by human actors, the process cycle time, the error rates and the overall customer experience are adversely affected.

Traditionally, banking industry was recognized as a conservative industry, where the stable business environment has made it very resistant to change [87]. However, today’s highly competitive marketplace and the volatile business environment has put pressure on the banking industry to get digitalized and accelerate its operations [87]. As a result, new innovative business processes are introduced [88], forcing banks to transform their traditional IT “back office” role to a tech-savvy mind-set [87].

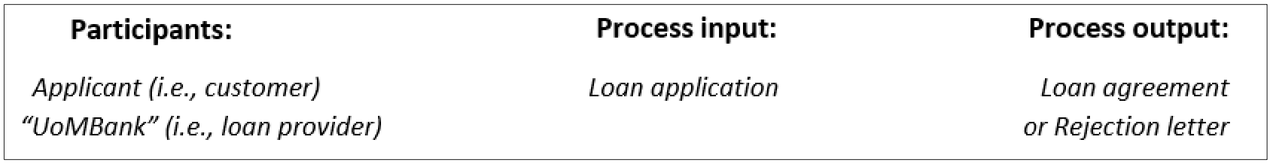
In this regard, process automation at the banking industry emerges as the art and science of automating repetitive work within bureaucratic processes. Executing automatically activities that would otherwise be enacted by human actors, banks can increase their agility and productivity, let alone survive in today’s market, where everything needs to be executed faster and with greater flexibility [65]. Considering that application-to-approval processes in banks are highly repetitive, standardized and rich in information, this makes them as well-suited candidates for workflow automation. In such a case, workflow automation can improve their visibility, traceability, agility and quality, refining at the same time the customer experience [2]. In the following section, a textual description of a loan application-to-approval process at the fictitious “UoMBank” is introduced, capturing meticulously the entire end-to-end process and decision logic.

6.2 The “UoMBank” loan application-to-approval process

For the premise of this thesis, a fictitious loan application-to-approval process at an imaginary bank, namely the “UoMBank”, was developed with the aim to reflect real end-to-end loan application processes that take place in contemporary banking institutions. Thus, the aim was to highlight various aspects that emerge around an end-to-end loan application-to-approval process, such as the process logic (i.e., What has to be enacted next?), business decisions (e.g., Is an applicant eligible to get a loan granted?), human resources involved in the evaluation of the application (i.e., different resources are involved based on the loan type and the application stage), systems that support the entire process (i.e., How different systems can be integrated?), evaluation strategies (e.g., Credit Score calculation, Credit Risk assessment, applicant’s probability of default, bank’s expected losses, etc.) and documents that might be circulated within the process and generated as the process lifecycle unfolds (e.g., Applicant’s documents, Loan Agreement, Rejection letter, etc.). In the following lines, the underlying process is introduced in a free-form textual description, before being transformed to a BPMN model in the following section.

Process description:

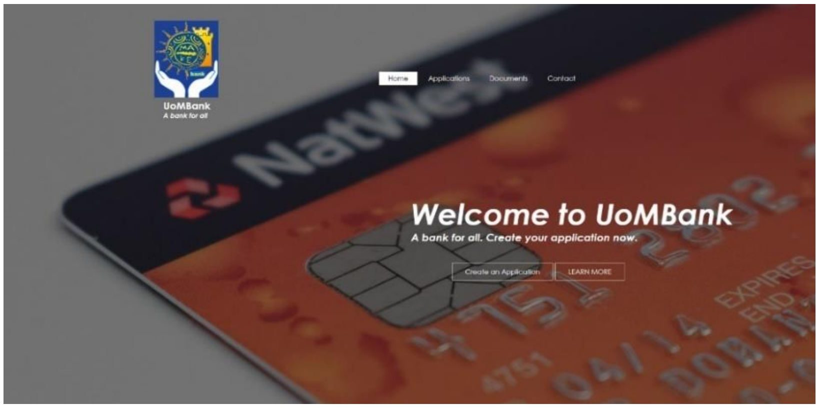
The underlying loan application-to-approval process takes place between the loan applicant (i.e., customer) and the loan provider, namely the fictitious “UoMBank”. The process is triggered, once a customer submits a loan application form, utilizing the “UoMBank’s” Website Portal, while it is completed once a loan agreement or a rejection letter is forwarded to the applicant. The high-level process information is displayed in Figure 92, before delving into the process details in the upcoming lines.



Participants:	Process input:	Process output:
Applicant (i.e., customer) *"UoMBank" (i.e., loan provider)*	*Loan application*	*Loan agreement* *or Rejection letter*

Figure 92 - High-level process information

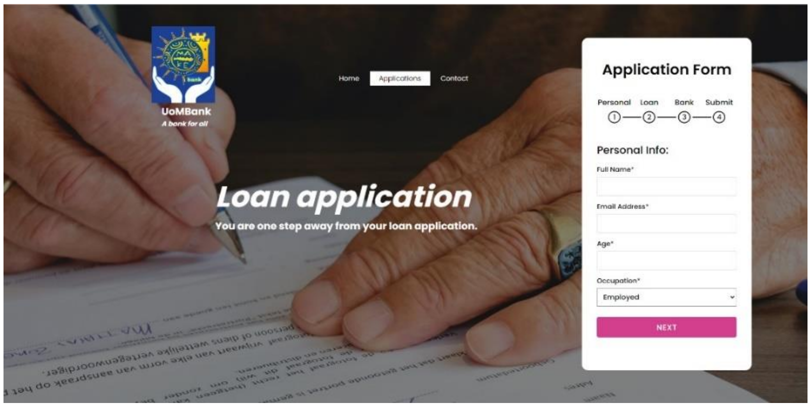
With the purpose of getting a loan granted, applicants enter the "UoMBank's" website portal (Figure 93), where they can create their applications for personal, auto, mortgage and business loans, provided by "UoMBank". Filling an application form (Figure 94) with various kind of information, such as their personal information (Full name, Email Address, Age, Occupation), loan information (Requested loan type, Loan Amount, Loan Term), financial information (VAT number, Monthly income) and bank-related information (Old customer at "UoMBank", Customer ID), they are able to forward their loan application to "UoMBank", upon submitting their application form.



Welcome to UoMBank

A bank for all. Create your application now.

Figure 93 - "UoMBank" Website Portal



Loan application

You are one step away from your loan application.

Application Form

Personal Info:

Full Name*

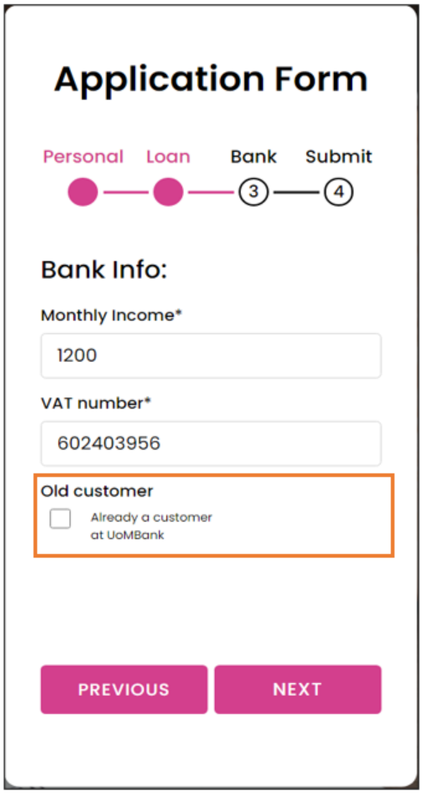
Email Address*

Age*

Occupation*
Employed

Figure 94 - "UoMBank" Website Portal: Loan application Form

Once the loan application is received from the "UoMBank's" perspective, the underlying loan application-to-approval process is triggered. Initially, a confirmation email is automatically forwarded to the applicant, denoting that the application has been successfully received. Starting to assess the input data received by the application form, "UoMBank" examines if the applicant is already a bank's existing customer. For this purpose, an automatic check, running in the background, is conducted in order to investigate if the applicant has selected the "Old customer" checkbox, during their application form submission (Figure 95).



Bank Info:

Monthly Income*

1200

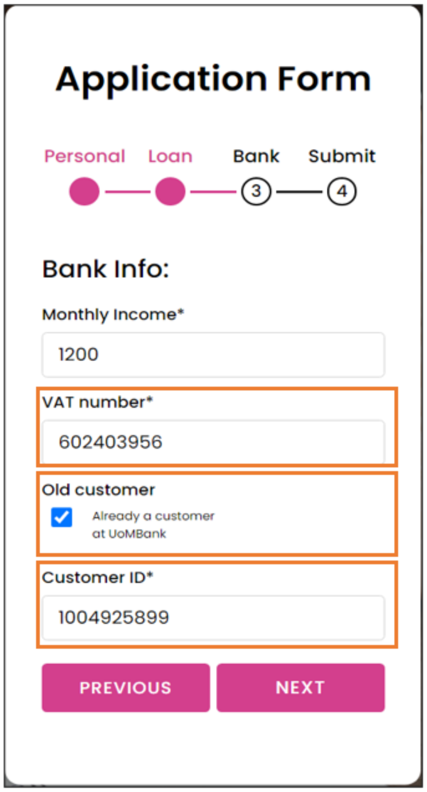
VAT number*

602403956

Old customer

☐ Already a customer at UoMBank

Figure 95 - Loan application form (1)



Bank Info:

Monthly Income*

1200

VAT number*

602403956

Old customer

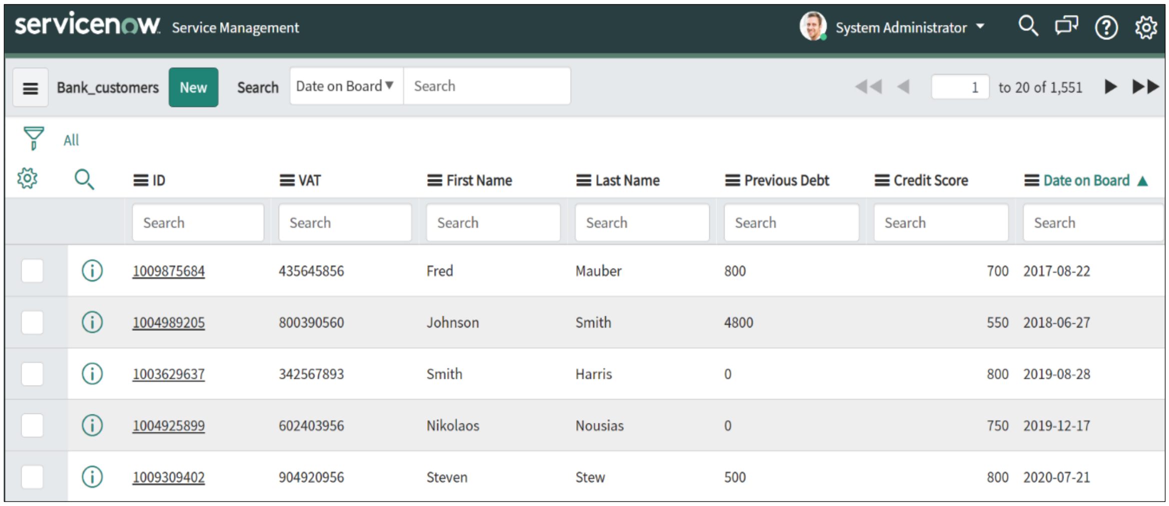
☑ Already a customer at UoMBank

Customer ID*

1004925899

Figure 96 - Loan application form (2)

Thereby, retrieving applicants' VAT number and their Customer ID (Figure 96), a REST API call to a bank's third system, namely the "ServiceNow" system (Figure 97), is automatically conducted for existing customers in order to retrieve their previously stored debt amount. In such a case, a relative response is returned by the "ServiceNow" system, indicating either the customer's existence and their previous amount of debt or denoting that the applicant has yet to be recorded as an "UoMBank's" customer. Considering that such integration with the third system is executed automatically with programming code, in case of an error occurrence during the REST API call, a user is notified to perform the check manually and denote the applicant as a new or an already existing customer.



ID	VAT	First Name	Last Name	Previous Debt	Credit Score	Date on Board ▲
1009875684	435645856	Fred	Mauber	800	700	2017-08-22
1004989205	800390560	Johnson	Smith	4800	550	2018-06-27
1003629637	342567893	Smith	Harris	0	800	2019-08-28
1004925899	602403956	Nikolaos	Nousias	0	750	2019-12-17
1009309402	904920956	Steven	Stew	500	800	2020-07-21

Figure 97 - "UoMBank's" third system: The "ServiceNow" system

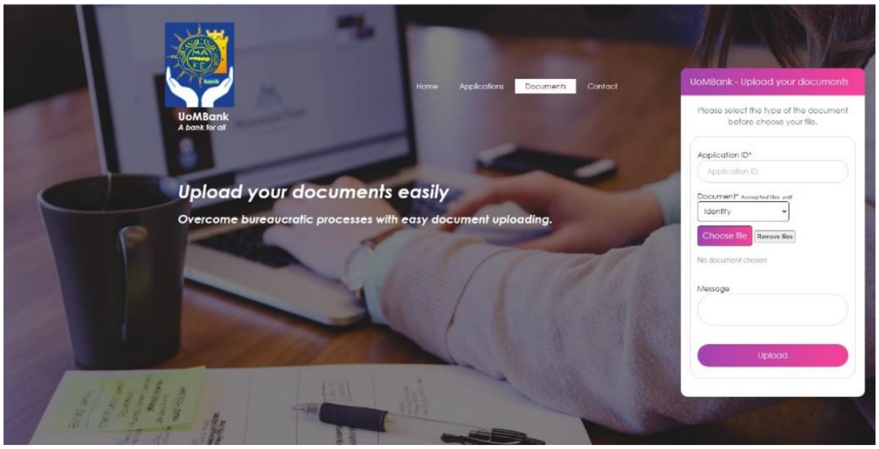
On the basis of the previous check, additional series of activities are conducted solely for applicants who are new customers in order to thoroughly inspect and verify their application details. That is to say, such activities are not performed for applicants who are already customers at "UoMBank", since the validity of their details was formerly assessed on their on-board date. To this extent, after receiving an application of a new customer, the application is automatically assigned to a different "UoMBank's" employee, based on the requested loan type. According to the following table (Table 8), an assignment decision is made.

Loan Type	Department	Employee - Assignee
Personal	Personal loan department	Mr. Smith Johnson
Auto	Auto loan department	Mrs. Harris Miller
Mortgage	Mortgage loan department	Mr. Stew Patriksen
Business	Business loan department	Mrs. Faye Rohn

Table 8 - Application assignment rules based on the loan type

Once the application is automatically assigned, the relative "UoMBank's" employee evaluates the received application details. Interestingly, the bank executive can either approve the application, requesting optionally missing or additional documents, or reject it, causing the termination of the entire loan application-to-approval process. In case a 24 hours interval has elapsed without the user task being completed, the employee is automatically reminded for their pending task. When an employee initially approves the application, yet requests additional documents in order the application to continue be processed, an email is forwarded automatically to the applicant, requesting missing or additional details, previously specified by the “UoMBank’s” employee. Taking into account that the email is sent over internet messaging protocols, in case of an error occurrence, the formerly involved employee is automatically notified to contact the customer manually.

At this point, the loan application-to-approval process is not further executed until the applicant provides the requested information or automatically terminates in case a five-days interval has elapsed, without receiving the requested details. From the applicant’s perspective, the applicant receives the forwarded email message, before uploading their documents at the UoMBank’s website portal (Figure 98, 99). Once the applicant provides the additional details, the process is further triggered, prompting the formerly involved employee to check again the application and the received documents’ validity. This process is repeatedly executed until the application is found complete, or terminated by the employee, rejecting the loan granting.



Upload your documents easily

Overcome bureaucratic processes with easy document uploading.

UoMBank - Upload your documents

Please select the type of the document before choose your file.

Application ID*

Application ID

Document* Accepted files .pdf

Identity

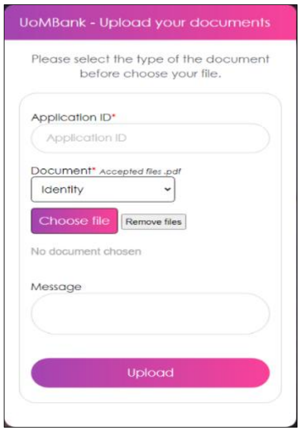
Choose file Remove files

No document chosen

Message

Upload

Figure 98 - UoMBank Service Portal: Document uploading


UoMBank - Upload your documents

Please select the type of the document before choose your file.

Application ID*

Application ID

Document* Accepted files .pdf

Identity

Choose file Remove files

No document chosen

Message

Upload

Figure 99 - Document uploading form

Having presented the aforementioned activities, the subsequent tasks are performed for all applicants, irrespective of being existing or new “UoMBank’s” customers. At first, on the basis of applicant’s age and previous debts, the process is terminated once the applicant’s age is less than 21 or greater than 71 years old, or the amount of debt exceeds the level of €3000. In any other case, the loan application is further evaluated, where the applicant’s creditworthiness is determined, denoting, thus, their eligibility to get a loan granted. For this purpose, the applicant’s credit score is calculated, scoring their monthly income, their previous debts and their occupation type (Table 9), before aggregating such individual scores. A total credit score over 650 denotes the applicant’s creditworthiness and entails its loan eligibility, while in any other case the customer is characterized as “ineligible” for getting a loan granted. In such situation, a rejection letter, informing the customer for their ineligibility, is automatically generated, before being attached to an automatically forwarded email.

Monthly Income	Score	Previous debts	Score	Occupation type	Score
<500	100	<500	250	Employed	250
[500,800)	150	[500,800)	220		
[800,1000)	180	[800,1500)	180	Self-employed	220
[1000,1200)	210	[1500,2500)	140		
>=1200	250	>=2500	100	Unemployed	150

Table 9 - Credit score assessment

On the other hand, the application continues to be processed, where the bank’s Credit Risk is evaluated. Considering that a customer might default and fail to fulfill their obligations, it is imperative for “UoMBank” to estimate the Credit Risk level, before readjusting the initially offered interest rate, that will reflect a risk premium and counterweight the applicant’s default risk. In this regard, after predicting the amount of expected losses out of the total initial bank’s exposure (i.e., loan amount), the Credit Risk is characterized as “low”, “moderate” or “high” (Table 10). Concretely, expecting to lose more than the 18% of the requested loan amount, the Credit Risk is characterized as “high” and the initially offered interest rate should be readjusted in order “UoMBank” to be compensated for the high exposure to customer’s default risk.

Expected losses/Loan Amount	Bank's Credit Risk
[0, 12%)	low
[12%, 18%)	moderate
>=18%	high

Table 10 - Expected Losses assessment

For this purpose, the following equation (1) [89], [90] is introduced in order to indicate the bank's expected losses once the customer defaults. Decomposing its factors, the bank's expected losses (EL) are expected to be affected by the initial Exposure At Default (EAD), namely the requested loan amount, the customer's Probability of Default (PD), namely the likelihood that a customer would be unable to cover the loan payments, as well as the bank's Loss Given Default (LGD), namely the portion of the bank's funds that are expected to be definitely lost if a customer defaults.

$$\mathrm{EL} = \mathrm{EAD} * \mathrm{PD} * \mathrm{LGD}\ (1),$$

$$EL: Bank's\ Expected\ Losses$$
$$EAD: Bank's\ Exposure\ At\ Default$$
$$PD: Customer's\ Probability\ of\ Default$$
$$LGD: Loss\ Given\ Default$$

Upon estimating the customer's Probability of Default (PD), a predictive model based on historical data is introduced (Table 11). In this way, after calculating the ratio of applicant's monthly loan payments to their monthly income, a probability of default is estimated, considering whether the applicant will earn enough money, not only to pay the monthly loan payments, but also cover all the living expenses. Concretely, for a ratio of 0.75, "UoMBank" estimates that it is 60% probable that this applicant will default, taking into account that the 75% of their monthly income should be spent solely for covering the monthly loan payment, without including their living cost.

Monthly Loan Payment/Monthly Income	Customer's Probability of Default (PD)
[0, 0.3)	10%
[0.3,0.4)	15%
[0.4,0.5)	25%
[0.5,0.6)	30%
[0.6,0.7)	50%
[>=0.7,1)	60%
>=1	90%

Table 11 - Probability of Default (PD) assessment

Additionally, the Loss Given Default (LGD) is another catalytic factor, utilized for the evaluation of the expected losses. Interestingly, "UoMBank" never loses its entire exposure, since guarantees from borrowers are provided, while it has the right to resell a property in order to receive its money back. Thus, LGD constitutes an estimation of the portion of the bank's funds that will be definitely lost if a default occurs. For this reason, utilizing historical data, "UoMBank" defines a default LGD value for each requested loan type (Table 12). Considering a case where a €200.000 mortgage loan is granted and the customer defaults, "UoMBank" will not lose the entire exposure, since it has the right to resell the real estate property for a specific amount of money (e.g., €160.000). Hence, the LGD is calculated as the percentage of lost to the initial exposure (20%=40000/200000).

Loan type	Loss Given Default (LGD)
Personal	50%
Auto	30%
Mortgage	30%
Business	40%

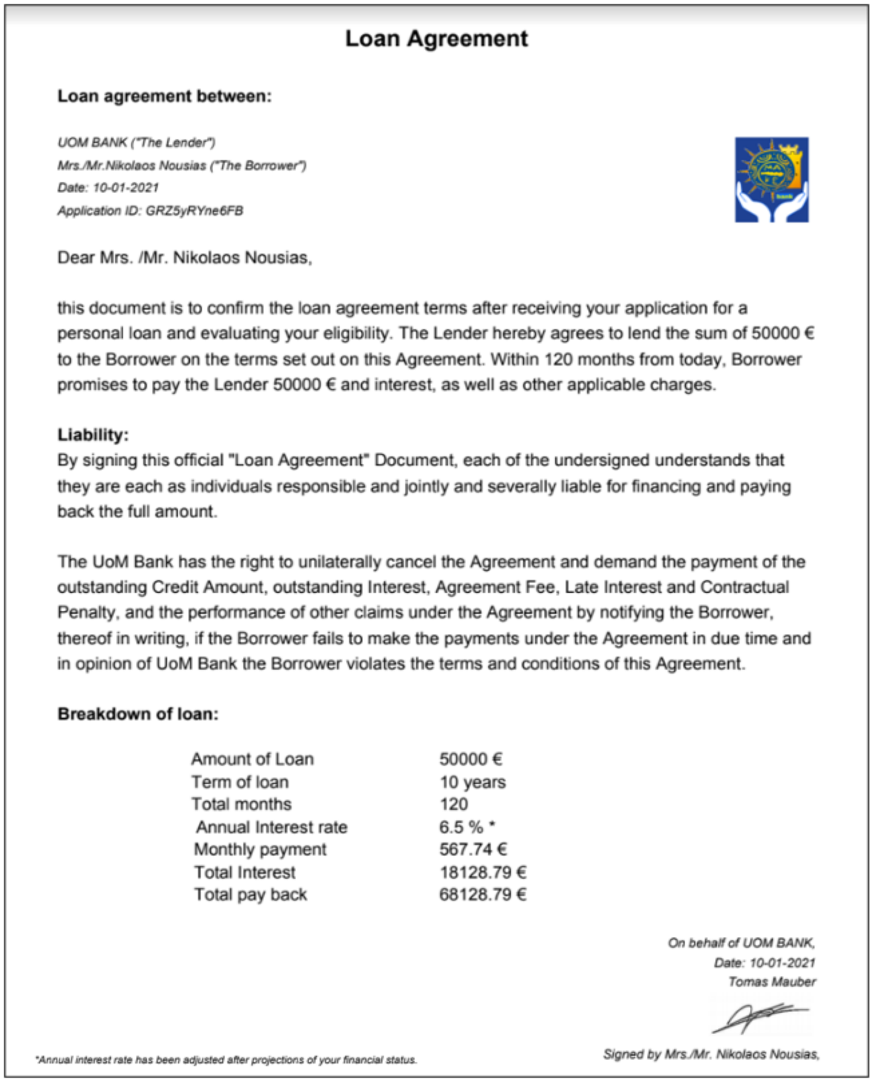
Table 12 - Loss Given Default (LGD) assessment

Thereafter, having estimated the bank's Credit Risk, the initially offered interest rate might be readjusted, aiming to compensate the bank for the additional risk that might take on. More specifically, in case of a low Credit Risk, the offered interest rate remains at its initial level. In turn, in case of a moderate Credit Risk, the interest rate is automatically readjusted based on the ratio of the bank's expected losses to the requested loan amount (Table 13). On the other hand, for a high Credit Risk level, a bank executive is automatically notified to analyze the risk and propose a new interest rate. To this extent, "UoMBank's" employee can alternatively reject the application in case the excessive bank's risk is characterized as a deterrent factor for an applicant to get a loan granted. In such scenario, a rejection document is automatically generated, before being attached to an automatically forwarded email.

Expected Losses/ Loan Amount	Interest rate increase
>=12% and <14%	+0.5%
>=14% and <16%	+1%
>=16% and <18%	+1.5%

Table 13 - Interest rate increase

At the end, having estimated the applicant's loan eligibility, the bank's credit risk and the finally offered interest rate, a loan agreement is automatically generated (Figure 100), before being attached to an automatically forwarded email. Altogether, at the time the proposed loan agreement is forwarded to the applicant, the loan application-to-approval process automatically ends, while the acceptance of the proposed loan agreement and the subsequent activities fall outside the scope of the underlying process. However, taking into account that an applicant might contact the bank at any time, the following assumptions need to be made.



Loan Agreement

Loan agreement between:

UOM BANK ("The Lender")
Mrs./Mr.Nikolaos Nousias ("The Borrower")
Date: 10-01-2021
Application ID: GRZ5yRYne6FB

Dear Mrs. /Mr. Nikolaos Nousias,

this document is to confirm the loan agreement terms after receiving your application for a personal loan and evaluating your eligibility. The Lender hereby agrees to lend the sum of 50000 € to the Borrower on the terms set out on this Agreement. Within 120 months from today, Borrower promises to pay the Lender 50000 € and interest, as well as other applicable charges.

Liability:
By signing this official "Loan Agreement" Document, each of the undersigned understands that they are each as individuals responsible and jointly and severally liable for financing and paying back the full amount.

The UoM Bank has the right to unilaterally cancel the Agreement and demand the payment of the outstanding Credit Amount, outstanding Interest, Agreement Fee, Late Interest and Contractual Penalty, and the performance of other claims under the Agreement by notifying the Borrower, thereof in writing, if the Borrower fails to make the payments under the Agreement in due time and in opinion of UoM Bank the Borrower violates the terms and conditions of this Agreement.

Breakdown of loan:

Amount of Loan	50000 €
Term of loan	10 years
Total months	120
Annual Interest rate	6.5 % *
Monthly payment	567.74 €
Total Interest	18128.79 €
Total pay back	68128.79 €

On behalf of UOM BANK,
Date: 10-01-2021
Tomas Mauber

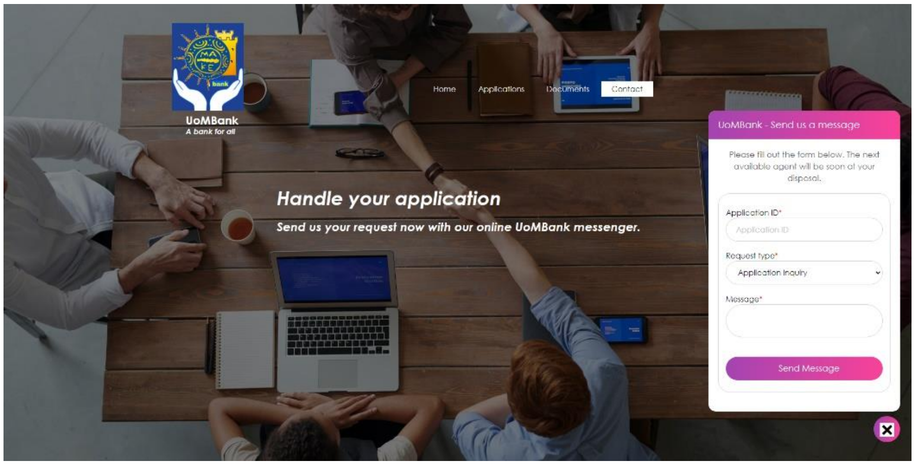
Signed by Mrs./Mr. Nikolaos Nousias,

**Annual interest rate has been adjusted after projections of your financial status.*

Figure 100 - Proposed loan agreement

Basic assumptions:

Considering that the applicant has not a concrete insight as the application gets processed, the applicant might make an inquiry or cancel the application assessment at any time of the process. Utilizing the "UoMBank messenger", available on the website portal (Figure 101), an applicant can select between an inquiry and cancellation request, forwarding their message to "UoMBank", upon the form submission (Figure 102).



