Model Repository as a Service in a Model Analytics Workflow

Zioual, Anass

Award date:
2021

Department of Mathematics and Computer Science
Software Engineering and Technology

Model Repository as a Service in a Model Analytics Workflow

Master Thesis

Anass Zioual

Supervisors:
Prof. Dr. Mark van den Brand
Dr. Önder Babur
MSc. Mahdi Saeedi Nikoo

16-6-2021

Abstract

This thesis defines an architecture for a Model Analytics Automation System (MAAS) along with an implementation of a running prototype. MAAS is a piece of software running on top of the Arrowhead Framework as a set of several systems whose purpose are to collaborate to provide automation tasks in the model analytics workflow of researchers. The main functions of MAAS are to collect process models from software repositories through repository mining, validate collected process models against their metamodel, provide filtering options on their metadata, and offer operations to transform models to a target format. The presented architecture adopts the SOA-based local cloud concepts of the Arrowhead Framework, providing several advantages. On the one hand, it adds an abstraction layer enabling MAAS to be extended with additional services to automate additional steps in the model analytics workflow. On the other hand, the architecture permitted flexibility by incorporating several software design patterns. Hence, software engineers can easily extend the core services with additional services.

We evaluated MAAS through a case study taking a single-case, holistic-design. First, a group of researchers having a background in the Arrowhead Framework and model management and analytics used MAAS to collect and prepare a number of process models for analytics. Next, they were asked to fill in a modified Technology Acceptance Model (mTAM) questionnaire. Then, the participants were interviewed. Lastly, we conducted a benchmark comparing the manual approach with the automated approach using MAAS. We found that using MAAS significantly reduces the effort it takes to prepare models for analytics. Future studies recommend incorporating the Arrowhead choreographer system into MAAS and designing and implementing a distributed crawler, allowing horizontal scaling for increased workloads.

Acknowledgments

First and foremost, I would like to thank Dr. Önder Babur for his excellent guidance and support. He has been of tremendous value providing good feedback and innovating solutions for the problems we were trying to solve. I would also like to extend my gratitude to Mahdi Saeedi Nikoo for sharing his expertise and knowledge in the Arrowhead Framework throughout the project. I would further like to thank Prof. Dr. Mark van den Brand for his valuable feedback and supervising this thesis. Our discussions and meetings made this thesis possible.

Second, I would like to express my gratitude to Weslley Torres, who helped me structure the evaluation and gave helpful insights throughout the project. I am also grateful to Wan-Yi Tang for his valuable critique. And I would also like to thank the Arrowhead Framework team for their support and for building the Framework, which allowed me to work on this project.

Lastly, I would like to thank my mother and family for their continuous support and always being there for me.

This research is supported by ECSEL, the Electronic Components and Systems for European Leadership Joint Undertaking under grant agreement No 826452 (Arrowhead Tools project), supported by the European Union Horizon 2020 research and innovation programme and by the member states.

Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
BPM	Business Process Management
BPMN	Business Process Model Notation
CPF	Canonical Process Format
CRUD	Create, Read, Update, and Delete
DBMS	Database Management System
EPC	Event-driven Process Chain
EPML	EPC Markup Language
GUI	Graphical User Interface
MAAS	Model Analytics Automation System
MDE	Model-Driven Engineering
OMG	Object Management Group
REST	Representation State Transfer
SaaS	Software-as-a-Service
SOA	Service-Oriented Architecture
SoS	System of Systems
SQL	Structured Query Language
SUS	System Usability Scale
SysML	The Systems Modeling Language
TAM	Technology Acceptance Model
mTAM	modified TAM
NPS	Net Promoter Score
PU	Perceived Usefulness
PEU	Perceived Ease-of-Use
UML	Unified Modeling Language
URL	Uniform Resource Locator
YAWL	Yet Another Workflow Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language
XSD	XML Schema Definition

CHAPTER 1

Introduction

In this chapter, an introduction to the topic that motivates this thesis is given in Section 1.1. Afterwards, a definition of the problem scope is provided in Section 1.2, followed by the structure of the thesis in Section 1.3.

1.1 Motivation

Data analytics is an increasingly important tool for large organizations. Data analytics refers to the discovery, interpretation, and communication of meaningful patterns in data [47] (e.g., text, video, audio, etc.). It comprises the processes, tools, and techniques to examine data with the goal of gaining useful information. Creating business value from data is at an all-time high, and contemporary organizations are embedding analytics to transform information into insight in order to attain enterprise operational efficiency [33].

With model analytics, models used in various domains, such as in the engineering and business domains, are taken as input in the analytical process in order to improve the decision-making process of organizations [47]. Models are a simplified representation of specific parts of the considered domain. They leave out details irrelevant to a given set of criteria while preserving properties of interest concerning a set of requirements. In the engineering domain, models can relate to the intermediate artifacts used in order to realize a system. While in the business domain, models (also referred to as business process models) describe a set of activities achieving a common goal. Models can be specified by various modeling languages. A modeling language is an artificial language used to represent knowledge or information about a domain. They are defined in terms of their metamodels, which state how models are defined, and the rules and constraints their models must conform to. Examples of two widely used modeling languages are the Business Process Model Notation (BPMN) and the Unified Modeling Language (UML).

Similar to data analytics, the general model analytics workflow consists of collecting, preparing, and finally analyzing models and related modeling artifacts. Collection and preparation of large amounts of data for analysis are typical in many domains, such as mining biological data for bioinformatics [42], and mining software repositories for source code [56]. In particular, mining repositories for software artifacts has increased in a steady trend throughout the last years. On the one hand, currently available data mining techniques and tools have made it possible to retrieve large volumes of data from a rich and heterogeneous spectrum of domains. On the other hand, platforms such as GitHub, BitBucket, and GitLab which provide a large source of this data, have been increasing in popularity and usage [21].

Analyzing large volumes of modeling artifacts have been one of the main focuses in the research field. Several efforts have been initiated to mine public repositories for UML and BPMN models [21, 19]. Further steps have been initiated to store and manage large sets of models, and related artifacts [6, 31, 18]. However, there is a common lack of research in tools that unify the model collection and preparation techniques with today's model storage and management techniques. Moreover, there is limited research in automating the steps in model analytics. All issues above hinder researchers in performing empirical research in their respective areas. This thesis aims to address this issue by introducing an architecture and implementation of the Model Analytics Automation System (MAAS) running in the Arrowhead Framework as a set of collaborating systems, providing model storage and retrieval capabilities as well as an automatic approach in collecting and preparing models for analysis.

1.2 Project Scope and Constraints

The model analytics workflow encompasses various steps (see Section 2.1), ranging from collecting modeling artifacts, by leveraging techniques such as repository mining, to the actual analysis of the artifacts. Designing a system that completely automates the model analytics workflow is challenging and a time-consuming task. Considering the limitations of the master's thesis, it would not be possible to automate the analytical process of researchers completely. Therefore, it is necessary to select parts of the workflow in terms of their uniqueness as specific services provided by MAAS. Furthermore, the services should be useful to the stakeholders (researchers) of MAAS. This was discussed with the main stakeholder, and based on his proposal, four *core* services/components were selected:

- **Repository Mining.** A repository miner component used to mine software repositories such as GitHub for potential process models (e.g. BPMN and EPML).
- **Filtering and Validation.** The component provides services to filter process models on their metadata, and the ability to validate process models.
- **Model-to-Model Transformation.** The transformation of a process model from a source language to a target language. In particular, BPMN to EPML and the other way around.
- **Model Storage.** The ability to store and retrieve process models and related metadata.

This thesis will focus on the four selected services and will derive an architecture of the involved components. These components will be designed according to the findings presented in Chapter 4. Moreover, the architecture must be *generic* and *extensible*, supporting the automation of additional *steps* in the model analytical workflow of researchers by extending the system with additional services.

An external constraint imposed by the stakeholders on MAAS's architecture is that it has to run in the Arrowhead Framework as a system or systems. The framework provides fundamental functionality to efficiently support development, deployment, and operation of inter-connected, cooperative systems based on the Service-Oriented Architecture (SOA) [50]. The framework is further elaborated in Section 2.3.

1.3 Structure of the Thesis

The thesis is structured as follows:

- **Chapter 2** gives a detailed overview of the concepts regarding model management and analytics as well as an introduction to the Architecture of the Arrowhead Framework.
- **Chapter 3** describes the research design of this project and the steps to be undertaken throughout its execution.
- **Chapter 4** reviews related work available in the context of model repositories, repository mining, and model transformation.
- **Chapter 5** describes the goals, requirements, and the architecture of the model analytics automation system. First, we present the goals of the system. Next, an overview of the architecture is given. The overview includes some of the important components of the model analytics automation system.
- **Chapter 6** covers the implementation of the model analytics automation system based on the ideas discussed in the previous chapter. The construction of the implementation includes the design and implementation of the various subsystems that form the complete model analytics automation system.
- **Chapter 7** summarizes the evaluation of the initial prototype version of MAAS. First, the evaluation methodology is presented. Next, the evaluation results are given. Finally, a discussion and a summary of the evaluation is presented.
- **Chapter 8** concludes the thesis with a summary of the conclusions extracted from the research results. Furthermore, the chapter includes some notes about the future work.

CHAPTER 2

Background and Terminology

In this chapter, background information on model analytics and the Arrowhead Framework is provided and the terminology related to these areas. First, an introduction of data and model analytics is given in Section 2.1. The main components of the BPMN and EPC modeling languages are presented in Section 2.2. Lastly, the chapter concludes with a brief introduction to the Arrowhead Framework, including its architecture and main concepts in Section 2.3.

2.1 Data Analytics

Data analytics is the process of discovering, interpreting, and communicating meaningful patterns in data [47]. It comprises a broad selection of processes, tools, and techniques to examine data in order to gain useful information to improve the decision-making process of businesses.

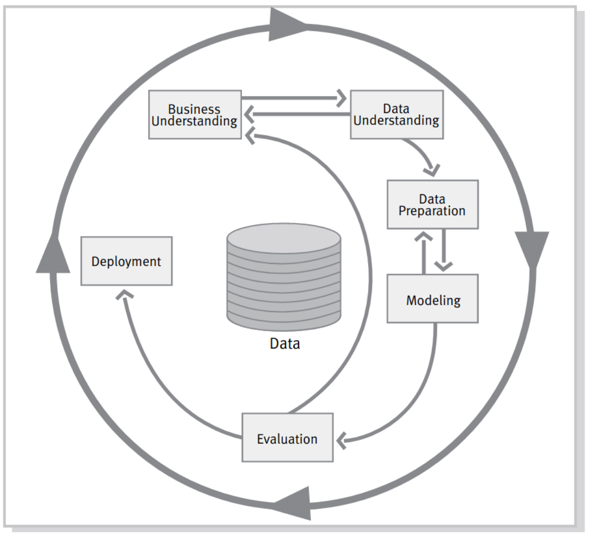


Figure 2.1: CRISP-DM data analysis reference model [38].

Figure 2.1 illustrates the CRISP-DM [38] data analysis model, which is a well known data analytics methodology used in a wide range of research domains. Data collection can be defined as: "the process of gathering and measuring information on variables of interest, which enables one to answer research questions, test hypotheses, and evaluate outcomes" [37]. One of the initial phases in the process is *Data understanding*. Throughout this phase, data is collected from various sources. The data understanding phase proceeds with the activities to get an initial understanding of the data. The phase is followed with *Data preparation*, comprising the steps needed to construct the final dataset. The steps include data cleaning, transformation, and selection of attributes and records required for the analysis. Consequently, a model or a collection of models using several modeling techniques are built to conduct the actual data analysis.

An example of a data analysis process is a team of researchers who want to perform relationship analysis between the different repositories on GitHub. Following the CRISP-DM model illustrated in Figure 2.1, one of the initial steps is data understanding. Researchers use a data collection technique, such as repository mining to collect raw data from software repositories. After collecting raw data, researchers prepare the final dataset through data cleaning, transformation, and feature selection to reduce the number of input attributes. Lastly, various statistical analysis techniques like linear regression can be used to build a model, which the researchers use to answer a set of research questions.

2.1.1 Model Analytics

As mentioned earlier, data analytics takes as source input data, such as text, audio, and video. Model analytics on the other hand takes models as input with the goal to extract useful information to improve the decision-making process of businesses. Model analytics can therefore be seen as a specialization of data analytics, with a particular focus on modeling artifacts [47]. Hence, the general activities found in data analytics, such as data collection, preparation, modeling can be applied in model analytics.

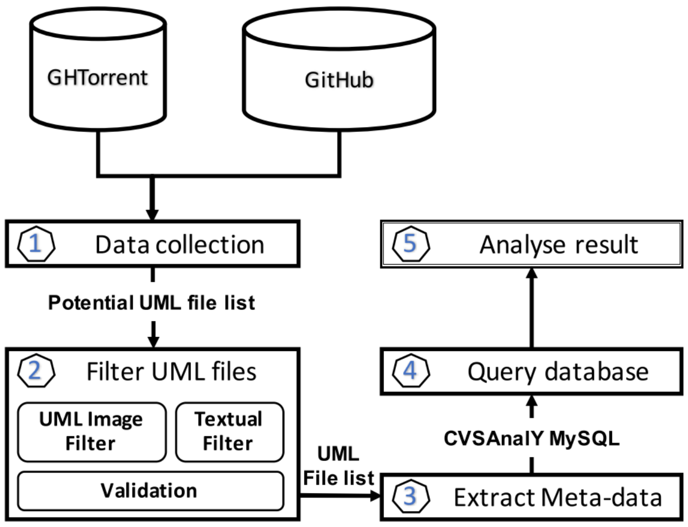


FIGURE 2.2: Overview of the UML collection and analysis process [19].

Figure 2.2 depicts the process used in [19] to collect and prepare large sets of UML models for analysis. As shown, the process consists of three general activities:

1. *Data collection.* A repository miner is used to mine GitHub for potential UML models.
2. *Filtering and Validation.* The collected models are validated if they contain UML notation.
3. *Data extraction and Analysis.* The steps are undertaken to prepare the final dataset in order to perform analysis.

In order to analyze models for useful information, we see that the general steps in the data analytics workflow are indeed applicable in model analytics. However, existing techniques more geared towards textual data, such as repository mining for source code, might not be directly applicable in this context.

2.2 Business Process Modeling Languages

A process modeling language can be defined as: "A language providing appropriate syntax and semantics to precisely specify business process requirements, in order to support automated process verification, validation, simulation and process automation" [34]. The aim of process modeling is to provide specifications independent from the implementation of such specifications at a general, abstract level [34].

There are two predominant types of process modeling formalisms, namely *graph-based* formalisms and *rule-based* formalisms [34]. Graph-based modeling languages leverage the concepts in graph theory such as nodes and edges in order to define process models. On the other hand, rule-based modeling languages are based on formal logic. In this thesis, we will focus our attention on two widely used graph-based modeling languages, namely BPMN and EPC.

2.2.1 BPMN: Business Process Model and Notation

Business Process Modeling Notation (BPMN) is a business process modeling methodology developed by the Business Process Modeling Initiative (BPMI) [52] and has been revised several times. The first version of the language (i.e. version 1.0) was released in 2003. At the moment of writing, the most recent version of the language is BPMN 2.0, released in 2011. This version introduced some notable changes, such as a metamodel that defines a set of rules and constructs required to create a particular BPMN process model and an interchange format in both XMI and XSD [51]. Furthermore, the language has been standardized by OMG [1] since version 1.1, released in 2008.

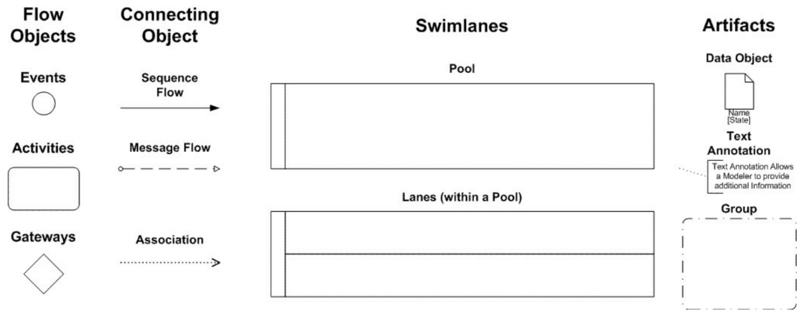


FIGURE 2.3: Basic elements of BPMN [39].

Figure 2.3 illustrates the basic elements of BPMN. BPMN provides five basic constructs for modeling, that is, *Data Objects*, *Flow Objects*, *Association Objects*, *Swim Lanes*, and *Artifacts* [26]. Furthermore, users can extend the *basic* constructs by designing artifacts composed of several basic elements. For instance, a sub-process that contains another pool or lane [49].

2.2.2 EPC: Event-driven Process Chain

Event-driven Process Chain (EPC) is a business process modeling language. The main language constructs of EPC are, *Functions*, *Events*, and *Connectors* [26]. Users can use the main constructs to model a certain business process. However, compatible tools extend the language with more elements, allowing the construction of more expressive business process models. The language supports various interchange formats. The most notable are the proprietary formats VDX and AML used by Visio and ARIS and the open-source and tool-independent format EPML. In this thesis, we will focus on the EPML format.

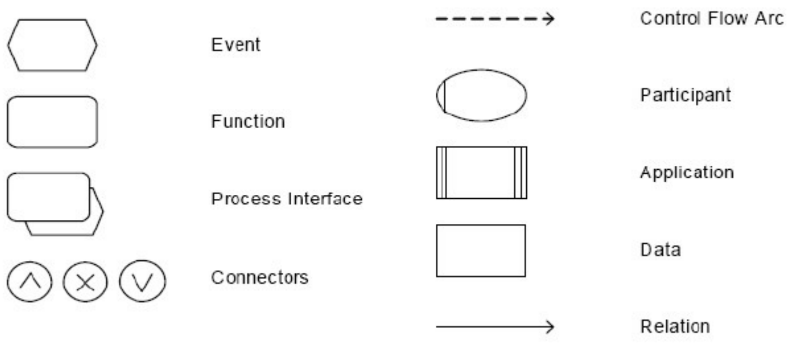


FIGURE 2.4: The core elements of EPC [25].

2.3 Arrowhead Framework

The Arrowhead Framework provides an architecture addressing the capability of building extensive Internet of Things (IoT) based systems built on the underlying principles of Systems of Systems (SoS) with an emphasis on [11]:

- Real-time data handling for low latency communication within automation systems.
- Security on automation systems and data.
- Automation system engineering to improve production efficiency and flexibility.
- Scalability of automation systems to support larger automation systems.

At the moment of writing, the most recent version of the framework is 4.2.0, and the project offers implementations of the framework in both the Java and C++ programming languages. The Java implementation is based on the Spring-Boot Framework for building RESTful web services [3]. Moreover, the Arrowhead Framework project provides client skeleton code for numerous programming languages, including: Java, C++, Python, NodeJS, and Kalix.

2.3.1 Service Oriented Architecture

The Service-Oriented Architecture (SOA) refers to a software design style where a distributed application or system is primarily built as a composition of several distinct *services* [45]. In the Arrowhead Framework, a service is utilized to exchange data between a providing system and a consuming system. Furthermore, the Arrowhead Framework is built around the principles of SOA [7]. This allows the framework to address the data interchangeability concerns between the systems and devices. This concept is further illustrated in Figure 2.5. (1) A service provider registers its services through the service registry, (2) allowing a service consumer to discover and utilize the provided service.

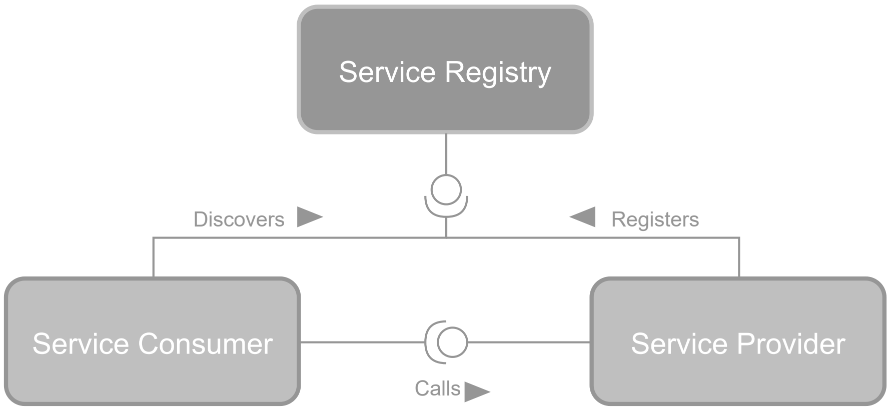


FIGURE 2.5: Service exchange between provider and consumer in SOA.

2.3.2 Systems and Devices

In the context of the Arrowhead Framework, a *system* is what is supposed to provide or consume services [7]. Furthermore, a system can be both a provider of one or multiple services and at the same time a consumer of one or multiple services. In the Arrowhead Framework, there is a strong distinction between software-based systems and hardware-based ones. To elaborate, a system is implemented as software running on hardware, which is referred to as a *device*. Stated differently, a software application responsible for providing or consuming services is called the system, whereas the hardware capable of hosting such software application is called the device. Furthermore, a device is capable of hosting multiple Arrowhead Framework systems.

2.3.3 System of Systems

A System of Systems (SoS) can be defined as [23]:

"large-scale integrated systems which are heterogeneous and independently operable on their own but are networked together for a common goal".

In the context of the Arrowhead Framework, an SoS consists of a set of systems, which are governed by the core Arrowhead local cloud systems and collaborate to perform a certain automation task [7]. Therefore, a local cloud is also an SoS in this context. The collaboration can also be extended to multiple local clouds that exchange services and information between each other, effectively providing a solution for more complex automation tasks.

2.3.4 Local Cloud

A *local cloud* can be viewed as a self-contained network, which may include systems and devices to perform certain largely independent automation tasks. Each local cloud is materialized by including the three mandatory Arrowhead *core* systems and at least one application system. A key characteristic of the local cloud is security from inbound and outbound network traffic from the open internet. A local cloud provides a boundary, aiming to "protect" the interior of the local cloud comprising the systems and devices from external network traffic [7].

In a local cloud, systems collaborate to perform a particular automation task. Communication between devices within a local cloud is also referred to as *intra-cloud* communication. A local cloud is mainly characterized as a set of systems that collaborate to form one complex automation system. Therefore, a local cloud essentially becomes an SoS [7]. Moreover, through *inter-cloud* communication, that is when two systems reside in two different local clouds and exchange information, this also becomes an SoS.

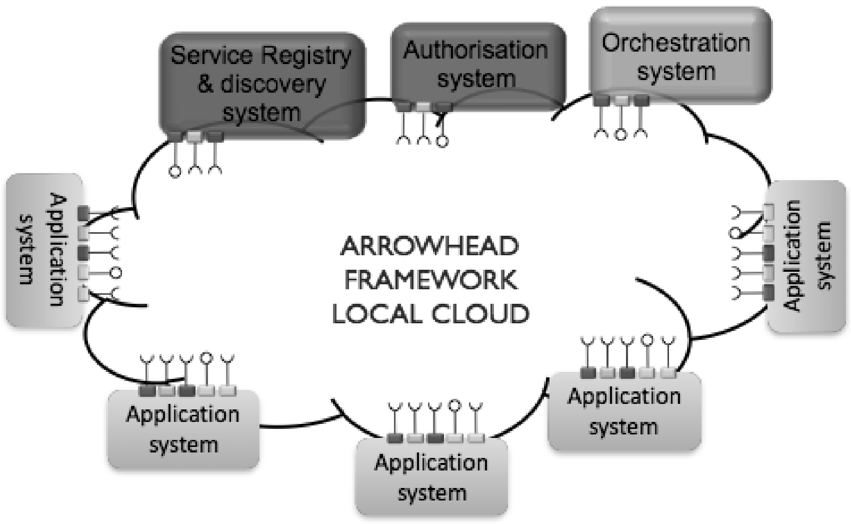


FIGURE 2.6: A simple example of a local cloud [7].

2.3.5 Core Systems

In the Arrowhead local cloud, systems need to utilize several mandatory core Arrowhead components before they can exchange services and data [7] among each other. In the case of a service provider system, it needs to utilize the *ServiceRegistry* and *Authorization* core systems. Whereas the service consumer system needs to utilize all three core systems. The three core systems of a local cloud are:

- *Authorization.* The Authorization component enables the authentication and authorization of services consumers given a set of rules. A service provider can determine if a service consumer can use the provided service based on these rules.
- *ServiceRegistry.* The ServiceRegistry component aims to provide storage capabilities for the currently active services registered to the local cloud. It also enables the services to be discoverable for the service consumers.
- *Orchestration.* The Orchestration component aims to enable the re-use of existing services to compose new services and functionality. Moreover, the component also

aims to provide advanced service discovery by discovering and pairing the service providers and consumers.

CHAPTER 3

Research Design

As we shall see Chapter 4 the problem we find in the state of the art is that all features regarding model collection, preparation, and storage are scattered among different tools and platforms. Furthermore, there is limited research in fully automating the analytical workflow of researchers. Therefore, our aim is to design a tool that can unify these features under one platform in order to automate the process of collecting and preparing models for analytics. Hence, this chapter describes the methodology used to develop and evaluate our work. First, an overview of the research design is provided in Section 3.1. Then, the research objective and questions are presented, including how the research methodology applies to this master thesis in Section 3.2.

3.1 Research Design Overview

This research project consists of a literature review and design research. The literature review aims to identify the latest research in model management and analytics, whose understanding will ultimately aid in designing an effective artifact supporting the model analytics workflow. Throughout the course of this thesis, we have followed the research design methodology as presented by Wieringa [53] which is aimed at conducting design science in information systems and software engineering research. According to Wieringa, design science comprises two activities that should be iterated: the design of an artifact that should improve something for some stakeholders and investigate the effectiveness of the designed artifact in a particular context. Hence, the goal of a design science project is to (re)design an artifact in such a way that it contributes to the stakeholders' desired outcomes.

The design cycle as presented in [53] describes the general methodology in a design science project and it comprises of the phases: *problem investigation*, *treatment design*, and *treatment validation*. In addition to the design cycle, Wieringa presents a larger cycle called the engineering cycle, which includes the steps of the design cycle and two additional steps: *treatment validation* and *implementation evaluation*.

The phases in the design cycle, as well as the engineering cycle, are shown in Figure 3.1. The first three phases correspond to the design cycle, while the engineering cycle encompasses all five phases. Depending on the outcomes of the validation or evaluation phases, the cycle has to be iterated again until it produces the desired outcome.

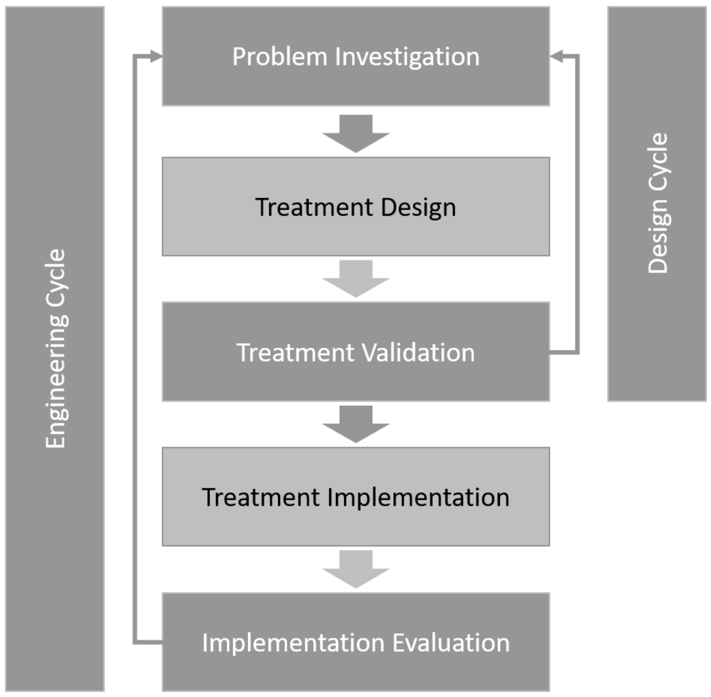


FIGURE 3.1: The engineering and design cycle phases.

3.2 Research Objective and Questions

To formulate the research objective, the template for defining design problems, as shown in Figure 3.2 was used. A *design problem* can be defined as: "a problem to (re)design an artifact so that it better contributes to the achievement of some goal" [53].

Improve <a problem context>
by <(re)designing an artifact>
that satisfies <some requirements>
in order to <help stakeholders achieve some goals>

FIGURE 3.2: Template for design problems [53].

Using this template, the objective of this research project can be formulated as:

"**Improve** the workflow in model analytics **by** designing a system **that satisfies** the requirement of running on the Arrowhead Framework **in order to** provide model analytics workflow automation".

In order to design a treatment achieving the desired outcomes in the problem context, we need to answer two knowledge questions:

First, an artifact enabling model management and model analytical workflow automation has to be designed. Moreover, the latest work in model repositories and model analytics has to be identified. Understanding these will aid in designing an effective treatment for the problem context. The first research question is therefore defined as:

> *RQ 1: How can we develop or integrate a model analytics automation system in the Arrowhead Framework?*

The second research question aims at determining whether the designed artifact produces the desired effects in the problem context. This includes the automation of the steps found in the model analytics workflow and the usability and performance aspects of the system. The second knowledge question is defined as follows:

> *RQ 2: How effective is the model analytics automation system in reducing the effort it takes to prepare models for analytics?*

To answer this knowledge question, we defined several sub-questions:

> *RQ 2.1: Can we reduce the time it takes to prepare models for analytics?*

Model analytics and data analytics, in general, encompass various activities prior to the actual analysis of data. For instance, data needs to be collected from multiple sources, and often this data needs to be cleaned and transformed. This process can be tedious and very time-consuming for researchers. The proposed system aims to streamline the process of collecting and preparing models; it is therefore expected to reduce the effort and time it takes to prepare models for analysis. We compare the performance of the manual approach and the automated approach using the system. In particular the *model preparation time* between the two approaches.

> *RQ 2.2: How easy is it to interact with the system?*

The system's primary aim is to reduce the effort it takes to collect and prepare models, and therefore enhance the researcher's job in empirical research. Therefore, it is important to design and implement a simple and easy to learn user interface so that users can effectively operate the system. With this question, we aim to measure the *usability* aspects of the system. While usability is a broad term and difficult to quantify, we defined usability more specifically following the TAM [32] model as described in Section 7.1.1.

The first research question is a combination of a knowledge question and a design problem [53]. It aims at identifying recent work in model management and analytics whose understanding can be used to design a tool that achieves the desired goals of the stakeholders. At the same time, the remaining research questions are knowledge questions that aim to evaluate the designed artifact. The application of the engineering cycle phases to this research project are shown in Table 3.1.

RESEARCH FOCUS	METHOD	CHAPTER
Problem investigation		
State of the art	Literature review	4
Treatment design		
RQ 1: How can we develop or integrate a model analytics automation system in the Arrowhead Framework?	Literature review Architecture design	4, 5
Treatment implementation		
Prototype implementation	Tool design	6
Evaluation		
RQ 2.1: Can we reduce the time it takes to prepare models for analytics?	Interview Benchmark	7
RQ 2.2: How easy is it to interact with the system?	Interview Questionnaire	7

TABLE 3.1: The engineering cycle phases applied in this research project.

CHAPTER 4

Review of Related Work

This thesis aims to design and integrate a system that runs on top of the Arrowhead Framework and enables the automatic execution of some of the steps in the model analytics workflow. Hence, this chapter explores related literature that provide assistance in model analytics. In particular, related work on the selected functionalities/services presented in Section 1.2.

First, an analysis of current model repositories is conducted to know which features they provide. We observe that they provide some form of automation to improve the management of models. In particular, they provide model versioning, check-in/check-out, collaborative modeling, model analytics, and extensibility. Furthermore, a comparison of the model repositories is provided. After the review on model repositories, a more detailed analysis of, in our opinion, the most relevant model repository is given: Apromore.

Second, collecting large numbers of models and preparing them for analysis are some of the main steps in model analytics as well as a feature proposed by the main stakeholder (see Section 1.2). Hence, a review of repository mining techniques will be conducted, specifically, crawling and scraping software repositories for process models.

Lastly, an analysis of some model transformation techniques is given. We will see that there are two methods for model transformation: *direct transformation* and *indirect transformation*. We will focus our attention on two particular model transformations: BPMN to EPML and the other way around.

4.1 Model Repositories

In recent years there have been several promising approaches regarding model repository technology. The studies in [55, 15] present a survey of process model repositories that vary in model management and persistence techniques. The study in [12] explores collaborative model repositories and their related obstacles and challenges. The study in [40] presents a survey of existing tools that employ model repositories as a service. This section reviews existing model repository technology related to MDE and BPM.

ModelBus [20]: is an SOA-based framework that facilitates the integration of heterogeneous modeling tools. The framework aims to provide model awareness to Service-Oriented Architectures. A prototype version was developed. However, it did not conform to some technical specifications. A new and revised version of ModelBus has been developed and is hosted as an open-source project. The tool provides web service interfaces that can be extended by the application developer, allowing existing tools to be integrated into the framework. Once the tool's integration process is complete, the tool's provided services become available for other tools in the framework to consume. The tool employs an interaction pattern, consisting of the modeling service, modeling consumer, and repository components. The interaction pattern's central component is a model repository, enabling model sharing between the integrated tools of the framework. The framework comes with a built-in model repository, providing functionalities for model versioning, partial checkout, and merging of model versions and fragments. The consumer and service components communicate with the repository through a skeleton and stub interface. The skeleton and stub interfaces allow models to be transmitted by a reference instead of a model. It is also possible for application developers to implement this interface to integrate a custom model repository.

CDO - Connected Data Objects [48]: is a pure Java model repository. It is designed to support both EMF models and metamodels. CDO offers some interesting repository functionalities, including collaborative modeling, model scalability, and model persistence with customizable database back-ends. CDO leverages the client-server architecture, supporting EMF-based client applications and providing model repositories on the server. The repository component comprises two layers, a generic layer that enables clients to interact with the repository and a database layer, allowing them to connect with various databases, such as relational- or object databases. One of the main focuses of the repository is to provide high scalability. This is achieved by employing lazy loading accompanied by prefetching techniques. These techniques enable on-demand model loading and model caching, resulting in improved performance while accessing large-scale and complex models.

GenMyModel [4]: is a cloud-based modeling environment, supporting BPMN, UML, EMF, Database, and Flowchart models. The tool provides a web-based interface with model management capabilities. It provides querying functionalities, allowing users to search for existing community projects by name. A filtering functionality is also provided, enabling users to filter by project types, such as UML or BPMN. Viewing search results is similar to other model repositories. It is possible to view more detailed information about a project, by selecting it from the list. Selecting a project provides a visualization of the project's model. Similarly to source-code repositories, it is also possible to clone the selected project, allowing users to reuse community-made models. Moreover, collaborative modeling is also supported, enabling users to work in real time on the same model. Furthermore, by leveraging an XMI interchange format, models developed on other platforms can be integrated into the platform. Hence, the tool can also be considered as a generic collaborative modeling platform.

MDEForge [6]: is an extensible modeling environment. While it was possible to test an early version of the tool, the project has been dismissed and taken offline. However, it is still possible to deploy a local instance of the repository. MDEForge aims to support the MDE community by providing functionalities to discover and reuse existing modeling artifacts. A feature that sets this repository apart from the others is the possibility to manage any modeling artifact including models, metamodels, transformations, and editors. This tool provides a web-based graphical interface to interact with the model repository. It supports searching and browsing the repository for already developed modeling artifacts. The repository leverages a three-tier architecture comprising the core, extension, and a REST API to expose the provided services. The core layer provides basic CRUD services for the modeling artifacts. Users can extend the repository with additional functionalities through community-made extensions, such as model metrics or advanced model querying.

AMOR - Adaptable Model Versioning [2]: is a framework that attempts to leverage version control systems in the area of MDE. A model repository is part of the AMOR framework. Features of the repository include conflict detection, intelligent conflict resolution, and adaptable model versioning. Precise conflict detection is described as the avoidance of previously undetected conflicts, and wrongly indicated conflicts. Intelligent conflict resolution is employed to represent the conflicting changes along with a proposal of resolution steps. Existing versioning techniques are either generic or tailored explicitly for a modeling language. Therefore, with adaptable model versioning, the repository aims to provide users with the ability to balance generality and specificity.

EMFStore [27]: is a model repository aimed to address the problem of model versioning. The repository is based on the Eclipse Modeling Framework and employs the client-server architecture, where the repository is deployed on the server. The repository supports operation-based change tracking, conflict detection, and merging. Operation-based change tracking is a particular case of change-based change tracking. In contrast to change-based change tracking, where each change of a model is recorded while it occurs in the repository, operation-based change tracking records the transformation operations on the state of a model. Operation-based conflict detection is adopted to detect conflicting operations on an attribute level. In other words, conflict detection marks operations conflicting if they change the same values of the same attribute to a different result. Operation-based merging is used to resolve conflicts.

WebGME [35]: is a web- and cloud-based (meta)modeling tool, that aims to provide scalability and collaborative modeling. The primary purpose of the tool is to support the design of large-scale and complex models. The tool is implemented using the client-server architecture, where the front- and back-end are deployed on the client and server respectively. The back-end consists of a model repository and a set of web-services exposing the models through a REST API accessible by the client. Moreover, the back-end can also be extended by plug-ins, allowing users to integrate additional tools. The tool leverages version control techniques as well as object-oriented concepts, such as inheritance and composition. Furthermore, novel modeling techniques are introduced to model crosscutting concerns.

APROMORE [31]: is a business process model repository. The repository is implemented as an open-source Software-as-a-Service (SaaS). It provides users with a wide range of services, such as analyzing and managing process models. The architecture follows a three-tier architecture composed of the enterprise (presentation), basic (business logic), and intermediate (middleware) layers. The repository provides a canonical process format, allowing heterogeneous modeling languages to be treated the same. Thus, the model repository is not only constricted to business process models, allowing it to be used in other domains.

ChronoSphere [18]: is a graph-based EMF model repository. The model repository aims to address the problem of managing large and complex models, by leveraging domain-driven modeling concepts with scalable graph-based model storage techniques and a custom model querying language. Moreover, advanced model versioning techniques are also provided through the use of a versioned graph, enabling users to make use of advanced conflict resolution and merging techniques.

Enterprise Architect [44]: is a modeling platform for heterogeneous modeling languages including BPMN, UML, and SysML. The tool provides a wide range of services supporting the software development process, from the analysis of requirements to the maintenance of models. Enterprise Architect provides a desktop application to interact with the model repository. It is possible to view and connect to repositories either locally, on a server, or the cloud. The desktop interface allows users to query models by attributes within multiple model repositories. Query results are presented in the form of a list. It is possible to select a model from the query results, resulting in a more detailed view of the chosen model. Furthermore, collaboration features are also provided in the form of discussions between users as well as reviewing of models. The tool provides design patterns for a wide range of UML diagrams, allowing users to reuse existing models without having to start from the ground up.

Table 4.1: A comparison of technologies employing model repositories

Technology	**Managed Artifacts**	**Main Purpose**	**Model Persistence**	**MaaS**	**Open-source**	**Extensible**
ModelBus [20]	Models Metamodels Transformation rules	Tool integration	RDBMS (built-in)	✓	✓	✓
CDO [48]	Models Metamodels	Scalability	RDBMS Documents Filesystem	✓	✓	✓
Enterprise Architect [44]	Models	Collaborative modeling	RDBMS	✓	✗	✓
GenMyModel [4]	Models	Collaborative modeling	RDBMS	✗	✗	✗
MDEForge [6]	Models Metamodels Transformation rules Editors	Storage	Documents	✓	✓	✓
AMOR [2]	Models Metamodels	Versioning	Filesystem	✗	✗	✓
EMFStore [27]	Models Metamodels	Versioning	Filesystem	✓	✓	✗
WebGME [35]	Models Metamodels	Scalability	RDBMS	✓	✓	✓
APROMORE [31]	Models Metamodels	Storage	RDBMS	✓	✓	✓
ChronoSphere [18]	Models Metamodels	Scalability	Graph	✓	✓	✓

4.2 APROMORE: Advanced Process Model Repository

Apromore (Advanced Process Model Repository) proposed by La Rosa et al. [31], is an open-source platform for SaaS (Software-as-a-Service) that resolves, to a greater or lesser extent, some of the problems laid out in Chapter 1. It "enables storing, retrieving, transforming, and analyzing the content of process models".

One of its main characteristics is its ability to treat process models represented by different modeling languages alike. The idea behind this is to translate process models to a common format, called the *Canonical Process Format*, which brings a set of benefits along with it, such as eliminating the need of having different algorithms for each process format.

4.2.1 Architecture

In this section the architecture of Apromore is presented.

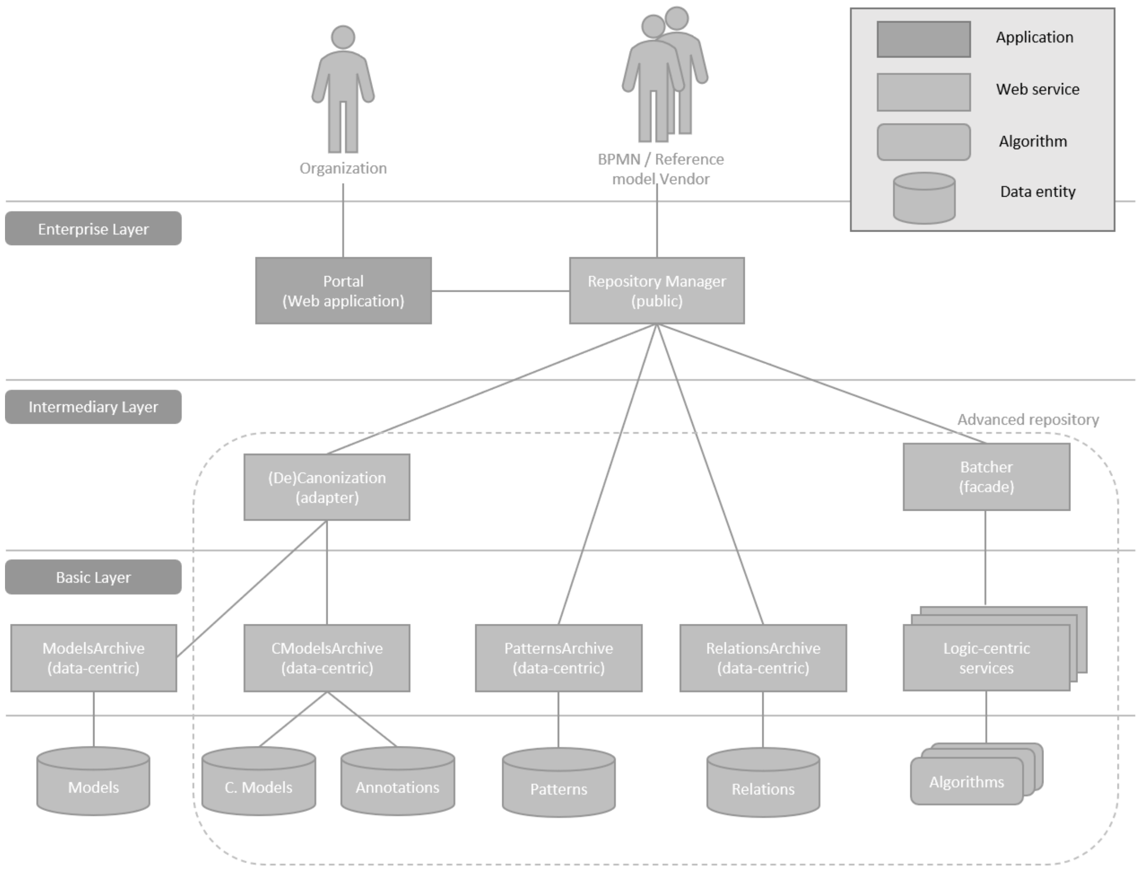


Figure 4.1: Architecture of the APROMORE process model repository [31].

Apromore is based on a three-tier architecture composed of the following layers:

- **Enterprise layer.** The enterprise layer acts as the *front-end* of the model repository. It contains the repository manager service, which provides the common functionalities of the model repository: model querying, import/export, model versioning, and security. The service can therefore be seen as the unique access point to the underlying layers of the model repository. Furthermore, through the repository manager, it is possible to access the services that provide model analytics (process similarity, clone detection, process merging, etc.) and storing models in the canonical process format.

- **Intermediate layer.** The intermediate layer exposes a set of services to the repository manager, including the *Batcher* and the *Canonization adapter*. The batcher service allows the repository manager to use the logic-centric services provided in the basic layer. In addition to this, users can batch a set of algorithms through a simple script. For instance, retrieve a collection of models, perform clone detection, and visualize the result. The canonization adapter allows the repository manager to access models both in their original and canonical format. In addition, conversions capabilities are also provided. For instance, the conversion from YAWL to BPMN.
- **Basic layer.** The heart of the architecture is the basic layer, which encapsulates the business and data logic of the model repository. The business logic contains the algorithms that perform actions on large collections of process models; typical of this entity are matching algorithms, merging algorithms, and individualization algorithms. The data logic consists of a set of data-centric services which enable access to the underlying data of the repository. The five main entities within the data logic level are:
    1. *Models archive*: this entity contains process models in their original format (i.e. BPMN, EPML, YAWL, etc.).
    2. *Canonical models archive*: this entity contains the canonical format of each corresponding model in the models archive.
    3. *Annotations archive*: for each model, the metadata about its representation is added (i.e. line thickness, positions, etc.).
    4. *Patterns archive*: the pattern archive contains a set of model definitions or patterns which can be reused by users for defining other process models.
    5. *Relations archive*: relations between process models in their canonical format, or relations between process models and their extensions are stored here.