Handle your application

Send us your request now with our online UoMBank messenger.

UoMBank - Send us a message

Please fill out the form below. The next available agent will be soon at your disposal.

Application ID*

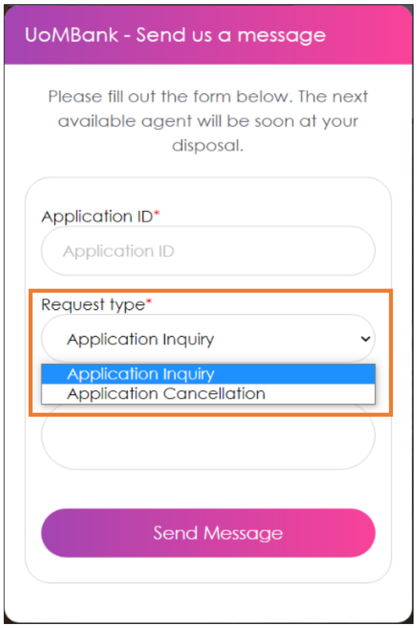
Application ID

Request type*

Application Inquiry

Message*

Figure 101 - "UoMBank" Website Portal: Application handling



UoMBank - Send us a message

Please fill out the form below. The next available agent will be soon at your disposal.

Application ID*

Application ID

Request type*

Application Inquiry

Application Inquiry
Application Cancellation

Figure 102 - "UoMBank" messenger

From "UoMBank's" perspective, in case of an inquiry request, a confirmation email is automatically forwarded to the applicant, denoting that the inquiry has been successfully received. Subsequently, a bank executive is automatically assigned to handle the inquiry, while gets notified every 24 hours for a pending task, in case the inquiry has yet to be successfully answered. Alternatively, in case of a cancellation request, a confirmation of cancellation is automatically forwarded to the applicant, before the loan application process being automatically terminated.

In the following section, the previously introduced textual description and its corresponding basic assumptions, are rendered in a BPMN representation, capturing the process in a graphical way and elucidating any ambiguities arising from the free-form text.

6.3 Modeling the loan process with BPMN

Having meticulously described the "UoMBank's" loan application-to-approval process, it is of utmost importance to communicate and impart its logic in an unambiguous and comprehensible manner. To this extent, process models and graphical notations are deemed to prevail over textual descriptions of business processes, since any misinterpretations and ambiguities, arising from the freeform textual descriptions, are well mitigated [5]. In this regard, a BPMN 2.0 representation of the aforementioned loan application-to-approval process, is provided in Figure 103.

More specifically, the BPMN model is comprised of two pools, namely the bank's and the applicant's pool, where special attention is given to their interactions, utilizing the BPMN message flows. Given the fact that the loan application-to-approval process takes place within the "UoMBank's" boundaries, it is imperative to illustrate the process from the bank's perspective, rather than shedding light into the applicant's way of acting. For this reason, the textual description has been transformed to BPMN elements inside the bank's pool, utilizing a plethora of concepts that have been previously introduced in Chapter 3.

Notably, a series of automated tasks have been utilized, denoting the objective to automate such process. Service tasks, script tasks and send tasks, are well adopted for the premise of this process, due to their nature to be implemented with programming code and enacted automatically by a workflow engine. However, even if eliminating the human intervention is the ultimate aim in the context of process and decision automation, user tasks are inevitably utilized where human-decision-making is imperative for getting the loan application processed or when human actors need to perform manually any automated tasks that might fail. Considering the powerful BPMN symbol armory, intermediate and boundary events, exclusive and event-based gateways, subprocesses and event subprocesses, to exemplify some, are additionally utilized in order to increase the expressive power of the model and reflect the textual description in the best possible way.

The underlying process is captured according to the BPMN 2.0 specification in the following figure, while for the sake of the desired level of granularity, its "Check applicant's details" subprocess, is rendered in a separate model, illustrated in Figure 104, decreasing the complexity of the main process.

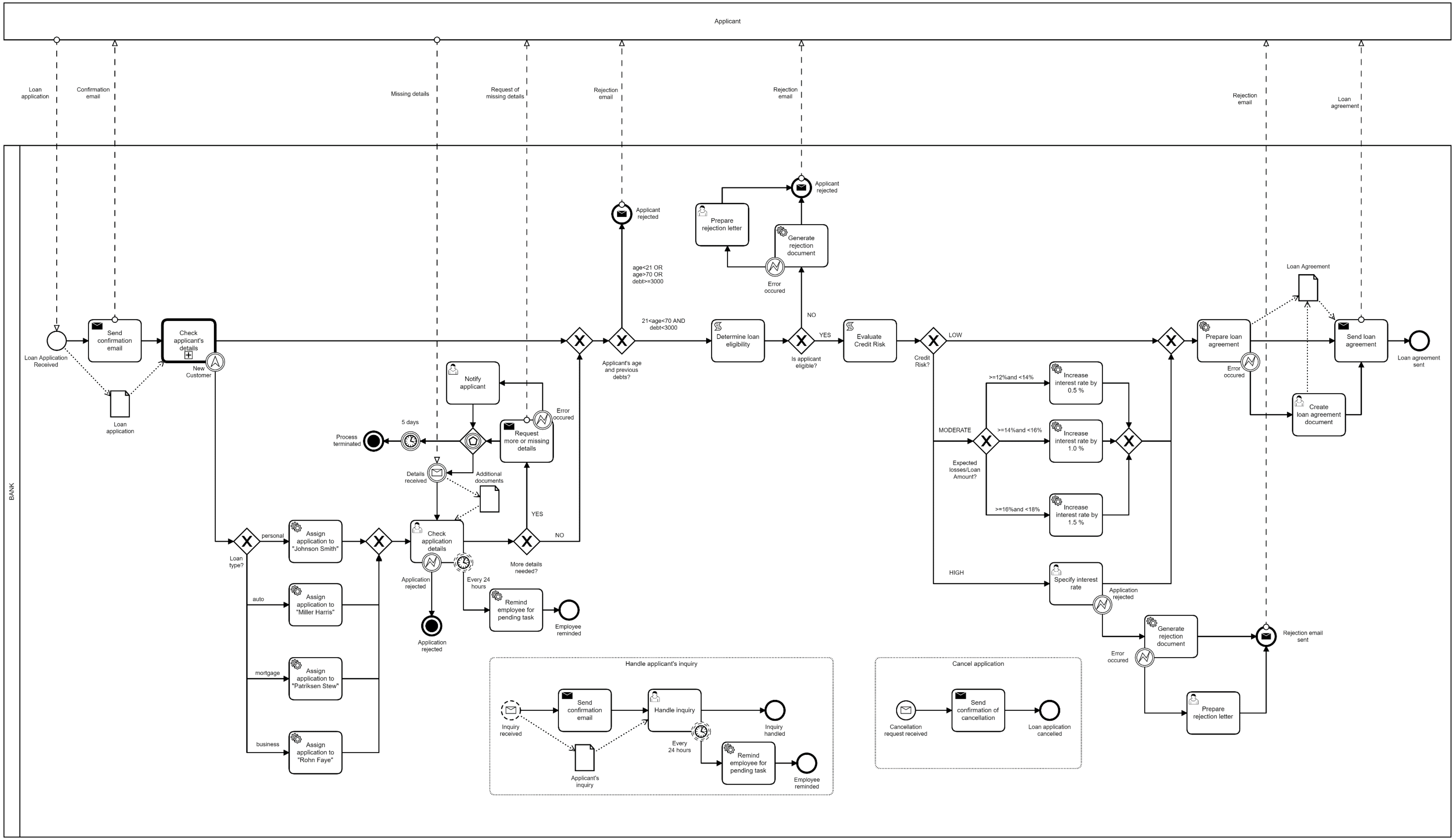
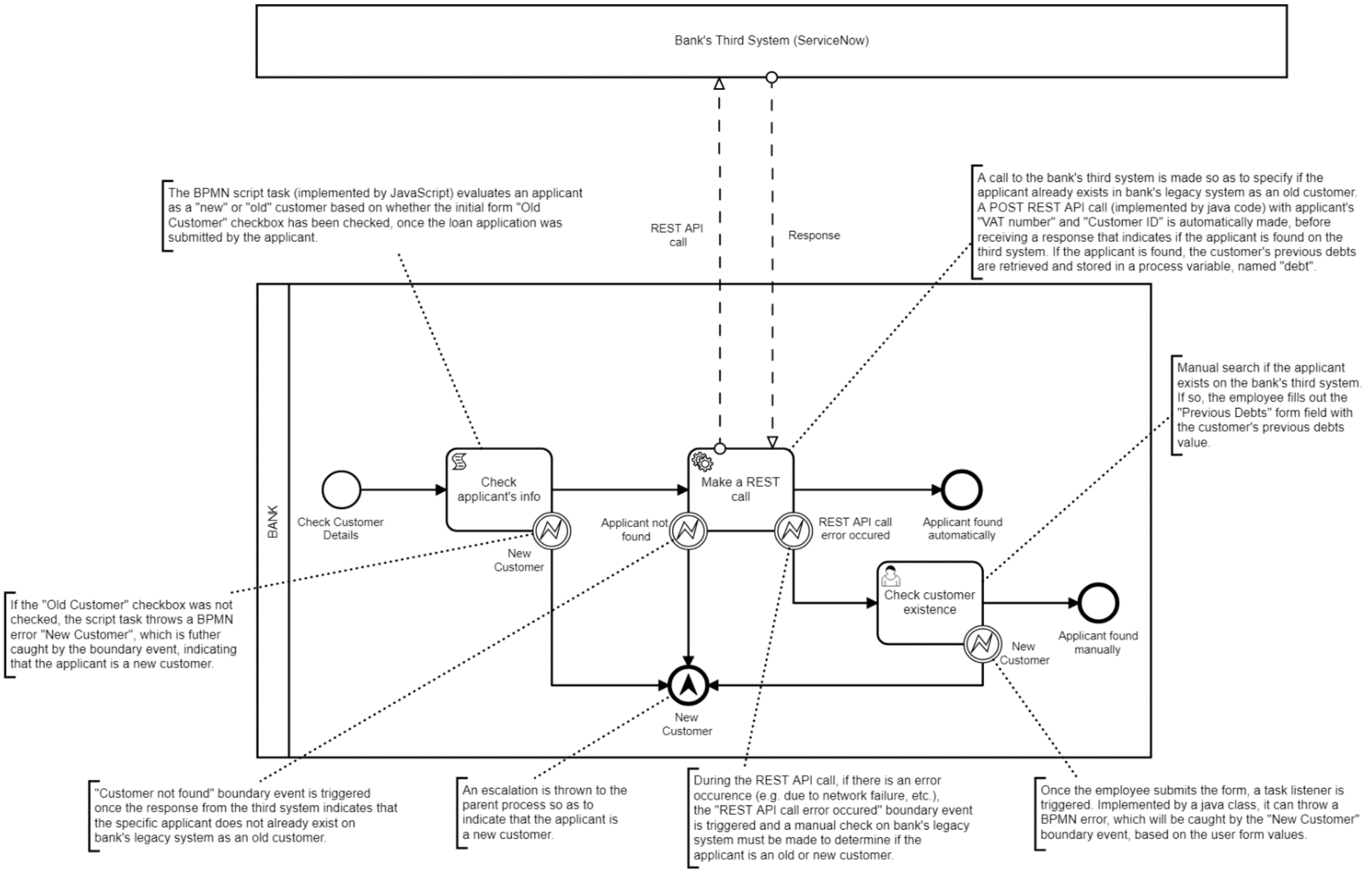


Figure 103 - BPMN process representation



The BPMN script task (implemented by JavaScript) evaluates an applicant as a "new" or "old" customer based on whether the initial form "Old Customer" checkbox has been checked, once the loan application was submitted by the applicant.

A call to the bank's third system is made so as to specify if the applicant already exists in bank's legacy system as an old customer. A POST REST API call (implemented by java code) with applicant's "VAT number" and "Customer ID" is automatically made, before receiving a response that indicates if the applicant is found on the third system. If the applicant is found, the customer's previous debts are retrieved and stored in a process variable, named "debt".

Manual search if the applicant exists on the bank's third system. If so, the employee fills out the "Previous Debts" form field with the customer's previous debts value.

If the "Old Customer" checkbox was not checked, the script task throws a BPMN error "New Customer", which is futher caught by the boundary event, indicating that the applicant is a new customer.

"Customer not found" boundary event is triggered once the response from the third system indicates that the specific applicant does not already exist on bank's legacy system as an old customer.

An escalation is thrown to the parent process so as to indicate that the applicant is a new customer.

During the REST API call, if there is an error occurence (e.g. due to network failure, etc.), the "REST API call error occured" boundary event is triggered and a manual check on bank's legacy system must be made to determine if the applicant is an old or new customer.

Once the employee submits the form, a task listener is triggered. Implemented by a java class, it can throw a BPMN error, which will be caught by the "New Customer" boundary event, based on the user form values.

Figure 104 - Subprocess: Integration with bank's third system

6.4 Identifying business decisions in the process model: BPMN and DMN integration

Undoubtedly, transforming directly a textual description to a BPMN process representation, a risk of convoluting the process and the decision logic frequently lurks. In this way, repetitive operational decisions are hidden behind BPMN constructs, posing a threat to the maintainability, scalability and understandability of both processes and decisions. Typically utilized, BPMN gateways and script tasks, imitate the decision logic by capturing decisions in a hard-coded manner. Thus, the traceability of the decision-making process is dramatically decreased, while any change in the decision logic entails modification on the process model logic. For this purpose, it is imperative to identify business decisions in the process model, externalize them in separate DMN models and invoke them as services by the means of BPMN business rule tasks. In this regard, in Figure 105, business decisions are identified in the process model, before delving into each of them in the following lines.

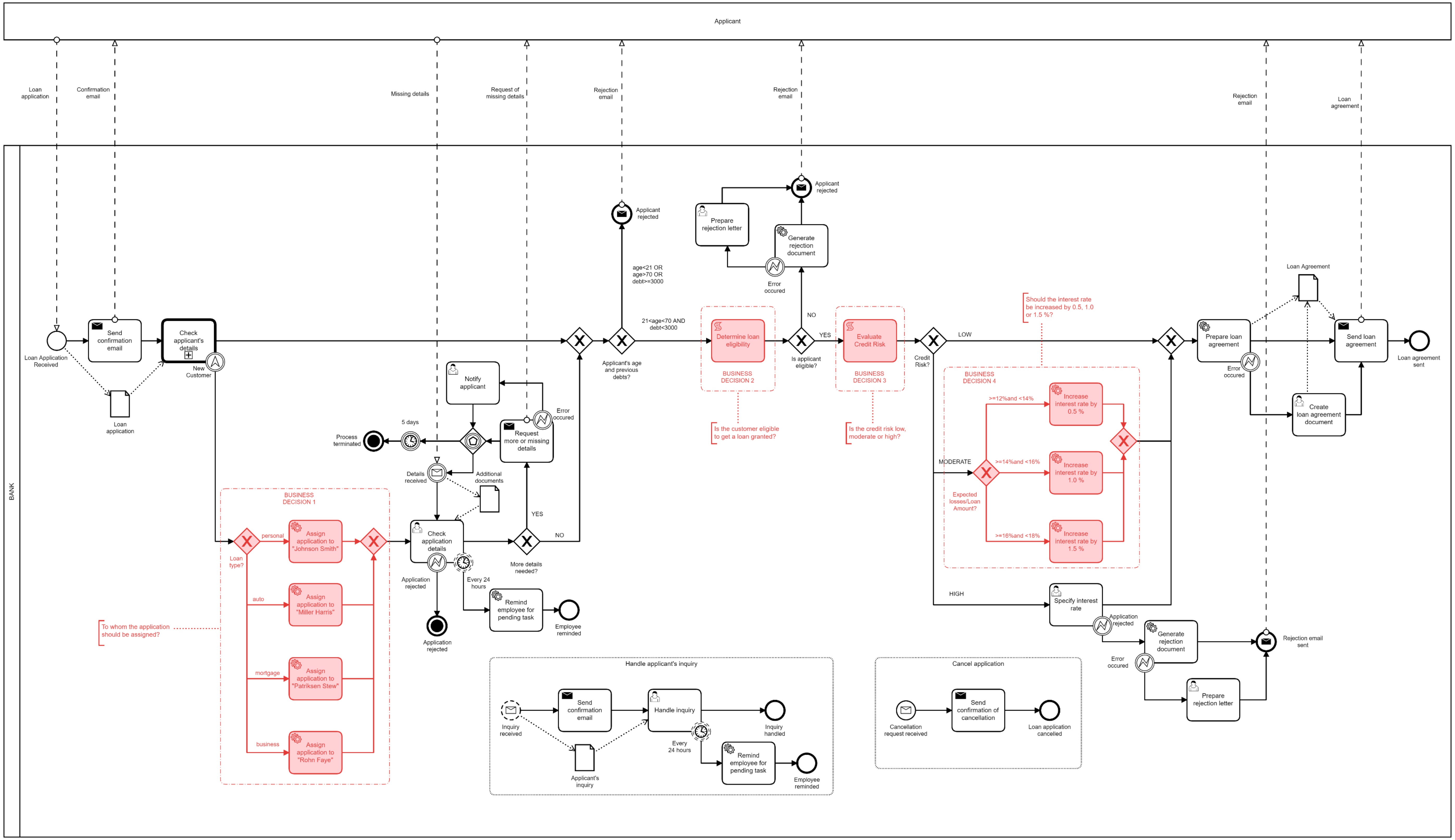


Figure 105 - Identifying business decisions in the process model

Business Decision 1:

The first business decision, identified in the process model, is associated with a decision that "UoMBank" needs to make each time that an application of a new customer is received. Considering the textual description and the application assignment rules, illustrated in Table 8, depending on the requested loan type, a different "UoMBank's" employee is assigned to check the application details. However, capturing the aforementioned decision logic with BPMN exclusive gateways (Figure 106), the complexity of the model escalates, while any modifications of the decision logic entail readjustment of the exclusive gateways and their associated sequence flows.

In this regard, the underlying decision logic can be rendered in a BPMN and DMN integration approach, where the decision logic is externalized in a separate model. As a result, the requested loan type can serve as an input data on a decision table that determines the assignee, before such decision is invoked by the means of a business rule task in the BPMN model (Figure 107). Adhering additionally to the principles of process and decision modeling integration, namely the 5PDM as introduced in Chapter 4, BPMN data objects are utilized in the process model in order to denote decision inputs and outputs and render their consistent integration.

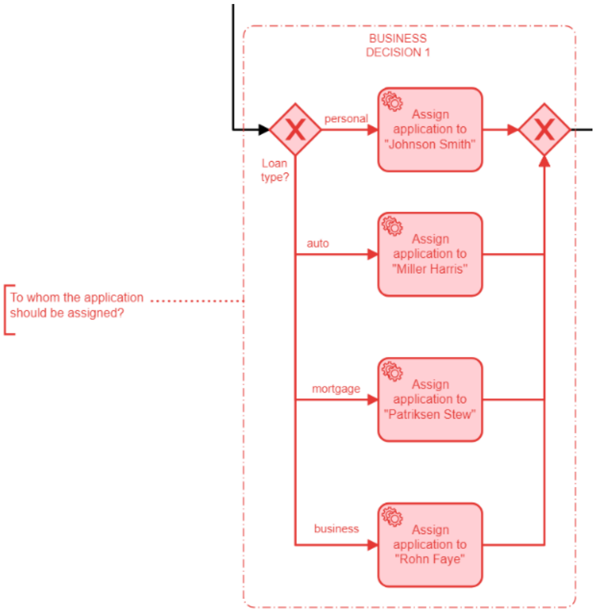
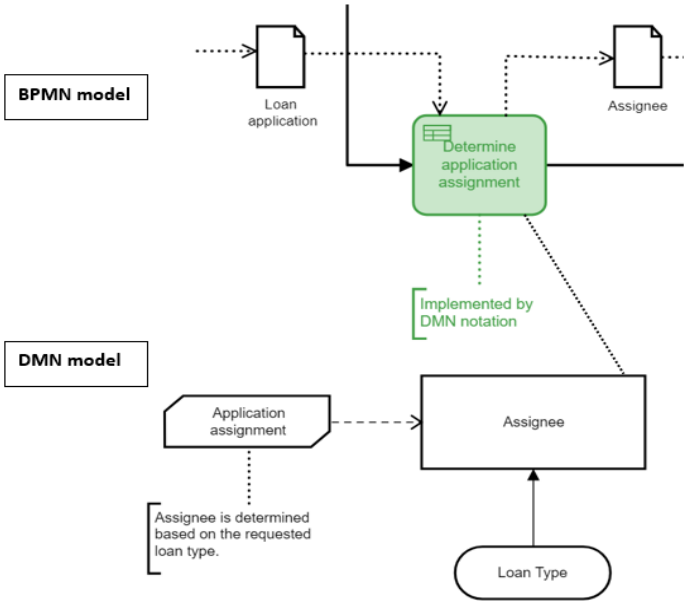
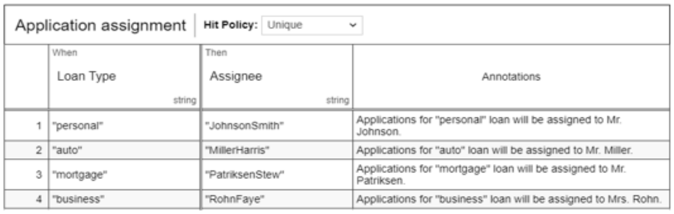

BUSINESS DECISION 1
Loan type?
personal
auto
mortgage
business
Assign application to "Johnson Smith"
Assign application to "Miller Harris"
Assign application to "Patriksen Stew"
Assign application to "Rohn Faye"
To whom the application should be assigned?

Figure 106 - Business decision 1


BPMN model
Loan application
Assignee
Determine application assignment
Implemented by DMN notation
DMN model
Application assignment
Assignee
Assignee is determined based on the requested loan type.
Loan Type



Application assignment | Hit Policy: Unique

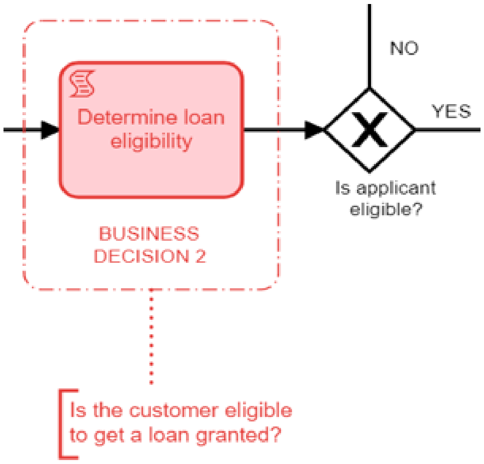
	When Loan Type string	Then Assignee string	Annotations
1	"personal"	"JohnsonSmith"	Applications for "personal" loan will be assigned to Mr. Johnson.
2	"auto"	"MillerHarris"	Applications for "auto" loan will be assigned to Mr. Miller.
3	"mortgage"	"PatriksenStew"	Applications for "mortgage" loan will be assigned to Mr. Patriksen.
4	"business"	"RohnFaye"	Applications for "business" loan will be assigned to Mrs. Rohn.

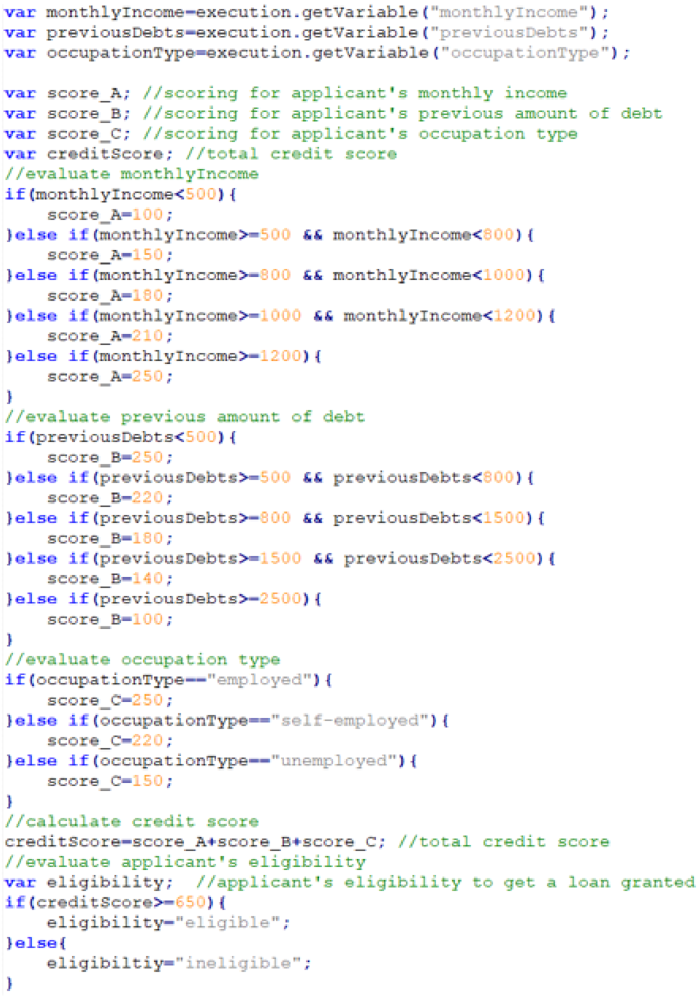
Figure 107 - Business decision 1: BPMN and DMN integration approach

Business Decision 2:

The second business decision, identified in the process model, is associated with a decision that "UoMBank" needs to make in order to determine if an applicant is eligible to get a loan granted. Hidden in a BPMN script task (Figure 108), the decision is captured and made, utilizing a scripting language, that a workflow engine is able to interpret. However, this is more about coding rather than modeling [56]. Considering that decisions are buried into lines of code, decisions outcomes are cumbersome to be evaluated, while any modifications of the decision logic entail the entire script recoding.