4.2.2 Canonical Process Format

The Canonical Process Format (CPF) provides a common unambiguous standard of business process models represented in different notations or different abstractions levels, with the intent that all process models can be treated as processes modelled in the same language. The idea is to include only the structural characteristics of a process model that are common and recurrent in most modeling languages. Since CPF provides a language purely for the structural type, graphical aspects of models, such as lines, positions, and shapes, are deemed unnecessary and are stored separately in the form of *annotations*. The information stored as annotations is only used when a model needs to be converted back to its original format.

The authors identify the following benefits of using the CPF:

- *Standardization:* there is no need to have different versions of algorithms for different types of models. In other words, algorithms working on the CPF, can be leveraged to operate on models written in different languages, given that there exists a translation from the original language to the CPF.

- *Efficiency:* the time of translating a model from one language to another is reduced significantly. Moreover, models can be indexed on their canonical format elements, which improves the overall system efficiency of querying models.
- *Interchangeability:* the non-structural information of models captured as annotations can be leveraged to transform CPF to the original language, but also other languages. Furthermore, it is possible to switch between different visual representations while keeping the same structure of process models.
- *Reusability:* reusable business process patterns are also stored in the CPF, allowing users to define new processes without the need of having to design a process from the ground up.
- *Flexibility:* the elements of the CPF are defined by an inheritance mechanism, enabling processes to be seen, from a higher abstraction level, as directed graphs. This allows performing operations at different levels of granularity.

Therefore, by leveraging the CPF, eliminates the need of having to implement different versions of the same algorithm for each modeling language. Figure 4.2 how some of the common BPMN and EPC constructs are translated to CPF.

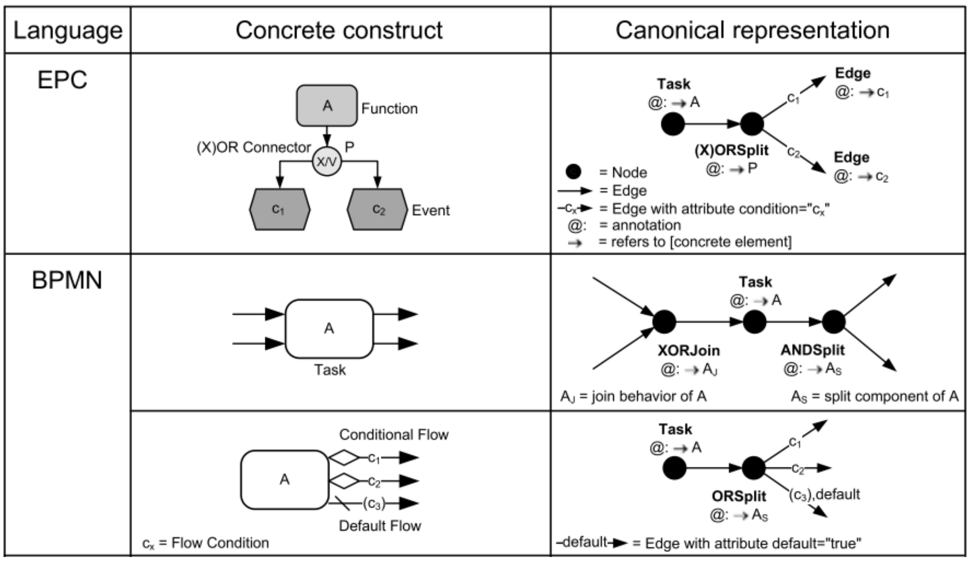


Figure 4.2: Common language constructs in CPF [31].

4.3 Software Repository Mining

Software Repository Mining that is the systematic way of collecting, processing, and analyzing information of software artifacts from software repositories, has grown in popularity over the last years [21]. Mining the data provided by *GitHub* for instance can uncover some interesting information about software engineering projects. While the research field is mainly centred around mining source code artifacts, other artifacts such as modeling artifacts can benefits from the same data mining techniques as well [19].

The work in [43, 19], presents a semi-automated approach to retrieve UML models from GitHub. The authors follow a systematic approach comprising the steps: GitHub Mining, Identification of UML models, Verification of UML models, and Metadata extraction. This approach resulted in the *Lindholm dataset*, consisting of over 93,000 publicly available UML models. As an inspiration, the study in [21], followed a similar systematic approach to mine GitHub for BPMN models. Figure 4.3, illustrates the mining process.

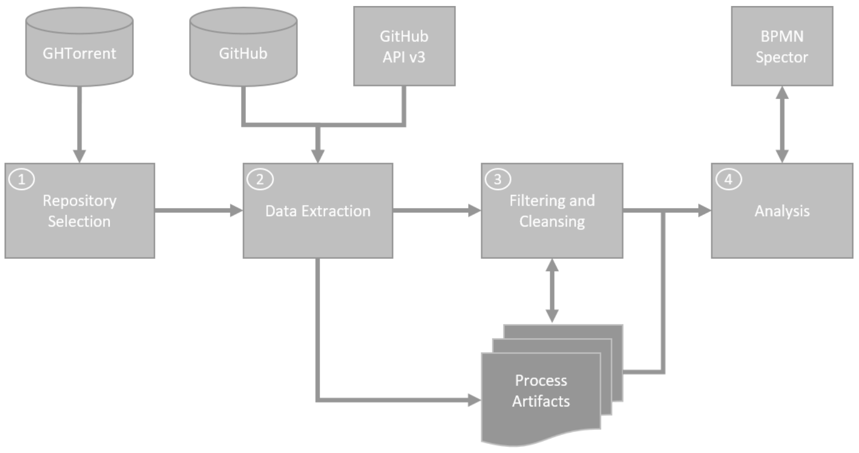


FIGURE 4.3: Overview of the BPMN mining process [21].

As can be seen, the mining process consists of the following steps:

1. **Repository Selection:** the first step in the mining process is the selection of repositories to mine. The authors consulted the GHTorrent [17] database in order to select 10% of GitHub repositories that are non-forked and non-deleted.

2. **Data Extraction:** from the pool of selected repositories, the *GitHub API* was queried for the repositories file-tree. The authors noted that the throughput was severely bottlenecked by using the *GitHub API*. To overcome this problem, they used several user credentials. Using this approach reduced time it took for this step to complete from 100 days to 31 days.

3. **Filtering and Cleansing:** since BPMN files can come if different file formats, for instance as an image or as XML files. They kept models that only matched the BPMN 2.0 metamodel, resulting in a corpus of 8,904 BPMN models.

4. **Analysis:** the last step in the mining process is the analysis, which the authors conducted to answer a set of research questions.

In [9] the authors present the GHS (GitHub Search) dataset containing data of 507,871 repositories on GitHub written in 9 programming languages. With this dataset, the authors aim to support researchers in their empirical studies by providing 25 characteristics of each mined project. Furthermore, the data of each mined project is continuously updated. The authors use a custom mining tool, leveraging the GitHub Search API in combination with an HTML page crawler to mine additional information from GitHub projects. The architecture of the tool is depicted in [9].

The *GitHub API Invoker* is used to collect a number of characteristics from a list of repositories written in a certain programming language, and created or updated during a specific time period. The authors use the latter feature to segment the GitHub Search query in several time intervals in order to overcome the 1,000 search results limit per request.

The *GitHub Website Crawler* is used to collect additional repository characteristics. The component parses HTML pages through the usage of CSS selectors. For this task, the authors relied on the *jsoup* and *Selenium* libraries. The former provided the ability to parse information from static HTML pages. However, due to the dynamic content of several GitHub pages, *jsoup* failed to retrieve some information of interest. The authors have therefore relied on *Selenium* in order to extract information from dynamic pages.

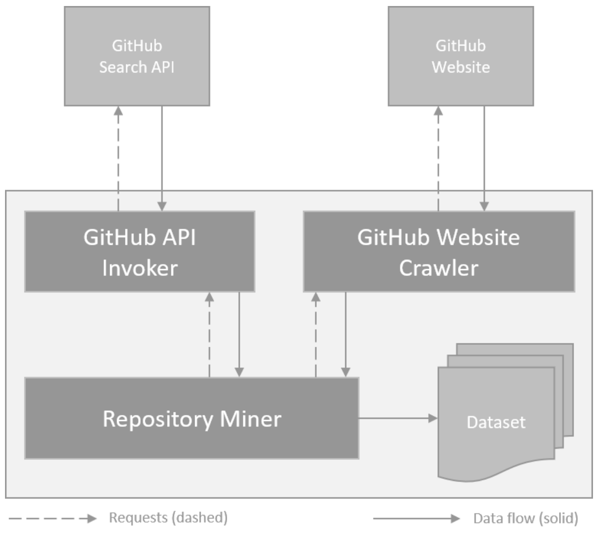


FIGURE 4.4: Architecture of GHS [9].

The *Repository Miner* is the core component of the GHS tool. The component can be seen as an orchestrator, commanding the *GitHub API Crawler* and *GitHub Website Crawler* to collect information from GitHub repositories. Furthermore, it leverages a periodic mining algorithm triggered every six hours to update the collected data in GHS.

4.4 Transforming Models

Business process modeling has become a significant part in the industry, mainly to analyze, document, and optimize workflows [49]. Currently, Event-Driven Process Chains (EPC) process models are widely used in the industry. With the current rise of the Business Process Modeling Notation (BPMN), there is a significant need from the industry for the transformation of EPC to BPMN [29, 49].

In this section, we will review recent works on model transformations. In particular, transforming EPC to BPMN and the other way around. We are aware that EPC includes various heterogeneous formats, such as VDX and AML, which are proprietary formats constrained to the tools Visio and ARIS respectively. We will therefore limit our review on a tool-independent format, referred to as the EPC Markup Language (EPML).

There are two general approaches for transforming process models. They can either be transformed *directly* or *indirectly* [49]. With direct transformation, model elements from one language are directly translated to the other language. On the other hand, indirect transformation uses an intermediate language to translate model elements from one language to the other.

4.4.1 Direct Transformation

Direct transformation is the direct or “one-to-one” mapping from model elements defined in one language to the other. For instance, a direct mapping from an EPC *function* element to the BPMN *task* element. An advantage of this method is that no structural or semantical information is lost, given that there is a direct mapping for every model element in the two original languages. The study [49] presents a direct approach for mapping EPC to BPMN. However, because of the semantical differences between the two languages, a one-to-one mapping was not possible. And thus, only some of the elements of EPC, which the authors refer to as the *core* elements, are directly translated to BPMN.

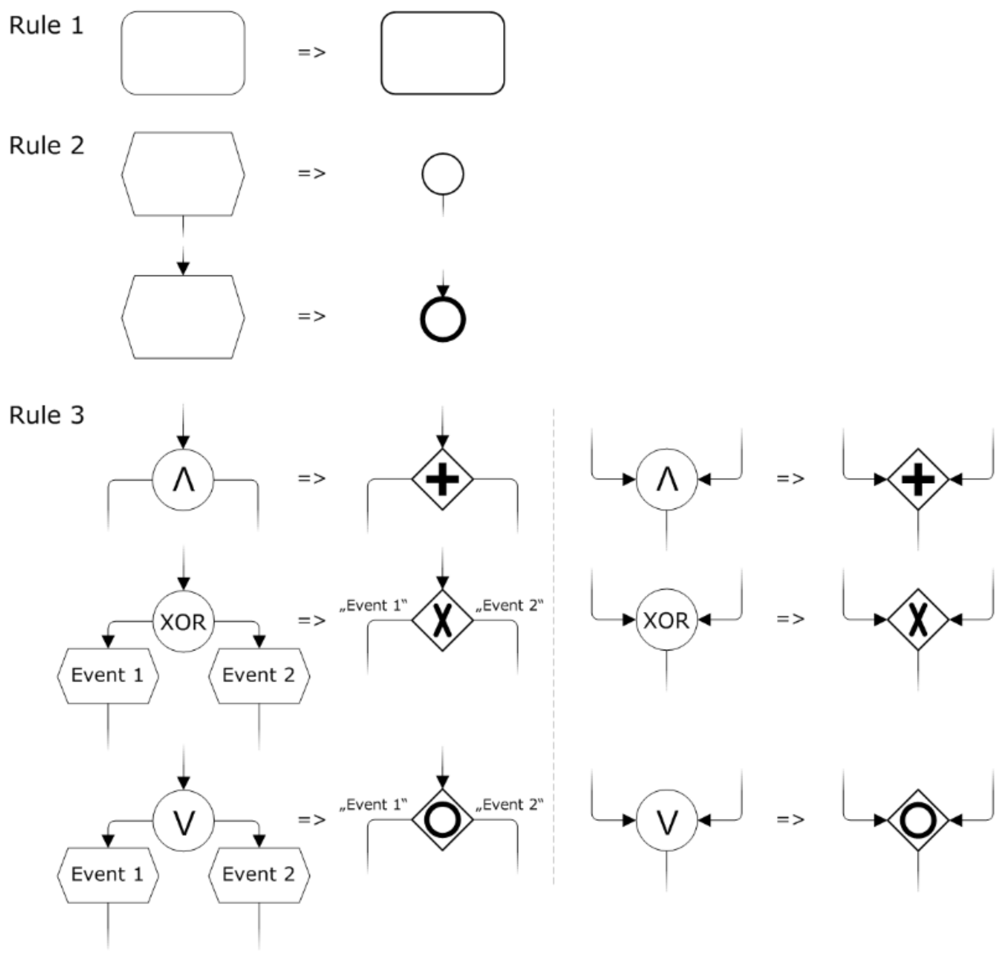


FIGURE 4.5: Direct transformation from core EPC to BPMN [49]

The work in [29] expands on this and presents a tool for two-way conversion between EPC and BPMN. In other words, a mapping from EPC to BPMN and the other way around. The authors intended to enable model interchangeability between different modeling toolsets, since one modeling toolset for EPC models can have more advantages than a modeling toolset for BPMN models. Similar to [49], it was not possible to map every element from BPMN to EPC, and the transformation was limited to the *core* elements of both languages only.

4.4.2 Indirect Transformation

Indirect mapping uses an intermediate language to translate EPC to BPMN. For instance, EPC is mapped to a Petri net, which is consequently translated to BPMN. An advantage of this approach is that already existing model mappers can be used to map the two notations. However, this method has some drawbacks. In particular, the expressive power of the intermediate language can be lower compared to EPC and BPMN. Therefore, some of the structural and semantical information can get lost during the transformation [49]. In this context, the CPF presented in Section 4.2.2 enables indirect transformation from one language to the other. A model represented in the EPC language is first mapped to the CPF and later translated to BPMN. However, as discussed earlier, indirect mapping results in a significant structural and semantical information loss if the intermediate language has a lower expressive power compared to the target and source languages. To minimize the information loss, the authors in [31] store non-structural information of models, such as graphical information separately as *annotations*. The information stored in the annotations is later combined with the structural information stored as a CPF to transform models to other languages.

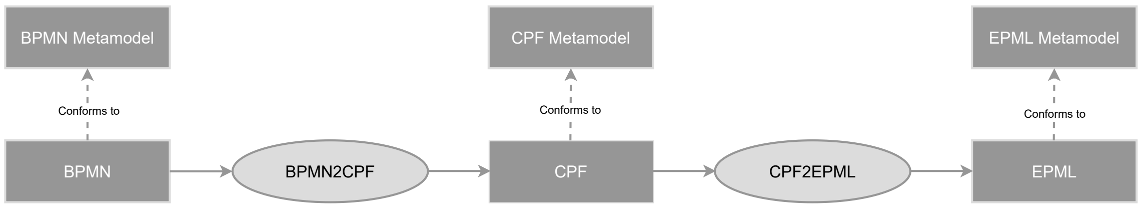


FIGURE 4.6: Example of indirect transformation.

Figure 4.6 illustrates an explanatory indirect model transformation chain. In particular, *BPMN2CPF* is a model transformation that converts models conforming to the BPMN metamodel to models confirming to the CPF metamodel. In addition, *CPF2EPML* is a model transformation that converts models conforming to the CPF metamodel to models conforming to the EPML metamodel.

4.5 Conclusion

We analyzed state of the art regarding the proposed services in Section 1.2. On the one hand, we presented several model repositories and the features they offer. These features include check-in/check-out, model versioning, collaborative modeling, model analytics, and extensibility. These features have been implemented as prototypes as well as production level solutions. On the other hand, we explained different techniques to mine data from software repositories and to perform model transformations. After the analysis of the related work, we conclude that the majority of proposed tools and techniques do not provide a fully automated solution to assist researchers in their model analytics workflow. Furthermore, there is a lack of research in unifying the features under one platform.

CHAPTER 5

Overview of the System

This chapter presents a detailed overview of the Model Analytics Automation System (MAAS) so that subsequent chapters can be easily understood. Firstly, the goal of MAAS is stated as well as the requirements it is constrained by. Next, an explanation of the architecture of the system and its core elements is given. Finally, information in order to integrate the system in the Arrowhead Framework, is provided.

5.1 Goals and Requirements

As stated throughout Chapter 1, modeling tools are not at the level yet of providing complete model analytics workflow automation to the user. In Chapter 4, we have seen several technologies that offer some of these functionalities to the user. However, after the literature review, we concluded that all the presented functionalities are scattered among various tools and technologies.

The goal of this thesis is to design and implement a tool that can integrate the services presented in Section 1.2. Furthermore, the tool should run in the Arrowhead Framework as a system composed of several subsystems. In Figure 5.1 we can see a simplified schema of the system's architecture. The system follows a multi-layer architecture, allocating the different responsibilities into different layers. The architecture comprises the following layers in general:

Presentation (*User Interface*): the main responsibility of this layer is to transform results to something the users can understand and provide an interface that can be used to easily interact with the functions provided by the system.

Logic (*Workflow Execution*): this layer comprises the business logic to operate the system. It acts as an intermediate layer, moving data between the presentation and data layers. This layer handles user requests by loading and saving information in the data layer.

Data (*Model Storage/Retrieval*): information is stored and retrieved from the data layer, which is connected to a particular database back-end.

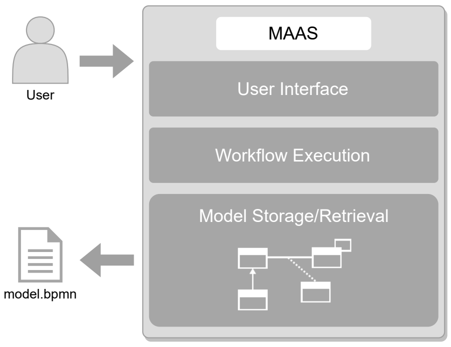


FIGURE 5.1: Simplified schema of the Model Analytics Automation System (MAAS) architecture.

We have already seen that the model analytics workflow encompasses various steps to prepare models for analysis. The tool should therefore be *generic* such that *new functionalities* can be integrated by extending existing components or adding new components. Hence, the tool should also allow other components to be plugged to extend its capabilities and thus automate more tasks in the researcher's work.

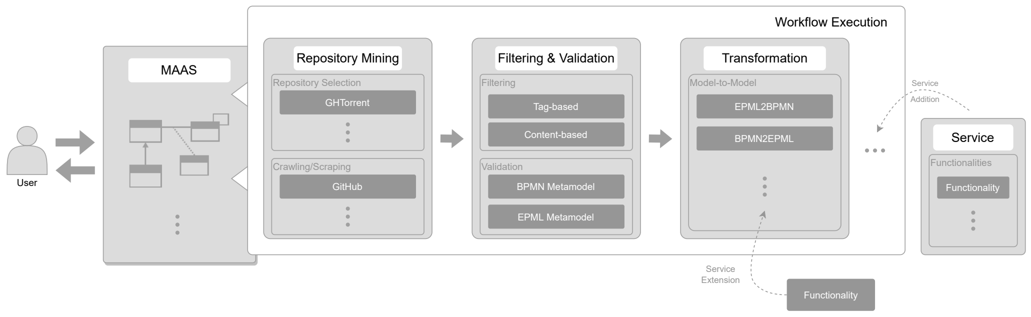


FIGURE 5.2: Overview of MAAS with the ability to plug and extend services.

Figure 5.2, presents a generalized overview of the tool. As shown, MAAS integrates a set of services that can be plugged or removed. Furthermore, existing services can be *extended* with new *functionalities*. A few concrete core services are illustrated.

- **Repository Mining**: this service is responsible for crawling/scraping software repositories such as GitHub for potential process models.
- **Filtering & Validation**: consist of validation and filtering services for various process models.
- **Transformation**: comprises the functionalities to perform model transformations.

5.2 The Model Analytics Automation System

The goal pursued by the Model Analytics Automation System (MAAS) is to improve the model analytics workflow. In order to contribute to this goal, MAAS provides automation of some of the *steps* found in the *model analytics workflow*. This section presents the architecture to achieve this goal.

First, an overview of the architecture is given. This overview includes the various required layers to materialize the tool and the interaction between the different components at a very abstract level.

Then, a detailed explanation for the various layers and their components is given. This includes the intercommunication between components inside the layers.

5.2.1 Architecture Overview

This section presents a more detailed overview of MAAS's architecture. As sketched in Figure 5.3 MAAS is based on a three-tier architecture:

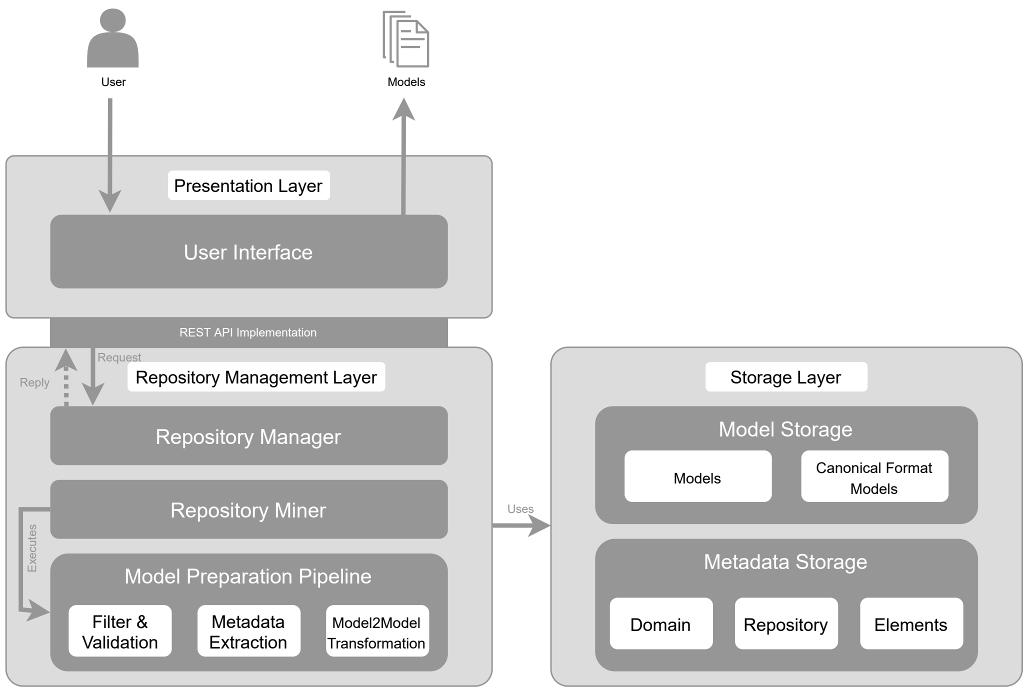


FIGURE 5.3: Details of MAAS's architecture.

The Presentation Layer. The presentation layer provides the interface to the model analytics automation system, which allows users to interact with the underlying subsystems through a GUI. The presentation layer connects to the repository management layer through a RESTful interface. The core services of MAAS are provided by the repository management layer, which are accessible through the RESTful interface. Furthermore, the repository management layer acts as a single point of entry to the services in the underlying layers.

The Repository Management Layer. In addition to the repository manager, which is responsible for the functionalities to communicate with the model repository, the repository management layer consists of two additional essential entities that are called the *repository miner* and the *model preparation pipeline*. The repository miner is responsible for crawling and scraping software repositories for potential process models. This entity executes the model preparation pipeline before any process models are stored in the repository.The first component of the model preparation pipeline is the *filtering and validation* component, which provides model cleaning and validation services. The second component is the *metadata extractor*, which collects additional metadata about process models. The last component is the *model2model transformer*, enabling model to model transformation. The three components exchange information with the storage layer to store the intermediate artifacts produced in the model preparation pipeline.

The Storage Layer. Lastly, the storage layer contains the data of the software architecture, where a set of services are collaborating to deal with the data in the system. It stores process models and related metadata. Process models are stored both in their original format and *canonical* format. In this case, we can significantly improve the transformation efficiency by reducing the number of intermediate transformations. Furthermore, in most cases, a transformation from one language to another leads to information loss. Therefore, having a reference to the original model eliminates the problems caused by the conversion from one format to the original one.

5.3 Arrowhead Framework

A mandatory requirement that MAAS has to meet is that it has to run on top of the Arrowhead Framework as a system, allowing other systems in a local cloud to consume the provided services to automate the steps in model analytics. MAAS is tightly related to the SOA and SoS concepts of the Arrowhead Framework. To elaborate, we have a tool comprising a set of services that collaborate to automate some tasks in model analytics. Moreover, services can be extended or added to extend the functionalities of the tool. Translated to the Arrowhead Framework, the tool can be seen as a system running on the *local cloud* consisting of various *sub systems* collaborating as an SoS to achieve a particular automation task.

It is, therefore quite reasonable to think that the architecture of the Arrowhead local cloud is designed to accommodate for the *modular* and *service-oriented* nature of the system, like the one shown in Figure 5.4. The architecture allocates the different responsibilities of MAAS into different (sub)systems that collaborate in order to provide automation services.

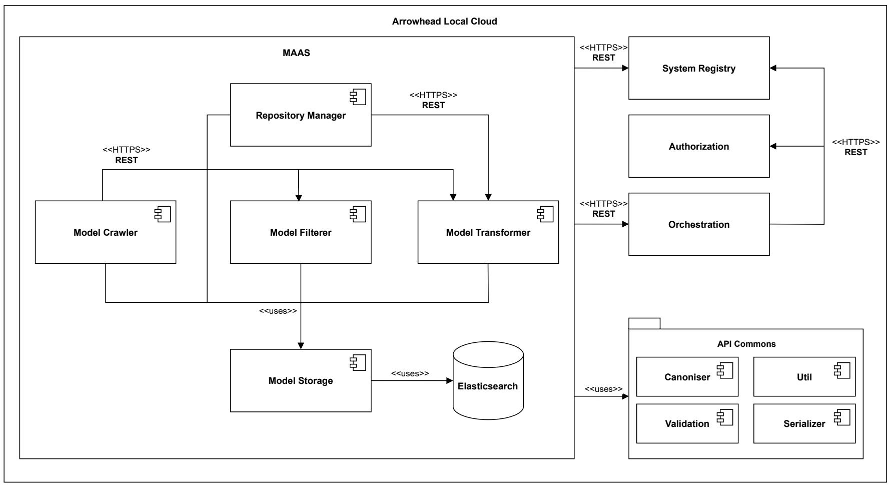


FIGURE 5.4: An overview of the systems and packages of MAAS running on a Arrowhead local cloud. The arrows represent a *uses* relationship. Moreover, the *API Commons* package represents the components that are part of MAAS, and shared by the various subsystems within MAAS.

CHAPTER 6

System Implementation

This chapter describes the implementation and design decisions for the prototype version of MAAS. The prototype is implemented using a multi-layer architecture consisting of the repository management, data, and presentation layers, which can be accessed through the following repository [57]. Although Arrowhead Framework offers implementations in both the Java and C++ programming languages, the former has a higher maturity level compared to the latter. We will therefore present the implementation of MAAS based on the Java implementation of the Arrowhead Framework. However, in order to generalize the implementation to other languages, we will discuss the design and implementation as generic as possible.

First, the design and implementation of the repository management layer are presented. The implementation is organized using the *Repository Miner* system and the subsystems that form the *Model Preparation Pipeline*. The former comprises the functionalities that enable repository mining for process models, whilst the latter consist of multiple systems that prepare models for analytics.

Second, details on the storage layer are given. It consists of information regarding the database back-end to store models and associated metadata, followed by an explanation of the data structure and conceptional schema of the repository system.

Finally, the chapter concludes with implementation details of the presentation layer. The layer consists of a RESTful interface exposing the service provided by the repository management layer and a web interface acting as the *front-end* of MAAS.

6.1 The Repository Management Layer

This section will detail how the cores services of MAAS are actually implemented and how the repository management layer acts as an intermediate layer between the presentation and model storage layers.

6.1.1 The Repository Model Miner

In Section 5.2, we stated that one of the main components in the repository management layers is the repository miner. This component collects models from software repositories in GitHub of a specific modeling language specified by the user. In this section, we present two methods of crawling models from GitHub. The first method leverages GHTorrent to select candidate repositories. The second method uses GitHub Search to query GitHub for process models. In the remainder of this section, we will detail the implementation of the crawler components and the design decisions.

GitHub provides a public REST API for retrieving a list of available repositories accompanied with their metadata. However, the API imposes some limitations on the number of queries for a given time frame. For non authenticated users, this limitation is 60 queries per hour, and for authenticated users, the number of queries per hour is 5000. The API can severely bottleneck crawlers leveraging the GitHub API, and they are usually implemented using a large set of authenticated user account credentials [21].

In order to solve this problem, our model crawler implementation does not use the GitHub API to retrieve information about the repositories but an HTML page parsing approach through CSS selectors. For this task, the crawler uses the *jsoup* library [24]. However, due to the dynamic nature in GitHub pages, information about some of the required elements needed by the model crawler is not always available, since it takes some time to load the remaining elements of a GitHub page, which *jsoup* fails to do so because its primarily a static HTML page parser. We have therefore relied on the *HtmlUnit* library [22], which allows us to parse dynamic pages. However, using *HtmlUnit* introduces a significant performance drawback. To reduce the performance drawback, we only use *HtmlUnit* were *jsoup* fails.

The HTML page parsing approach through CSS selectors may impose some problems in the near future when GitHub decides to change its UI substantially. Our repository mining component might require a future update if this is the case. To reduce the maintenance cost of our crawler, we followed the same procedure as in [9] by making the CSS selectors as generic as possible. In addition, we have generalized our implementation of the crawler and made some key components modular, which will be further detailed in the remainder of this section.

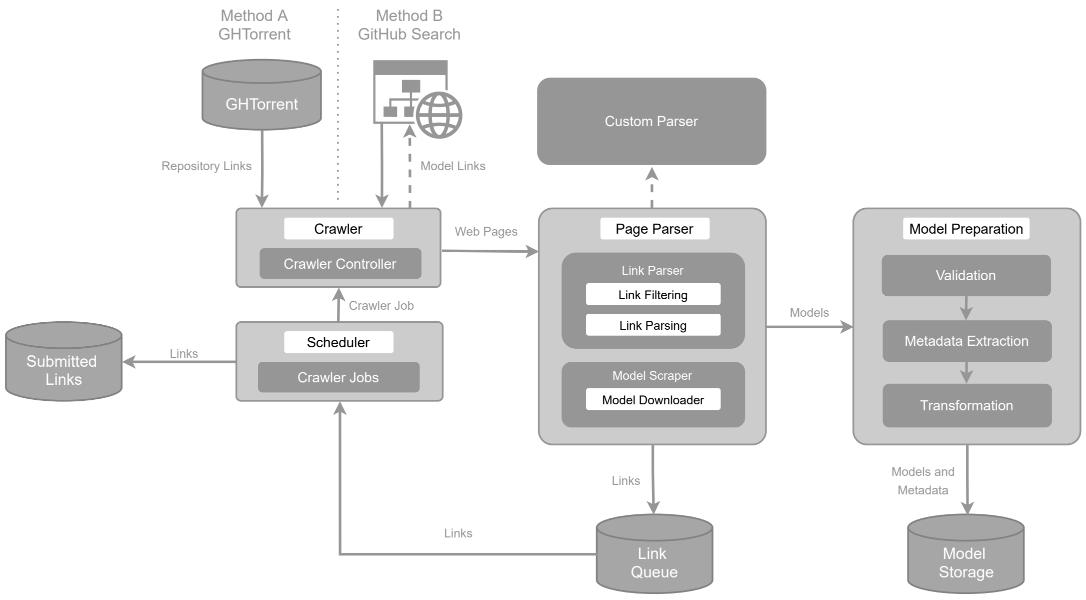


Figure 6.1: Architecture overview of the repository miner depicting the two methods of crawling.

To improve the model and metadata collection performance, we conduct multi-threading, which allows the crawler to download and parse multiple HTML pages in parallel. The architecture of the multi-threaded crawler is depicted in Figure 6.1. As illustrated, the general architecture consists of the following key entities:

- The *Crawler* is the central unit of the repository miner. It is responsible for initiating multiple *crawling jobs* exposed by an API. The crawler supports two methods of crawling, which are discussed in the remainder of this section.
- The *Page Parser* is responsible for parsing HTML pages and extracting outgoing links and potential process models. Links are filtered based on a set of criteria (e.g. links ending with ".bpmn/.epml") and are sent to the *Link Queue* for later processing by the crawler. A *Model Scraper* is used to extract the contents of a potential process model. Lastly, the contents of potential process models are transferred to the *Model Preparation Pipeline.* The component is *generic*, allowing for particular parsing *specializations* (e.g. a parser for the BitBucket software repository).
- The *Scheduler* divides the workload over the multiple CPU cores/threads by retrieving the first URL in the queue and scheduling it for a crawling job sent to the *crawler.* In order to overcome the problem of circular crawling, the component keeps track of the *submitted links* during the crawling process.
- The *Model Preparation Pipeline* component is used to "prepare" models for analysis. It comprises the steps of validation, metadata extraction, and transformation. During the *validation* step, process models are validated against their metamodels (i.e. BPMN 2.0 and EPML 2.0). Next, the metadata of process models is extracted, such as the canonical process format elements. Lastly, *Transformation* is used to translate models to the canonical process format. Consequently, resulting in two formats process models: the original format and the canonical process format.

In the remainder of the section, we will discuss our two methods of model crawling. The first method is inspired by the mining approach discussed in [21]. However, *HtmlUnit* imposed a significant performance drawback. Therefore, we have implemented a heuristic method, which resulted in the crawler collecting models in a reasonable amount of time.

GHTorrent Method

Our first repository mining method was inspired by the approach discussed in [21]. The approach consists of two steps, for which the first step requires some manual work, while the system completely automates the last step: (1) Select a sample of available *GitHub* repositories, (2) find and extract BPMN/EPML models as well as their repository metadata:

1. *Repository Selection.* The first step in our approach consists of collecting a list of available repositories from *GitHub.* Similar to [21], we used the *GHTorrent* database [17] to download the most recent database dump (*mysql-2019-06-01*, 103 GB). Finally, we queried the *projects* table to select a random sample of 10% of the repositories which are not forked and not deleted, resulting in a set of 7,435,722 repositories.
2. *Model and Metadata Extraction.* After the repository selection step, we used the crawler to examine the repositories for any BPMN/EPML files by "feeding" it the list of 7,435,722 URLs to the repositories. For each URL, the crawler uses *HtmlUnit*

to parse the dynamic content of the HTML pages, followed by *jsoup* to extract the URLs of the folders in the repository's "file-tree" by using CSS selectors. The extracted URLs would then be put in the queue for later processing by the crawler. The crawler uses *depth-first* search to traverse the repository's file-tree and locate any potential BPMN/EPML models. Once it locates a potential process model (i.e. an URL ending with **.bpmn**), the crawler extracts the model and validates it against its metamodel. Finally, if the model conforms to its metamodel, it is stored in the database along with its metadata and canonical process format.

As discussed earlier, *HtmlUnit* imposes a severe bottleneck on this method. Based on our observations on smaller selections of repositories (i.e. 100, 1,000), this method alone would take ~1456 days to complete. We have therefore opted for a heuristic method, which is orders of magnitudes faster.

GitHub Search Method

In order to increase the throughput of our model crawler, we, therefore, conducted model extraction using a heuristic method. This method uses the *Search* feature of *GitHub* to query the repositories for files ending with ".bpmn" or ".epml". For instance, if we want a list of BPMN files, the following request is triggered:

```
https://github.com/search?q=extension:bpmn
```

This request results in an HTML page containing at most 10 "hits" of the search query with the ability to "paginate" to the next page for more results. For each item in the list, we insert the URL in the queue. Additionally, the URL to the next page is inserted in the queue as well. Similar to [9], to overcome the 1,000 search results per request imposed by GitHub, we segment our query in multiple sub-queries. In particular, we segment our queries in multiple file size intervals. In other words, if we want a list of BPMN files, having file size between a given interval (i.e. 0 and 250 Kilobytes), we use the following search request:

```
https://github.com/search?q=extension:bpmn+size:0..250
```

If the request returns more than 1,000 files for a given size interval, we split the interval in half, and the new size intervals are inserted in the queue, later processed by the crawler. Otherwise, if the interval results in 1,000 or fewer results, the crawler iterates over the resulting list as usual. The crawler extracts the model and the repository's metadata for each potential process model (i.e. an URL ending with ".bpmn/.epml").

Implementing the Model Crawler

The class diagram of the model crawling service is illustrated in Figure 6.2. The entry point of the service is through the *ModelCrawlService*, which exposes a set of crawling functionalities provided by the *Crawler*.

The class *ModelCrawlService* provides the functionalities for starting and stopping a crawler. Moreover, it handles the creation of the *Crawler* class. Furthermore, the class is responsible for injecting a *PageProcessor* in the *Crawler* class.

The *Crawler* class is responsible for crawling a given list of URLs. The class uses an injected instance of the *PageProcessor* to extract the outgoing links from an HTML page. Furthermore, it is responsible for the creation of the *CrawlerThreadPoolExecutor*.

The *CrawlerThreadPoolExecutor* is a subclass of the *ThreadPoolExecutor* provided by the Java library. The class acts as the *Scheduler* of the crawling service, which is responsible for scheduling tasks in the *CrawlerJob* class.

The *PageProcessor* is the base class of the *GitHubPageProcessor* and *GitHubSearch-PageProcessor* which are responsible for the *GHTorrent* and *GitHub Search* methods respectively. The *PageProcessor* class can be seen as the *Parser* component of the crawling service.

The *CrawlerStatistics* class holds basic statistics of the crawler (i.e. the number of collected models, number of non-valid models, and number of collected page links).

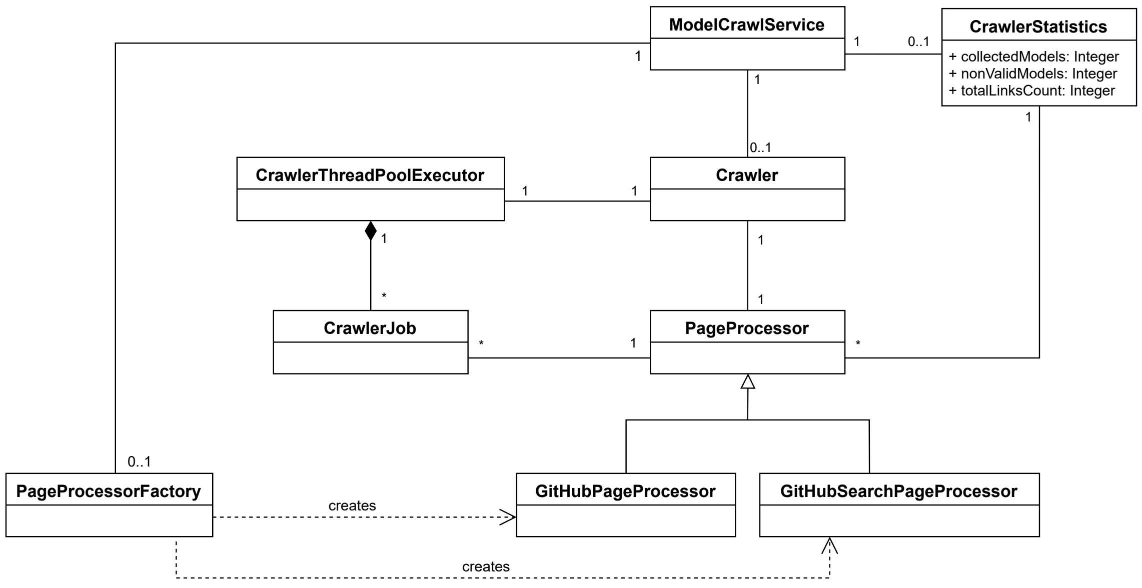

ModelCrawlService
CrawlerStatistics
+ collectedModels: Integer
+ nonValidModels: Integer
+ totalLinksCount: Integer
CrawlerThreadPoolExecutor
Crawler
CrawlerJob
PageProcessor
PageProcessorFactory
GitHubPageProcessor
GitHubSearchPageProcessor
creates
creates
1
0..1
*

FIGURE 6.2: Class diagram of the model crawling service.

6.1.2 Filtering Models

In Chapter 5.2, we presented the subsystems that encompass the model analytics automation system. One of these subsystems is the model filtering system, which provides a set of services to filter models based on user-provided search criteria. This section describes how the model filtering system should be set and how it should be implemented in the Arrowhead Framework.

The model filtering system provides users with the following type of model filtering:

- General filtering.
- Model elements filtering.
- Repository filtering.

General filtering can be applied to select models containing a specific string in their name (e.g. “shipping” will return all models containing this string), or having a specific extension (e.g. “.bpmn” or “.epml”).

Models can also be filtered by their canonical format model elements. The user can issue a predicate followed by an integer to filter on the model elements (e.g. “models having more than five elements of the event type”). The user can choose from the following types of predicates: “greater than”, “lesser than”, “not equals”, and “equals”. Moreover, the ability to “chain” conditions is provided. With this, the user can initiate filtering conditions having the following form:

“$numberOfEdges > 5$ AND $numberOfTasks < 3$ AND $numberOfORJoins = 1$”.

Through repository filtering, models can also be filtered on their metadata of the repository (e.g. number of stars or number of forks). Similar to the filtering by model elements, repository filtering can be leveraged through predicates.

Constructing Filtering Conditions

To determine the *filtering conditions*, we require two components in the model filtering system to set the filtering of models in motion:

- When the system receives a filtering request, a *Parsing Component* must parse and extract the conditions from the request.
- When the parsing of the conditions is complete, a *Query Translation Component* must map the conditions into a database query.

In order to retrieve models based on a set of conditions, the above components must be implemented. The problem is that the filtering requests are *dynamic.* Users can initiate a filtering request based on an arbitrary set of predicates and model attributes. Moreover, conditions can be chained together, and they can be of any arbitrary size. Hence, manually coding all possible combinations of filtering conditions is not feasible for this thesis nor a *sound* design decision.

To overcome this problem, we applied the *Builder* [16] design pattern. The builder pattern is classified as a *creational pattern*, as it describes how classes should be instantiated. The pattern allows a step by step construction of objects, which is deemed suitable for the dynamic construction of filtering conditions.

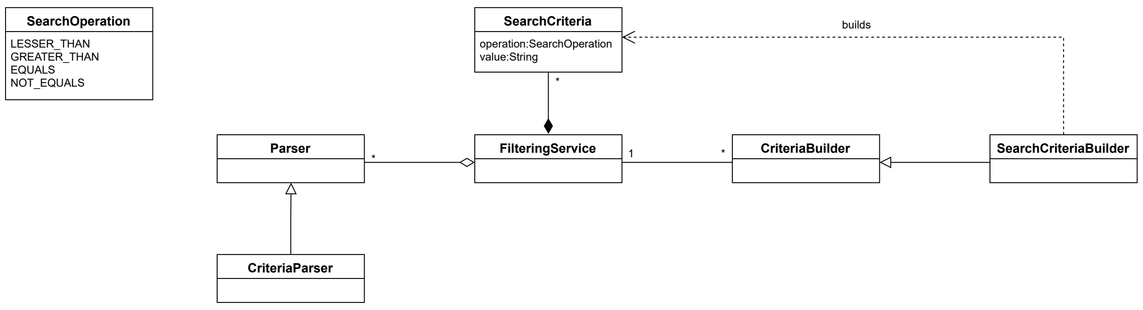


FIGURE 6.3: Class diagram of the filtering service.

Furthermore, because parsing algorithms tend to be complex in general, it is desirable to keep them contained or completely independent from the general system. It is suggested by [16] to isolate complex algorithms by encapsulating them in objects, resulting in easier interchangeability between the various algorithms at run-time. Therefore, to make the algorithms interchangeable at run-time, a *generic* and a *concrete* parsing class are defined.

6.1.3 Validating Models

We expect to encounter a wide range of process models during the repository mining process, including models with the same file formats modeled in a different modeling language versions (i.e. BPMN 1.0 and BPMN 2.0). Fortunately, most process models define a standard XML-based interchange format, providing tools to import and export models using this format [21]. This thesis focuses on the BPMN and EPML formats, and process models of these formats are validated against their BPMN 2.0 and EPML 2.0 metamodel, respectively.

To validate process models against their metamodel, we need to address the following problem: given that *Validation Requests* occurs, how do we instantiate the correct type of *Model Validator* once it has been triggered.

Determining Model Validators

In order to instantiate the right *Model Validator* given a model type, we recognized that the *Factory Method* [16] design pattern provided the ideal structure for this design problem. This pattern abstracts the creation process, such that a specific *type* of an object can be determined at run-time, given the parameters passed to the factory method. Figure 6.4 shows the classes of the validator component.