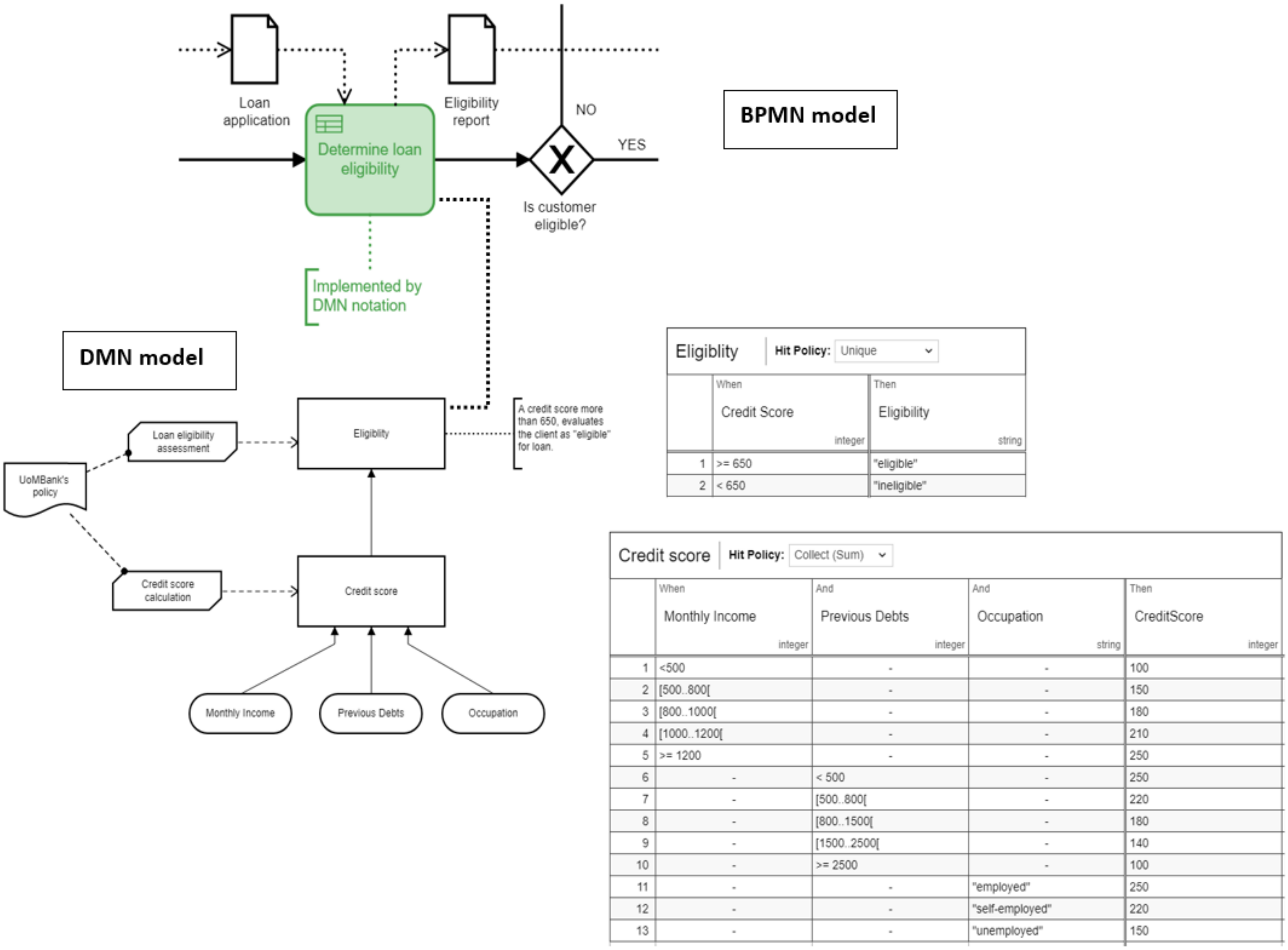
Evaluating and scoring the applicant's monthly income, their previous amount of debt and their occupation type, as presented in the textual description and in Table 9, the applicant's credit score is determined, before the eligibility is decided. As a result, externalizing such decision in a DMN model and invoking it with a BPMN business rule task (Figure 109), the traceability of decision-making process is well refined. In this sense, the applicant's credit score is estimated utilizing a collect sum hit policy in a first-level decision, before such score serves as an input to a higher-level decision, where the eligibility decision is made.


Determine loan eligibility
BUSINESS DECISION 2
NO
YES
Is applicant eligible?
Is the customer eligible to get a loan granted?



```
var monthlyIncome=execution.getVariable("monthlyIncome");
var previousDebts=execution.getVariable("previousDebts");
var occupationType=execution.getVariable("occupationType");

var score_A; //scoring for applicant's monthly income
var score_B; //scoring for applicant's previous amount of debt
var score_C; //scoring for applicant's occupation type
var creditScore; //total credit score
//evaluate monthlyIncome
if(monthlyIncome<500){
    score_A=100;
}else if(monthlyIncome>=500 && monthlyIncome<800){
    score_A=150;
}else if(monthlyIncome>=800 && monthlyIncome<1000){
    score_A=180;
}else if(monthlyIncome>=1000 && monthlyIncome<1200){
    score_A=210;
}else if(monthlyIncome>=1200){
    score_A=250;
}
//evaluate previous amount of debt
if(previousDebts<500){
    score_B=250;
}else if(previousDebts>=500 && previousDebts<800){
    score_B=220;
}else if(previousDebts>=800 && previousDebts<1500){
    score_B=180;
}else if(previousDebts>=1500 && previousDebts<2500){
    score_B=140;
}else if(previousDebts>=2500){
    score_B=100;
}
//evaluate occupation type
if(occupationType=="employed"){
    score_C=250;
}else if(occupationType=="self-employed"){
    score_C=220;
}else if(occupationType=="unemployed"){
    score_C=150;
}
//calculate credit score
creditScore=score_A+score_B+score_C; //total credit score
//evaluate applicant's eligibility
var eligibility;  //applicant's eligibility to get a loan granted
if(creditScore>=650){
    eligibility="eligible";
}else{
    eligibiltiy="ineligible";
}
```

Figure 108 - Business decision 2



Eligiblity **Hit Policy:** Unique

	When Credit Score integer	Then Eligibility string
1	>= 650	"eligible"
2	< 650	"ineligible"

Credit score **Hit Policy:** Collect (Sum)

	When Monthly Income integer	And Previous Debts integer	And Occupation string	Then CreditScore integer
1	<500	-	-	100
2	[500..800[	-	-	150
3	[800..1000[	-	-	180
4	[1000..1200[	-	-	210
5	>= 1200	-	-	250
6	-	< 500	-	250
7	-	[500..800[	-	220
8	-	[800..1500[	-	180
9	-	[1500..2500[	-	140
10	-	>= 2500	-	100
11	-	-	"employed"	250
12	-	-	"self-employed"	220
13	-	-	"unemployed"	150

Figure 109 - Business decision 2: BPMN and DMN integration approach

Business Decision 3:

The third business decision, identified in the process model, is associated with a decision that "UoMBank" needs to make in order to determine if the bank's Credit Risk is low, moderate or high. Identically with the second business decision, such decision is hidden in the background of a script task (Figure 110), estimating the credit risk level in case an applicant defaults. For this purpose, the decision logic is buried into lines of code, where the credit risk is defined after calculating the percentage of expected losses out of the bank's initial exposure, as presented in the textual description of the process.

However, taking into account that the applicant's probability of loss and the bank's loss given default are based on predictive models that utilize historical data, any modification of the aforementioned models and the applied logic, entail a change inside the scripting code. In this regard, externalizing such decision logic in a DMN model, "UoMBank's" Credit Risk can be centrally decided as low, moderate or high, before such decision outcome serves as an input to the BPMN model (Figure 111). For this purpose, a DRD model is introduced, indicating the way that lower-level information is aggregated, before a higher-level decision is made. Interestingly, in such an approach, the decision logic is captured in a graphical way, where business analysts without programming knowledge are able to interpret effortlessly.

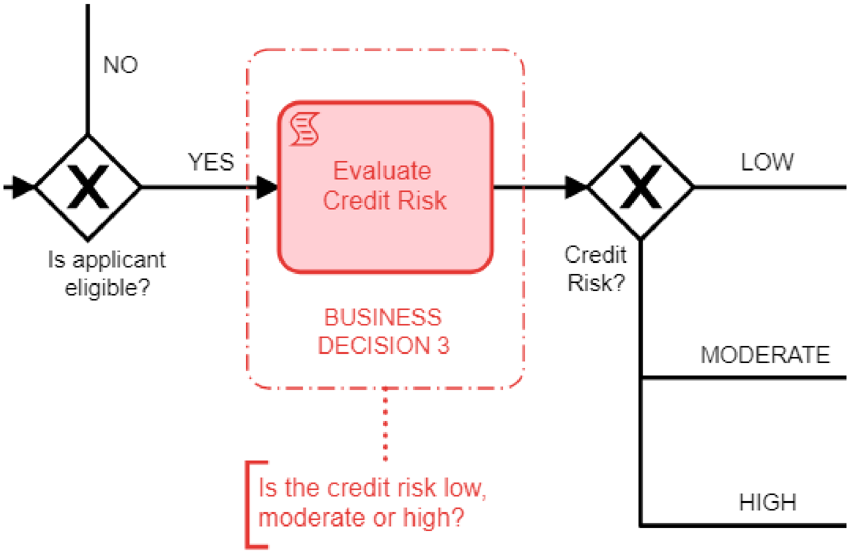
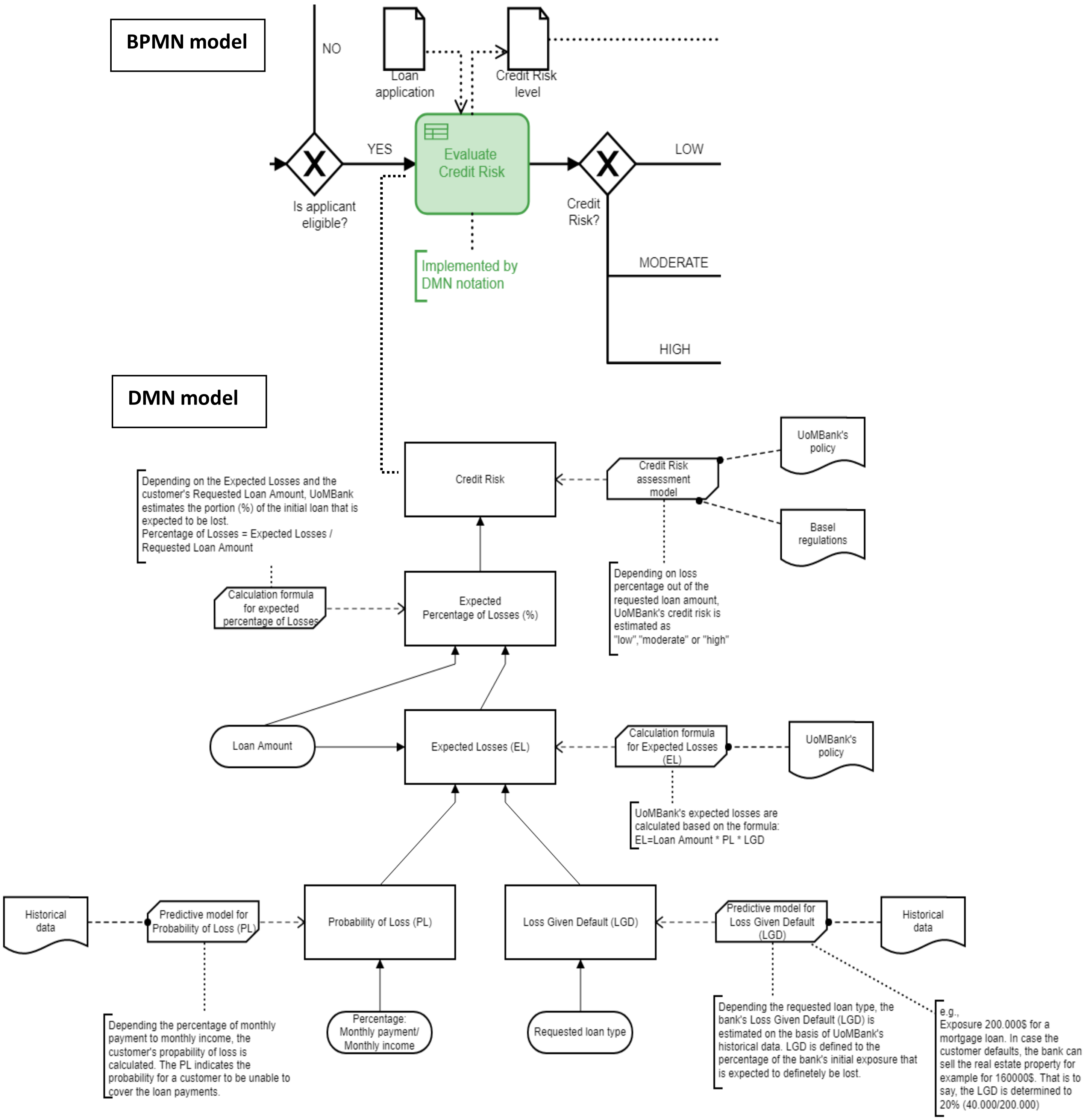


Figure 110 - Business decision 3


BPMN model
NO
Loan application
Credit Risk level
YES
Evaluate Credit Risk
LOW
Is applicant eligible?
Credit Risk?
Implemented by DMN notation
MODERATE
HIGH
DMN model
Credit Risk
UoMBank's policy
Credit Risk assessment model
Basel regulations
Calculation formula for expected percentage of Losses
Expected Percentage of Losses (%)
Loan Amount
Expected Losses (EL)
Calculation formula for Expected Losses (EL)
UoMBank's policy
Historical data
Predictive model for Probability of Loss (PL)
Probability of Loss (PL)
Loss Given Default (LGD)
Predictive model for Loss Given Default (LGD)
Historical data
Percentage: Monthly payment/ Monthly income
Requested loan type

Depending on the Expected Losses and the customer's Requested Loan Amount, UoMBank estimates the portion (%) of the initial loan that is expected to be lost.
Percentage of Losses = Expected Losses / Requested Loan Amount

Depending on loss percentage out of the requested loan amount, UoMBank's credit risk is estimated as "low","moderate" or "high"

UoMBank's expected losses are calculated based on the formula:
EL=Loan Amount * PL * LGD

Depending the percentage of monthly payment to monthly income, the customer's propability of loss is calculated. The PL indicates the probability for a customer to be unable to cover the loan payments.

Depending the requested loan type, the bank's Loss Given Default (LGD) is estimated on the basis of UoMBank's historical data. LGD is defined to the percentage of the bank's initial exposure that is expected to definetely be lost.

e.g.,
Exposure 200.000$ for a mortgage loan. In case the customer defaults, the bank can sell the real estate property for example for 160000$. That is to say, the LGD is determined to 20% (40.000/200.000)

Figure 111 - Business decision 3: BPMN and DMN integration approach

Business decision 4:

Last, but not least, the fourth business decision, identified in the process model, is associated with a decision that "UoMBank" needs to make in order to readjust the initially offered interest rate, once the Credit Risk is characterized as moderate. In this sense, the decision logic is convoluted with the process logic of the model, since decision conditions are hard-coded and rendered as labels in a gateway's outgoing sequence flows (Figure 112). As a result, once a new condition is added or an old one needs to be removed, the process model needs to be adapted accordingly in order to reflect the desired decision logic.

Considering that such decision is based on the percentage of expected losses to the total loan amount, this information can be utilized as an input data to a DMN decision table. Evaluating the table's conditions, the interest rate increase can be determined, before returning as an input to the BPMN model level (Figure 113).

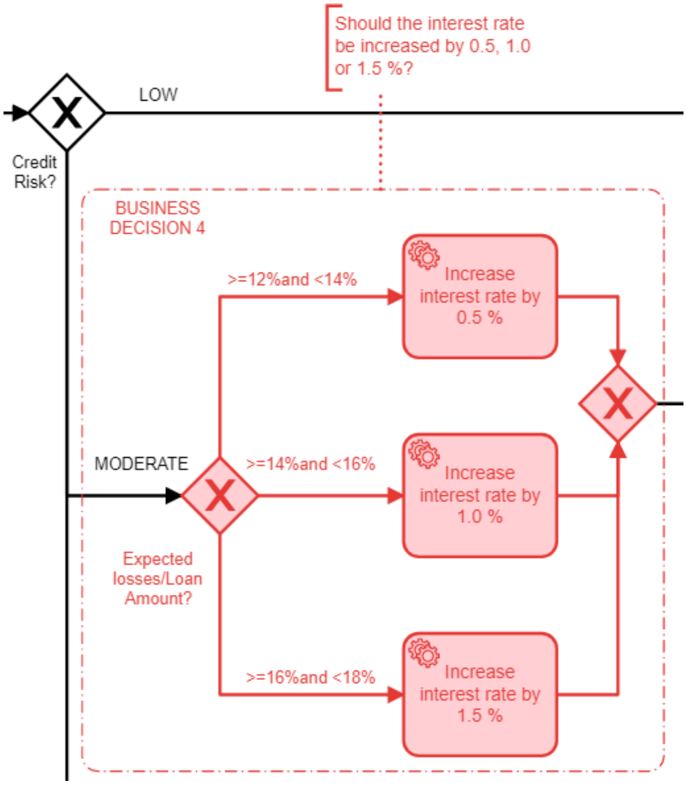
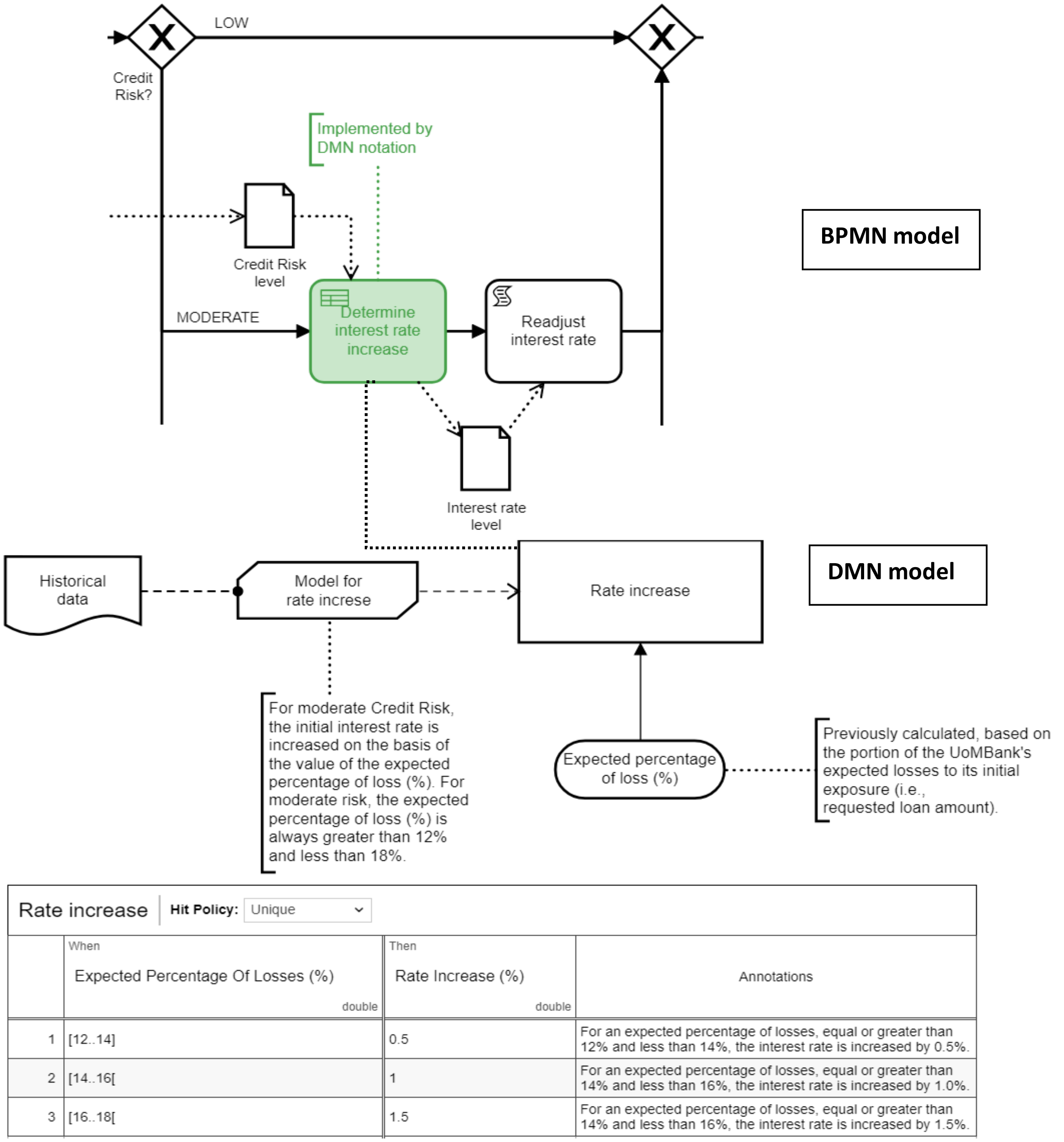

Should the interest rate be increased by 0.5, 1.0 or 1.5 %?
LOW
Credit Risk?
BUSINESS DECISION 4
>=12%and <14%
Increase interest rate by 0.5 %
MODERATE
>=14%and <16%
Increase interest rate by 1.0 %
Expected losses/Loan Amount?
>=16%and <18%
Increase interest rate by 1.5 %

Figure 112 - Business decision 4



For moderate Credit Risk, the initial interest rate is increased on the basis of the value of the expected percentage of loss (%). For moderate risk, the expected percentage of loss (%) is always greater than 12% and less than 18%.

Previously calculated, based on the portion of the UoMBank's expected losses to its initial exposure (i.e., requested loan amount).

Rate increase | Hit Policy: Unique

	When Expected Percentage Of Losses (%) double	Then Rate Increase (%) double	Annotations
1	[12..14]	0.5	For an expected percentage of losses, equal or greater than 12% and less than 14%, the interest rate is increased by 0.5%.
2	[14..16[	1	For an expected percentage of losses, equal or greater than 14% and less than 16%, the interest rate is increased by 1.0%.
3	[16..18[	1.5	For an expected percentage of losses, equal or greater than 14% and less than 16%, the interest rate is increased by 1.5%.

Figure 113 - Business decision 4: BPMN and DMN integration

Having identified the business decisions in the process model and externalize them to separate DMN models, in Figure 114, the underlying loan application-to-approval process is rendered in the context of integrating BPMN and DMN notations, before such model serves as an input for the automation of the process logic and its decision-making parts.

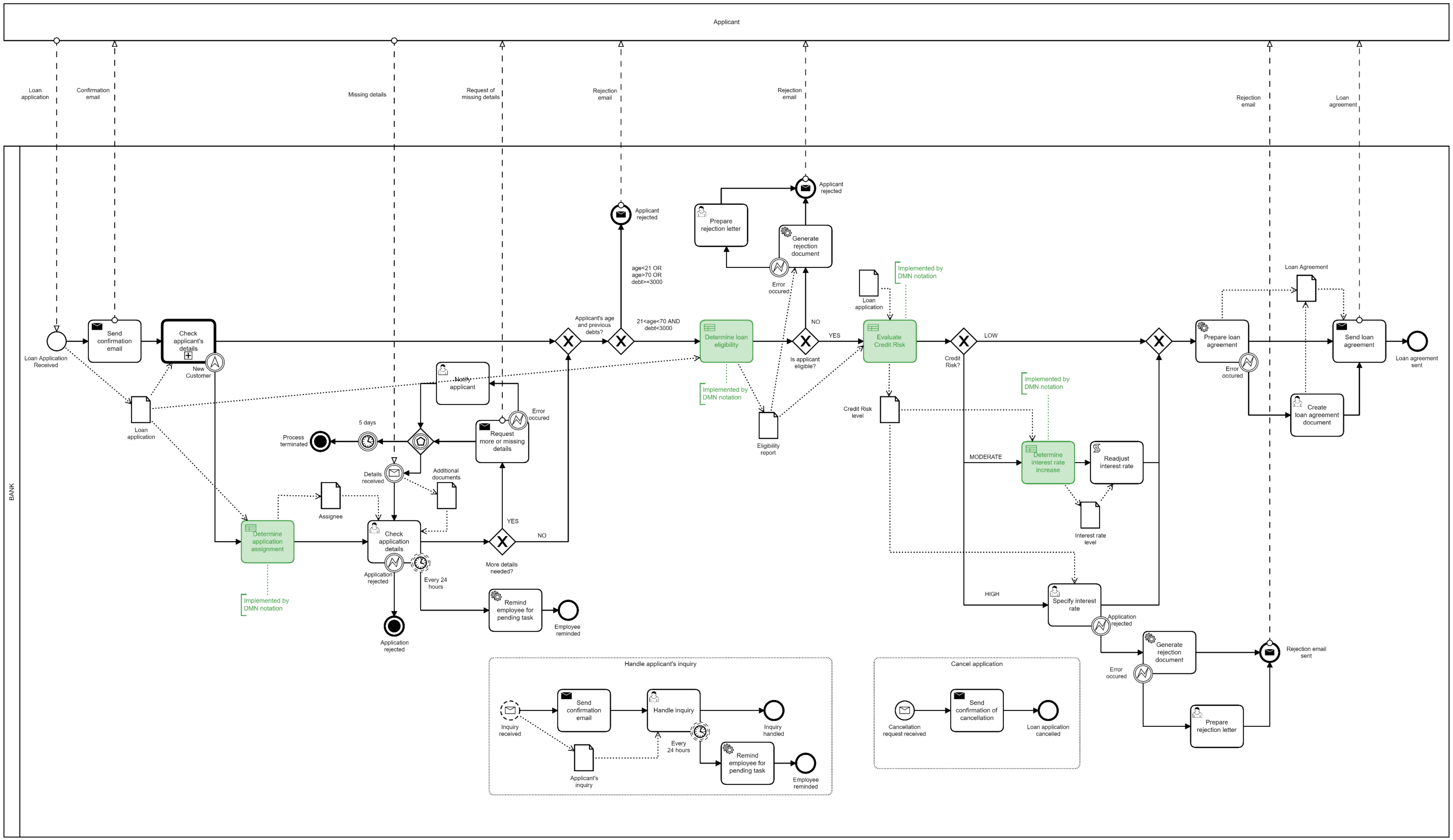


Figure 114 - Process representation: A BPMN and DMN integration approach

6.5 Workflow automation: Specifying the execution semantics

With the ultimate aim to automate the underlying loan application-to-approval process at the fictitious "UoMBank", the previously created model must be enriched with execution semantics. In this regard, it is of utmost importance to consider not only the BPMN and DMN specifications, but also the different implementations that vendors might offer for each of the aforementioned notations' elements. For the premise of this thesis, the Camunda BPM ecosystem and its underlying workflow engine, introduced in Chapter 5, are utilized in order to enact the process automatically and investigate the automation merits.

Considering that automation aims to eliminate the human intervention, special focus is given on the identification of process parts, where automation can deliver high value, decrease the work load and accelerate the process execution. Concentrating, thus, on restructuring human capital, integrating applications and deploying software applications throughout an organization [66], [67], business process automation can make the automatic enactment of an end-to-end business process a reality rather than a wishful thinking. In this regard, workflow automation in the underlying loan application-to-approval process, can be regarded as four-fold, focusing on automating the process logic, the decision-making, the system integration and the activities execution (Figure 115).