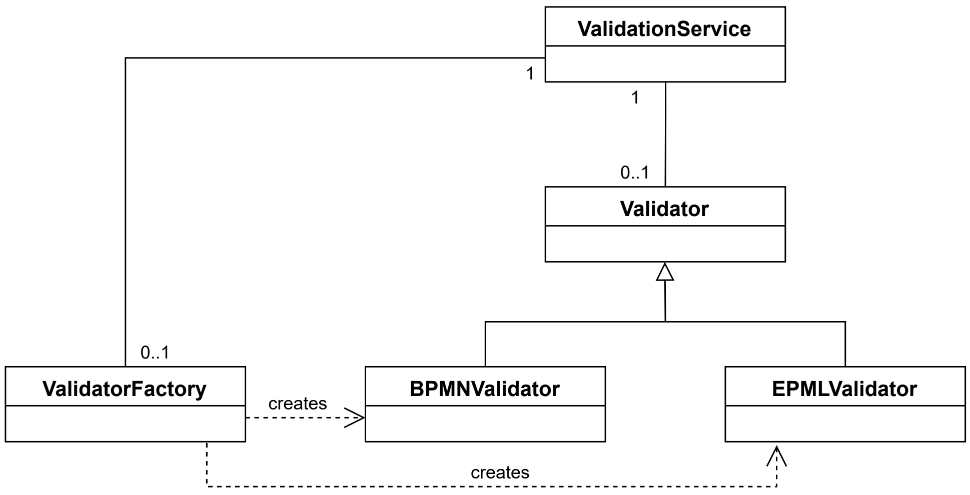


FIGURE 6.4: Class diagram of the validation services.

6.1.4 Transforming Models

The goal of the *Transformation* service is to convert a process model from its *source* metamodel to a *target* metamodel. While considering how to implement the required transformation services, we examined the Apromore codebase and discovered that a set of *plugins* existed which supplied the functionalities required to transform process models to a target language. These plugins will be referred to from now on as the *canoniser* plugins, and they provide the features presented in Section 4.2.2.

In light of this, it made more sense to reuse the functionalities of the existing *canoniser* plugins, rather than developing our own process model transformation service, which is out of scope for this thesis. However, although the plugins provided some useful functionalities to transform models, they did not represent a completely ready-made solution. The canoniser plugins provided a number of methods through their interfaces. To transform a model from its source language to the target, numerous method calls would be required in a particular sequence.

In order to tackle this problem, we realized that the *facade* [16] pattern was a good fit for this particular situation. The purpose of this pattern is to create a simplified representation of a more complex subsystem. This representation can also be described as a "wrapper" for the functionalities of the complex subsystem beneath the facade.

To transform a BPMN to an EPML model, a number of method calls must be performed upon the canoniser plugins. We realized that encapsulating the method calls together in one place and provide a *single point of access* would *simplify* the transformation process, rather than performing a series of method calls directly upon the canoniser plugins themselves.

To implement the facade pattern, we created a new class classed *TransformationFacade*, which wrapped the functionalities of the *canoniser* plugins to perform transformations.

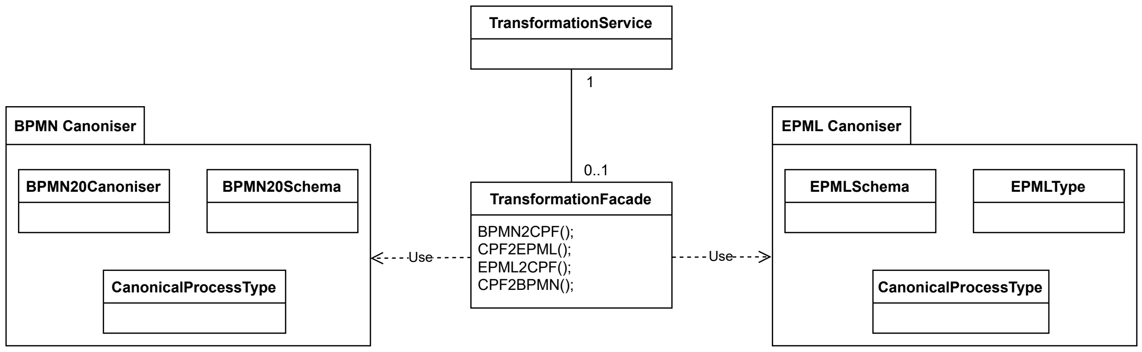


FIGURE 6.5: Class diagram of the transformation service.

As illustrated in Figure 6.5, the *TransformationFacade* provides multiple methods for initiating a particular transformation. When one of the methods are called, they perform numerous method calls to the underlying canoniser plugins, in order to perform the actions needed to transform models. Hence, the complex sequence of method calls is completely obscured from the clients using the facade, which is, in this case, the *TransformationService.*

6.2 The Storage Layer

The storage layer provides persistence capabilities for process models and associated metadata. As discussed earlier, process models are stored both in their original and canonical process format for the reasons laid out in Section 4.2.2. We are interested in storing large volumes of models and additional metadata. While well known traditional RDBMS solutions exist such as MySQL and PostgresSQL, which are easy to set up and query. They are not well suited for Big Data analytics, requiring support for large quantities of data, and accommodations for the real-time characteristics of Big Data [28]. In the presence of large amounts of data, an RDBMS requires the creation of numerous indices, which imposes some significant performance drawbacks during the data updating process.

Taking the above mentioned problems into consideration, we propose a document-based storage approach utilizing *Elasticsearch* as a database back-end, an open-source search engine, that is designed to be scalable, distributive, and real-time capable [14].

6.2.1 Elasticsearch

Elasticsearch is an open-source search engine, providing full-text search. It is designed to be scalable, distributive, and real-time capable. A running Elasticsearch instance is called a *node*, and two or more nodes may form a Elasticsearch *cluster* [28]. While RDBMSs and ElasticSearch may differ in various ways, many core concepts of a traditional RDBMS are analogous to the concepts in the Elasticsearch world as depicted in Figure 6.6. A *field* is like a *column* in a RDBMS: it stores a value of a certain data type. However, a field may store multiple values, essentially becoming a list, whereas a column is restricted to one value only. Analogous to a RDBMS *table row* is a *document* in Elasticsearch. A *document* is essentially a JSON object in Elasticsearch. It may include multiple *fields*, similar to *rows* which may include multiple *columns*. A *document type* is analogous to an RDBMS *table*, since it defines the fields that can be specified for a certain document. *Mappings* are similar to *schema* definitions in SQL databases. They define all the document types within an index. Lastly, an *index* is like a *database* in a RDBMS: providing storage, search, and update capabilities for different types of documents.

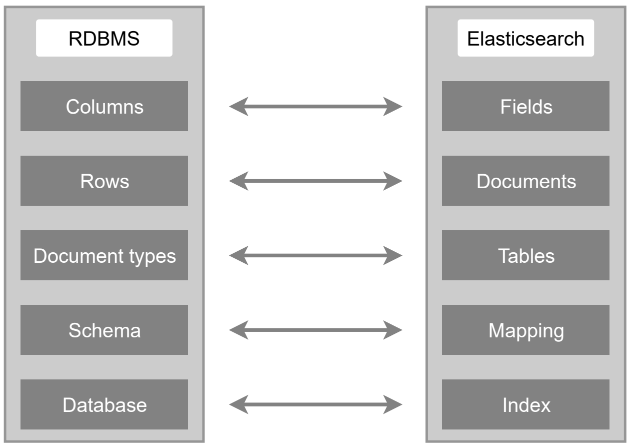


FIGURE 6.6: RDBMS concepts translated to Elasticsearch.

6.2.2 Models and Metadata

Similar to a *schema* in an RDBMS, a *mapping* defines the fields and datatypes that reside in an index. The mapping consists of two index definitions: an index for the models and an index for domains. The model index consists of the following fields:

- id: the primary key (PK) of the model table assigned by the database.
- name: the name of a model.
- file_name: the file name of the model.
- description: a long or short description of the model.
- version: the version number.
- modeling_language: the file name of the model.
- path: original path to the model.
- model: the contents of the model.

- c_model: the canonical format of the model.
- elements: metadata about the canonical elements.
- repository: metadata about the original repository.
- domains: domain metadata.

The domain index consists of the following fields:

- id: the primary key (PK) of the domain assigned by the database.
- name: name of the domain.
- tags: a list of domain tags.

Figure 6.7 depicts the conceptual schema of the storage layer. A *Repository* may contain multiple *Models* and *Domains*. A *Model* consists of its original file contents and the canonical process type format. It is composed of different versions and may contain multiple domains. A *Version* is a specialization of a *Model*. A *Domain* is composed of multiple *tags* and may reference zero or multiple models.

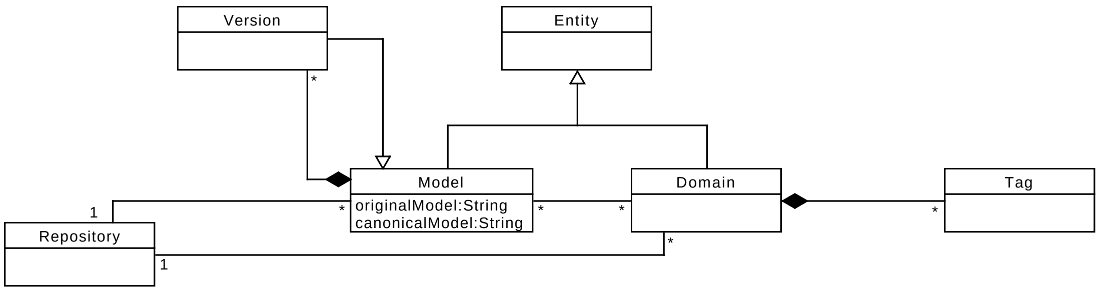


FIGURE 6.7: Conceptional schema of the storage layer.

6.3 The Presentation Layer

The presentation layer provides the user with the ability to interact with MAAS. It presents a Graphical User Interface (GUI), allowing users to execute commands and queries on the underlying system. The GUI communicates with the repository management layer through a REST API. Hence, two ways of interacting with MAAS are (1) through the GUI, (2) directly through the REST API. In this section, the implementation of the presentation layer is presented.

6.3.1 Implementation

The implementation of the presentation layer is quite simple. We only need to expose the functionalities provided by the model repository. In other words, we only need to connect the layer with the REST API. The implementation of the presentation layer uses a simple web-based interface since implementing a GUI found in real model repositories is not the goal of this thesis, and it takes a lot of time to realize. Hence, the GUI of the model repository system is basic, which only displays the minimal UI components required to perform CRUD actions and other commands on the system.

The interaction with the system is done using the GUI or directly through the REST API. The user can interact with the following aspects of the system:

Model Crawling

The API exposes services to initiate or terminate an already running crawling job. To initiate a crawling job, the user is presented with a set of options. First, the user can select a crawling method, which is further detailed in Section 6.1.1. Next, the user is presented with the option to select a model format to crawl (e.g. BPMN or EPML). Once the required options are set, the user can initiate a crawling job. The following is a list of functionalities exposed by the crawling API:

- *StartCrawler*: This function tells the crawler system to start mining GitHub for process models in the specified format. It supports both the GHTorrent and the GitHub Search methods. It also instructs the system to prepare models for analysis. For the BPMN mining example, the *StartCrawler* function instructs the collected models to undergo a model preparation pipeline, by consulting the *filtering* and *transformation* systems in the local cloud.
- *StopCrawler*: This function instructs the system to terminate all currently running *Crawler jobs.*
- *Status*: It is used to get the latest update regarding the number of collected models, links, and whether the crawler is idle or not.

Model Filtering

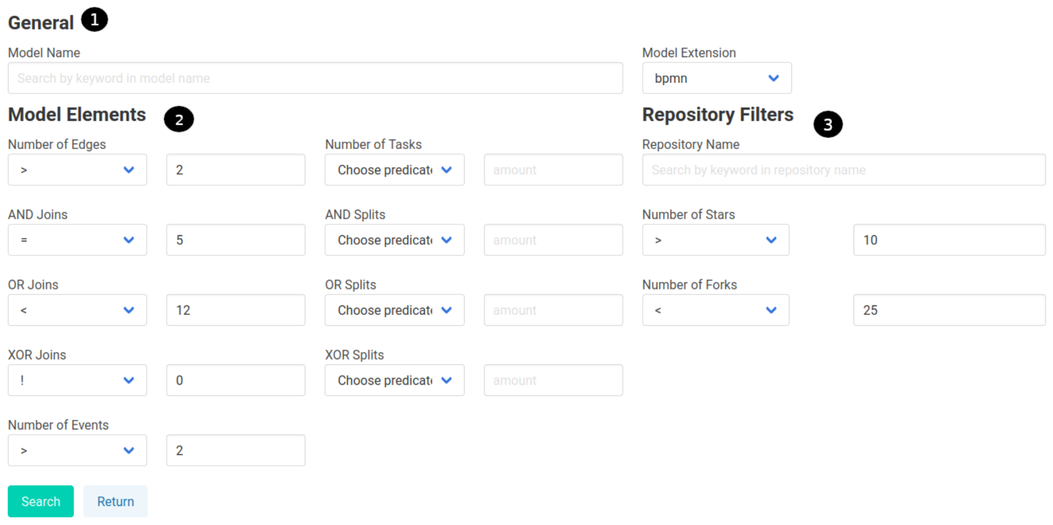
Another service exposed by the API is the model filtering service. The web interface of the model filtering service is presented in Figure 6.8. As shown, users can initiate a filtering request based on different types of criteria.

General filters (1), can be used to filter models on their name. For example, the name “medical” will return all models containing this string. It is also possible to filter models on their model extensions (e.g. files ending with “.bpmn/.epml”).

Models can also be filtered based on their canonical process format elements. Users can specify a certain predicate (e.g. “<”, “>”, “=”, “!”) followed by a number, giving the possibility to perform the following search query: “numberOfEdges > 0 AND numberOfEvents = 3”.

Finally, users can also filter models on their repository metadata (e.g. number of stars, forks) (3) and repository name.

By pressing the “Search” button, the *filtering system* will be instructed to collect and return models satisfying the search criteria, giving users the possibility to downloaded the models as a bundled archive.


General 1

Model Name: Search by keyword in model name

Model Extension: bpmn

Model Elements 2

Number of Edges	>	2	Number of Tasks	Choose predicate	amount
AND Joins	=	5	AND Splits	Choose predicate	amount
OR Joins	<	12	OR Splits	Choose predicate	amount
XOR Joins	!	0	XOR Splits	Choose predicate	amount
Number of Events	>	2			

Repository Filters 3

Repository Name: Search by keyword in repository name

Number of Stars	>	10
Number of Forks	<	25

FIGURE 6.8: Screenshot of the GUI's model filterer.

Model Management

In our proposed system, model management refers to performing CRUD actions on a set of models. Following is a list of functionalities exposed by the API:

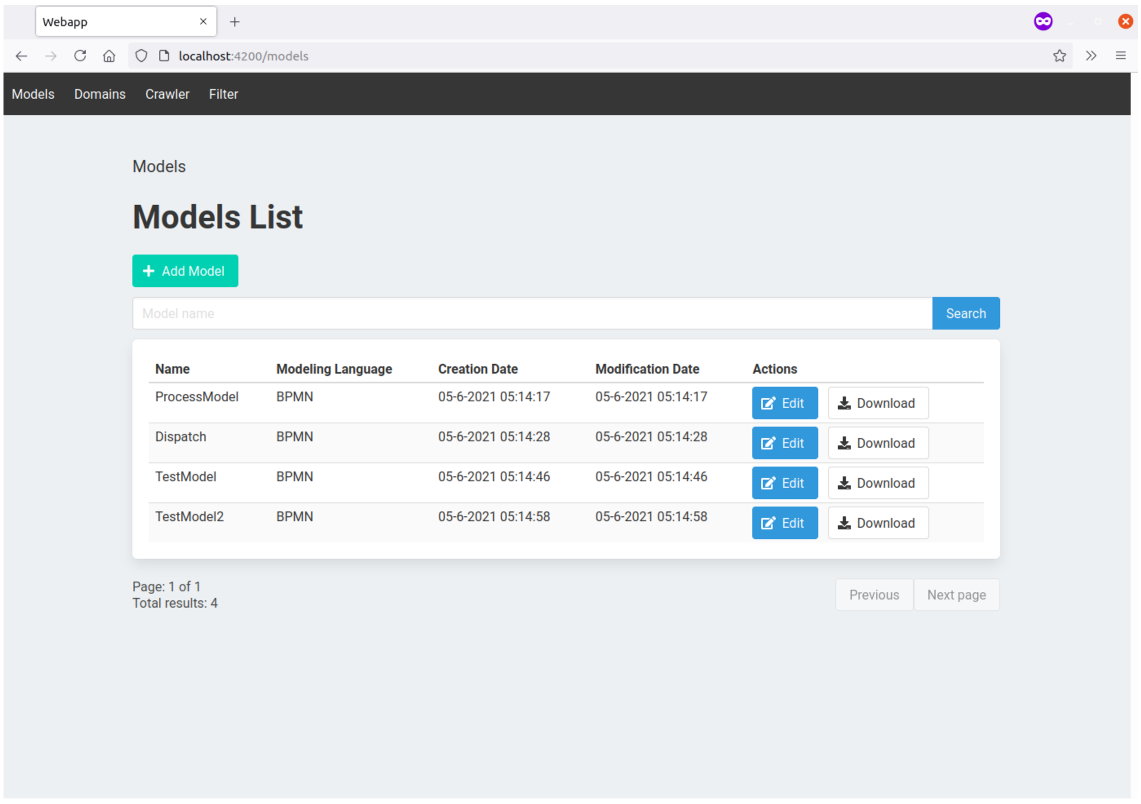
- *Create*: The create function is used to upload a model along with additional metadata to the model repository.
- *Get*: This function is used to search for the most recent version of a model given a unique identifier (i.e. "id"). In addition, a version history of the model is also given.
- *List*: It is used to get a collection of the most recent versions of available models in the repository.
- *Search*: Search is used to find a model by its "name" attribute. It is not required to provide the full name of the model. Therefore, it is possible to fetch a model by its partial name.
- *Update*: This function updates an already available instance of a model given its unique identifier.
- *Delete*: The delete function is used to remove an existing model from the repository given its unique identifier.

The model management service will generate a JSON response whenever a call to one of these functionalities is made. An example of the generated output of the *Get* function:

```
"id": 38,
"name": "Dispatch-of-goods",
"file_name" : "file1.bpmn"
"description": "A bpmn diagram",
"modeling_language": "bpmn",
"model": ...
```

```
"versions": [
    {
        "id": 86,
        "name": "Dispatch-of-goods",
        "file_name" : "file1.bpmn"
        "description": "A bpmn diagram",
        "modeling_language": "bpmn",
        "model": ...,
        "version_number": 1.0,
        ...
    },
...
...
]
```

Figure 6.9 illustrates the web interface section for managing models. The interface allows users to query models by the "name" attribute. Query results are presented in the form of a list. By pressing the "Edit" button, it is possible to view the details of the model (i.e. metadata) and apply some modifications. By pressing the "Add Model" button, users can upload their own process models and associated metadata.



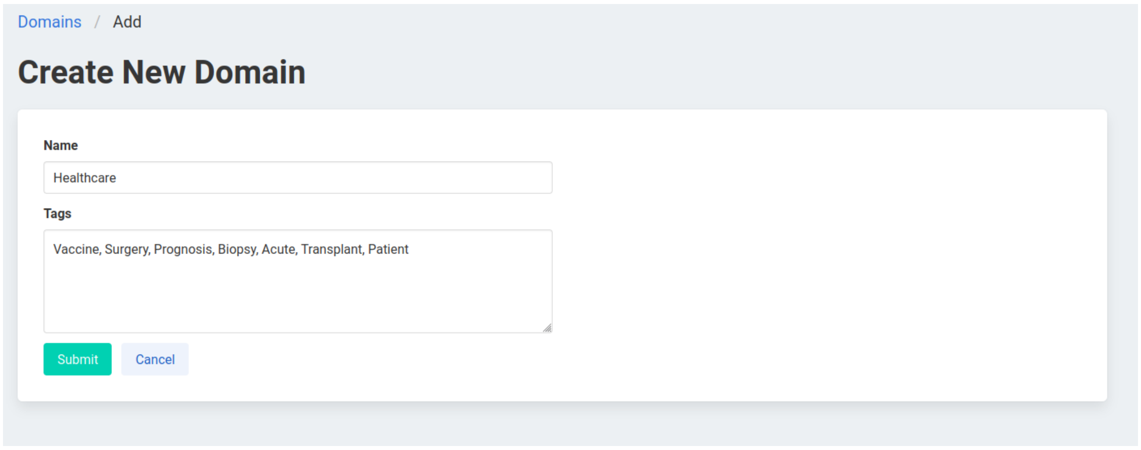
Name	Modeling Language	Creation Date	Modification Date	Actions
ProcessModel	BPMN	05-6-2021 05:14:17	05-6-2021 05:14:17	Edit Download
Dispatch	BPMN	05-6-2021 05:14:28	05-6-2021 05:14:28	Edit Download
TestModel	BPMN	05-6-2021 05:14:46	05-6-2021 05:14:46	Edit Download
TestModel2	BPMN	05-6-2021 05:14:58	05-6-2021 05:14:58	Edit Download

Page: 1 of 1
Total results: 4

FIGURE 6.9: Screenshot of the GUI's models list page.

Domain Management

Similar to model management, domain management refers to performing CRUD actions on a set of domains. Users can specify a number of application domains (e.g. healthcare, banking, and retail) through the domains interface along with associated tags/labels. The system uses the labels to compute a similarity check between the domains and the collected process models. Figure 6.10 shows an example of defining a new application domain including associated tags.


Domains / Add

Create New Domain

Name

Healthcare

Tags

Vaccine, Surgery, Prognosis, Biopsy, Acute, Transplant, Patient

FIGURE 6.10: Example of an application domain consisting of a set of related tags.

CHAPTER 7

Evaluation

This chapter describes how the different aspects of MAAS are evaluated. In particular, Section 7.1 describes the evaluation methodology. Then, in Section 7.2 the evaluation results are provided. Lastly, Sections 7.3 presents the observations based on the evaluation results.

7.1 Methodology

In order to investigate what impact MAAS has in the model analytics workflow of researchers, we decided to conduct a case study taking a single-case, holistic design [54]. This means that MAAS will be the global unit of analysis. Our approach uses a combination of qualitative and quantitative data analysis, inspired by the work in [36]. In particular, we collect qualitative data through interviews and quantitative data through a questionnaire. With the interview results, we aim to understand MAAS's strong/weak points and why the participants had particular experiences. In contrast, the questionnaire results will serve to understand how the participants experienced the usability aspects of MAAS. In addition to this, a benchmark will be conducted, comparing the manual approach with the new approach using MAAS. Overall, with the results of this case study, we aim to derive new hypotheses and build theories regarding the effectiveness of MAAS in the model analytics workflow of researchers.

7.1.1 The Case Study

As described above, the case study of MAAS consists of a questionnaire, interview, and a benchmark. The interview questions and questionnaire can both be found in Appendix A. The questionnaire consists of 12 questions on a seven-point Likert scale about MAAS that are based on the Technology Acceptance Model (TAM) [32, 36]. Furthermore, we modified each question to talk about MAAS more specifically. In addition to the 12 questions, we incorporated a Net Promoter Score (NPS) question [41] to get a better understanding of whether or not researchers would recommend MAAS to others: "How likely is it that you would recommend MAAS to others?".

The TAM questionnaire consists of six perceived usefulness (PU) and six perceived ease-of-use (PEU) questions. PU is defined as "The degree to which a person believes that using a particular system would enhance their job performance." [10]. On the other hand, PEU refers to "The degree to which a person believes that using a particular system would be free of effort." [10]. The purpose of TAM is to predict future use instead of rating the experience of actual use [36]. We will therefore use the modified TAM (mTAM) proposed by Lah et al. [32]. In contrast to TAM, in mTAM, respondents can indicate a rating regarding actual user experience instead of anticipated use. Similar to TAM, the mTAM questionnaire consists of six PU and six PUE questions, which can be found in Appendix A.1.

A semi-structured interview will serve as the main input of the case study. This means that questions are planned before the interviews but not necessarily asked in the same order as they are listed. In addition, probing questions will be asked with the aim to get more specific or in-depth information about a subject. By following the interview guidelines [46], we designed 11 interview questions listed in Appendix A.2. With the interview questions, we aim to understand MAAS's strong/weak points regarding usability, workflow automation, and performance.

Participants were selected based on their experience with the Arrowhead Framework and model management and analytics. Furthermore, participants were asked via email to participate (voluntarily) in the case study. This amounted to three participants with hands-on experience with the Arrowhead Framework and some background in model management and analytics. Two of the participants are PhD candidates, and one participant is a PDEng student. All of the participants have varying knowledge about the concepts of the Arrowhead Framework. An online "tutorial" session was held with the aim of getting the participants on the same level of understanding. During this session, a demo of MAAS running on the Arrowhead Framework was given, and participants were asked to perform some tasks using MAAS with the possibility to ask questions. After the session, participants were given four days using MAAS to crawl BPMN models from GitHub, filtering the crawled models based on some criteria, and transforming the filtered models to the EPML format.

The mTAM questionnaire was distributed to the three participants via email as a Word file which the participants could fill in. The interviews were held online through Microsoft Teams, and every interview took approximately 30-40 minutes. All interviews were recorded with the permission of the participants, and after each interview session, audio recordings were transcribed manually, aiming to be as complete as possible. In some cases, filler words, interjections, and stutters were excluded as they did not directly contribute to the evaluation. However, the remainder of the interview transcripts were left verbatim. This resulted in a transcript length of 10 pages on average per interview, which can be found in Appendix B.2. Furthermore, audio transcripts were sent to the corresponding interviewees with the opportunity to give some feedback.

The interview transcripts were analyzed using the card sorting method [8, 36], consisting of the *preparation phase*, *execution phase*, and *analysis phase*. First, during the preparation phase, we selected statements/paragraphs about MAAS and turned these into so-called *cards*. In some cases, irrelevant statements, e.g., in-between conversations not relevant to answering the research questions were omitted. Next, during the execution phase, we put the extracted cards in an Excel sheet and sorted the cards into meaningful groups. In general, there are two types of card sorting. *Open* card sorting does not have predefined groups, i.e., the groups are identified during the card sorting process. *Closed* card sorting on the other hand does have predefined groups. In our case, we used the open card sorting method. Lastly, for the analysis phases, we constructed a hierarchical model of the identified themes, aiming to represent the mental model of the participants regarding MAAS.

7.1.2 Threats to Validity

In order to assess the validity of our empirical study, we use the following validity aspects [54, 13]: internal validity, external validity, conclusion validity, and construct validity. Each aspect is discussed further in more detail below.

Internal validity considers the study design itself and is used to determine whether the results really do follow from the data. In our case, a factor to consider is the fact that the participants have varying knowledge about the Arrowhead Framework and model management and analytics in general. For instance, one participant may be more adept with certain tools related to model analytics compared to another participant. This difference in knowledge may affect how the participants feel about the usability and overall effectiveness of MAAS. To reduce this knowledge gap as much as possible, we hosted an online tutorial session where we asked the participants to perform a set of tasks using MAAS. In addition, participants were allowed to ask questions regarding the tool and the Arrowhead Framework.

External validity focuses on whether claims regarding the generality of the results are justified. In our case, the target audience of MAAS are researchers, and the case study was conducted with two PhD candidates having backgrounds in computer science and one PDEng student having a background in mechanical engineering. In general, the population of researchers have different characteristics and come from different backgrounds. Because of this, we cannot be certain whether the results of the study can be generalized to researchers in general. Another factor to consider is the low number of participants in the study, which may affect the generalizability of the study as well.

Conclusion validity considers whether the conclusions reached in a study are correct. A factor to consider here is the card sorting method used to analyze the interview transcripts. In particular, we selected paragraphs/statements about MAAS and turned these into cards. Consequently, we identified themes that best represented the card statements. A risk to consider here is that the card sorter can introduce some bias in categorizing the cards, e.g., *fishing* for a specific result. To reduce this bias, we conducted card sorting with another software engineer.

Construct validity considers whether the theoretical constructs are interpreted and measured correctly. In our case, this refers to the question of whether our study measures the effectiveness of MAAS in the model analytics workflow of researchers. First of all, this question is very broad and not really quantifiable. We have, therefore, defined two empirical sub-questions as presented in Section 3.2. One question considers the *performance* aspect and another considers the *usability* aspects of MAAS. For both questions, we used a combination of qualitative and quantitative research techniques. A risk to consider is the experience level of the user and the specifications of the test machine for the benchmark, which may affect the results. However, we do think that the benchmark provides an indication of how the manual approach compares to the automated approach. Furthermore, we constructed the interview questions according to the guidelines [46], but there is no guarantee that they are the right questions.

7.2 Evaluation Results

The interview transcriptions and questionnaire results can be found in Appendix B. In Section 7.2.1 we describe our findings based on the questionnaire and benchmark results. Lastly, in Section 7.2.2 we explain how we analyzed the audio transcriptions and report our findings.

7.2.1 Quantitative Results

Using the questionnaire results (see Appendix B.1), we computed mTAM scores for PU and PEU following the approach in [32]. MAAS scored 94 on PU and 67 on PUE, leading to an overall mTAM score of 81. Unfortunately Lah et al. [32] do not specify a way to interpret the mTAM values. On the other hand, mTAM is intended to be similar to the System Usability Scale (SUS), and we use corresponding guidelines to interpret SUS scores [5]. By following these guidelines, MAAS with an mTAM score of 81 would be rated *acceptable* on a acceptable/not acceptable scale. Furthermore, MAAS would get a B on an American grading scale, and a *excellent* on an adjective scale.

As for the NPS question, two participants answered with an eight, and one participant answered with a nine, meaning that two participants are *neutral*, that is, being neither promoters nor detractors and one participant is a promoter [41]. This translates to an NPS score of 33, which can be considered as *good*.

Regarding the benchmark, we measured the *model preparation time* between the manual approach and the automated approach using MAAS. We define *model preparation time* as the execution time to collect ten process models from GitHub, validating the process models against their metamodel, perform filtering, and finally transforming the filtered models to a target format. We collected ten process models from GitHub manually and utilized Apromore for storage for the manual approach. Consequently, we used Apromore to validate the collected process models, conduct filtering and finally transforming the filtered models to a target format. In contrast, for the automated approach, we incorporated MAAS to perform the collection/preparation of the ten process models. A comparison between the two methods is depicted in Table 7.1.

	Using MAAS	Preparation Time
Preparing 10 process models	NO	10 min
Preparing 10 process models	YES	1 min

Table 7.1: Comparison between the two methods of collecting/preparing process models.

7.2.2 Qualitative Results

As mentioned in Section 7.1.1, the interview transcriptions were analyzed using the open card sorting method. During the card sorting session, categories we rearranged, and in some cases, deleted or merged if they were redundant. This process amounted to 66 cards grouped into 29 categories. At the highest level, we found three main categories: tops, tips, and neutral statements. Furthermore, we found subcategories and even subsubcategories within the high-level categories. According to the card sorting method, a hierarchical structure was constructed depicting the identified categories and their relations, which are illustrated in Figure 7.1.

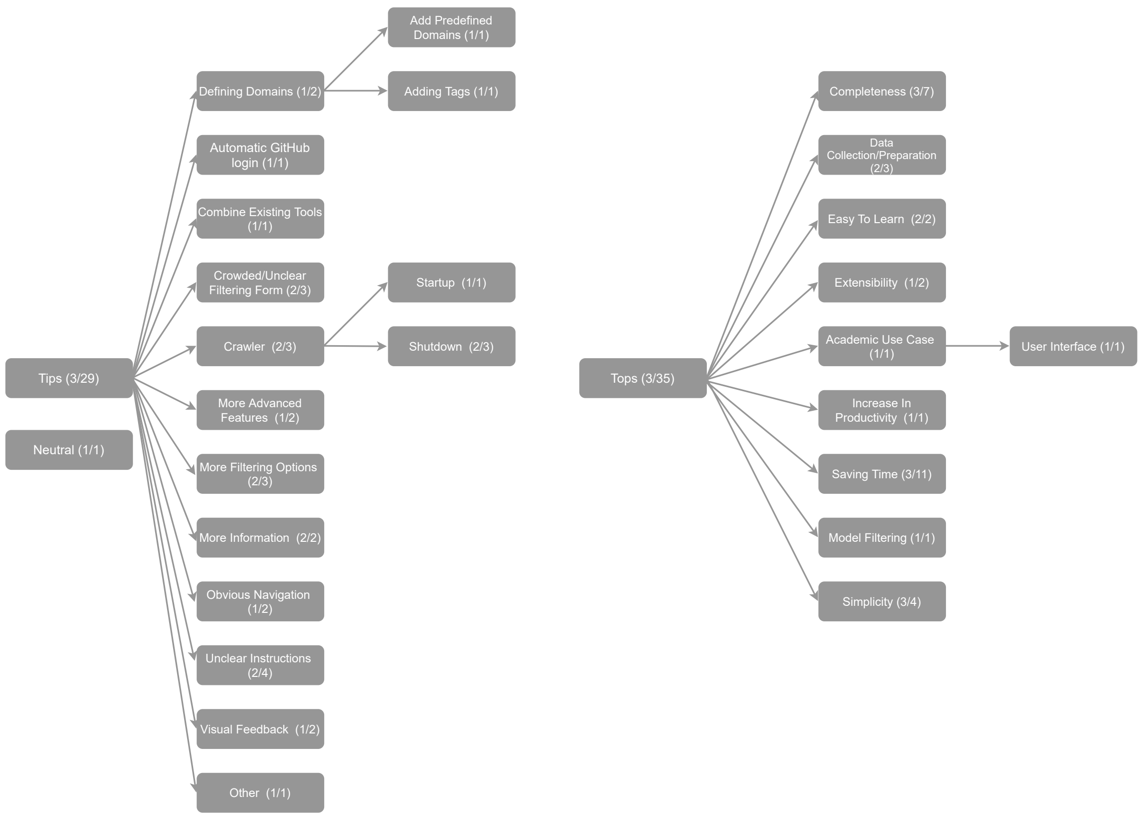


FIGURE 7.1: An overview of the identified categories after the card sorting session. Numbers in parentheses represent: *number of participants/number of statements*.

7.3 Discussion

Based on the quantitative and qualitative results of the case study, we can make the following observations about MAAS:

- Generally, participants are satisfied with MAAS (35 tops compared to 29 tips and an mTAM score of 81). They believe that MAAS incorporates most of the required functionality and that it saves a significant amount of time in collecting/preparing models for analytics as demonstrated by the benchmark.
- Most tips refer to the user interface of MAAS and nice-to-have features such as tooltips, visual feedback, and hints. In contrast, most tops refer to the effectiveness of MAAS regarding model collection and preparation. These observations are reflected in the questionnaire results, where PU scored significantly higher than PUE.
- Participants feel that MAAS is easy to use and learn. More specifically, the user interface is simple, and the provided options do not overwhelm the user.
- Most participants had trouble with operating the model crawler, in particular with terminating an already running crawler.
- Overall, participants would have liked more filtering options and the ability to hide segments of the filtering form as they felt the user interface was too crowded/populated.

- There is less consensus when it comes to tips compared with tops. More specifically, there are tops that all three participants mention. On the other hand, no tips are mentioned by all participants.
- The tip regarding automatic GitHub login is something we cannot solve at the moment due to how GitHub handles user authentication.

The case study results provide some valuable insights regarding the effectiveness of MAAS in the model analytics workflow of researchers. The results suggest that MAAS is easy to use and it saves a significant amount of time in the model analytics workflow. However, due to the low number of participants, whether the evaluation results can be generalized is yet to be seen. In light of this, we hypothesize that: “MAAS is perceived as an effective tool in reducing the effort it takes to prepare models for analytics”.

CHAPTER 8

Conclusion and Future Work

This chapter provides conclusions based on the literature study, implementation, and evaluation results. The chapter concludes with the potential future works for the continuation of this thesis.

8.1 Conclusion

This section iterates over the research questions defined in Section 3.2 and answers those research questions based on the literature study, implementation, and evaluation results.

RQ 1: How can we develop or integrate a model analytics automation system in the Arrowhead Framework?

Based on the literature study on the state of art model repositories, it can be concluded that there are several tools for managing and storing models. In particular, they provide check-in/check-out, model versioning, collaborative modeling, and model analytics features. We have also looked at several tools that implement features that are also of interest, namely, repository mining, model validation, and model transformation. We designed an architecture for our model analytics automation system and implemented a running prototype on the Arrowhead Framework. Our proposed system is elaborated in Chapters 5 and 6.

The architecture of MAAS is based on a three-tier architecture consisting of the presentation, repository management, and storage layers. The presentation layer provides the interface of MAAS, allowing users to interact with the underlying model repository system and model preparation pipeline. The repository management layer is the central entity of MAAS, consisting of the business logic of the core components. Furthermore, it acts as an intermediate layer between the presentation and storage layers. The storage layer contains the data of the software architecture, where a set of services collaborate to deal with the system's data.

Our approach consists of adapting the SOA-based local automation cloud architecture of the Arrowhead Framework, which provided many advantages. First, it adds an abstraction layer by which new systems providing additional services can be included into MAAS and all the systems in the local cloud. Hence, a software engineer could design and implement a new system to automate additional steps or add new features enhancing the model analytics workflow. Furthermore, we implemented several *core* systems, providing services to crawl, store, validate, filter, and transform models, all running as individual systems in the local cloud, exchanging services to perform an automation task. The architecture of these systems permits *flexibility* by applying several software design patterns such as the *factory method*. Hence, software engineers can extend these systems with additional features such as page parsers for different software repositories. And last but not least, the architecture is able to scale up to meet increased workloads, by incorporating stronger or additional devices in the local cloud.

RQ 2: How effective is the system in reducing the effort it takes to prepare models for analytics?

This thesis aims to design and implement an effective treatment supporting the model analytics workflow of researchers. To determine whether the treatment produces the desired effects in a real-world scenario, we conducted empirical research. More precisely, we are interested in how effective the treatment reduces the effort it takes to prepare models for analytics. To answer this empirical question, a single-case study taking a holistic design was conducted. This means that the treatment was the global unit of analysis. This empirical research question was divided into two sub-questions which are answered in the following.

RQ 2.1: Can we reduce the time it takes to prepare models for analytics?

Based on the interview and benchmark results, we can answer that we can significantly reduce the time it takes to prepare models for analytics. Instead of manually collecting process models from software repositories, validating, filtering, and finally transforming them, we can automate the complete process through the system. Thus the execution time to collect/prepare models is significantly reduced, as demonstrated by the benchmark comparing the manual approach with the automated approach. Furthermore, the majority of participants indicated that the system would reduce a great amount of effort in preparing models for analytics and thus effectively improving the job performance of researchers.

RQ 2.2: How easy is it to interact with the system?

The questionnaire and interview results have shown that the system is simple and easy to interact with. Furthermore, participants were generally positive about the system (35 tops compared to 29 tips). Moreover, an mTAM score 81 indicates that the participants are overall satisfied with the usability aspects of the system. However, participants had widely different recommendations and nice-to-have features regarding the user interface. It, therefore, seems unattainable to develop a user interface that meets the requirements of everyone involved with model management and analytics. However, we do believe that the interface provides sufficient flexibility and options that make interacting with the underlying system easier to perform a particular job.

8.2 Recommendations for Future Work

This thesis has been subject to time limitations that bound its achievements. For future work, there are several directions to continue this research.

1. As mentioned in Section 5.3, we only used the core Arrowhead systems to implement the systems that form the model preparation pipeline of MAAS. In the future, we recommend incorporating the choreographer system [30] into MAAS. This system makes it possible to execute pre-defined workflows by levering the orchestration and service consumption of the Arrowhead Framework [3]. Each workflow consists of three mandatory components: plans, actions, and steps. With plans, it is possible to define a workflow by name, consisting of a set of actions grouping together several steps. Through these workflows, it is possible to execute a number of functionalities provided by the systems in a particular sequence.

2. As described in Section 6.1.1, we successfully designed and implemented a model crawler leveraging multi-threading to run crawling jobs in parallel. However, this design is bounded to one machine/device only. Hence, the design can only scale up vertically and not horizontally by adding a more powerful machine. In order to scale horizontally, we recommend designing a distributive crawler. Essentially, allowing the system to handle the increased workload by adding more devices to the local cloud.

3. We have implemented a heuristic domain mapping mechanism, which extracts all the node names from a process model and computes a similarity score between the defined domains. However, this method is not accurate in mapping domains. Therefore, we recommend incorporating more sophisticated techniques in mapping domains with process models such as machine learning.

4. Overall, based on the evaluation results, we would recommend improving the user interface. In particular, improving the user-friendliness of the interface, by incorporating clear instructions, hints, and visual feedback. While the current implementation of the interface provides the required functionality to operate MAAS, it is very basic and requires some technical knowledge from the user. Furthermore, we recommend streamlining the user authentication process of the GitHub crawler.

5. We evaluated the usability aspects of MAAS using a semi-structured interview and an mTAM questionnaire. However, because of the low number of participants, we cannot claim with certainty that the results can be generalized to a broader audience. A further improvement would be to replicate the interview and questionnaire with a larger sample size.

APPENDIX A

Case Study Documents

A.1 Questionnaire

Think about the tasks that you performed with the Model Analytics Automation System (MAAS) while you answer these questions. Some statements refer to a “job”. For these statements, you can interpret model analysis (e.g. clone detection on large quantities of process models) as a job. Please read each statement carefully and indicate how strongly you agree or disagree with the statement (**1 = extremely disagree** and **7 = extremely agree**) by putting an X in the appropriate table cell.

	1	**2**	**3**	**4**	**5**	**6**	**7**
Using MAAS in my job enables me to accomplish tasks more quickly.							
Using MAAS improves my job performance.							
Using MAAS in my job increases my productivity.							
Using MAAS enhances my effectiveness on the job.							
Using MAAS makes it easier to do my job.							
I have found MAAS useful in my job.							
Learning to operate MAAS was easy for me.							
I found it easy to get MAAS to do what I want it to do.							
My interaction with MAAS has been clear and understandable.							
I found MAAS to be flexible to interact with.							
It was easy for me to become skillful at using MAAS.							
I found MAAS easy to use.							

Could you answer the following question, where **0 = not likely** and **10 = very likely**.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
How likely would you recommend MAAS to others?											

A.2 List of Interview Questions

1. **Background.**
   (a) Can you tell me a bit about yourself and your experience with the Arrowhead Framework?
2. **Usability.**
   (a) How easy or hard was it using MAAS?
   (b) How easy or hard was it learning to use MAAS?
   (c) How useful do you think MAAS might be for preparing models for analytics?
   (d) Are the UI elements a good choice for the data management aspects of the system (i.e. models, domains)? What would you add, change?
   (e) Is the proposed "Filtering form" a useful approach for model filtering? Why/Why not?
3. **Workflow Automation.**
   (a) How do you think MAAS is going to impact the model analytics workflow of researchers?
   (b) Would you say that the provided core systems (providing: crawling, validation, transformation, filtering) are sufficient in covering the steps in the model analytics workflow of researchers? What steps are still missing? What additional steps would you like to be automated by the system?
4. **Performance.**
   (a) How much time do you think you have saved by using MAAS, instead of having to use a manual approach?
   (b) Do you think that MAAS (providing: crawling, validation, transformation, filtering) will reduce the effort it takes for researchers to prepare models for analytics? Why/How?
5. **Other.**
   (a) Any additional feedback that you might have for MAAS?

APPENDIX B

Case Study Results

B.1 Questionnaire Results

Table B.1 presents the results of the questionnaire. Each column refers to a particular questionnaire question as follows:

- Q1: Using MAAS in my job enables me to accomplish tasks more quickly.
- Q2: Using MAAS improves my job performance.
- Q3: Using MAAS in my job increases my productivity.
- Q4: Using MAAS enhances my effectiveness on the job.
- Q5: Using MAAS makes it easier to do my job.
- Q6: I have found MAAS useful in my job.
- Q7: Learning to operate MAAS was easy for me.
- Q8: I found it easy to get MAAS to do what I want it to do.
- Q9: My interaction with MAAS has been clear and understandable.
- Q10: I found MAAS to be flexible to interact with.
- Q11: It was easy for me to become skillful at using MAAS.
- Q12: I found MAAS easy to use.
- Q13: How likely would you recommend MAAS to others?

The values for questions 1-12 range from **1 = extremely disagree** and **7 = extremely agree**. The values for question 13 range from **0 = not likely** and **10 = very likely**.

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13
A	7	6	6	6	6	6	6	4	5	4	6	5	9
B	7	5	7	7	7	7	5	5	4	3	4	5	8
C	7	7	7	7	7	7	6	6	6	5	5	6	8

Table B.1: The results of the questionnaire.

B.2 Interview Results

This section presents the audio transcriptions of the three interviews.

B.2.1 Interview Results: Participant A

Interviewer: Okay, it's recording. I will just go over the questions. One by one. The first question is about your background. So can you tell me about yourself and your experience with the Arrowhead Framework?

Participant A: Yeah, I actually, my let's say, my first interaction with the Arrowhead Framework was when I started my PhD in 2019. And that's, that was actually part of my position, I mean, defined as my position, PhD position and where I wanted to cooperate in this project. First case was between theory and Philips, so there was an industrial case being done at Philips, but for some reason, we could not make progress there. But we started to build our own academic use cases. And, yeah, from there, I started to learn the framework to work on that. Get some hands on experience. Yeah, we, I mean, I've been working on the framework from that time. Not I've not been, I mean, using it, like, all the time, but at the beginning, especially it, I spent a lot of time on it to learn what's going on. But over time, still, we've been using it. There have been a lot of updates on the framework. I've been attending the different workshops, meetings. For the framework. Yeah, I've been following. I mean, technically, like, watching the progress, what is happening? Trying to adapt.

Interviewer: Yeah. And you already built some services or systems on the framework?