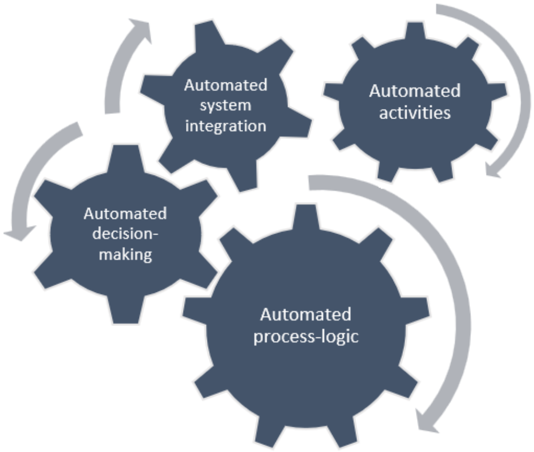
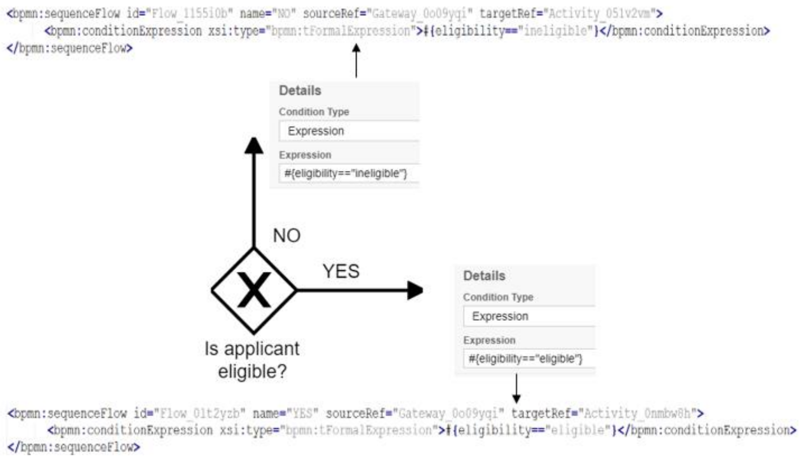


Figure 115 – Workflow automation

At first, aiming to automate the process logic of the model, routing decisions are explicitly specified, indicating the way that the workflow engine will determine the process path during the process enactment. Defining, thus, JUEL expressions in Camunda Modeler, exclusive gateways are enriched with execution semantics that encapsulate the desired process logic

(Figure 116). Therefore, during runtime, the workflow engine evaluates the predefined expressions and automatically defines what has to be enacted next.



```
<bpmn:sequenceFlow id="Flow_115510b" name="NO" sourceRef="Gateway_0o09yqi" targetRef="Activity_051v2vm">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">#{eligibility=="ineligible"}</bpmn:conditionExpression>
</bpmn:sequenceFlow>
```

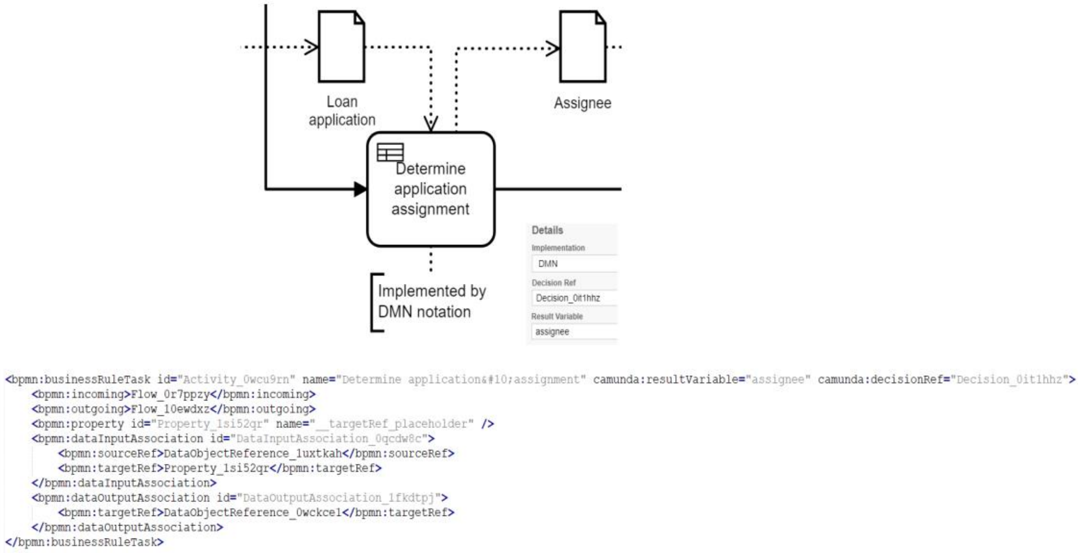
Details
Condition Type
Expression
Expression
#{eligibility=="ineligible"}

Details
Condition Type
Expression
Expression
#{eligibility=="eligible"}

```
<bpmn:sequenceFlow id="Flow_01t2yzb" name="YES" sourceRef="Gateway_0o09yqi" targetRef="Activity_0nmbw8h">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">#{eligibility=="eligible"}</bpmn:conditionExpression>
</bpmn:sequenceFlow>
```

Figure 116 - Implementing routing decisions

In turn, focusing on automated decision-making, business rule tasks are explicitly specified to be implemented by DMN notation (Figure 117). Referencing explicitly the unique identifier of a higher-level decision of a DRD model, the process instance is able to invoke an external decision logic, before its result being available to the process execution level.

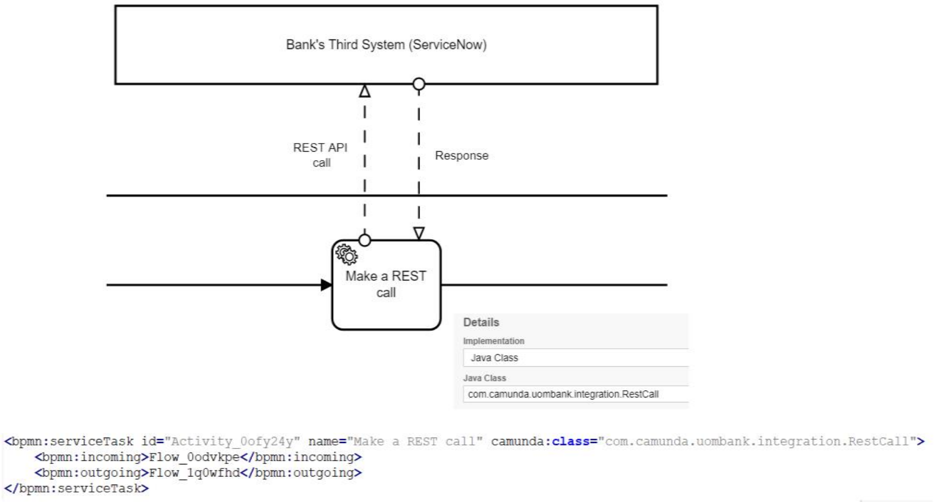


Details
Implementation
DMN
Decision Ref
Decision_0it1hhz
Result Variable
assignee

```
<bpmn:businessRuleTask id="Activity_0wcu9rn" name="Determine application&#10;assignment" camunda:resultVariable="assignee" camunda:decisionRef="Decision_0it1hhz">
    <bpmn:incoming>Flow_0r7ppzy</bpmn:incoming>
    <bpmn:outgoing>Flow_10ewdxz</bpmn:outgoing>
    <bpmn:property id="Property_1si52qr" name="__targetRef_placeholder" />
    <bpmn:dataInputAssociation id="DataInputAssociation_0qcdw8c">
        <bpmn:sourceRef>DataObjectReference_1uxtkah</bpmn:sourceRef>
        <bpmn:targetRef>Property_1si52qr</bpmn:targetRef>
    </bpmn:dataInputAssociation>
    <bpmn:dataOutputAssociation id="DataOutputAssociation_1fkdtpj">
        <bpmn:targetRef>DataObjectReference_0wckce1</bpmn:targetRef>
    </bpmn:dataOutputAssociation>
</bpmn:businessRuleTask>
```

Figure 117 - Implementing decision-making

Additionally, considering that various enterprise systems might be leveraged as the process lifecycle unfolds, it is imperative to establish a consistent integration among them. In this regard, workflow automation can facilitate such initiatives, since contemporary systems provide an interface that the workflow engine of a BPMS is able to invoke. In the underlying process, implementing a service task by a Java Delegate (Figure 118), the integration with the third system is automatically enacted, once the workflow engine interprets the code inside the Java Class (Figure 119).



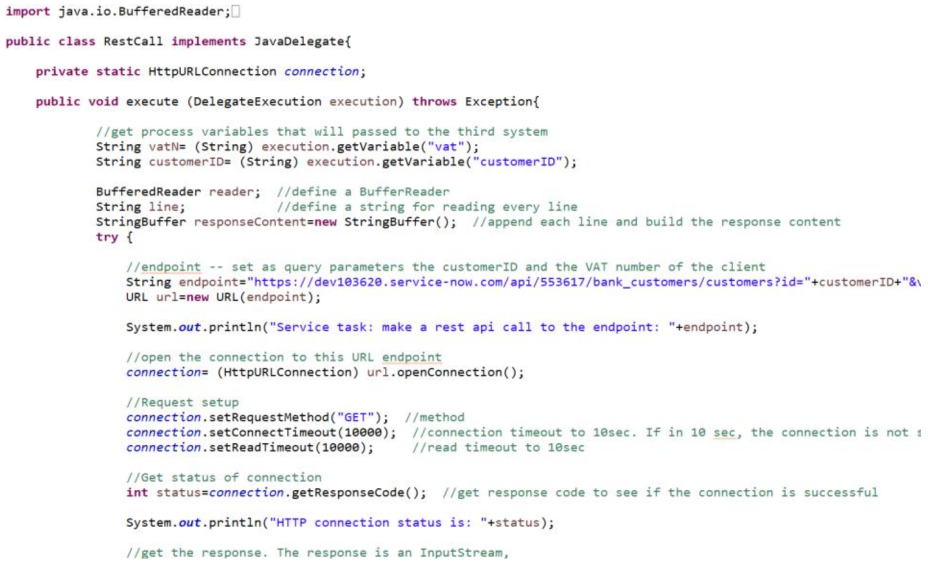
Details

Implementation: Java Class

Java Class: com.camunda.uombank.integration.RestCall

```
<bpmn:serviceTask id="Activity_0ofy24y" name="Make a REST call" camunda:class="com.camunda.uombank.integration.RestCall">
    <bpmn:incoming>Flow_0odvkpe</bpmn:incoming>
    <bpmn:outgoing>Flow_1q0wfhd</bpmn:outgoing>
</bpmn:serviceTask>
```

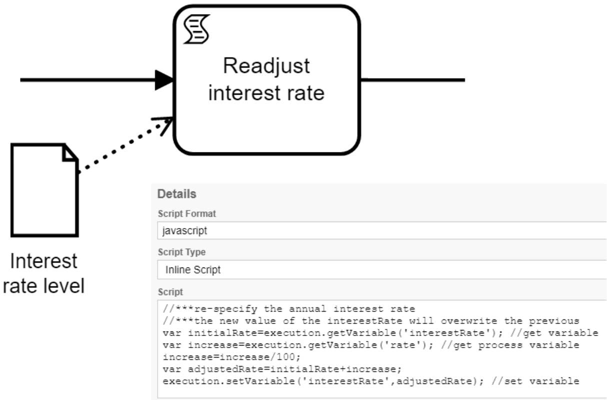
Figure 118 - Implementing service tasks



```
import java.io.BufferedReader;□

public class RestCall implements JavaDelegate{

    private static HttpURLConnection connection;

    public void execute (DelegateExecution execution) throws Exception{

            //get process variables that will passed to the third system
            String vatN= (String) execution.getVariable("vat");
            String customerID= (String) execution.getVariable("customerID");

            BufferedReader reader;  //define a BufferReader
            String line;            //define a string for reading every line
            StringBuffer responseContent=new StringBuffer();  //append each line and build the response content
            try {

                //endpoint -- set as query parameters the customerID and the VAT number of the client
                String endpoint="https://dev103620.service-now.com/api/553617/bank_customers/customers?id="+customerID+"&
                URL url=new URL(endpoint);

                System.out.println("Service task: make a rest api call to the endpoint: "+endpoint);

                //open the connection to this URL endpoint
                connection= (HttpURLConnection) url.openConnection();

                //Request setup
                connection.setRequestMethod("GET");  //method
                connection.setConnectTimeout(10000);  //connection timeout to 10sec. If in 10 sec, the connection is not
                connection.setReadTimeout(10000);     //read timeout to 10sec

                //Get status of connection
                int status=connection.getResponseCode();  //get response code to see if the connection is successful

                System.out.println("HTTP connection status is: "+status);

                //get the response. The response is an InputStream,
```

Figure 119 - Java Delegate

Undoubtedly, an integral part of workflow automation is to execute automated activities that would otherwise be enacted by human actors. Utilizing script tasks or service tasks, an activity can be automated, paving the way for the elimination of human intervention. In the underlying process, automating activities with script tasks, a JavaScript block of code is established in the tasks' background and executed automatically by the workflow engine during runtime (Figure 120). Interestingly, comparing to the case where a service task needs to invoke an external Java Class, script tasks are developed and coded within the process model, without needing to invoke an external block of code.



Details

Script Format

javascript

Script Type

Inline Script

Script

```
//***re-specify the annual interest rate
//***the new value of the interestRate will overwrite the previous
var initialRate=execution.getVariable('interestRate'); //get variable
var increase=execution.getVariable('rate'); //get process variable
increase=increase/100;
var adjustedRate=initialRate+increase;
execution.setVariable('interestRate',adjustedRate); //set variable
```

```
<bpmn:scriptTask id="Activity_1fj3pqd" name="Readjust&#10;interest rate" scriptFormat="javascript">
    <bpmn:incoming>Flow_0xbq9dx</bpmn:incoming>
    <bpmn:outgoing>Flow_1p0ag7f</bpmn:outgoing>
    <bpmn:property id="Property_1q6vwcg" name="__targetRef_placeholder" />
    <bpmn:dataInputAssociation id="DataInputAssociation_118sjpv">
      <bpmn:sourceRef>DataObjectReference_1vce3op</bpmn:sourceRef>
      <bpmn:targetRef>Property_1q6vwcg</bpmn:targetRef>
    </bpmn:dataInputAssociation>
    <bpmn:script>
        //***re-specify the annual interest rate
        //***the new value of the interestRate will overwrite the previous
        var initialRate=execution.getVariable('interestRate'); //get process variable
        var increase=execution.getVariable('rate'); //get process variable
        increase=increase/100;
        var adjustedRate=initialRate+increase;
        execution.setVariable('interestRate',adjustedRate); //set process variable
    </bpmn:script>
</bpmn:scriptTask>
```

Figure 120 - Implementing script tasks

Altogether, specifying the execution semantics for each of the model's elements, the process model along with its all dependencies, namely decision models, java classes and java libraries, need to be deployed into the workflow engine, where it will be automatically enacted. In the following section, the deployment procedure of the process model is introduced.

6.6 Process deployment into the workflow engine

After specifying the execution semantics of the process elements and explicitly define the ways of their implementation, the process model along with its all dependencies, such as the decision models, the invoked Java Classes and the utilized Java libraries, need to be encapsulated into a single application file (i.e., .war file), that will be subsequently deployed into the Camunda's application server. To this extent, a Maven project is created, considering that Maven constitutes a build management tool tailored to Java-based project development in a more effortless and straightforward way. Addressing two fundamental aspects of building software, namely the way that the software is built, as well as its dependencies [91], Maven projects are deemed to facilitate an application development.

Interestingly, the focal pillar of a Maven project is its pom.xml file, which contains all the project's configuration, such as its dependencies, libraries, required plug-ins, etc., indispensable to the project execution. Upon the build of a project, Maven reads the pom.xml file and dynamically downloads and adds all the included dependencies, Java libraries and Maven plug-ins, from central, remote or local repositories. Importantly, in case a dependency is not available in a local repository, Maven automatically downloads it from a remote repository into the local one [92].

Additionally, Maven archetypes are utilized in order to standardize the way that a Maven project is configured. Contemplating that each archetype provides pre-packacked dependencies, an archetype constitutes a template for standardized Maven project creation. In this regard, Camunda's Archetype can be utilized in order to automatically include all the required dependencies, needed to the development of Java Classes, within the application file [75].

Containing not only the source code, namely the Java Classes, but also any other resources required for the process enactment, such as .bpmn and .dmn files, a Maven project is compiled and packaged as a single output .war file, before being deployed to Camunda's application server. Currently, a plethora of Integrated Development Environments (IDEs), such as the Eclipse IDE, facilitate the interoperability of Maven with the IDE's build mechanisms and source editing tools. To this extent, Appendix A renders how a Maven project can be created in Eclipse IDE, before Appendix B introduces the way that a Maven project is installed and deployed as a single .war file into Camunda's application server. Overall, the aforementioned concepts are illustrated in Figure 121, before presenting the overall demo architecture in the following section.

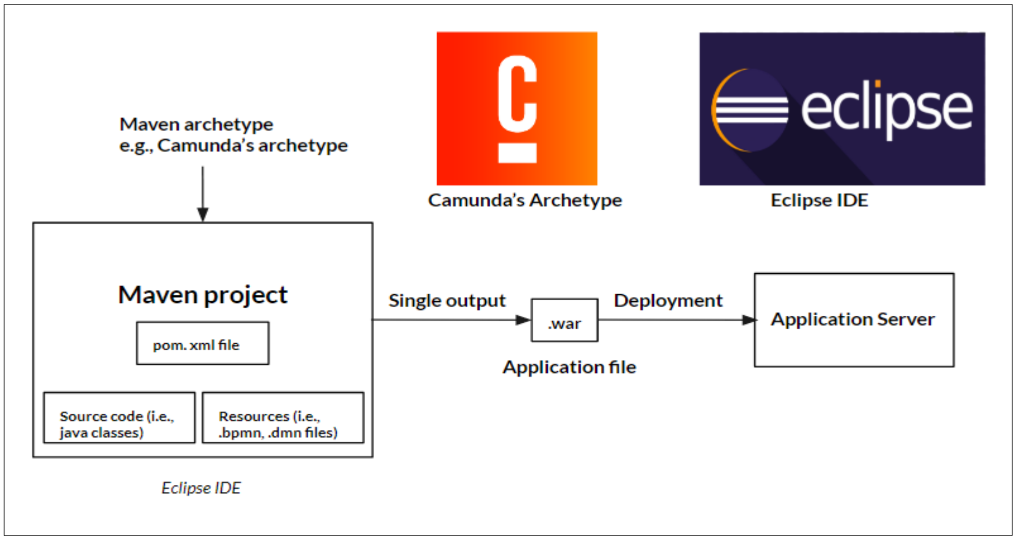


Figure 121 – Maven project

6.7 Overall architecture

In the current section, the overall demo architecture is presented, shedding light into the ways that its different components communicate with each other, as well as focusing on the role of Camunda BPM ecosystem and the utilization of its fundamental components as the process lifecycle unfolds. At first, modeling a business process within Camunda Modeler or Cawemo, a conceptual model is engendered, rendering the process logic in a graphical, let alone intuitive way. Enriching such model with executable semantics and implementing its service tasks with Java Classes, a Maven project, utilizing the Camunda's archetype, is created. Packaging all process and decision models into such project, a single .war file is generated after installing the Maven project. Deploying, thereafter, the application file to the Camunda's application server, the process is automatically enacted by its workflow engine.

In this regard, considering that for the premise of the underlying process, applicants utilize the "UoMBank's" website portal, it is of utmost importance to discover how a process instance can be generated and evaluated automatically, once a loan application form is submitted from an end user. Upon a form submission, thus, an asynchronous REST API call to the workflow engine is conducted triggering a process instance generation. Packaging all the applicant's information as a JSON string variable, all data are transferred in the backend, before being stored as process variables. Upon a process instance completion, a relative response is returned to the end user, enhancing, thus, the user experience and facilitating the communication of the front end and backend environments.

Ultimately, utilizing default Camunda's webapplications (i.e., Camunda Cockpit, Camunda Tasklist) as "UoMBank's" backend systems, the process monitoring of individual process instances, namely loan applications, as well as the task handling and user task completion, are significantly facilitated.

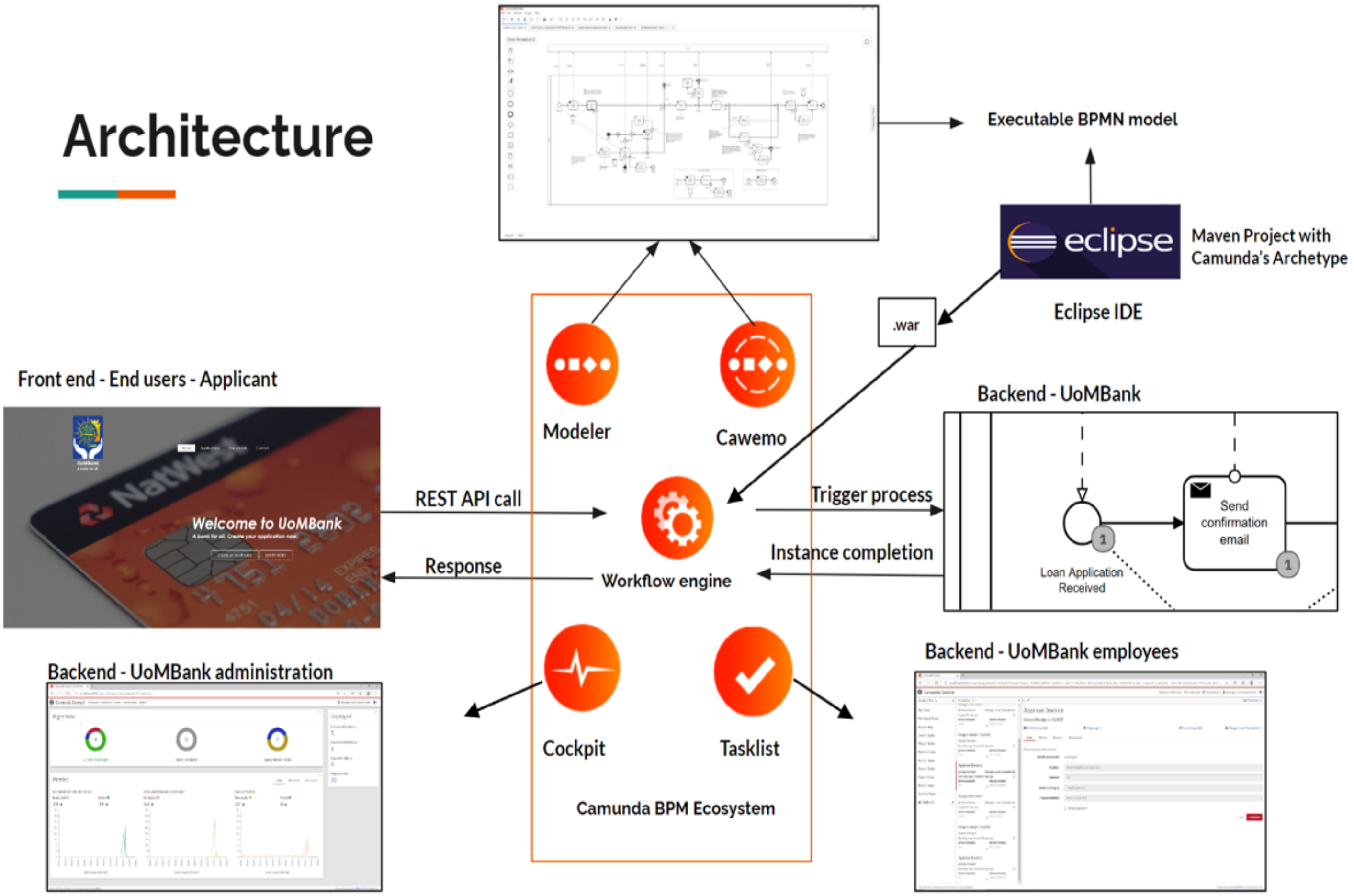


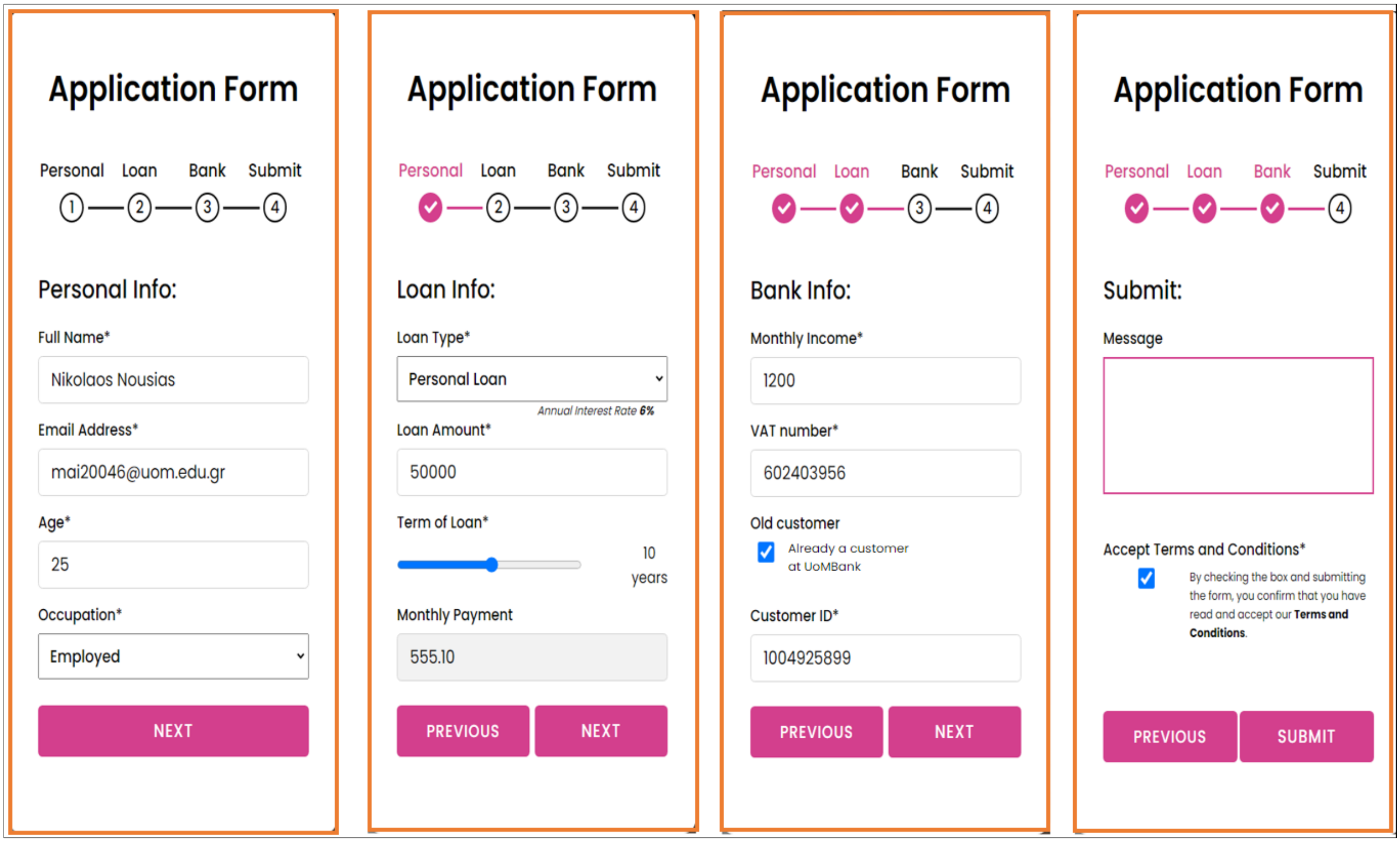
Figure 122 - Demo architecture

6.8 Execution scenario

Having presented the loan application-to-approval process, the overall architecture and the way that its different components interact with each other, in the current section an execution scenario is developed in order to illustrate how the process unfolds both from the applicant's and the "UoMBank's" perspective. In this regard, special focus is given on the customer experience upon submitting a loan application form, let alone shedding light on what is happening behind the scenes and determines if an applicant is eligible to get a loan granted.

Applicant's perspective:

1 At first, the applicant enters the "UoMBank's" website portal, where a loan application form can be submitted on their behalf (Figure 123).



Application Form

Personal ① Loan ② Bank ③ Submit ④

Personal Info:

Full Name*: Nikolaos Nousias

Email Address*: mai20046@uom.edu.gr

Age*: 25

Occupation*: Employed

NEXT

Application Form

Personal ✓ Loan ② Bank ③ Submit ④

Loan Info:

Loan Type*: Personal Loan

Annual Interest Rate ***6%***

Loan Amount*: 50000

Term of Loan*: 10 years

Monthly Payment: 555.10

PREVIOUS NEXT

Application Form

Personal ✓ Loan ✓ Bank ③ Submit ④

Bank Info:

Monthly Income*: 1200

VAT number*: 602403956

Old customer

☑ Already a customer at UoMBank

Customer ID*: 1004925899

PREVIOUS NEXT

Application Form

Personal ✓ Loan ✓ Bank ✓ Submit ④

Submit:

Message

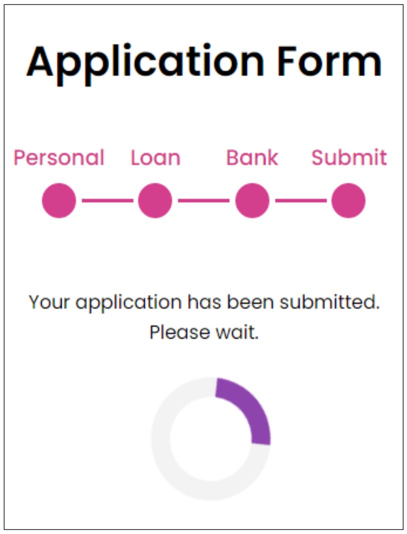
Accept Terms and Conditions*

☑ By checking the box and submitting the form, you confirm that you have read and accept our **Terms and Conditions**.

PREVIOUS SUBMIT

Figure 123 - Loan application form

2 After filling the required information, the applicant submits the form and receives a loading message, indicating the processing of the form (Figure 124).