Participant A: Yeah, it was not like real systems. But so, we had two academic use cases that we started with. One wdents working on similar profile to provide users, designers, a platform so that they could design their systems. In Yeah, in there. We had some progress. I mean, I also was involved. We were working closely with him. We could we did not produce like a concrete system. It was about design.

Interviewer: Okay, so the architecture.

Participant A: Yeah, the architectural part, system of systems and everything. For the other projects that you're also working on now, part of it was about this model ecosystem. Model and analytics, workflow. And you're working on one part of the repository. But there's also this dashboard to visualize the models, and also the model analytics system that you built. Yeah. I mainly focused on the dashboard, and <*other researcher name*> is responsible for the analytics part because that was his own research. Yeah, I had some experience building a system there. And working with the toolchain, it was not very advanced. But still, we had some working.

Interviewer: And your experience with model analytics and model management?

Participant A: Yeah, that's actually my research in PhD, analytics and models, finding similarities, clones in models, and I'm working with this special type of models, called process models, business process models, you know, about BPMN and EPML. And yeah. The goal is to find efficient and let's say, write algorithms, tools. Develop, right tools for detecting these similarities as clones, in a good way, and then doing further analysis on these.

Interviewer: Okay. Yeah, so, I think that's, that's enough for the background. Let's go through the usability aspects of the tool. So how, how easy or hard was it using the system?

Participant A: Actually, it is. I think it is straightforward. But very difficult. The way you have made it into tabs, these different tabs of modules based on the functionality. This is good. This makes it easier to follow what is what you want to do. But I had a bit problem. I think with the connections between them. I did not use the tool. Much. But yeah, that was my first impression, actually, that the links could be better. How you move through these steps. Back and forth.

Interviewer: So do you mean from the crawler to the filtering part or?

Participant A: Yeah, maybe all of them?

Interviewer: Okay. Yeah.

Participant A: I think it kinda needs to be related in some way. And I just have it open. Now I'm looking at the UI to see what else is there. Like, these are minor things, but I don't know. I mean, generally, I can say it is good. It is easy to follow. We can see the options. What is there? Of course, you expected more.

Interviewer: Yeah.

Participant A: Yeah, you always expect the more advanced things. But I understand this is not on that level. I mean, this is not supposed to be at that level.

Interviewer: Yeah, the web application was supposed to be there. Yeah. Basic not the main part of the thesis.

Participant A: So yeah, if you want to comment on the UI, I mean, you can find different things. Why this is here, this should be like this. This should be like that. I can make such comments, but I don't know if it's useful for you.

Interviewer: Yeah, just the general. The main, the main thing, I guess, the yeah, what was the hardest thing to use?

Participant A: Yeah. At the beginning, for me, the hardest part was this crawler. Yeah. So it is fully automated. You don't need to configure anything. That's the good part of it. Outside of it. But here, the problem is to get it started.

Interviewer: Yeah, because we have this dummy account already set up in the background. But it doesn't work sometimes if you use it on a different machine, or on a different device. Because, yeah, that's how GitHub is set up. Yeah, there is no other way to do it. So then you have to use your own account credentials.

Participant A: As from a user's perspective, this is not really a good thing. But again, just we look at this as an academic case, and we are just looking at how the system functions. I think we are not focusing much on the, let's say, the user interface part. So in that sense, I think that is acceptable.

Interviewer: Yeah. Okay. And how easy or hard was it learning to use the system? So the first one was about using, and the second one was about learning to use the system?

Participant A: Yeah, learning. I think it was quick. I think you can learn like the whole thing in a few minutes. Yeah, you don't need to spend hours on it. So yeah. This is easy, I think. I mean, yeah, easy, because you don't find a lot. I mean, too many options and features on the UI. Yeah. Maybe a big part of it is happening, like in the background?

Interviewer: Yeah. A lot of steps are automated,

Participant A: Automated. So actually, there are many options that you can deal with.

Interviewer: Yeah. Wouldn't you have liked to see more options? Or do you think at the moment, it's fine for the system to automate, these options for you?

Participant A: Yeah, I of course, I want it to be automated. Automation is key here. But I still would expect more options.

Interviewer: Yeah. And what kind of options did you like to see?

Participant A: Yeah, I mean, like, for each of them, for example, for the crawler, if I were to expect a bit more status, let's say options and live updates. Like for the filtering, a bit more instructions on how to use, yeah, total I mean, in general, I, I find the instructions a bit weak. Yeah. What is happening? I remember there was something, I don't know it was in filtering, or I think it was in filtering that you filter something, and then you search for it, it makes it a zip file, then you transform the files into other files.

Interviewer: Yeah, you can set the target language to transform.

Participant A: Yeah, for me, I mean, I could not understand this from the UI that this has happened.

Interviewer: So, it should be more user friendly in that part.

Participant A: Yeah. So yeah, the user should really be able to see that what is really happening at the moment, the connections. Yeah, maybe some little hints. Here, and there.

Interviewer: Some labels?

Participant A: Some labels, yeah. To put for the user to know that you, for example, some explanations. Here, we have this domain tab, for example, but I have no idea what it means. So I want to see this one paragraph here about what it means. So such things. And the connections between models like you're defining domains, and then where are you going to use these things? Do you want to see that in the models list? And then when you click on a model, things like that.

Interviewer: Yeah. So those are the main things, right?

Participant A: Yeah.

Interviewer: Okay, let's go to the next question. Yeah, so you already have some knowledge about model analytics. And so, yeah, I think also model crawling. Do you have some experience with that?

Participant A: I did not have previous experience.

Interviewer: Okay. Let's just say model analytics. So, how useful Do you think the system might be for preparing models for analytics? How useful do you think that might be?

Participant A: Yeah, I think it can be very useful if it does really what it says. Like, GitHub, that you're crawling, it is a big jungle. I mean, you really get lost in there. Yeah. If this tool really finds the right models for you. And yeah, it passes this all through these steps. You know, it, validates, transforms, stores the model in the right way. So we use it later, then I think this is a very helpful thing for a researcher. Okay, because, yeah, you provide the data. So data is the key. In your research, that's the main input. Yeah. So yeah, if you prepare this reasonably and acceptably for the user. I mean, based on what the user expects, then that is really great. I mean, this system is working well, I think.

Interviewer: yeah. So, yeah, we already discussed this, about the elements and the labels. So you have already seen the domain model page and the filtering page, right? Yeah, the UI elements, as you see them currently in the system. Do you think these are a good choice of UI elements for the CRUD aspects of the system?

Participant A: Yeah, you actually have, I think, the basic functionality on the UI. Yeah, maybe. I mean, you can always do a better job on the UI. This is I mean. I think there is no end to it.

Interviewer: Yeah, yeah. You can keep improving.

Participant A: Yeah, maybe. Also, I don't know if this is relevant, but maybe somehow show the flow to the user also on the UI that you have. These are what is happening, and these are what is supposed to happen. Where are we at the moment?

Interviewer: Okay. Yeah.

Participant A: Yeah, I think this this could also be nice.

Interviewer: So, show the steps or?

Participant A: Yeah, kind of the steps.

Interviewer: Yes, it would be nice to have. So, you have seen the domain page, and how to add domains? What do you think about adding tags to a domain? At the moment, we are just defining a comma-separated string. Do you think this could be better?

Participant A: This could be better, I think. Yeah, of course, this is like the basic, the most basic way of defining domains. And yeah, one thing is that I think there should be predefined domains. That could be nice.

Interviewer: Predefined domains? So?

Participant A: I mean, just a few examples. For example, health, automotive, let's say telecommunications.

Interviewer: Something like some default domains, then you can add additional domains if you want.

Participant A: Yeah.

Interviewer: Yeah, okay.

Participant A: And, yeah, and the link tags like this. I think this is very basic. It can be improved, like, I think having it like as a comma-separated form like this is not very user friendly. And yeah, I don't know how else differently this can be entered. But I can think of... I think there are better ways.

Interviewer: Yeah.

Participant A: I think the tags could be improved.

Interviewer: Yes, maybe a dynamic form is a better way to represent tags?

Participant A: Dynamic form?

Interviewer: Yeah. So maybe you can press on an Add button, and then it adds another text box for you to put in a tag, so perhaps that's a better design choice?

Participant A: But yeah, in that case, if you want to add a 100 tags, or a 1000 tags this can become problematic. Also, for the tags, maybe you could have you could read it from a file maybe not necessarily.

Interviewer: Yeah. But this file, should also follow some form? It should also be comma-separated, right?

Participant A: Yeah. Yeah, I was thinking maybe the user could use another tool for that. And, you know.

Interviewer: Yeah. And then make a comma-separated file.

Participant A: For example, in a normal text file, or CSV file, or an Excel file, like all these different files could be supported.

Interviewer: Yeah, that's good. Yeah, let's just move to the next question because we have, I think, 20 minutes. So you have seen this filtering form? Where you can filter models based on the partial name, extension and elements with some predicates and a number. So, yeah. Do you think this approach is useful for model filtering? And, why/why not? What would you change?

Participant A: I think the UI could be improved a bit. So here, I see. Three main parts, I think, general model elements and repository filtering.

Interviewer: That's right.

Participant A: But yeah, how these are connected. You can't find out from here, maybe? Yeah, maybe you see general and model elements as one section that are related. But maybe you don't have much idea about repository filters, how this is. What is the role of this year? I mean, I think this could be designed in a better way. And it is a bit crowded. I think. The UI for the model elements, I mean, maybe you could make it simpler. Like here, it seems that you're repeating a lot of these fields.

Interviewer: Yeah, another idea is to have one text box. And then you can do your query, you can type your query in there, but that will be. Yeah, you have to check a lot of things.

Participant A: Yeah.

Interviewer: So yeah, this was I think, the easiest approach from the UI point of view.

Participant A: Yeah, I can see that. But yeah, with that, you can filter models based on its content structure. This is really a nice thing. I personally, I think this is useful. For my research, if it works the way I expect.

Interviewer: Yeah.

Participant A: Yeah, the idea is very nice.

Interviewer: So that's it about the usability. Let's go to the other questions. So, you have used this system, right? And how do you think the system is going to impact the model analytics workflow for researchers? If we are going to apply it in, in the real world, how is it going to impact model analytics workflow?

Participant A: So it's definitely about the speed? The process?

Interviewer: Yeah, about the process itself, the steps. So in model analytics, we usually do some data collection and then preparation before the analysis, right? So, if we introduce this tool, how is it going to impact the workflow of researchers?

Participant A: Yeah. So, normally you do this manually, right? You mean that right?

Interviewer: Yeah, there are some semi-automated steps, like, crawling. But there are some other steps that you need to do manually, like transformation or filtering, maybe also validation. And the system automates that for you.

Participant A: Yeah, these are all automated. All these steps. If there is not a system like this, you need to do it all by hand. Or maybe there are some tools, but still, you have to do this, like, separate the connections between them. Having them all in one place, it is makes the job much easier for the researcher. But, this is very customized for this model analytics system.

Interviewer: Yeah, but it's based on the SOA architecture of the Arrowhead Framework. So you can add other services to extend the system. So it doesn't have to be constrained on this. So you can add additional, maybe you can automate additional steps in the future. So yeah.

Participant A: Yeah. So I like that idea that the design is flexible and scalable. As you say, you can add other modules based on your need to the system. But the existing one that we have this is, I mean, customized for our own purpose. So from that point of view, it makes it easier for the researcher to do their job.

Interviewer: Okay. Yeah, but don't you think that the validation and repository mining are some common steps that every researcher takes?

Participant A: So I think validation needs to be there. Unless you really trust this data source that you that you know, it really has the right files and data for you. Yeah. In your case, maybe you don't need this validation. But in real life, you don't have such action to deal with lots of different models from different sources. Yeah, so you want to test since we're dealing with academic cases, we need diversity in our inputs. Diversity comes from open source, open access data on the internet out there. You really need this validation. So that is a must-have, I think. Otherwise, I mean, there are consequences for you later, maybe. If it is not a valid model, then maybe later, your algorithm will be killed at some point. Yeah. Also, the same as for crawling. You want to find models from different places, and projects online? So yeah, if you have no input problem, then maybe you don't need any crawling from online resources. But this is, again, I think it's something that most researchers need.

Interviewer: Yeah. Let's, let's go to the next question. So you have seen that the system consists of some subsystems running on the local cloud. So we have crawling, filtering, transformation, and repository systems. Do you think that these systems are sufficient? in covering the steps of model analytics? Or what is still missing? Or what additional steps would like to be automated by the system?

Participant A: Yeah, I think these covered the basic, basic steps, like, you want to get the right input, right? I mean, that's the goal here. This is all about preparing your data at the right data for your input for your tool. So if you can provide this data, you know, in the desired format for the tool, then I think you're doing your job. But you can always have all these different functionalities features. On top of that, yeah, maybe you can extend to each of these features. For example, for crawling queue, you want to specify not only for GitHub but also other types of repositories. And the same for filtering, I want to customize my filtering differently.

Or transformation, also. I mean, these can be changed. For our research now, this is enough. I think. This basic functionality works fine. I don't know what else can be there for now.

Interviewer: So it's fine at the moment.

Participant A: Yeah.

Interviewer: Okay, the next questions are about performance. So, how much time do you think you have saved by using the system instead of using a manual approach?

Participant A: Yeah, the thing is that I have not actually used it to do a real job for me getting models and see the results. I mean, for this, you need to spend some time on it. But I see. I mean, by running the tool for a short while and see, okay. It is actually working. I mean, but the options that are listed on the UI, they seem to be useful. So if it works fine, then I think it is going to save a lot of time. Yeah, I don't know, like, how we can give numbers here. I think maybe.

Interviewer: Yeah, just some, do you think weeks days or maybe months?

Participant A: Yeah, I don't know. Maybe we should maybe decompose it into parts. Like for each part, for example for crawling, that is one I think, difficult step, for a researcher. If it is done manually, then you need to do this. I mean, just you go to the UI, GitHub UI, for example, you type in these keywords, go over these files in the repository. So it takes hours to get a few collections of a number of models. You easily spend hours on it. And the same for filtering I mean, if this is done manually, this is a huge process actually, you do this for every single model, you have to open it up, you need to check it for and for some models, for some quantities, even It is impossible to do this manually. Yeah, transformation. That is also, it is not possible to do it manually. I mean, for every model. Also the validation Yeah. These parts, I mean, I think, in small quantities, you can do this manually. And for part of it, crawling and filtering, but validation transformation. I don't think this is the can be done manually. A tool is needed here. But yeah, even for these little, mini small number of models, you have to spend a lot of time on it. Simply days, I can say, okay, but thinking of models of big quantity, like let's say 1000s of models, under 1000s of models. You cannot really compare to manual. It's not possible like that.

Interviewer: Yeah. Yeah. So, we need to be a bit fast because of the time. Do you think that the system? Yeah, maybe this same question, but yeah, do you think that the system will reduce the effort it takes for the researchers to prepare models for analytics?

Participant A: Yes. I think so. Yeah, based on what I saw from the system, I think it's, it is able to do the job. Yeah, I can't really say it is going to be useful, for example. Not useful, I mean, it is exactly what I want in my example research. But I know that it is. It is very helpful. Yeah, in general, I can see that there's a good use for it. It can be useful. But yeah, I was thinking of my own case, and because I have not tested the system, I can't give like very concrete answers. But definitely, it will reduce the effort. That's for sure.

Interviewer: That's good. So, yeah. Do you have any other feedback for the system?

Participant A: No, I don't think so. I mean, I think we discussed a lot on the system and different parts, how it can be improved. I find it a good system overall. I can say. Yeah. As I mean, as a basic model repository system. Not very advanced. I think it is a very good start. Yeah, you can build on it. I mean, it is. Yeah, one thing is I find it that they are I mean, people develop systems, but it is difficult to continue on it. Yeah, I don't know your I know a bit about your design, but I'm not sure about the implementation of how this is done. But writing good, clear code. I think this is very important. So that others can get continue on this. Otherwise, yeah, it is just some nice tool that is nice to have. Hopefully, we can continue on this one.

Interviewer: Yeah, there are some improvements there. I need to. I need to clean up a lot. But yeah, so that was the interview. I will stop recording.

B.2.2 Interview Results: Participant B

Interviewer: Okay, so the first question is about your background. So, can you tell me a bit about yourself and your experience with the Arrowhead Framework?

Participant B: Okay, so generally, my background is in mechatronics system design, so mostly about mechanical engineering. And I had this one year project regarding the Arrowhead Framework with Philips. So what I'm doing is to implement the arrowhead framework to have the interaction between multiple local clouds. So I am quite familiar with the structure and also how to implement the arrowhead framework.

Interviewer: Okay. So you know, of the concepts of local clouds

Participant B: Yeah, the local clouds and all the service registry orchestrator and authorization, these core systems and how they interact with the application system. So that for that part, I'm quite familiar.

Interviewer: Okay, that's nice. Yeah. So that's it about the background. The other questions are regarding the usability of the system.

Participant B: Yeah.

Interviewer: So how easy or hard was it using the system?

Participant B: So for the system? Yeah, I think you provided it. I think it's quite straightforward. The user interface is quite clear for me. And so overall, I would say it's easy and clear to use, of course, there are some I would say, like, I think there are some steps that will lead to some point that there's no turning back. Well, you probably want to, let's say to stop the crawler. It's like that. It's a bit. Maybe some drawback there. But overall, I will say it's quite easy to use.

Interviewer: So there needs to be the ability to stop the crawler, then initiate another crawling request?

Participant B: Yeah, I think that's for now. Right? I had to stop the instance of the crawler. And besides, I, to be honest, I don't really understand how it works. The crawler is going through the whole GitHub repository?

Interviewer: Yes, it, it uses the GitHub search method to search for files ending with a specific extension, like BPMN, and then it just goes over all these files and downloads, scrapes these models.

Participant B: So in the scope of the whole GitHub?

Interviewer: Yeah.

Participant B: So and, I realized that every time I do this search, the result will be the same. So it's going through the same path, is that correct?

Interviewer: No, that depends on the search. The search doesn't always return the same order of the results, it's always different. But you will get the same amount of models, but not in the same order.

Participant B: Okay.

Interviewer: And it also takes the modifications of models into consideration. So maybe models are deleted or added. So yeah.

Participant B: Oh, yeah that part? Yes, I just want to clarify.

Interviewer: Yeah. Let's go to the next question. So yeah, how easy or hard was it learning to use the system? So not using but learning?

Participant B: Yeah. To learn it was, I would say, also quite straightforward, especially that you arrange this demonstration session. So it's quite clear for me, as long as I have the system launched in my environment, and then everything's fine.

Interviewer: Were there any problems you encounterd?

Participant B: During the launch of the system?

Interviewer: Yeah, yeah.

Participant B: I think that the manual you provided is already clear enough. And, and you also did make that adjustment regarding that typo you made. And, so besides that, everything's fine. It's a, the only, how to say it, ambiguous part from my side is what these terms really are. I don't really know what a crawler is, but after I did some research I got to understand them and that's fine.

Interviewer: Yeah. Okay. So maybe the terminology part could be better? And how useful Do you think the system will be useful for preparing models for analytics?

Participant B: You may like to get as many models as possible. Is that correct?

Interviewer: Yeah. And then preparing them for analytics?

Participant B: specific format in BPMN, or EPML?

Interviewer: Yeah. And transformation, validation.

Participant B: In that case, I would say, it's really useful. To be honest, without this kind of system, I don't know how people can get, like, huge amount of models out of GitHub, and from the perspective of the transformation from BPMN to EPML. And I saw that it's done. And I have to say, I don't know how to verify if it's correct, but I believe so. It is. Overall, will say it is useful.

Interviewer: Okay. Yeah. So usually, people go manually over the steps. So there are crawling tools for crawling repositories. So that can happen automatically. But the other steps need to be done manually, like the transformation and validation of models. So there is no other tool to compare it to.

Participant B: And, and if you want to, if you're looking for a specific type of model, you just use you call it the tag thing. Yeah, the extension. Yeah. And so yeah, I think then it will be quite nice to use this system.

Interviewer: Yeah. So it will be useful in your opinion?

Participant B: In my opinion, it will be.

Interviewer: Okay. Another question about usability is. So, you have seen these UI elements. For crawling or creating new models? Do you think they are the right choice of UI elements? What would you add or change?

Participant B: In terms of preparing models for analytics?

Interviewer: Yeah, but yeah, overall, the user interface of the web application, you already have seen some text fields for creating new models. And you saw UI controls for crawling, doing some filtering.

Participant B: Yeah.

Interviewer: Yeah, what should be changed about this?

Participant B: Maybe I can think I sent you a screenshot before. I just want to review it again. How can I check this out? Assuming all these types of models, domains, crawler filter upload, and the corresponding user interface so for the models part, I think you had this option to add models?

Interviewer: Yeah.

Participant B: And did you have the option to delete them?

Interviewer: Yes. If you go to the Edit page, the update pages of a model, then you can delete them.

Participant B: Okay. Yeah, I'm just, yeah, I only have a snapshot from the application. I am not running the system in the background. And, for the domain part. The domain part is the tag thing I mentioned.

Interviewer: Yeah. We had this big text box, and then you could separate the string with the format.

Participant B: And, yeah, and the crawler part. Yeah, the only, let's say the problem with the crawler, I was gonna say, just how can I stop it? Without shutting the complete crawler system down?

Interviewer: Yeah, okay. And information-wise, do you? Do you want more information from the crawler?

Participant B: Do you mean? At least for now I can see from the crawler that you can see how many links are collected, how many models are collected, and how many models are skipped. I will say, I think it's satisfactory for me. I, I'm not sure in terms of this model, model preparing, like what else, what the other information is needed that to me, for now, information is sufficient. And also, the user interface is quite clear, you got different blocks with different colors in a really clean, clear form style, so it's fine.

Interviewer: Okay. Yeah, let's go to the next one. So you have seen this filtering form?

Participant B: Yeah.

Interviewer: Do you think this is a useful approach of filtering? So, you have the option to choose a predicate, like, larger than, then choose an integer to filter on some elements of these models. Do you think that's a useful approach for filtering?

Participant B: Yeah, I think so. As long as you can, we can have a certain I mean, you can assign a specific criteria to retrieve the needed information, and then it's a perfect function for this kind of system. Okay, and I think you also have the option to, to say that you want to download it in the BPMN or EPML format. And the transformation is done automatically. So this is really nice.

Interviewer: Okay, that's fine. Let's go to the next one. How do you think the system is going to impact the model analytics workflow of researchers?

Participant B: Well, yeah, to my knowledge, as you mentioned, there is no such such kind of system that even there's no something similar in the market or in the anywhere. And I would say, without this kind of system, if you have to do like, a huge amount of model analytics on your own, or let's say manually, then I think this kind of automatic system will definitely have a huge impact. Especially if I, I mean, if I if I am the researcher who wants to, let's say collect a huge amount of models and to check if these different models have some similarity or compare them and to have a system to automatically optimize a process of collecting data, it will be perfect.