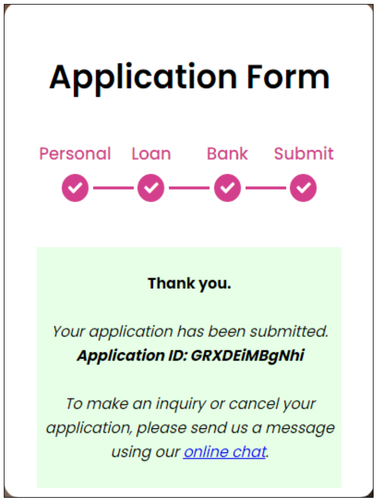


Application Form

Your application has been submitted.
Please wait.

Figure 124 - Form loading

3 Within seconds, the form is updated with a success message, denoting the efficacious form submission (Figure 125).



Application Form

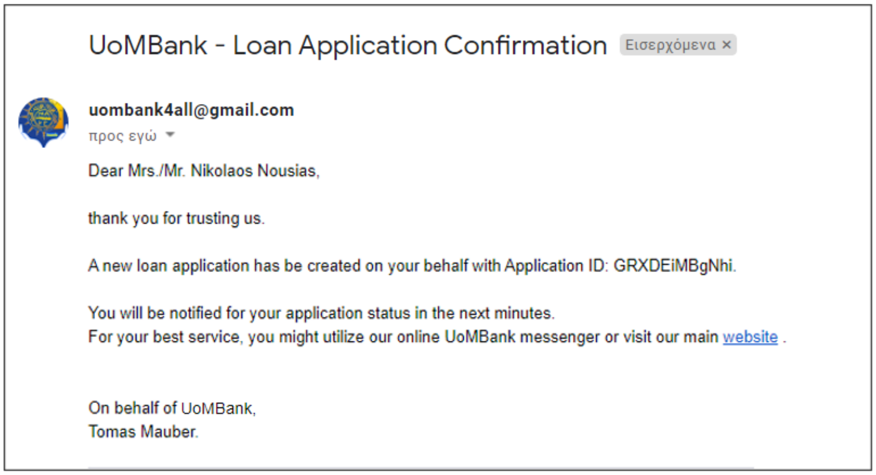
Thank you.

Your application has been submitted.
Application ID: GRXDEiMBgNhi

To make an inquiry or cancel your application, please send us a message using our online chat.

Figure 125 - Successful loan application form submission

4 At the same time, the applicant receives automatically a confirmation email from “UoMBank” at the provided email address (Figure 126).



UoMBank - Loan Application Confirmation

uombank4all@gmail.com

Dear Mrs./Mr. Nikolaos Nousias,

thank you for trusting us.

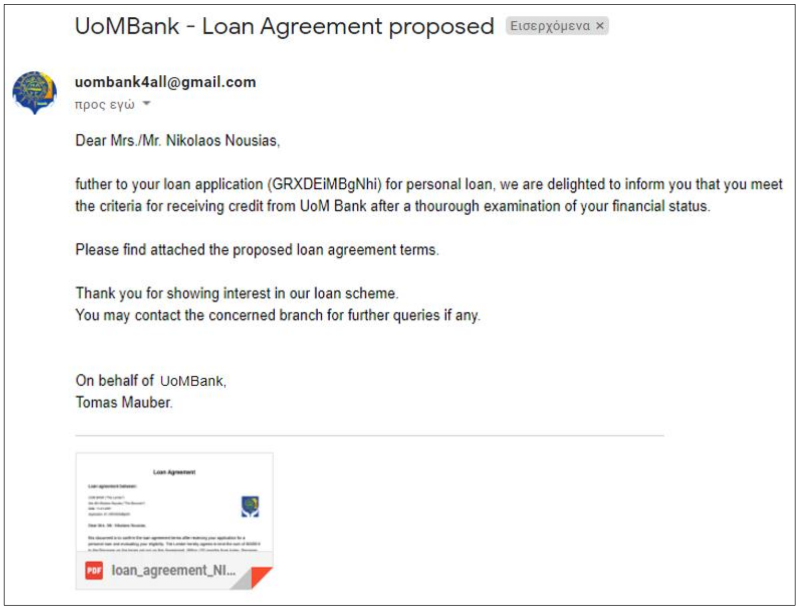
A new loan application has be created on your behalf with Application ID: GRXDEiMBgNhi.

You will be notified for your application status in the next minutes.
For your best service, you might utilize our online UoMBank messenger or visit our main website .

On behalf of UoMBank,
Tomas Mauber.

Figure 126 - Confirmation email

5 Within seconds, a second email is forwarded to the applicant’s email address, where the loan agreement terms are attached, indicating the customer’s eligibility for getting a loan granted (Figure 127).



UoMBank - Loan Agreement proposed

uombank4all@gmail.com

Dear Mrs./Mr. Nikolaos Nousias,

futher to your loan application (GRXDEiMBgNhi) for personal loan, we are delighted to inform you that you meet the criteria for receiving credit from UoM Bank after a thourough examination of your financial status.

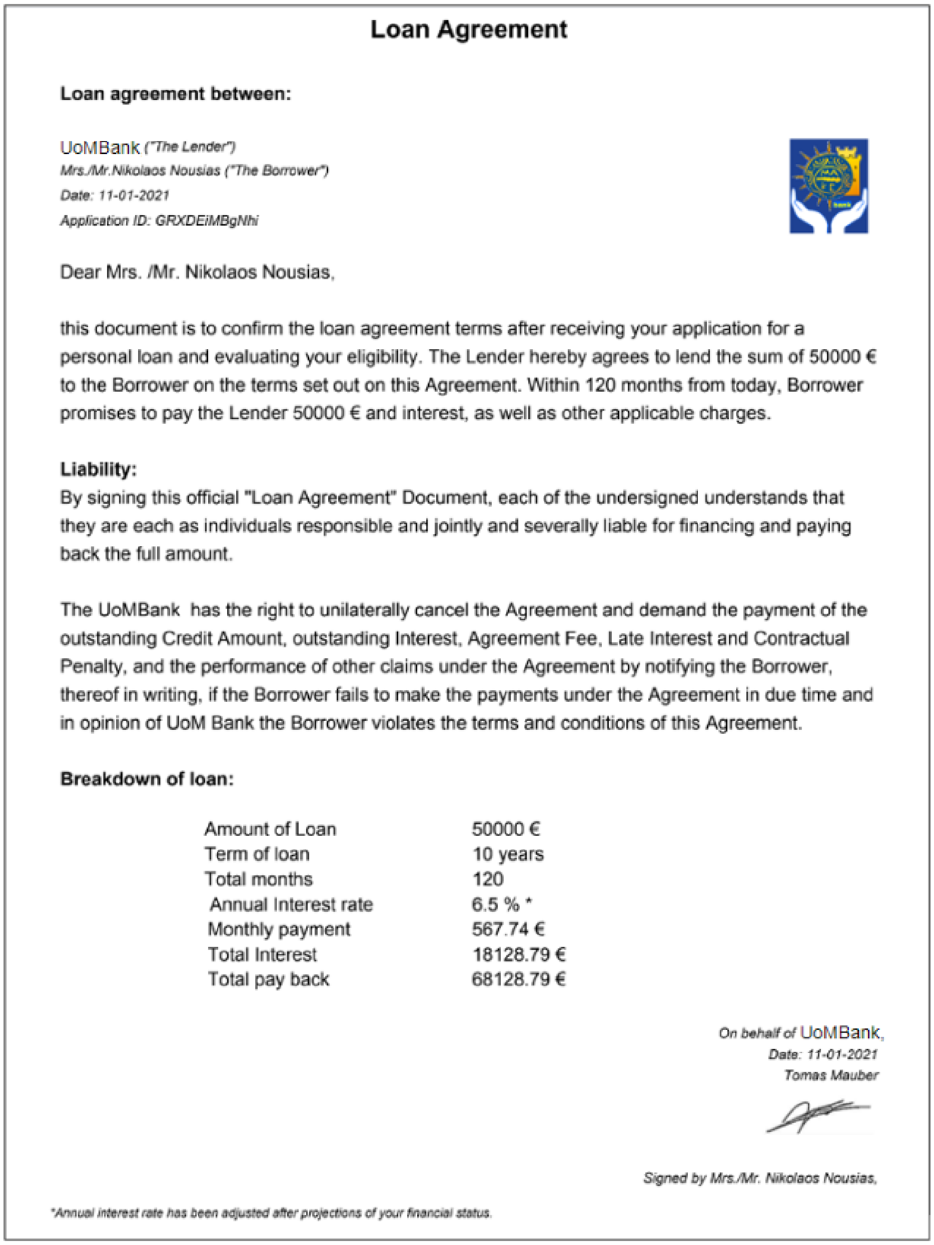
Please find attached the proposed loan agreement terms.

Thank you for showing interest in our loan scheme.
You may contact the concerned branch for further queries if any.

On behalf of UoMBank,
Tomas Mauber.

Figure 127 - Proposed loan agreement email

6 Subsequently, the applicant views the proposed loan agreement terms, as well as the breakdown of the requested loan (Figure 128). However, the acceptance of the proposed loan agreement and the subsequent activities fall outside the scope of the underlying process and are not presented further.



Loan Agreement

Loan agreement between:

UoMBank *("The Lender")*
Mrs./Mr.Nikolaos Nousias ("The Borrower")
Date: 11-01-2021
Application ID: GRXDEiMBgNhi

Dear Mrs. /Mr. Nikolaos Nousias,

this document is to confirm the loan agreement terms after receiving your application for a personal loan and evaluating your eligibility. The Lender hereby agrees to lend the sum of 50000 € to the Borrower on the terms set out on this Agreement. Within 120 months from today, Borrower promises to pay the Lender 50000 € and interest, as well as other applicable charges.

Liability:
By signing this official "Loan Agreement" Document, each of the undersigned understands that they are each as individuals responsible and jointly and severally liable for financing and paying back the full amount.

The UoMBank has the right to unilaterally cancel the Agreement and demand the payment of the outstanding Credit Amount, outstanding Interest, Agreement Fee, Late Interest and Contractual Penalty, and the performance of other claims under the Agreement by notifying the Borrower, thereof in writing, if the Borrower fails to make the payments under the Agreement in due time and in opinion of UoM Bank the Borrower violates the terms and conditions of this Agreement.

Breakdown of loan:

Amount of Loan	50000 €
Term of loan	10 years
Total months	120
Annual Interest rate	6.5 % *
Monthly payment	567.74 €
Total Interest	18128.79 €
Total pay back	68128.79 €

On behalf of UoMBank,
Date: 11-01-2021
Tomas Mauber

Signed by Mrs./Mr. Nikolaos Nousias,

**Annual interest rate has been adjusted after projections of your financial status.*

Figure 128 - Proposed loan agreement document

Having thoroughly presented the way that the loan application process is performed from the end user's perspective, it is of utmost importance to examine what is actually happening behind the scenes.

"UoMBank's" perspective:

1 At first, once the loan application form is submitted, an automatic asynchronous REST API call to the workflow engine is triggered (Figure 129). In this regard, fetching the applicant's data and converting them to a JSON string variable (i.e., variable "data" in Figure 131)), the application is wrapped into the body of a POST REST API call, directing to the Camunda's engine endpoint. In turn, a new instance of the loan application-to-approval process is asynchronously generated in the backend system (Figure 130), namely the Camunda's Cockpit environment, while the front-end form and the client's session are still active. On process instance completion, a relative message is returned to the applicant.

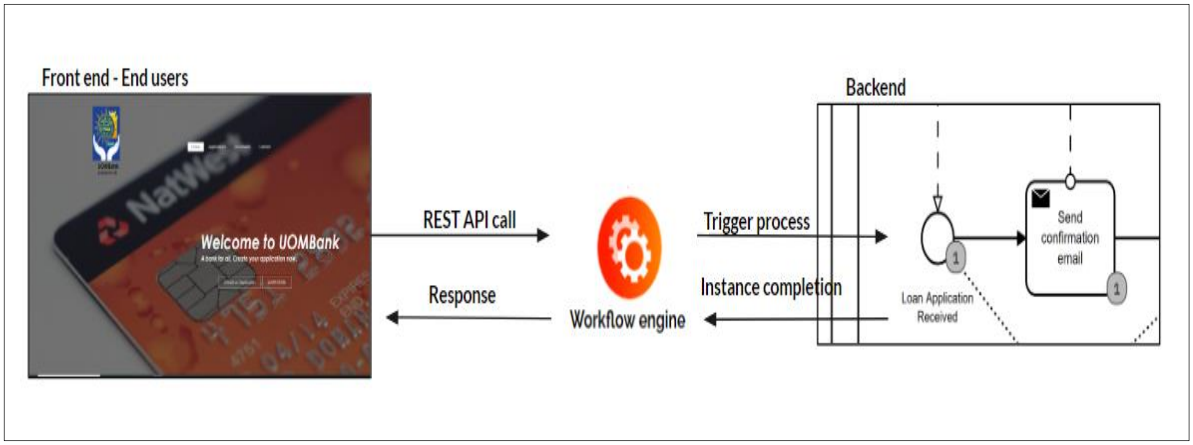


Figure 129 - Trigger a process instance

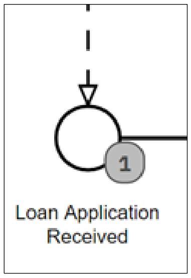
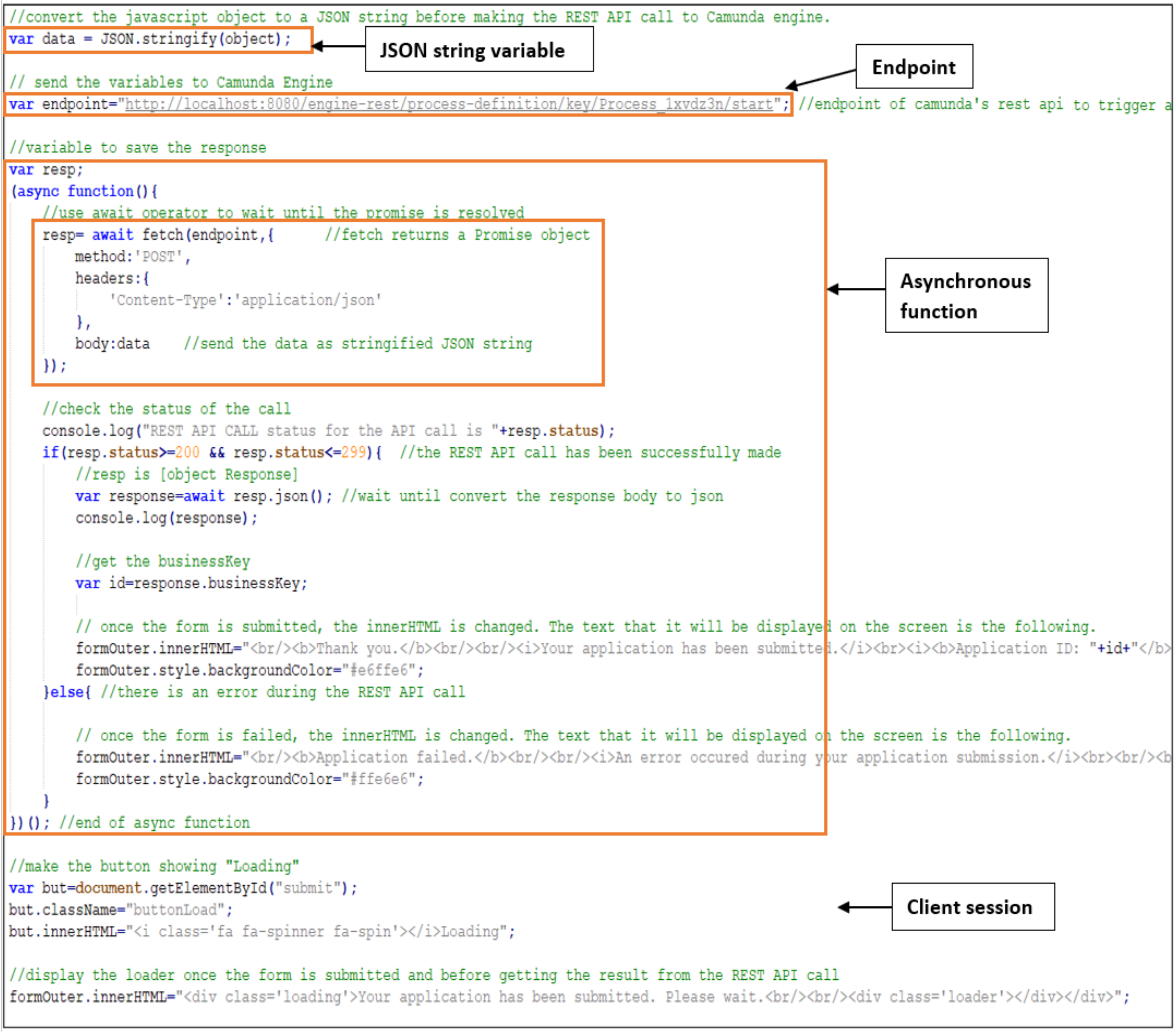


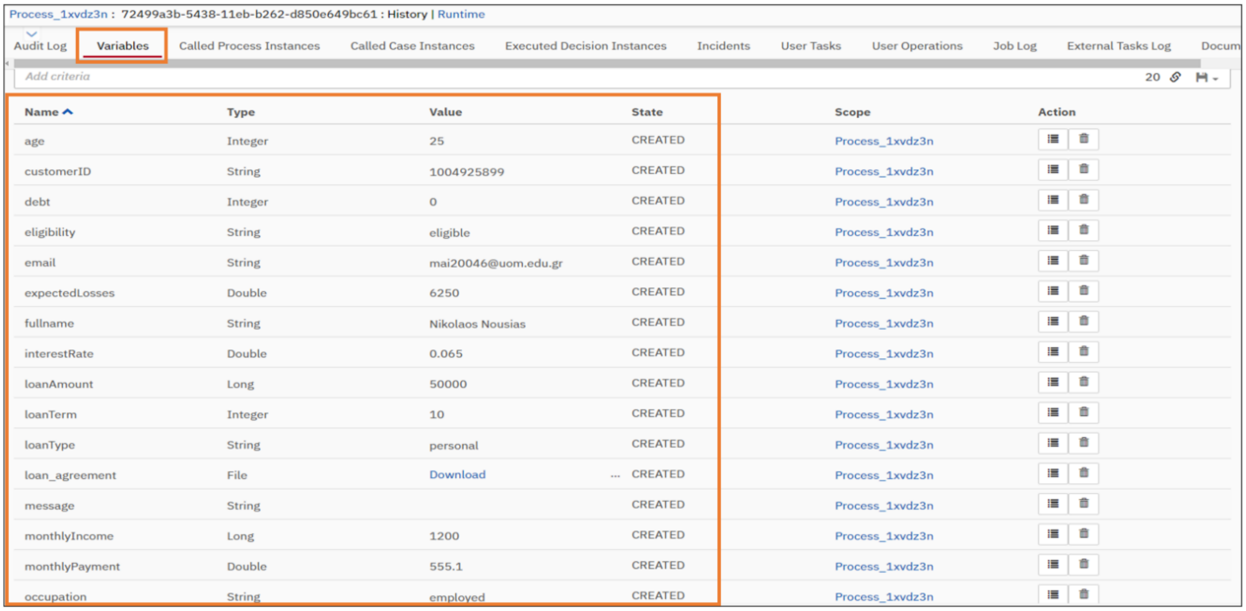
Figure 130 - Process instance generation



```
//convert the javascript object to a JSON string before making the REST API call to Camunda engine.
var data = JSON.stringify(object);

// send the variables to Camunda Engine
var endpoint="http://localhost:8080/engine-rest/process-definition/key/Process_1xvdz3n/start"; //endpoint of camunda's rest api to trigger a

//variable to save the response
var resp;
(async function(){
    //use await operator to wait until the promise is resolved
    resp= await fetch(endpoint,{       //fetch returns a Promise object
        method:'POST',
        headers:{
            'Content-Type':'application/json'
        },
        body:data    //send the data as stringified JSON string
    });

    //check the status of the call
    console.log("REST API CALL status for the API call is "+resp.status);
    if(resp.status>=200 && resp.status<=299){  //the REST API call has been successfully made
        //resp is [object Response]
        var response=await resp.json(); //wait until convert the response body to json
        console.log(response);

        //get the businessKey
        var id=response.businessKey;

        // once the form is submitted, the innerHTML is changed. The text that it will be displayed on the screen is the following.
        formOuter.innerHTML="<br/><b>Thank you.</b><br/><br/><i>Your application has been submitted.</i><br><i><b>Application ID: "+id+"</b>
        formOuter.style.backgroundColor="#e6ffe6";
    }else{ //there is an error during the REST API call

        // once the form is failed, the innerHTML is changed. The text that it will be displayed on the screen is the following.
        formOuter.innerHTML="<br/><b>Application failed.</b><br/><br/><i>An error occured during your application submission.</i><br><br/><b
        formOuter.style.backgroundColor="#ffe6e6";
    }
})(); //end of async function

//make the button showing "Loading"
var but=document.getElementById("submit");
but.className="buttonLoad";
but.innerHTML="<i class='fa fa-spinner fa-spin'></i>Loading";

//display the loader once the form is submitted and before getting the result from the REST API call
formOuter.innerHTML="<div class='loading'>Your application has been submitted. Please wait.<br/><br/><div class='loader'></div></div>";
```

Figure 131 - Asynchronous REST API call

2 Getting provided all the application details, present in the body of the REST API call, applicant's data are associated with process variables inside the backend system (Figure 132). Thus, all application-related information is available to "UoMBank" and are ready to be evaluated as the process unfolds.

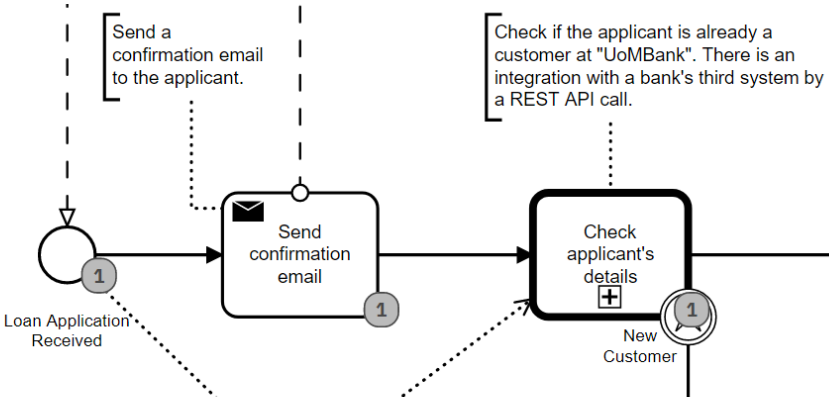


Process_1xvdz3n : 72499a3b-5438-11eb-b262-d850e649bc61 : History | Runtime

Name ^	Type	Value	State	Scope	Action
age	Integer	25	CREATED	Process_1xvdz3n	
customerID	String	1004925899	CREATED	Process_1xvdz3n	
debt	Integer	0	CREATED	Process_1xvdz3n	
eligibility	String	eligible	CREATED	Process_1xvdz3n	
email	String	mai20046@uom.edu.gr	CREATED	Process_1xvdz3n	
expectedLosses	Double	6250	CREATED	Process_1xvdz3n	
fullname	String	Nikolaos Nousias	CREATED	Process_1xvdz3n	
interestRate	Double	0.065	CREATED	Process_1xvdz3n	
loanAmount	Long	50000	CREATED	Process_1xvdz3n	
loanTerm	Integer	10	CREATED	Process_1xvdz3n	
loanType	String	personal	CREATED	Process_1xvdz3n	
loan_agreement	File	Download ...	CREATED	Process_1xvdz3n	
message	String		CREATED	Process_1xvdz3n	
monthlyIncome	Long	1200	CREATED	Process_1xvdz3n	
monthlyPayment	Double	555.1	CREATED	Process_1xvdz3n	
occupation	String	employed	CREATED	Process_1xvdz3n	

Figure 132 - Process variables

3 More specifically, a confirmation email is automatically forwarded to the applicant (Figure 133). Subsequently, a subprocess is triggered in order to check the applicant's existence on "UoMBank's" third system, namely the "ServiceNow" system, with the aim to retrieve the previous amount of debt in case the applicant is already an existing "UoMBank's" customer.

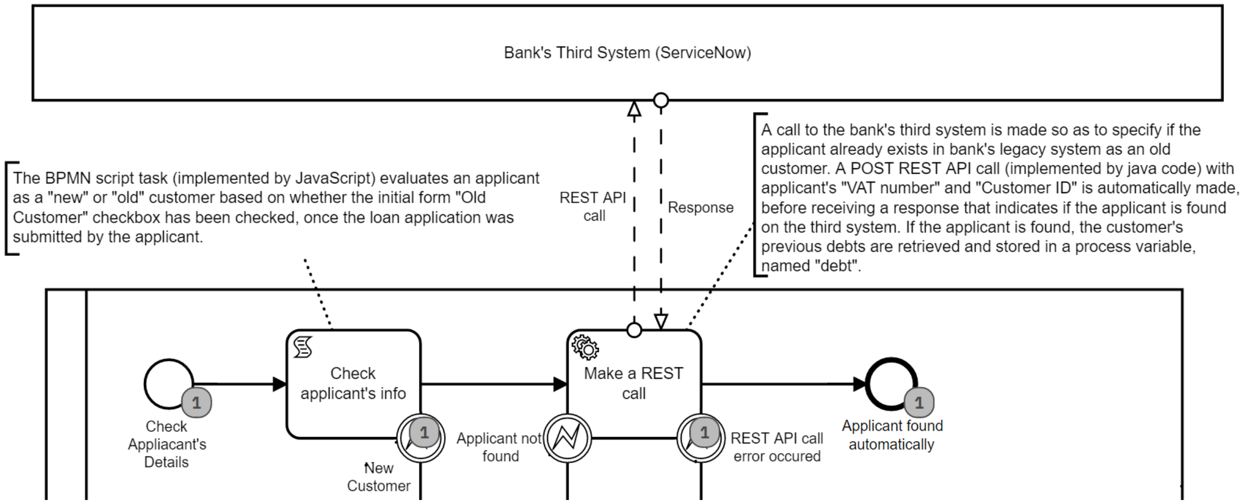


Send a confirmation email to the applicant.
Check if the applicant is already a customer at "UoMBank". There is an integration with a bank's third system by a REST API call.
Send confirmation email
Check applicant's details
1
1
1
Loan Application Received
New Customer

Figure 133 - Process instance execution

4 Conducting, thus, an automatic REST API call to the third system (Figure 134), the applicant's existence is verified with a relative response. Interestingly, the amount of the customer's previous debt, stored in "UoMBank's" third system (Figure 135), is automatically retrieved,

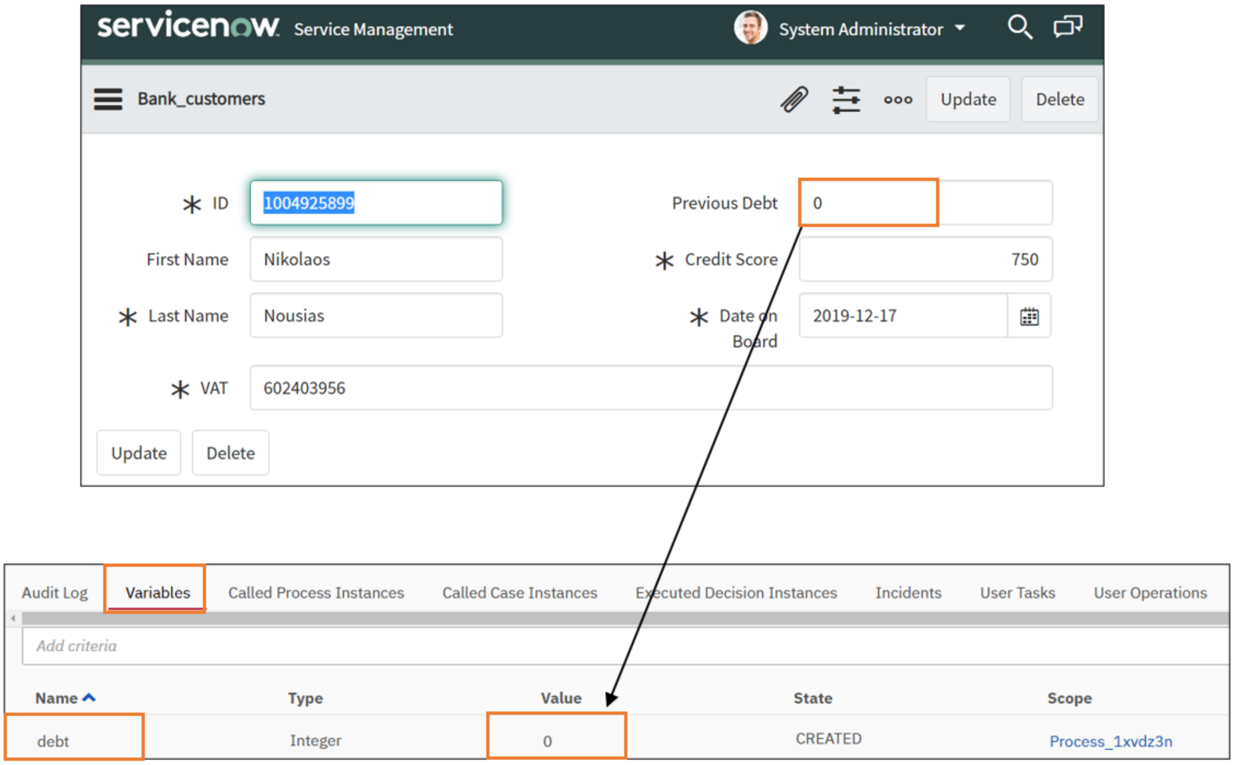
before being returned and assigned to a relative process variable within the backend system (i.e., variable "debt" (Figure 135)).


Bank's Third System (ServiceNow)
REST API call
Response
Check Appliacant's Details
Check applicant's info
New Customer
Applicant not found
Make a REST call
REST API call error occured
Applicant found automatically
1
1
1
1

The BPMN script task (implemented by JavaScript) evaluates an applicant as a "new" or "old" customer based on whether the initial form "Old Customer" checkbox has been checked, once the loan application was submitted by the applicant.

A call to the bank's third system is made so as to specify if the applicant already exists in bank's legacy system as an old customer. A POST REST API call (implemented by java code) with applicant's "VAT number" and "Customer ID" is automatically made, before receiving a response that indicates if the applicant is found on the third system. If the applicant is found, the customer's previous debts are retrieved and stored in a process variable, named "debt".

Figure 134 - REST API call to "UoMBank's" third system



* ID	1004925899	Previous Debt	0
First Name	Nikolaos	* Credit Score	750
* Last Name	Nousias	* Date on Board	2019-12-17
* VAT	602403956		

Name ^	Type	Value	State	Scope
debt	Integer	0	CREATED	Process_1xvdz3n

Figure 135 - Retrieving the applicant's previous debts

5 Having been successfully retrieved, the amount of previous debt and the applicant's age determine the continuation of the process execution. Considering that the underlying applicant has zero previous debts and a valid age, the process is further executed with an evaluation of applicant's eligibility to get a loan granted (Figure 136).

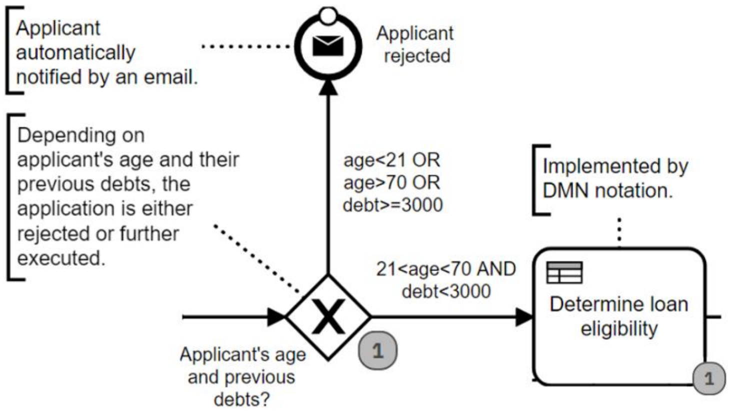
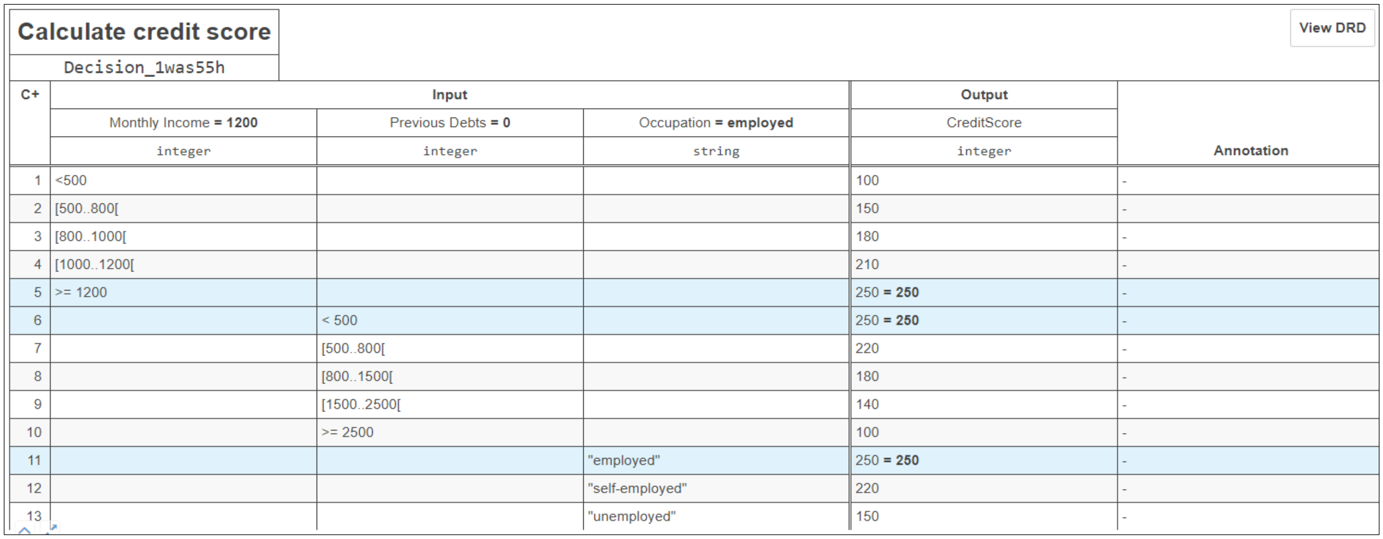

Applicant automatically notified by an email.
Applicant rejected
Depending on applicant's age and their previous debts, the application is either rejected or further executed.
age<21 OR age>70 OR debt>=3000
Implemented by DMN notation.
21<age<70 AND debt<3000
Determine loan eligibility
Applicant's age and previous debts?
1
1

Figure 136 – Routing decision based on applicant's age and previous amount of debt

6 Implemented by the DMN notation, "UoMBank" determines the applicant's eligibility after evaluating their credit score. In this regard, scoring the monthly income, the previous debts and the occupation type of the underlying applicant, a credit score of 750 is defined (Figure 137).



Calculate credit score

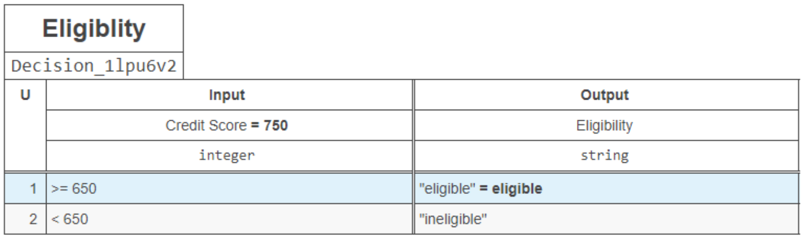
Decision_1was55h

View DRD

C+	Input			Output	
	Monthly Income = **1200**	Previous Debts = **0**	Occupation = **employed**	CreditScore	
	integer	integer	string	integer	**Annotation**
1	<500			100	-
2	[500..800[			150	-
3	[800..1000[			180	-
4	[1000..1200[			210	-
5	>= 1200			250 = **250**	-
6		< 500		250 = **250**	-
7		[500..800[		220	-
8		[800..1500[		180	-
9		[1500..2500[		140	-
10		>= 2500		100	-
11			"employed"	250 = **250**	-
12			"self-employed"	220	-
13			"unemployed"	150	-

Figure 137 – Calculating applicant's credit score

Considering that the applicant's credit score is further utilized as an input to a higher-level decision, where the applicant's eligibility is determined, the applicant is characterized as "eligible" due to a credit score greater than 650 (Figure 138). Importantly, the final decision outcome (i.e., "eligible") is returned back to the process execution and associated with a relative process variable (i.e., "eligibility") (Figure 139).

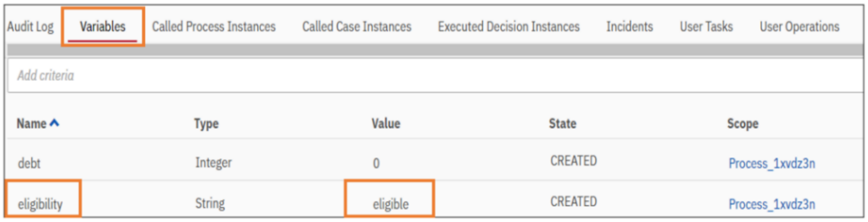


Eligiblity

Decision_1lpu6v2

U	Input	Output
	Credit Score = **750**	Eligibility
	integer	string
1	>= 650	"eligible" = **eligible**
2	< 650	"ineligible"

Figure 138 – Determining applicant's loan eligibility



Name ^	Type	Value	State	Scope
debt	Integer	0	CREATED	Process_1xvdz3n
eligibility	String	eligible	CREATED	Process_1xvdz3n

Figure 139 – Applicant's eligibility as a process variable

7 On the basis of customer's eligibility, an instant routing decision is performed in order to terminate or futher execute the process (Figure 140). In the current scenario, due to the applicant's eligibility to get a loan granted (i.e., eligibility=="eligible"), the process automatically proceeds to a Credit Risk evaluation.

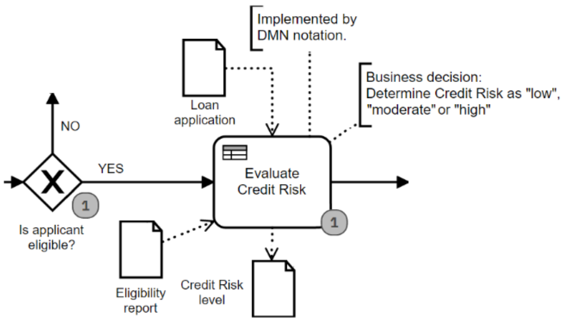
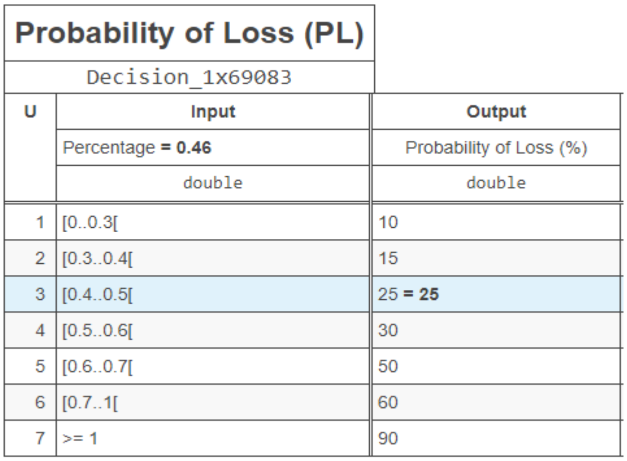


Figure 140 - Loan eligibility evaluation

8

At this stage, a series of decisions is automatically made, before the Credit Risk level is finally determined. Evaluating previously received input data (i.e., percentage of monthly loan payment to monthly income and requested loan type), the applicant's probability of loss (Figure 141) and the bank's loss given default (Figure 142) are automatically decided, before estimating the overall expected losses (Figure 143). Thereafter, calculating automatically the percentage of expected losses to the total requested loan amount (Figure 144), the Credit Risk level is finally determined (Figure 145), before being returned to the process execution and associated with a relative process variable (i.e., "creditRisk") (Figure 146).

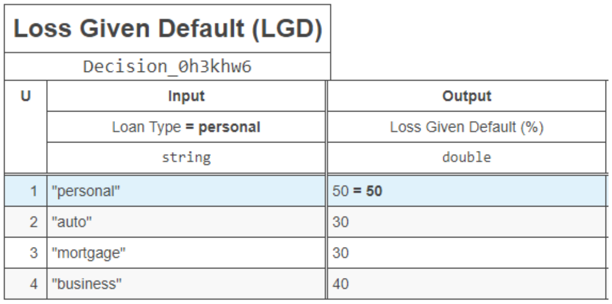


Probability of Loss (PL)

Decision_1x69083

U	Input	Output
	Percentage = **0.46**	Probability of Loss (%)
	double	double
1	[0..0.3[	10
2	[0.3..0.4[	15
3	[0.4..0.5[	25 = **25**
4	[0.5..0.6[	30
5	[0.6..0.7[	50
6	[0.7..1[	60
7	>= 1	90

Figure 141 - Estimating applicant's probability of loss

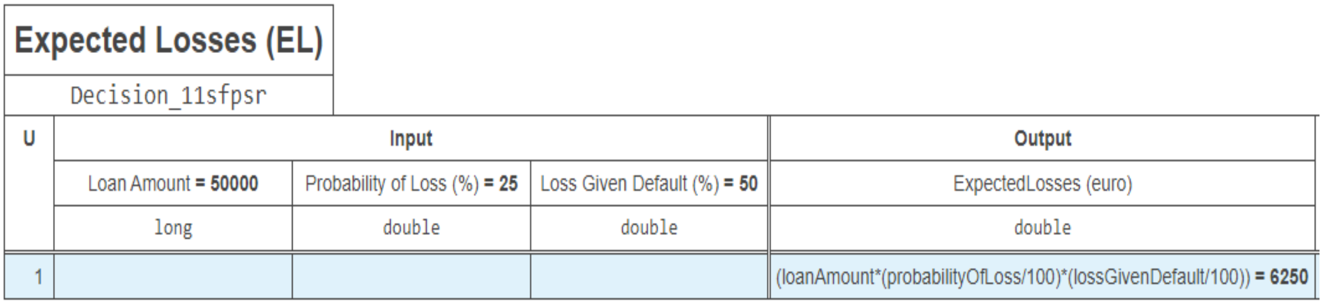


Loss Given Default (LGD)

Decision_0h3khw6

U	Input	Output
	Loan Type = **personal**	Loss Given Default (%)
	string	double
1	"personal"	50 = **50**
2	"auto"	30
3	"mortgage"	30
4	"business"	40

Figure 142 - Estimating "UoMBank's" loss given default

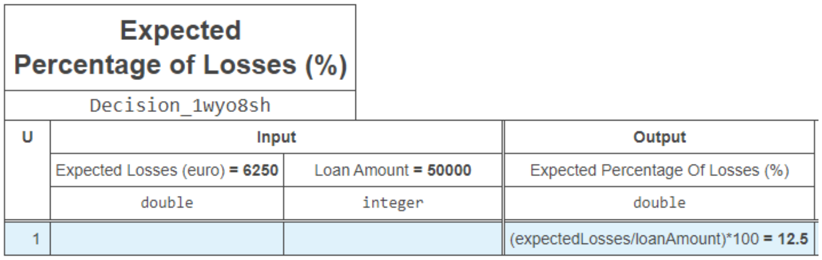


Expected Losses (EL)

Decision_11sfpsr

U	Input			Output
	Loan Amount = **50000**	Probability of Loss (%) = **25**	Loss Given Default (%) = **50**	ExpectedLosses (euro)
	long	double	double	double
1				(loanAmount*(probabilityOfLoss/100)*(lossGivenDefault/100)) = **6250**

Figure 143 - Estimating "UoMBank's" expected losses

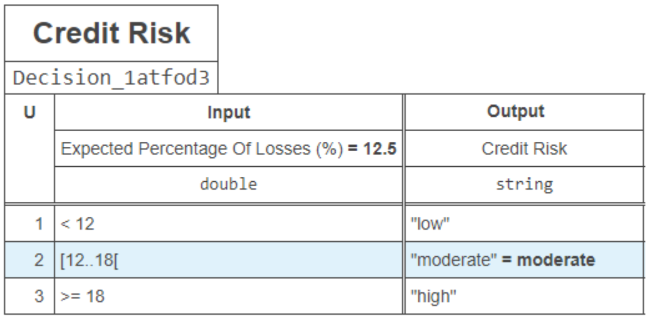


Expected Percentage of Losses (%)

Decision_1wyo8sh

U	Input		Output
	Expected Losses (euro) = **6250**	Loan Amount = **50000**	Expected Percentage Of Losses (%)
	double	integer	double
1			(expectedLosses/loanAmount)*100 = **12.5**

Figure 144 - Estimating "UoMBank's" percentage of expected losses

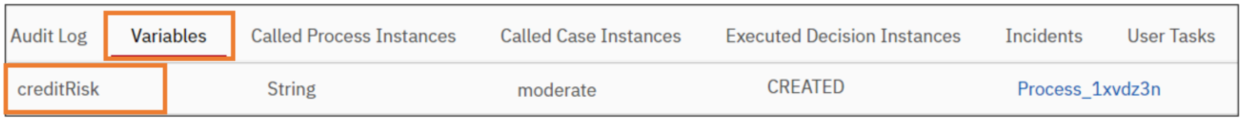


Credit Risk

Decision_1atfod3

U	Input	Output
	Expected Percentage Of Losses (%) = **12.5**	Credit Risk
	double	string
1	< 12	"low"
2	[12..18[	"moderate" = **moderate**
3	>= 18	"high"

Figure 145 - Estimating "UoMBank's" credit risk



creditRisk	String	moderate	CREATED	Process_1xvdz3n

Figure 146 - Credit risk as a process variable

9 On the basis of the previously estimated risk level, the flow of the process is automatically defined (Figure 147). In the current scenario, due to a moderate Credit Risk level, the interest rate is automatically readjusted based on a decided increase. Such decision, implemented by DMN notation (Figure 148), determines automatically the value by which the initially offered interest rate must be increased. For the underlying applicant, a 0.5 increase is decided, before the initially offered interest rate (i.e., 6.0% for the requested personal loan) being readjusted to its finally offered level (i.e., 6.5%) and associated with a relative process variable (Figure 149).

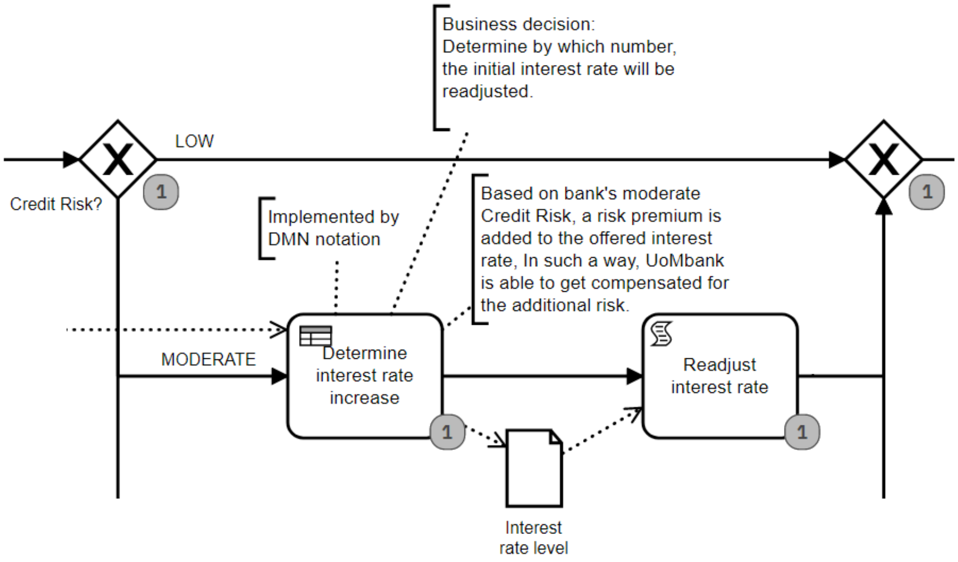
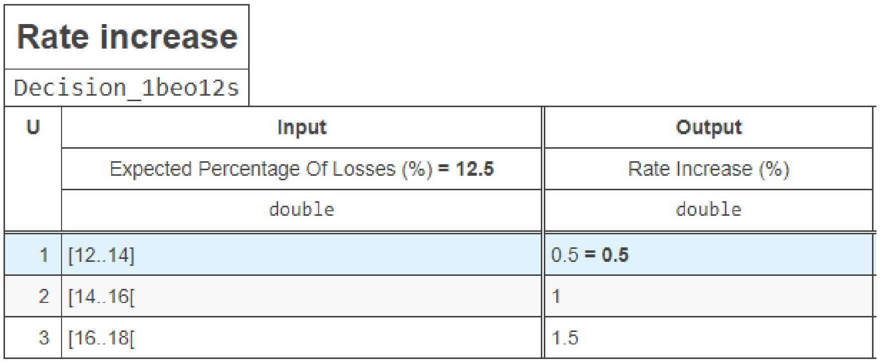


Figure 147 – Routing a process instance

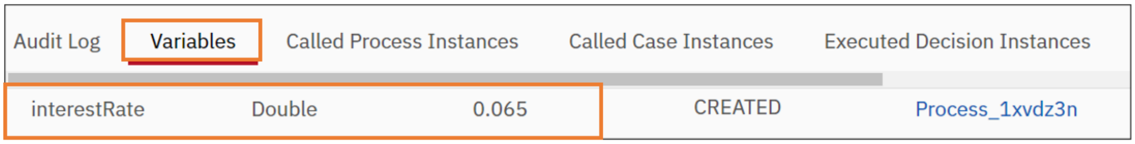


Rate increase

Decision_1beo12s

U	Input	Output
	Expected Percentage Of Losses (%) = **12.5**	Rate Increase (%)
	double	double
1	[12..14]	0.5 = **0.5**
2	[14..16[	1
3	[16..18[	1.5

Figure 148 - Interest rate readjustment



interestRate	Double	0.065	CREATED	Process_1xvdz3n

Figure 149 – Interest rate as a process variable

10 After the interest rate being readjusted, the loan agreement is automatically generated, before being attached to a forwarded to the applicant email (Figure 150).

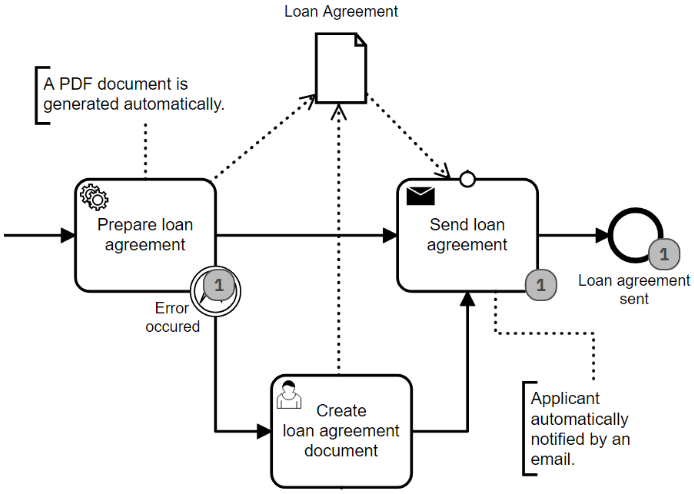

Loan Agreement
A PDF document is generated automatically.
Prepare loan agreement
1
Error occured
Send loan agreement
1
1
Loan agreement sent
Create loan agreement document
Applicant automatically notified by an email.

Figure 150 - Loan agreement sending

11 Once the loan agreement is automatically sent, the process lifecycle from the "UoMBank's" perspective is finished.

6.9 Expected benefits

Considering the end-to-end deployment of the loan application-to-approval process and the previously introduced execution scenario from both an applicant's and the "UoMBank's" perspective, the benefits of investing in workflow automation come in frontline. Undoubtedly, a workflow engine is able to accelerate a process execution, utilizing knowledge to route the process, as well as to transport work items, saving someone the time to even think about what should be done next [5]. Taking into account that a process running by a workflow engine is not only available as source code (XML, Java, etc.), but also as a diagram, workflow automation facilitates the transparency and the traceability of process instances, since the execution of the process instance is not buried deep in a software, but it is explicitly rendered in a graphical and intuitive manner [2]. Importantly, enforcing predefined rules and enact a process in a preordained way, workflow automation is running on the basis of the logic of a pre-deployed process model. Thus, error-rates are significantly decreased, while the process is executed without any concessions, avoiding situations where employees perform a business process in the way that it looks best to them [5]. Considering that lots of delays in business processes arising because of deficient decision-making processes, workflow automation brings decision-making in frontline, where faster and improved decisions can be made on the basis of ubiquitous information and its continuous circulation. Altogether, in today's rapidly changing environment that everything needs to be executed faster [65], workflow automation is able to increase the customer satisfaction and deliver high-value outcomes without affecting the offered quality.

6.10 Summary

This chapter presented an end-to-end deployment of a fictitious loan application-to-approval process in an imaginary bank, namely the "UoMBank". The fictitious process was developed in order to reflect real loan application-to-approval processes and consider how workflow automation can handle such processes. At first, a BPMN representation of the process was created, mitigating any ambiguities and misinterpretations arising from its free-form textual description. Identifying the decision logic inside the process model and externalizing the business decisions in separate DMN models, a BPMN and DMN integration approach, served as an input to the workflow automation initiatives. Thereafter, focusing on automating the process logic, the decision-making, the system integration and the activities execution, the execution semantics of the aforementioned models were established, paving the way for their automated enactment. Deploying the process model along with its all dependencies into the workflow engine, workflow automation comes in practice, where is deemed to accelerate such processes, refine traceability, decrease error rates, and increase customer satisfaction.

CHAPTER 7: Discussion & Conclusions

This chapter concludes the thesis, providing an overview of this research. Underlying the research contribution, the limitations and the future work that can push forward the research in the area of workflow automation, the chapter aims to provide the main findings and they key observations around the business process and decision automation, based on executable models, running by a workflow engine.

7.1 Thesis Overview

The aim of this thesis, as presented in Chapter 1, was to practically assess the workflow automation paradigm on the basis of executable notations, such as the BPMN and DMN notations. Given its practical scope, the purpose was to develop an end-to-end process, starting from its textual description and ending with its automated enactment by the means of a BPMN and DMN-based workflow engine.

Chapter 2 introduced the theoretical background of this thesis, denoting how the process thinking notion, the ubiquitous penetration of IT in business processes and the underlying digital transformation of contemporary organizations, have established the foundations of business process and decision automation. Aiming to establish the foundation behind the thesis's practical scope, the chapter presented the BPMN and DMN notations, shedding light on the Separation of Concerns (SoC) paradigm in business process modeling.

Chapter 3 presented the BPMN notation and its capabilities to render business processes in a graphical, yet executable way. The chapter introduced the complete BPMN symbol armory, before highlighting the ways that its various elements can be utilized within process models. Emphasizing advanced modeling concepts, such as the transaction subprocesses and the error-handling, the chapter presented advanced BPMN elements that are frequently omitted in BPMN modeling initiatives. The chapter concluded with the inability of BPMN notation to render the decision logic of business processes, giving prominence to the DMN notation for the modeling and the enactment of business decisions.

Chapter 4 introduced the DMN notation as the recent standard for rendering the decision logic of business processes in a graphical and executable way. The chapter presented its two levels of decision modeling, namely the decision requirements and the decision logic level, before illustrating the way that BPMN and DMN notations can be integrated. In this regard, the chapter introduced the Decision as a Service (DaaS) paradigm and the five principles for integrated

Process and Decision Modeling (5PDM), as presented in recent literature in order to render business processes decision-aware and decision-intelligent.

Chapter 5 presented the Business Process Management Systems (BPMSs) as the instruments for managing business processes, ranging from their modeling to their automated enactment. Introducing their origins, namely the Process Aware Information Systems (PAISs) and the Workflow Management Systems (WfMSs), special attention was given on their architecture and the ways that their components interact with each other. In this regard, a renowned BPMS, namely the Camunda BPM platform, was introduced, before being utilized for an end-to-end process deployment in the following chapter.

Chapter 6 fulfilled the practical scope of this thesis, presenting an end-to-end deployment of a loan application-to-approval process. Initially, the chapter presented the automation challenges in banking industry, summarized to acceleration and, ideally, automation of its bureaucratic processes. In this regard, a fictitious loan application-to-approval process was introduced, before its textual description being transformed to a BPMN process model. Subsequently, business decisions were identified in the process model and externalized to separate DMN models. Thereafter, specifying their execution semantics, the chapter highlighted how workflow automation comes in practice. In the end, the chapter concluded with an execution scenario of the underlying process, before presenting the expected benefits of workflow automation initiatives, summarized to accelerate business processes, refine traceability, decrease error rates and increase customer satisfaction.

7.2 Research contribution

The overall contribution of this research concerns the automation of an end-to-end business process, after specifying the execution semantics of its underlying BPMN model. The scope of the research focuses on identifying the automation capabilities of executable notations, such as the BPMN and DMN notations, that are directly executed by the workflow engine of a BPMS.

The research has provided an understanding about the business process spectrum within literature, as well as highlighted how business process and decision automation have emerged. In addition, it renders the complete power and expressiveness of BPMN notation, presenting complex processes' concepts, such as the transaction processes and error-handling, that can be explicitly modeled with advanced BPMN elements. In this sense, sophisticated elements that are frequently omitted in BPMN modeling initiatives, are thoroughly presented, aiming to elucidate their usage for the day-to-day engagement with the BPMN notation.

Moreover, this research demonstrates how business decisions can be separated from process models, and being treated as separate concerns in business process modeling. Providing a series of practical examples, it imparts methods of rendering business processes as decision-aware and decision intelligent.

Ultimately, this research can contribute to the already existing academic work around the workflow automation, by providing not only the theoretical concepts behind it, but also by introducing a practical example of a loan application-to-approval process. Considering the lack of practical examples of business processes within the literature, this research introduces an end-to-end process in the banking industry.

7.3 Research Limitations & Future work

A limitation of the underlying research derives from the fact that real business processes within the banking industry are not explicitly presented in literature. For this reason, a fictitious process at an imaginary bank, was introduced in order to tackle this limitation. However, considering the fact that real loan application-to-approval processes of contemporary banks are highly complex and involve many participants as the process lifecycle unfolds, the current research presents only a limited view on the process and decision automation of such processes. Additionally, taking into account that BPMN is mainly introduced for modeling purposes within literature, the establishment of its execution semantics in end-to-end processes has yet to be highly adopted within academic works.

However, considering the rise of process and decision automation, there is a plethora of opportunities and research challenges around the workflow automation initiatives. In this regard, hyped technologies, such as Machine Learning (ML) and Robotic Process Automation (RPA), can be seen as additional features of process and decision automation, rendering business processes fully-automated and eradicating excessively the human intervention. In this way, BPMN service tasks that in their background are implemented by programming code, can invoke a machine learning algorithm and leverage its knowledge, before utilize it for routing decisions and activities executions as the process lifecycle unfolds. In the same direction, RPA bots, constituting a powerful technology where screen-scrapping is required, can be invoked by BPMN models, automating activities that would otherwise be enacted by human actors. Taking into account that many legacy systems do not provide an interface which can be invoked automatically by the workflow engine of a BPMS, RPA bots automate highly-repetitive clerical work, reducing error-rates and execution cycles.

In addition, considering that more and more business processes become automated by the means of a workflow engine, research challenges arise on the context of managing automated processes. Given the increased number of process instances being executed, the monitoring of automated processes is significantly impeded. As a result, the exploitation and visualization of execution data that is explicitly gathered by a workflow engine of a BPMS, can be considered as a powerful instrument for monitoring the performance of automated processes. In this regard, establishing Key Performance Indicators (KPIs) related to the automation of a business process, can be considered as the main objective of a future work.

7.4 Conclusions

This thesis presented the workflow automation paradigm on the basis of executable notations, such as the BPMN and DMN notations. Considering that today's business conditions are ripe for major change, process and decision automation by the means of a workflow engine, provide the foundation for a robust process intelligence. Given the fact that business processes have to be performed in a preordained way, business processes executed by the process engine of a BPMS, leave less room for deviation from the desired process logic, which has been explicitly captured in an executable process model. Summarized to accelerate business processes, refine traceability, decrease error rates and increase customer satisfaction, workflow automation is deemed to constitute a first-class citizen in BPM initiatives, focusing on business processes that are characterized by high repetency, richness in information and standardization.

References

[1] W. M. P. Van der Aalst, “Re-engineering knock-out processes,” *Decision Support Systems*, vol. 30, no. 4, pp. 451–468, 2001, doi: 10.1016/S0167-9236(00)00136-6.

[2] J. Freund, B. Rücker, "Real-Life BPMN (4th edition): Includes an introduction to DMN," 2019

[3] D. Lübke and C. Pautasso, “Empirical Research in Executable Process Models,” in *Empirical Studies on the Development of Executable Business Processes*, D. Lübke and C. Pautasso, Eds. Cham: Springer International Publishing, 2019, pp. 3–12.

[4] M. Dumas, V. D. W. M.P. Aalst, and T. A. H.M. Hofstede, "*Process-aware information systems: bridging people and software through process technology," Wiley-Interscience,* 2005.

[5] M. Dumas, M. L. Rosa, J. Mendling, and H. Reijers, "*Fundamentals of Business Process Management*", 2nd ed. Berlin Heidelberg: Springer-Verlag, 2018.

[6] M. Lind, “Business process thinking in practice,” 1996.

[7] M. Kohlbacher, “The effects of process orientation: a literature review,” *Business Process Management Journal*, vol. 16, no. 1, pp. 135–152, Jan. 2010, doi: 10.1108/14637151011017985.

[8] H. A. Reijers, “Implementing BPM systems: the role of process orientation,” *Business Process Management Journal*, vol. 12, no. 4, pp. 389–409, Jan. 2006, doi: 10.1108/14637150610678041.

[9] W. Aalst, M. La Rosa, and F. Santoro, “Business Process Management: Don’t Forget to Improve the Process!,” *Business & Information Systems Engineering*, vol. 58, Oct. 2015, doi: 10.1007/s12599-015-0409-x.

[10] M. Weske, "*Business Process Management: Concepts, Languages, Architectures*", 2007.

[11] W. Aalst, “Aalst, W.M.P.: Business process management: a comprehensive survey". ISRN Softw. Eng. 1-37,” *ISRN Software Engineering*, Jan. 2012, doi: 10.1155/2013/507984.

[12] K. Kluza, K. Kaczor, G. J. Nalepa, and M. Ślażyński, “Opportunities for Business Process semantization in open-source process execution environments,” in *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Sep. 2015, pp. 1307–1314, doi: 10.15439/2015F250.

[13] A. Delgado, D. Calegari, P. Milanese, R. Falcon, and E. García, “A Systematic Approach for Evaluating BPM Systems: Case Studies on Open Source and Proprietary Tools,” in *Open Source Systems: Adoption and Impact*, Cham, 2015, pp. 81–90, doi: 10.1007/978-3-319-17837-0_8.

[14] M. Hammer, “What is Business Process Management?,” in *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*, J. vom Brocke and M. Rosemann, Eds. Berlin, Heidelberg: Springer, 2015, pp. 3–16.

[15] M. Lederer, J. Knapp, and P. Schott, “The digital future has many names—How business process management drives the digital transformation,” in *2017 6th International Conference on Industrial Technology and Management (ICITM)*, Mar. 2017, pp. 22–26, doi: 10.1109/ICITM.2017.7917889.

[16] J. Bloomberg, "The BPM Renaisssance", Intellyx 2020. [Online]. Available: https://intellyx.com/2020/03/17/the-bpm-renaissance/. [Accessed 3 Jan, 2021].

[17] A. Van Looy, “A quantitative and qualitative study of the link between business process management and digital innovation,” *Information and Management*, vol. 58, no. 2, 2021, doi: 10.1016/j.im.2020.103413.

[18] V. B. Vuksic, M. P. Bach, and O. Marjanovic, “Business Process Orientation in Croatian companies: A multi-site case study,” in *Proceedings of the ITI 2011, 33rd International Conference on Information Technology Interfaces*, Jun. 2011, pp. 51–58.

[19] K. Vergidis, “Business process optimisation using an evolutionary multi-objective framework,” Ph.D. dissertation, Cranfield University, 2008

[20] M. Hammer and J. Champy, “Reengineering the corporation: A manifesto for business revolution,” *Business Horizons*, vol. 36, no. 5, pp. 90–91, Sep. 1993, doi: 10.1016/S0007-6813(05)80064-3.

[21] K. Grolinger, M. A. M. Capretz, A. Cunha, and S. Tazi, “Integration of business process modeling and Web services: a survey,” *SOCA*, vol. 8, no. 2, pp. 105–128, Jun. 2014, doi: 10.1007/s11761-013-0138-2.

[22] D. Georgakopoulos, M. Hornick, and A. Sheth, “An overview of workflow management: From process modeling to workflow automation infrastructure,” *Distrib Parallel Databases*, vol. 3, no. 2, pp. 119–153, Apr. 1995, doi: 10.1007/BF01277643.

[23] A. Nikaj, M. Hewelt, and M. Weske, “Towards implementing REST-enabled business process choreographies,” *Lecture Notes in Business Information Processing*, vol. 320, pp. 223–235, 2018, doi: 10.1007/978-3-319-93931-5_16.

[24] M. Reichert and B. Weber, "*Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies,"* Berlin Heidelberg: Springer-Verlag, 2012.

[25] T. Davenport, "*Process Innovation: Reengineering Work through Information Technology*", 1993.

[26] I. Jacobson, M. Ericsson, and A. Jacobson, “The object advantage - business process reengineering with object technology,” 1994.

[27] E. A. Stohr and J. L. Zhao, “Workflow Automation: Overview and Research Issues,” *Information Systems Frontiers*, vol. 3, no. 3, pp. 281–296, Sep. 2001, doi: 10.1023/A:1011457324641.

[28] M. Chinosi and A. Trombetta, “BPMN: An introduction to the standard,” *Computer Standards & Interfaces*, vol. 34, no. 1, pp. 124–134, Jan. 2012, doi: 10.1016/j.csi.2011.06.002.

[29] J. Mendling, H. A. Reijers, and W. M. P. van der Aalst, “Seven process modeling guidelines (7PMG),” *Information and Software Technology*, vol. 52, no. 2, pp. 127–136, Feb. 2010, doi: 10.1016/j.infsof.2009.08.004.

[30] A. Lindsay, D. Downs, and K. Lunn, “Business processes—attempts to find a definition,” *Information and Software Technology*, vol. 45, no. 15, pp. 1015–1019, Dec. 2003, doi: 10.1016/S0950-5849(03)00129-0.

[31] D. Moody, “The ‘Physics’ of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering,” *IEEE Transactions on Software Engineering*, vol. 35, no. 6, pp. 756–779, Nov. 2009, doi: 10.1109/TSE.2009.67.

[32] J. Larkin and H. Simon, “Why a Diagram is (Sometimes) Worth Ten Thousand Words,” *Cogn. Sci.*, 1987, doi: 10.1111/j.1551-6708.1987.tb00863.x.

[33] K. Figl, J. Mendling, G. Tokdemir, and J. Vanthienen, “What we know and what we do not know about DMN,” *Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model.*, 2018, doi: 10.18417/EMISA.13.2.

[34] K. Batoulis, A. Meyer, E. Bazhenova, G. Decker, and M. Weske, “Extracting Decision Logic from Process Models,” in *Advanced Information Systems Engineering*, Cham, 2015, pp. 349–366, doi: 10.1007/978-3-319-19069-3_22.

[35] SmartDraw, LLC, [Online]. Available: https://www.smartdraw.com/. [Accessed Jan. 14,2021]

[36] H. Mili, G. Tremblay, G. Jaoude, E. Lefebvre, L. Elabed, and G. El-Boussaidi, “Business process modeling languages: Sorting through the alphabet soup.,” *ACM Comput. Surv.*, vol. 43, p. 4, Jan. 2010.

[37] R. Mayer, C. Menzel, M. Painter, P. Dewitte, T. Blinn, and P. Benjamin, “Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report,” Jan. 1995.

[38] J. Mendling and M. Nüttgens, "*Exchanging EPC Business Process Models with EPML*", 2004.

[39] N. Genon, P. Heymans, and D. Amyot, “Analysing the Cognitive Effectiveness of the BPMN 2.0 Visual Notation,” in *Software Language Engineering*, Berlin, Heidelberg, 2011, pp. 377–396, doi: 10.1007/978-3-642-19440-5_25.

[40] S. White. and D. Miers, “BPMN modeling and reference guide: understanding and using BPMN: develop rigorous yet understandable graphical representations of business processes”, 2008.

[41] H. Völzer, “An Overview of BPMN 2.0 and Its Potential Use,” 2010, doi: 10.1007/978-3-642-16298-5_3.

[42] M. Geiger, S. Harrer, J. Lenhard, and G. Wirtz, “BPMN 2.0: The state of support and implementation,” *Future Generation Computer Systems*, vol. 80, pp. 250–262, Mar. 2018, doi: 10.1016/j.future.2017.01.006.

[43] W. V. Aalst, A. Hofstede, B. Kiepuszewski, and A. Barros, “Workflow Patterns,” *Distributed and Parallel Databases*, 2004, doi: 10.1023/A:1022883727209.

[44] ISO/IEC,ISO/IEC 19510:2013 - Information technology- Object Management Group Business Process Model and Notation, November 2013, v2.0.2., [Online]. Available: https://www.omg.org/spec/BPMN/ISO/19510/PDF

[45] G. Aagesen and J. Krogstie, “BPMN 2.0 for Modeling Business Processes,” in *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*, J. vom Brocke and M. Rosemann, Eds. Berlin, Heidelberg: Springer, 2015, pp. 219–250.

[46] OMG, "Business process model and notation (BPMN 2.0)", [Online]. Available: https://www.omg.org/spec/BPMN/About-BPMN/

[47] C. Ouyang, W. Aalst, M. Dumas, and A. Ter, “From Business Process Models to Process-oriented Software Systems: The BPMN to BPEL Way,” Aug. 2009.

[48] P. Wohed, W. V. Aalst, M. Dumas, A. Hofstede, and N. Russell, “On the Suitability of BPMN for Business Process Modelling,” 2006, doi: 10.1007/11841760_12.

[49] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, “Workflow Patterns,” *Distributed and Parallel Databases*, vol. 14, no. 1, pp. 5–51, Jul. 2003, doi: 10.1023/A:1022883727209.

[50] L. Janssens, E. Bazhenova, J. Smedt, J. Vanthienen, and M. Denecker, “Consistent Integration of Decision (DMN) and Process (BPMN) Models,” 2016.

[51] F. Hasić, J. D. Smedt, and J. Vanthienen, “Redesigning Processes for Decision-Awareness: Strategies for Integrated Modelling,” in *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*, Sep. 2018, pp. 247–250, doi: 10.1109/QUATIC.2018.00043.

[52] L. Janssens, J. De Smedt, and J. Vanthienen, “Modeling and Enacting Enterprise Decisions,” in *Advanced Information Systems Engineering Workshops*, Cham, 2016, pp. 169–180, doi: 10.1007/978-3-319-39564-7_17.

[53] C. G. Schuetz, B. Neumayr, and M. Schrefl, “Multilevel Modeling for Business Process Automation,” in *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop*, Sep. 2015, pp. 51–60, doi: 10.1109/EDOCW.2015.30.

[54] F. Hasić, J. De Smedt, and J. Vanthienen, “Augmenting processes with decision intelligence: Principles for integrated modelling,” *Decision Support Systems*, vol. 107, pp. 1–12, Mar. 2018, doi: 10.1016/j.dss.2017.12.008.

[55] T. Biard, A. Le Mauff, M. Bigand, and J.-P. Bourey, “Separation of Decision Modeling from Business Process Modeling Using New ‘Decision Model and Notation’ (DMN) for Automating Operational Decision-Making,” in *Risks and Resilience of Collaborative Networks*, Cham, 2015, pp. 489–496, doi: 10.1007/978-3-319-24141-8_45.

[56] F. Hasić, E. Serral, and M. Snoeck, “Comparing BPMN to BPMN + DMN for IoT process modelling: a case-based inquiry,” *SAC*, 2020, doi: 10.1145/3341105.3373881.

[57] E. Bazhenova and M. Weske, “Optimal acquisition of input data for decision taking in business processes,” *SAC*, 2017, doi: 10.1145/3019612.3019615.

[58] OMG, "Decision Model and Notation (DMN 1.3)", [Online]. Available: https://www.omg.org/spec/DMN

[59] F. Hasić, J. De Smedt, and J. Vanthienen, “A Service-Oriented Architecture Design of Decision-Aware Information Systems: Decision as a Service,” in *On the Move to Meaningful Internet Systems. OTM 2017 Conferences*, Cham, 2017, pp. 353–361, doi: 10.1007/978-3-319-69462-7_23.

[60] M. Geiger, S. Harrer, J. Lenhard, M. Casar, A. Vorndran, and G. Wirtz, “BPMN Conformance in Open Source Engines,” in *2015 IEEE Symposium on Service-Oriented System Engineering*, Mar. 2015, pp. 21–30, doi: 10.1109/SOSE.2015.22.

[61] H. A. Reijers and W. M. P. van der Aalst, “The effectiveness of workflow management systems: Predictions and lessons learned,” *International Journal of Information Management*, vol. 25, no. 5, pp. 458–472, Oct. 2005, doi: 10.1016/j.ijinfomgt.2005.06.008.

[62] A. H. M. ter Hofstede, W. van der Aalst, M. Adams, and N. Russell, Eds., "*Modern Business Process Automation: YAWL and its Support Environment*," Berlin Heidelberg: Springer-Verlag, 2010.

[63] WfMC, "The Workflow Reference Model", WFMC-TC-1003, 1995, [Online]. Available: http://www.wfmc.org.

[64] M. Fares, A. Moufarrej, E. Jreij, J. Tekli, and W. Grosky, “Unsupervised word-level affect analysis and propagation in a lexical knowledge graph,” *Knowledge-Based Systems*, vol. 165, pp. 432–459, Feb. 2019, doi: 10.1016/j.knosys.2018.12.017.

[65] A. T. Leon, “Framework for the Automation of Business Processes,” *International Journal of Systems Engineering*, vol. 4, no. 1, Art. no. 1, May 2020, doi: 10.11648/j.ijse.20200401.11.

[66] S. Mohapatra, “BPR and Automation,” in *Business Process Reengineering: Automation Decision Points in Process Reengineering*, S. Mohapatra, Ed. Boston, MA: Springer US, 2013, pp. 213–219.

[67] S. Ahmad Sirohey, A. I. Hunjra, and B. Khalid, “Impact of Business Process Automation on Employees’ Efficiency,” Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 3228559, 2012.

[68] D. Etinger, S. D. Simic, and L. Buljubašic, “Automated decision-making with DMN: From decision trees to decision tables,” 2019, pp. 1309–1313, doi: 10.23919/MIPRO.2019.8756694.

[69] M. Castellanos, F. Casati, U. Dayal, and M.-C. Shan, “A Comprehensive and Automated Approach to Intelligent Business Processes Execution Analysis,” *Distributed and Parallel Databases*, vol. 16, no. 3, pp. 239–273, Nov. 2004, doi: 10.1023/B:DAPD.0000031635.88567.65.

[70] Camunda, "The state of process automation", 2020, [Online]. Available: https://camunda.com/state-of-process-automation/. [Accessed Jan 23, 2021].

[71] S. L. Chan, “Information technology in business processes,” *Business Process Management Journal*, vol. 6, no. 3, pp. 224–237, Jan. 2000, doi: 10.1108/14637150010325444.

[72] L. Hrustek, M. T. Furjan, and I. Pihir, “Influence of Digital Transformation Drivers on Business Model creation,” in *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2019, pp. 1304–1308, doi: 10.23919/MIPRO.2019.8756666.

[73] T. Clohessy, T. Acton, and L. Morgan, “The Impact of Cloud-Based Digital Transformation on ICT Service Providers’ Strategies,” 2017, doi: 10.18690/978-961-286-043-1.9.

[74] P. Soto-Acosta, “COVID-19 Pandemic: Shifting Digital Transformation to a High-Speed Gear,” *Information Systems Management*, vol. 37, no. 4, pp. 260–266, Oct. 2020, doi: 10.1080/10580530.2020.1814461.

[75] Camunda Documentation, [Online]. Available: https://docs.camunda.org. [Accessed 10 Aug, 2020]

[76] S. Jablonski and C. Bussler, "*Workflow Management: Modeling Concepts, Architecture, and Implementation,"* 1996.

[77] E. Bazhenova, F. Zerbato, B. Oliboni, and M. Weske, “From BPMN process models to DMN decision models,” *Information Systems*, vol. 83, pp. 69–88, Jul. 2019, doi: 10.1016/j.is.2019.02.001.

[78] F. Hasić, J. De Smedt, and J. Vanthienen, “An Illustration of Five Principles for Integrated Process and Decision Modelling (5PDM),” Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 3082752, Sep. 2017. doi: 10.2139/ssrn.3082752.

[79] K. Jensen and W. van der Aalst, Eds., "*Transactions on Petri Nets and Other Models of Concurrency II: Special Issue on Concurrency in Process-Aware Information Systems*," Berlin Heidelberg: Springer-Verlag, 2009.

[80] S. Nurcan, K. Gaaloul, R. Schmidt, S. Guerreiro, and Q. Ma, "*Enterprise, Business-Process and Information Systems Modeling", 16th International Workshop, BPMDS 2015, and 20th International Conference, EMMSAD 2015, held at CAiSE 2015, Stockholm, Sweden, June 8-9, 2015. Proceedings*, vol. 214. 2015.

[81] W. van der Aalst and K. van Hee, “Workflow Management: Models, Methods, and Systems,” The MIT Press, MIT Press Books, 2004.

[82] V. Ferme, A. Ivanchikj, C. Pautasso, M. Skouradaki, and F. Leymann, “IT-Centric Process Automation: Study About the Performance of BPMN 2.0 Engines,” in *Empirical Studies on the Development of Executable Business Processes*, D. Lübke and C. Pautasso, Eds. Cham: Springer International Publishing, 2019, pp. 167–197.

[83] Camunda, [Online]. Available: https://camunda.com/. [Accessed 10 Aug, 2020]

[84] TechCrunch, "Camunda hauls in $28M investment as workflow automation remains hot", [Online]. Available: https://techcrunch.com/2018/12/05/camunda-hauls-in-28m-investment-as-workflow-automation-remains-hot/. [Accessed Jan 25, 2021]

[85] Deloitte, " Technology Fast 500™ Europe, Middle East & Africa (EMEA) 2019 program and top-ranked companies", [Online]. Available: https://www2.deloitte.com/global/en/pages/technology-media-and-telecommunications/articles/technology-fast-500-emea.html. [Accessed Jan 25, 2021]

[86] Deloitte, "The fastest-growing technology companies in Germany", [Online]. Available: https://www2.deloitte.com/de/de/pages/technology-media-and-telecommunications/articles/fast-50-2019-germany-winners.html [Accessed Jan 25, 2021]

[87] V. Tornjanski, S. Marinkovic, G. Savoiu, and M. Čudanov, “A Need for Research Focus Shift: Banking Industry in the Age of Digital Disruption,” *Econophysics, Sociophysics & Other Multidisciplinary Sciences Journal (ESMSJ)*, vol. V, pp. 11–15, Jan. 2015.

[88] T. Ndlovu, A. Echchabi, M. Boulkeroua, E. Ndiweni, and W. Sibanda, “Digital technology disruption on bank business models,” *International Journal of Business Performance Management*, vol. 21, p. 184, Jan. 2020, doi: 10.1504/IJBPM.2020.10027639.

[89] Moody's Analytics, "Equity-at-Risk and Transfer Pricing: Annualised Expected Loss versus Cumulative Expected Loss", [Online]. Available: https://www.moodysanalytics.com/-/media/whitepaper/2018/annualised-vs-cumulative-el-11142018.pdf. [Accessed Dec 14, 2020]

[90] Capital, [Online]. Available: https://capital.com/expected-loss-definition. [Accessed Dec 14, 2020]

[91] "Apache Maven", *Wikipedia*, [Online]. Available: https://en.wikipedia.org/wiki/Apache_Maven. [Accessed Feb 17, 2021]

[92] Apache Software Foundation. "Apache Maven", [Online]. Available: https://maven.apache.org/. [Accessed Jan 25, 2021]

Appendix A: Creating a Maven project in Eclipse IDE

In the following lines, the creation of a Maven project in Eclipse IDE is meticulously rendered in order to optimally generate the loan application .war file. Navigating to the "Preferences" option (Figure 151) and searching for the "Maven" preference (Figure 152), the Camunda's Maven archetype can be established.

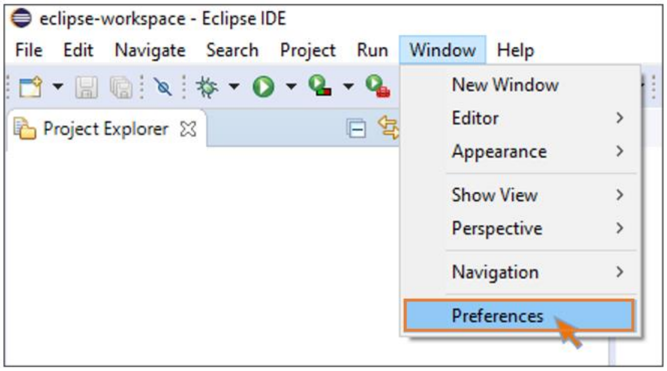
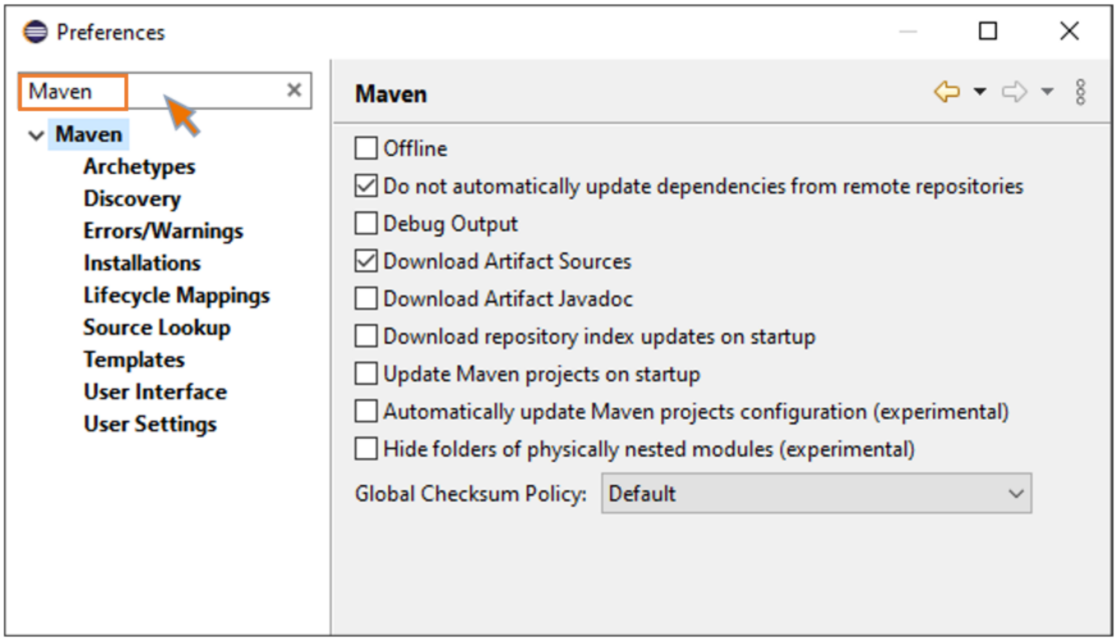


Figure 151 - Eclipse preferences



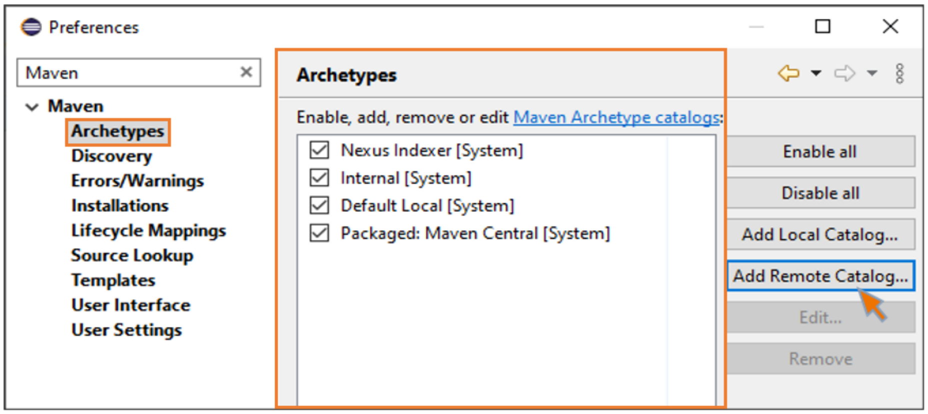
Maven

- □ Offline
- ☑ Do not automatically update dependencies from remote repositories
- □ Debug Output
- ☑ Download Artifact Sources
- □ Download Artifact Javadoc
- □ Download repository index updates on startup
- □ Update Maven projects on startup
- □ Automatically update Maven projects configuration (experimental)
- □ Hide folders of physically nested modules (experimental)

Global Checksum Policy: Default

Figure 152 - Maven preferences

Selecting the "Archetypes" option out of the Maven preferences, the list of available Maven Archetypes is displayed, while a remote repository can be additionally established, by clicking on the "Add Remote Catalog" button (Figure 153).

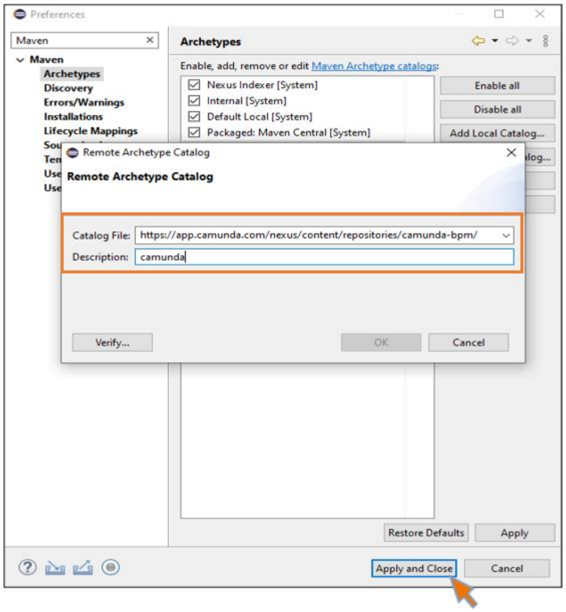


Enable, add, remove or edit Maven Archetype catalogs:

☑ Nexus Indexer [System]
☑ Internal [System]
☑ Default Local [System]
☑ Packaged: Maven Central [System]

Figure 153 - Maven archetypes

Utilizing the Camunda's repository URL as the catalog file (i.e., https://app.camunda.com/nexus/service/rest/repository/browse/camunda-bpm/), the Camunda's Maven Archetype is inserted (Figure 154).



Enable, add, remove or edit Maven Archetype catalogs:

☑ Nexus Indexer [System]
☑ Internal [System]
☑ Default Local [System]
☑ Packaged: Maven Central [System]

Catalog File: https://app.camunda.com/nexus/content/repositories/camunda-bpm/

Description: camunda

Figure 154 - Camunda Maven Archetype

Having applied the Camunda's archetype, a new Maven project can be initialized. Navigating to to the new "Project" option (Figure 155), a "Maven project" is established (Figure 156).

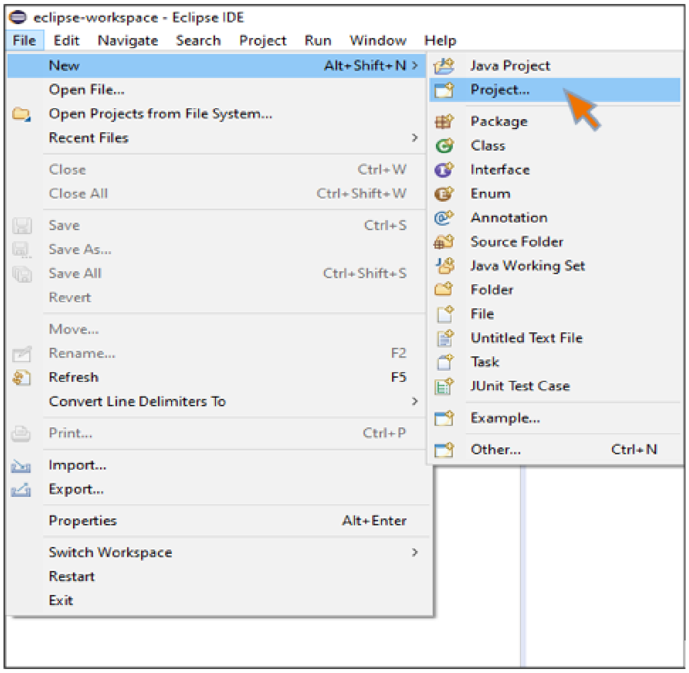
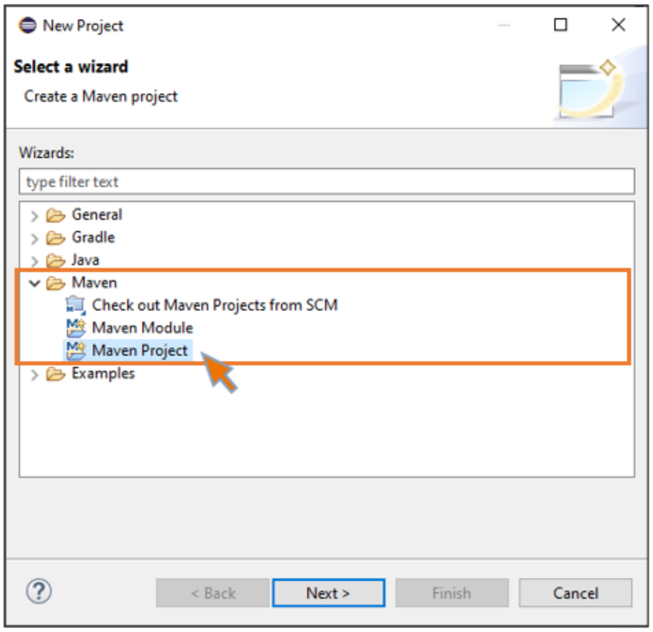


Figure 155 - New project

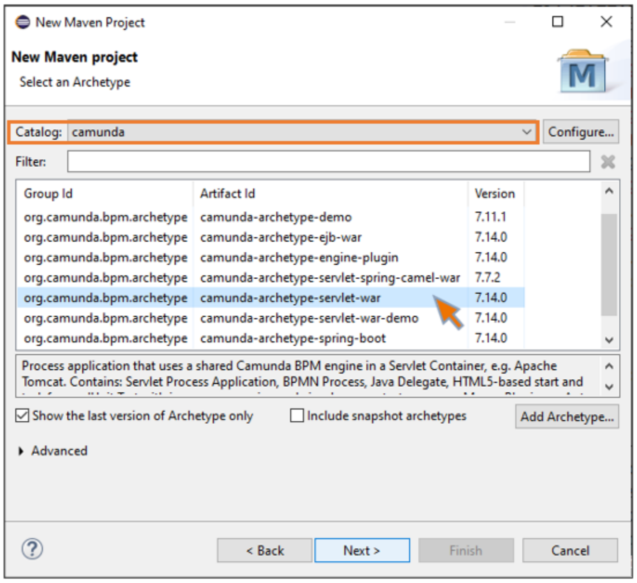

New Project

Select a wizard

Create a Maven project

Figure 156 - New Maven project

Selecting the previously inserted "camunda" archetype in the "catalog" drop-down list, a plethora of Artifact ids is rendered (Figure 157). Utilizing the "camunda-archetype-servlet-war" artifact id, the Maven project is created, after naming the project's Group and Artifact ids (Figure 158).



Catalog: camunda

Filter:

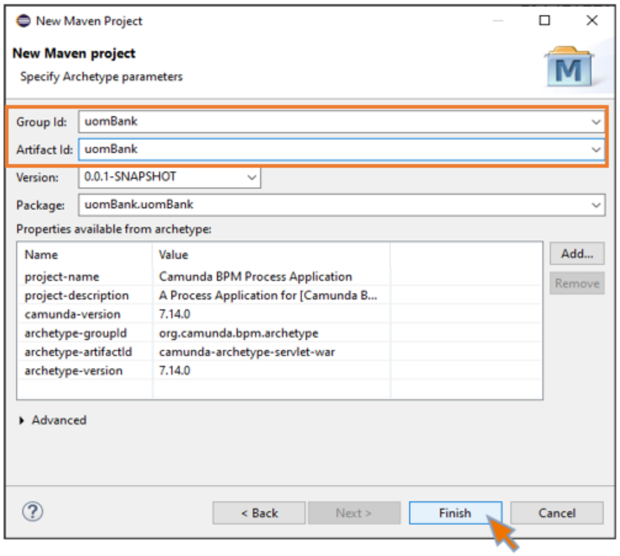
Group Id	Artifact Id	Version
org.camunda.bpm.archetype	camunda-archetype-demo	7.11.1
org.camunda.bpm.archetype	camunda-archetype-ejb-war	7.14.0
org.camunda.bpm.archetype	camunda-archetype-engine-plugin	7.14.0
org.camunda.bpm.archetype	camunda-archetype-servlet-spring-camel-war	7.7.2
org.camunda.bpm.archetype	camunda-archetype-servlet-war	7.14.0
org.camunda.bpm.archetype	camunda-archetype-servlet-war-demo	7.14.0
org.camunda.bpm.archetype	camunda-archetype-spring-boot	7.14.0

Process application that uses a shared Camunda BPM engine in a Servlet Container, e.g. Apache Tomcat. Contains: Servlet Process Application, BPMN Process, Java Delegate, HTML5-based start and

☑ Show the last version of Archetype only

□ Include snapshot archetypes

Figure 157 - Camunda's archetype



Group Id: uomBank

Artifact Id: uomBank

Version: 0.0.1-SNAPSHOT

Package: uomBank.uomBank

Properties available from archetype:

Name	Value
project-name	Camunda BPM Process Application
project-description	A Process Application for [Camunda B...
camunda-version	7.14.0
archetype-groupId	org.camunda.bpm.archetype
archetype-artifactId	camunda-archetype-servlet-war
archetype-version	7.14.0

Figure 158 - Maven project creation

Having efficaciously build the project, the new generated Maven project is listed under the "Project Explorer" section of Eclipse IDE (Figure 159).

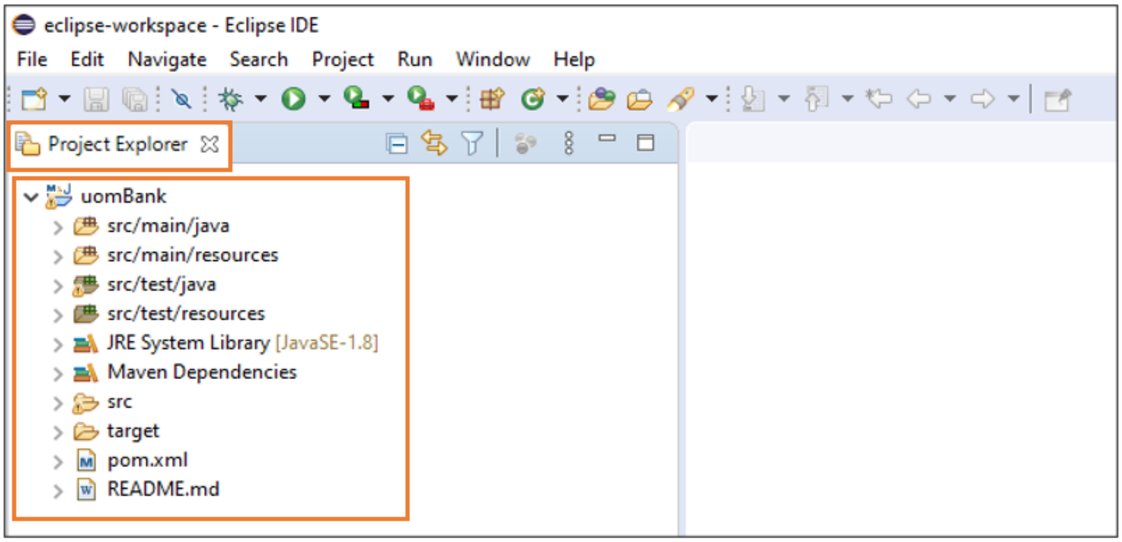


Figure 159 - Maven Project - Project Explorer

Appendix B: Installing and deploying the Maven project (.war file) into the Camunda BPM

In the following lines, the deployment of a Maven project as a .war file into the Camunda BPM ecosystem, is thoroughly rendered. All the executable process and decision models are inserted into the underlying Maven project, before being deployed as a single .war file into the application server. Right clicking on the project's name and selecting the "System explorer" view (Figure 160), the project is opened in the local system directory (Figure 161).

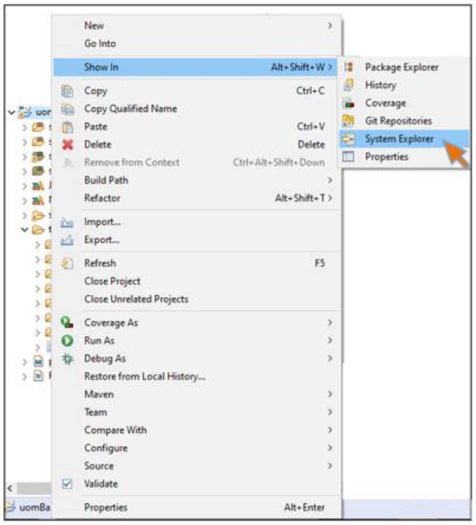
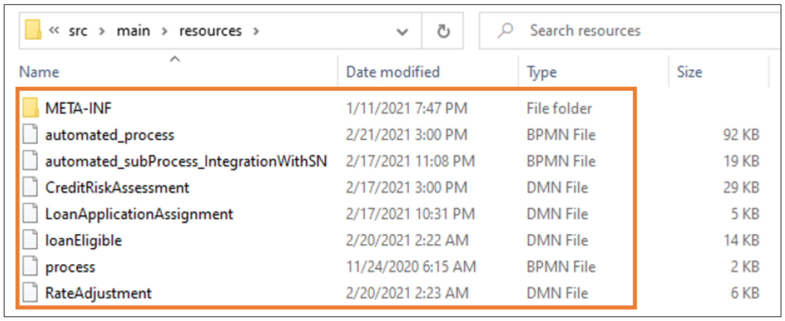


Figure 160 - System Explorer

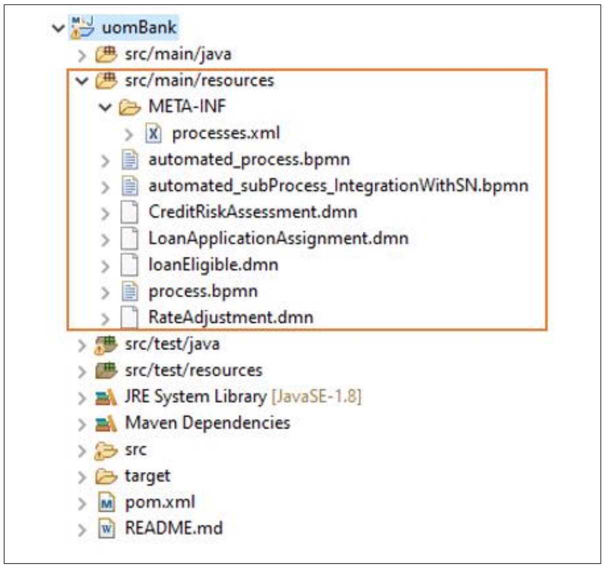
Navigating to the project's "resources" folder (i.e., path: projectName\src\main\resources), all BPMN and DMN files, indispensable to the process execution, are explicitly inserted (Figure 161).



Name	Date modified	Type	Size
META-INF	1/11/2021 7:47 PM	File folder	
automated_process	2/21/2021 3:00 PM	BPMN File	92 KB
automated_subProcess_IntegrationWithSN	2/17/2021 11:08 PM	BPMN File	19 KB
CreditRiskAssessment	2/17/2021 3:00 PM	DMN File	29 KB
LoanApplicationAssignment	2/17/2021 10:31 PM	DMN File	5 KB
loanEligible	2/20/2021 2:22 AM	DMN File	14 KB
process	11/24/2020 6:15 AM	BPMN File	2 KB
RateAdjustment	2/20/2021 2:23 AM	DMN File	6 KB

Figure 161 - Maven project in local directory

Returning to Eclipse IDE and refreshing the project (i.e., F5), the executable process and decision models are displayed under the Maven project's "resources" folder (i.e., path: src/main/resources) (Figure 162).



```
uomBank
  src/main/java
  src/main/resources
    META-INF
      processes.xml
    automated_process.bpmn
    automated_subProcess_IntegrationWithSN.bpmn
    CreditRiskAssessment.dmn
    LoanApplicationAssignment.dmn
    loanEligible.dmn
    process.bpmn
    RateAdjustment.dmn
  src/test/java
  src/test/resources
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  pom.xml
  README.md
```

Figure 162 - BPMN and DMN models in Maven project

Following the "Maven install" option, after right clicking on the project's name, the Maven project is installed (Figure 163).

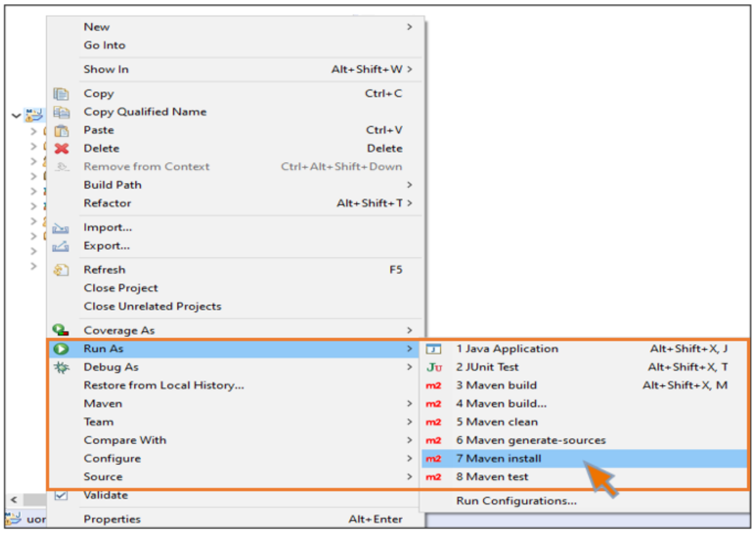
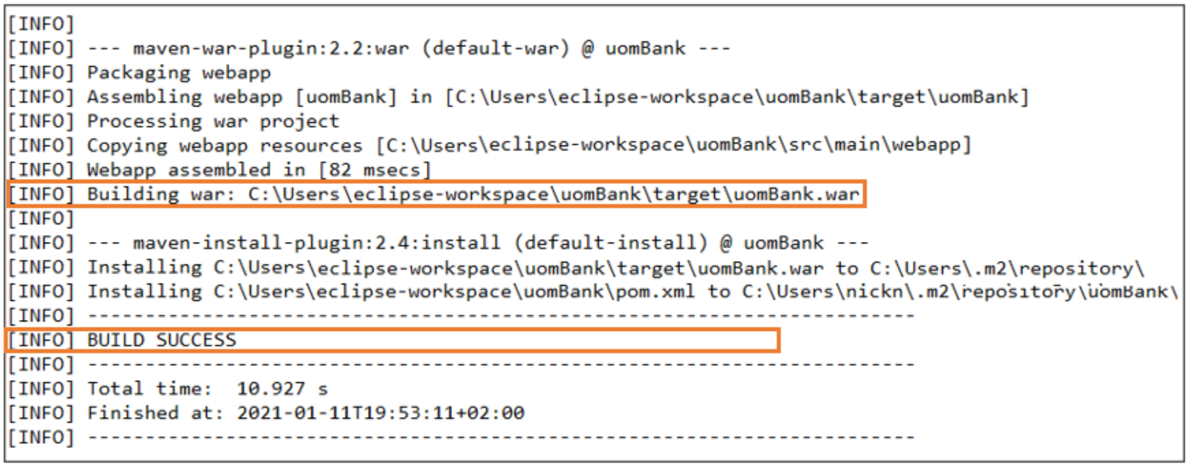


Figure 163 - Maven install

Having efficaciously installed the Maven project, a "BUILD SUCCESS" message is rendered (Figure 164), while the generated application file is listed under the project's "target" folder (Figure 165).



```
[INFO]
[INFO] --- maven-war-plugin:2.2:war (default-war) @ uomBank ---
[INFO] Packaging webapp
[INFO] Assembling webapp [uomBank] in [C:\Users\eclipse-workspace\uomBank\target\uomBank]
[INFO] Processing war project
[INFO] Copying webapp resources [C:\Users\eclipse-workspace\uomBank\src\main\webapp]
[INFO] Webapp assembled in [82 msecs]
[INFO] Building war: C:\Users\eclipse-workspace\uomBank\target\uomBank.war
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ uomBank ---
[INFO] Installing C:\Users\eclipse-workspace\uomBank\target\uomBank.war to C:\Users\.m2\repository\
[INFO] Installing C:\Users\eclipse-workspace\uomBank\pom.xml to C:\Users\nickn\.m2\repository\uomBank\
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  10.927 s
[INFO] Finished at: 2021-01-11T19:53:11+02:00
[INFO] ------------------------------------------------------------------------
```

Figure 164 - BUILD SUCCESS

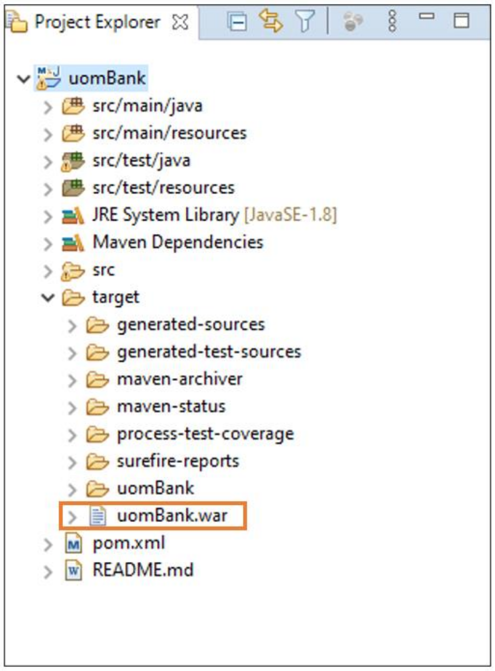
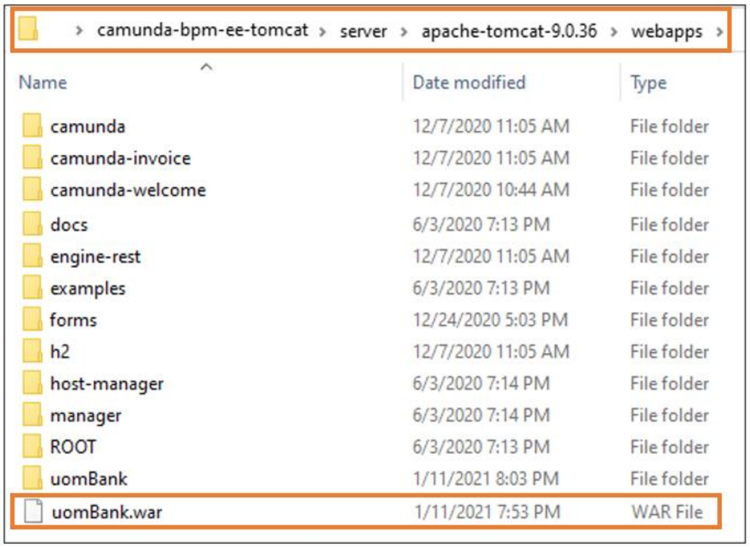


Figure 165 - Application .war file

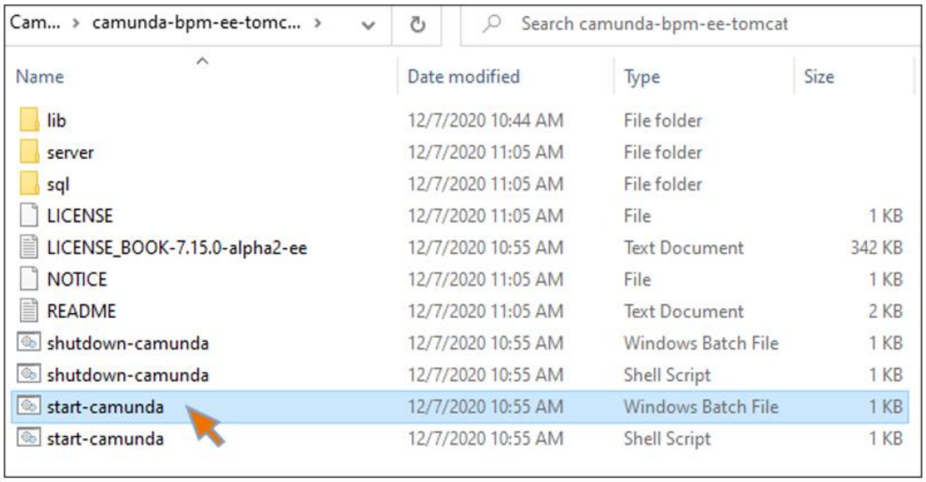
Navigating to Camunda’s server “webapps” folder (i.e., camunda-bpm\server\apache-tomcat-9.0.36\webapps), the previously generated application .war file is deployed and inserted into the server (Figure 166), before, subsequently, initializing the application server (Figure 167).



› camunda-bpm-ee-tomcat › server › apache-tomcat-9.0.36 › webapps ›

Name	Date modified	Type
camunda	12/7/2020 11:05 AM	File folder
camunda-invoice	12/7/2020 11:05 AM	File folder
camunda-welcome	12/7/2020 10:44 AM	File folder
docs	6/3/2020 7:13 PM	File folder
engine-rest	12/7/2020 11:05 AM	File folder
examples	6/3/2020 7:13 PM	File folder
forms	12/24/2020 5:03 PM	File folder
h2	12/7/2020 11:05 AM	File folder
host-manager	6/3/2020 7:14 PM	File folder
manager	6/3/2020 7:14 PM	File folder
ROOT	6/3/2020 7:13 PM	File folder
uomBank	1/11/2021 8:03 PM	File folder
uomBank.war	1/11/2021 7:53 PM	WAR File

Figure 166 - Application deployment



Cam... › camunda-bpm-ee-tomc... ›
Search camunda-bpm-ee-tomcat

Name	Date modified	Type	Size
lib	12/7/2020 10:44 AM	File folder	
server	12/7/2020 11:05 AM	File folder	
sql	12/7/2020 11:05 AM	File folder	
LICENSE	12/7/2020 11:05 AM	File	1 KB
LICENSE_BOOK-7.15.0-alpha2-ee	12/7/2020 10:55 AM	Text Document	342 KB
NOTICE	12/7/2020 11:05 AM	File	1 KB
README	12/7/2020 11:05 AM	Text Document	2 KB
shutdown-camunda	12/7/2020 10:55 AM	Windows Batch File	1 KB
shutdown-camunda	12/7/2020 10:55 AM	Shell Script	1 KB
start-camunda	12/7/2020 10:55 AM	Windows Batch File	1 KB
start-camunda	12/7/2020 10:55 AM	Shell Script	1 KB

Figure 167 - Starting the application server