Interviewer: Okay. And yeah, you have seen that we have these subsystems running in the local cloud of Arrowhead Framework. So we have this crawling filtering, transformation and repository systems. Do you think these are sufficient in covering the steps in the model analytics? Or, yeah, what do you think is missing? Or what additional steps would you like to be covered? What additional steps would you like to be automated by the system?

Participant B: Yeah, before that, what is repository exactly?

Interviewer: A repository is simply a place to store models. Furthermore, it allows you to retrieve models based on a database query. Just means of storing, storing retrieving data.

Participant B: Yeah, sufficient? Yeah, I, yeah, I cannot come up with another system that might be needed. Maybe you can give me something. But yeah.

Interviewer: Yeah, it's usually dependent. Some researchers think this is sufficient. And, yeah, others might want even more systems in the local cloud to automate more steps.

Participant B: I mean, like, I would say, if we can think about this in, in a real scenario, if a researcher, he wants to, say gather a bunch of models in a certain view, and then do some, analyzation. And then what he can do is, yeah, he has this crawler to gather the data. And then he even got the domains to see which type of data he's getting. And then, after retrieving a bunch of data, he can have this filtering system to filter out something not needed. And then he might need a certain format, and then that can be done with the transformation system. And then I will say it's sufficient. I'm like I say, I'm not experienced with this so I'm not sure. There is any other system that can add other value into this.

Interviewer: Yeah. Yeah, let's go to the last few questions. So yeah, these questions are about the performance, the speed of the system. I don't know if if I can ask you this question. Because you don't have that much knowledge about model analytics.

Participant B: I really hope I'm not wasting your time.

Interviewer: But I'm still going to ask. Yeah, so how much time do you think you saved by using the system instead of using a manual approach? Like, how much time do you think you have saved?

Participant B: So, I just want to clarify, so if, like without this system, and people really just manually search for the models? They are?

Interviewer: Yeah, you just go over GitHub, you download a model. Yes, let's just take one model, for example, you go to GitHub, you download this model from a certain repository. And then you want to prepare this model, so you validate it against a metamodel. And then, you want to maybe you want to extract some metadata, and then transform it to another language and other formats, like EPML, then store it in a database. So that kind of scenario you can think of.

Participant B: Yeah, I think that leads to another question I have from my side is with this crawler function, and how do you know that the crawling process is enough? Like you get a lot of models, then how do the researchers know, It is at a point that the amount of models is enough? I'm not sure about that. And also, like you mentioned, how to validate or verify these models are ”good” models.

Interviewer: Yeah. I understand what you mean. But let’s say hypothetically. That, like 1000 models are enough. Do you think you think, you will save more time with the system? Compared to the manual approach?

Participant B: Yeah, if I have a target amount of models, and with this, yeah, with this system, they will save a lot of time, and in terms of how much time will be saved? I don’t know. Just so yeah, the difference between a search for models for 1000 times manually and the difference? Yeah. And then to compare it with gather all the 1000 models with a crawler? The time difference will be like that. I don’t know. I don’t know how people do this manually for 10000s of models, then you use your own crawler?

Interviewer: There are already some repositories online, that already crawled GitHub to get some models, and then you can use this repository to query these models and then download them immediately. But then you have to do the additional steps manually, like validation against a metamodel and model transformation, and filtering.

Participant B: Okay. So, it sounds like the difference between your system MAAS, we can actually just go through the whole scope of GitHub, and then you get what you want. Yeah. And after that, you do these filtering, or transformations in your local site. On the other hand, without this, you just, you have to look for a specific repository on GitHub. And query, and make queries in that repository. And that difference in terms of time, say? Is it time-consuming to look for that exact repository? In a GitHub?

Interviewer: No, you don’t have to look for specific repositories on GitHub. In some cases, we have these databases you can find through a Google search that crawled repositories on GitHub. So they have already done some work for you.

Participant B: Okay.

Interviewer: But only for some types of models. And if yeah, for the other types, you might have to use your own crawler.

Participant B: Okay.

Interviewer: Your own model crawler. So, yeah, it depends on whether it exists or not. For UML and BPMN models, there are some databases you can use.

Participant B: Yeah?

Interviewer: But for the other types, like EPML, you really need to use a crawler to crawl these models.

Participant B: Okay, so, so there are some other options to automatically retrieve it, you have to I mean, you are kind of confined with something provided by the others.

Interviewer: Yeah, yeah.

Participant B: So how much time to think. So, in terms of how much time will be saved, I think it depends on what you are looking for. Is that correct?

Interviewer: Yes.

Participant B: And so if I, I put it into like two aspects. So, if you are looking for something, it's already available.

Interviewer: Yeah, but they are available, but they are in a raw format. So, they are not validated.

Participant B: And with your system they are validated?

Interviewer: With my system, they are crawled and then validated against a metamodel, automatically during the crawling. So, it checks whether it conforms to its metamodel, whether the elements are correct.

Participant B: I see. So, yeah what are you looking for, like, an exact time to say, maybe like a week or hours,

Interviewer: That's just an estimate. Do you think, a couple of days or what?

Participant B: Yeah, I don't know, the timescale they spend, as in this kind of scenario. So usually, would it be like days or weeks?

Interviewer: Yeah. I read some papers online about crawling. And people usually spend, like, one month to crawl. Although, okay. That's only the crawling without validating and filtering, you know, without validating stuff. But it took 30 days to crawl.

Participant B: Okay, so if we are looking to get a set of data, which is validated and filtered. And in that case, in comparison with using the existing crawler, and then validate and filter, on your own, to compare it with that, I think this will save time. At the scale of weeks, I guess.

Interviewer: Okay.

Participant B: Yeah, it's just, I don't really have a clear idea about this. So yes, yeah.

Interviewer: Yeah, no problem. Then we have a similar question. Do you think that the system will reduce the effort it takes for researchers to prepare models for analytics? Do you think it will reduce the total effort of researchers?

Participant B: Yeah, I do think so.

Interviewer: And, and why? Or how, why do you think it will reduce the effort?

Participant B: Yeah, from what I learned, I think the validation part and the transformation part are the main time-saving functions. So if you, yeah, for sure, you get other types of crawler, or you can even implement your own crawler. But then the part of the validation if you have that, if you have to do it manually, then it would be way more time consuming than it's being automatically done with the system. And the validation part is done automatically. So it's not shown in the user interface. Is that correct?

Interviewer: Yeah, it's automatically.

Participant B: So, all the collected results are already validated?

Interviewer: Yeah, yeah. What you see is validated.

Participant B: So those are not collected? I mean, the model skip is not validated or?

Interviewer: Yeah, model skipped are the models that are not valid. So they are not stored in the database.

Participant B: So then yeah, also, I would say, that's a nice feature, you see that there are that amount of data models are now validated. So and, and also the transformation part. So, I will say, from my perspective, I see that this system is addressing the process of collecting models. So, if, for instance, if we can maybe draw a, like a flow chart or activity diagram of the process, collecting models, and we can, for sure identify which part is quite time consuming, maybe the validation part, maybe the transformation part. And I can see that the system is automating those procedures, which indeed, even with my limited knowledge, I will assume that can be really helpful for the researchers to save their time.

Interviewer: Okay. Okay, that's good.

Participant B: I'm not trying to say something nice. It's just based on my knowledge if I'm able to identify something rather nice. I will say that.

Interviewer: Any additional feedback that you might have for the system?

Participant B: Yeah. I'm not sure if it is feasible. But is it possible to have some kind of progress bar thing?

Interviewer: For the crawling?

Participant B: Yeah, maybe? Crawl? Yeah, crawling.

Interviewer: Yeah, but I guess, for now, the problem is, you need to have a start and you have to, you need to know the minimum amount. So that's zero, and the maximum amount of models, but this maximum amount can change over time. You know what I mean? You don't have a fixed amount in time.

Participant B: I mean, is it possible that? I mean, as a researcher, do they? Are they looking for a specific amount of models? Or are they just trying to get as many?

Interviewer: Yeah, as many models as possible. That is the purpose of the crawler to get as many models as possible.

Participant B: Okay, so progress bar might not be feasible in that case? Because you don't know how many models there are there?

Interviewer: Yeah, it can change over time. Especially on GitHub, because there are many changes committed to the repositories. Some models are added, and some more models are deleted.

Participant B: Or modified?

Interviewer: Yeah, modified as well.

Participant B: I see.

Interviewer: But, yeah, if it's modified still. It still stays there. So, that's not the problem. It just if models are deleted, or added during the crawling process.

Participant B: Okay, so. And yeah, after the crawling process models are stored in the repository, or?

Interviewer: Yes, they are. In the Elasticsearch instance. So, in the repository and the repository uses Elasticsearch as a database back end to store these models.

Participant B: So the users are actually. Sorry, I mean, are those models actually stored on the user's machine?

Interviewer: Yes. But yeah, because we are running it locally. But you can change it to store it on another server if you want. Yeah, but at the moment, they are just stored where they are hosted where the systems are running.

Participant B: And so you're getting the models for that instance. And if the model has been modified or deleted after your crawler passed by It doesn't matter?

Interviewer: Do you mean when it's modified on GitHub or locally?

Participant B: On GitHub.

Interviewer: Yeah. So if it already passed by the model, and the model was modified on GitHub, then the system does not take into account those changes. It just gets the most recent version of the model.

Participant B: The one you?

Interviewer: Yeah. The one that you receive, so it doesn't take into account modifications.

Participant B: Yeah. I'm not sure. Would that be a problem for them? The researchers?

Interviewer: I don't think there's a feasible way to. I think that will be very hard to implement. I don't think it feasible for this thesis. And I think you need to connect to another database system that gets all the events from GitHub, and then maybe you can filter on these events. So maybe on updates done on BPMN files, for instance.

Participant B: You need to get like log data?

Interviewer: Yeah.

Participant B: And, see all these different, maybe modifications or all kinds of events?

Interviewer: Yeah. And then, you can sync, synchronize your models with these types of events. But I think that's another master thesis by itself.

Participant B: I think that you can immediately come up with a solution. That's pretty nice.

Interviewer: So, that's an idea that, yeah, I don't know if it's actually feasible. So was there anything else? Any other feedback? Or questions?.

Participant B: At the moment, I think it's quite clear for me, yeah.

Interviewer: Okay. Then I think that's it for the interview.

B.2.3 Interview Results: Participant C

Interviewer: So the first question is about your background. So yeah. Can you tell me a bit about yourself and your experience with Arrowhead Framework?

Participant C: Yes. Well, okay, I am a PhD student. Well, I am about to graduate. Actually, yeah, I'm going to defend my thesis in a couple of months. I'm going to defend my thesis on September. Yeah, it is about Model Management. Basically, we were investigating ways of identifying relationships between models from different domains. And my experience with Arrowhead is not that high. I started with this project in a couple of months ago. I know how Arrowhead Framework works. And I know the process and the communication between the services. And yeah, I mean, yeah, I don't have much experience with Arrowhead. But I have some, I'm going to extend the tool that I created during my PhD to use Arrowhead.

Interviewer: Okay. Yeah. And you are familiar with the local cloud concept of Arrowhead?

Participant C: Yes.

Interviewer: Okay. Yeah, that's the main thing about Arrowhead Framework. So that's good.

Participant C: I mean, I am not able to give a lecture. I am not. Yeah, but I know I am familiar.

Interviewer: That they are composed of systems?

Participant C: Yes.

Interviewer: Collaborating with each other?

Participant C: Yes.

Interviewer: Okay. And your experience with models, model analytics? Your knowledge?

Participant C: Well, with model analytics. It is difficult to define because I would say that there are several, how can I say, let's say, model analytics has several phases? I think, maybe in my opinion, I mean, you can extract several kinds of information from the models, at least, that's how I see the model analytics. And then, you can investigate several themes from model analytics. And then, in my experience, I am more, let's say, interested in how information related to relationships and how those models are interconnected.

Interviewer: Yes, that is also model analytics.

Participant C: Of course, there are much more than these. I mean, you can extract out some kind of meta models, meta meta models, yeah, get like other kinds of information, but the only thing, how to put some codes, the information that I am more interested, is related to the relationships between them between the models, especially models from different domains.

Interviewer: Okay.

Participant C: Sorry, from different engineering domains, like electronics and software, and then like software engineering, and then mechatronics. And now, I want to identify the relationships between, let's say, a Simulink MATLAB model and UML model, then Simulink belongs to the mechatronic domain and then UML belongs to a software engineering domain.

Interviewer: Yeah, yeah.

Participant C: Okay?

Interviewer: Yeah. Clear. And also, MDE, I think? You have a Model-Driven Engineering?

Participant C: Yes. Yes.

Interviewer: And you also have the other one? Model-Based System Engineering?

Participant C: Well, yeah. Honestly, for me, they're almost the same.

Interviewer: Yes, yeah. So, I think that's it for the background. But let's go to the usability aspect of the system. So yeah, how easy or hard was it using the system?

Participant C: Maybe it might be relevant for their previous question is that I am working on models for, let's say, five years. Like in this topic, like Model-Driven Engineering, etc. But I have also working experience, let's say like two or three years, like in the industry working with, but in the software engineering, like developing software, but also creating UML models, and also writing, like, requirements, documents, like, you know like the document that you write the requirements?

Interviewer: I think it's just requirements, right?

Participant C: Yeah. Sorry, yeah.

Interviewer: Okay. So that's actually it for the background. So we are here about usability. So you used the tool?

Participant C: Yeah.

Interviewer: So how hard was it for you using the tool? I mean, how easy or hard was it?

Participant C: It was really, let's say, simple. Also, to install the tool. It was. I didn't have problems. I mean, yes, I had one problem, but it was actually related to my machine. For some reason, Apple. Yeah, for some reason. Apple automatically uses the eight zero port for bash. And then I had to turn it off manually. Because it automatically turns on. Yeah. And then yeah. And then I had problems to identify these internally. But, it was just like it was my machine not really related to your tool.

Interviewer: Yeah, it's. I think it's the Docker, right? Because it uses this specific port?

Participant C: No, it is. I mean, if you check, do you use? I think you're using MAC?

Interviewer: Yes. Yeah. I'm using Mac and Linux at the same time. Yeah. They are almost the same.

Participant C: Yeah. Almost. If you go to the properties, you're going to find that the port eight zero has been used by something from Apple.

Interviewer: Okay.

Participant C: Yes. Yes, you can do it later. But you can see that.

Interviewer: Yeah, I can do it later.

Participant C: But in general, I think it was simple to install and also easy to use.

Interviewer: Okay, that's good.

Participant C: Yeah. I mean, of course, the user interface could be improved. That's true. Of course. Yeah. But as a proof of concept, I think it is good enough. And could easily see the features, the functionalities that your tool provides. And I think it's good.

Interviewer: Okay. Yeah, the user interface was not the main focus of the thesis. It was just to demonstrate these functionalities.

Participant C: I know that, don't worry.

Interviewer: Yeah, but yeah, it needs some improvement. Some work.

Participant C: Yeah.

Interviewer: I got that from the other interviews as well. So, yeah, I mean, yeah.

Participant C: I am also a software developer, and I have a lot of problems with the front end, like yeah, I am good with the back end, and with the front end, I am really bad.

Interviewer: Yeah, especially. Yeah, I think it's easier with HTML, but yeah, I don't know some of the CSS stuff.

Participant C: Exactly. I know when something is beautiful or ugly, but I don't know how to make it beautiful. Yeah, that's true I don't know if you got my point, but yeah.

Interviewer: Yeah. So let's continue. So how easy or hard was it learning to use this system?

Participant C: Yeah, I think it was easy. Let's say, I think it took maybe half an hour from your tutorial. I think it was easy.

Interviewer: Okay. And you didn't think that you needed more explanation of some of the parts?

Participant C: Yeah, actually, the definition of domain that your use is different than mine. And then I had, let's say, it was a bit confusing for me. But it was because I was, let's say biased on the definition of domains that I use, because the definition of domain that I use is the engineering domains. But the definition of domain that you use, as far as I understood, is more like health, like health care, or maybe, let's say, sports, or maybe like science, these kind of domains, not really engineering domains. It can be everything. Yeah, but despite that, I think it was. Yeah, it was easy to understand.

Interviewer: Okay. Okay. That's good. Yeah, I think this is a tough question. So, how useful Do you think the system might be for preparing models for analytics?

Participant C: Yeah, I mean, we can focus on those specific tasks. That could be: search for models, download the models and make that analysis. And I believe that doing that, doing those tasks, using your tool was really easy. And was also really like, it could be useful compared to doing that manually. So that's why I believe it can be useful.

Interviewer: Okay.

Participant C: Yeah.

Interviewer: Okay. Clear. Let's move on. So, you have seen some of the pages of the web application, you have seen how to create models, upload models, create new domains and starting the crawler. And you also have seen the interface elements that I used. Do you think these are the right choice of interface elements?

Participant C: Do you mind opening the system? Do you have your system open? No, you don't. Because, I don't remember much but.

Interviewer: Wait, let me open the system.

Participant C: You need to open everything right, or no?

Interviewer: Oh no, I think only the user interface.

Participant C: Okay.

Interviewer: Wait I think it is this folder. Yeah. It will take some time to start.

Participant C: It is okay.

Interviewer: Good, I think it is starting. localhost I think port 80. And it's working? Yeah. Okay. What did you want to see? Which one? So you can add the model here.

Participant C: No, sorry. Can you please return to your question? Ask the question again. So I can.

Interviewer: Yeah, so here's the question. So you see these UI elements on the pages? Do you think these are a good choice? Or do you think there are some improvements there?

Participant C: Ah, okay. So again, please go to the, yeah.

Interviewer: So, for instance, I want to add the model. Then you see these UI elements. Do you think these are a good choice for you?

Participant C: I think it's a good choice. And then maybe you can give some description of this path, because path of the model, what do you mean with this? Is it like a local path? Or is it like URL?

Interviewer: Yeah, it is actually the URL to the repository.

Participant C: Yeah. Maybe it could be like URL.

Interviewer: Yeah, we need to add some more hints.

Participant C: So back to the first page. The modeling language, the creation date, actions. Okay. Yeah, I think it's okay. Can you please go to domains?

Interviewer: Yeah. Domains is an interesting one. It's just the same as models list. But if you add a new domain, you need to give it a name, and then the tags are separated by a comma, you know? And that was the simplest design.

Participant C: Yeah, I think it's okay. Can you please go to crawler?

Interviewer: Uhm, yes.

Participant C: Can you please explain to me again the difference between links collected and models collected?

Interviewer: So, links are just the URLs collected by the crawler and models collected? Are the models that are validated and transformed, and stored in the repository.

Participant C: I see. Yeah, but it's it again, this is just a minor thing. It could be like to explain what these links mean. Yeah, yeah. But again, just really like, minor thing.

Interviewer: And uhm, where is the best position or section I can put this information?

Participant C: You can put like, I forgot the name. But you can. Like when you put your mouse on?

Interviewer: Oh, on hover?

Participant C: Yes. Yes, exactly.

Interviewer: Okay. Yeah, that's a good one.

Participant C: You can give like a really short explanation about this. Yeah, but again this is really, like, really minor thing. And can you please go to filter?

Interviewer: It doesn't fit the page on my laptop.

Participant C: That's okay.

Interviewer: I have a 13 inch MacBook so.

Participant C: Model extension? Well, again, it is just a minor thing. What you could do is just like, hide these model elements, and the repository, features, kind of hide. And then if the user wants to have, let's say, advanced search, and then they would click that plus button, and then this, what was I gonna say, these, not components but these, these fields would appear.

Interviewer: Yeah.

Participant C: But again, this is really like a minor thing.

Interviewer: Because it's a bit crowded, right?

Participant C: Yes. Yeah. When you see like uh, as the first thing, it, it might scare the user a little bit. Because you don't need to fill in all those things, right? You just need to fill in their name?

Interviewer: Yeah they are all optional. I got that as well from my other interviews that it is a bit too crowded, populated.

Participant C: Yeah.

Interviewer: Yeah.

Participant C: But again, I totally understand that it is just like a proof of concept. So you need to really take care of the small things. Of course, there is room for improvements everywhere, but yeah.

Interviewer: Yeah always.

Participant C: Always. Exactly.

Interviewer: Yeah, shall we move on?

Participant C: Yes.

Interviewer: Yeah. So here, we can upload multiple models instead of one as a zip file.

Participant C: Yeah, yeah. I think that's it.

Interviewer: Okay.

Participant C: Okay?

Interviewer: Let's move on to the next question. So you have seen this filtering form, do you think this is a useful approach for model filtering? So, you can choose some predicates here like equals, not equals, and then you can specify an amount, you know, do you think this is a useful approach for filtering?

Participant C: Uh, let me think. Okay, I will turn this question around? Do you think it is possible to miss any kind of specific kind of model, if I use these filters? Like do you think that using these filters I can find the model that I want?

Interviewer: I think so.

Participant C: Okay.

Interviewer: Yeah. Yeah.

Participant C: Like, uhm.

Interviewer: Let's say you want to filter on specific elements within models.

Participant C: Yeah, but let's say I just want. I mean, just like, I am guessing, let's say that I want to find those models created before 2020. Can I do this?

Interviewer: Before 2020? Yeah, that's, that can be another text box here. But, you can't do that at the moment.

Participant C: Yeah, then maybe like, the user not really the user, but the person who created, yeah the GitHub user. Can I also filter by them? You have like a repository name? Right?

Interviewer: Yeah, you can filter by the repository name, the number of starsm, and the number of forks.

Participant C: Yeah.

Interviewer: Yeah. But you also want to filter by the user?

Participant C: Yeah.

Interviewer: Or the owner of the repository? What do you mean?

Participant C: Yeah, maybe both? I mean.

Interviewer: Okay, okay.

Participant C: Yeah, I am just, let's say, brainstorming.

Interviewer: Yeah. There are always some more filtering options you can add.

Participant C: Yeah, I mean, I am just saying these things because you ask it right. I mean, you will wonder what is missing there. Then maybe, like the creation date, the name of the owner, or maybe the name of the person who created?

Interviewer: Yeah, these are some additional fields, filters. But what I'm mainly asking is the way of filtering. So I am choosing predicates here, and then I type a number to filter on these models. Maybe there are also other options, like a query language, like OCL or BPMN-Q.

Participant C: Ah I see. No, I honestly, I don't think the user should. I mean, of course, you could create, like, let's say, an option where the user would add, like, let's say, let's call it like a professional query, select the, you know, use the SQL, whatever. But, honestly, I don't think it is a good idea. Maybe the way that you are doing is fine.

Interviewer: Yeah, yeah.

Participant C: Yeah. Oh, sorry. I misunderstood your question.

Interviewer: Yeah. No, both answers are fine for me. Yeah. So let's move on. That's it about usability. Let's go to workflow automation. So, yeah, this is also a tricky question. So how do you think the system is going to impact the model analytics workflow of researchers?

Participant C: Yeah, I mean, yeah, as I mentioned before, it's tricky, because we would need to compare the research workflows, but considering only those only the steps that you're tool does, uh I think, your tool has potential to improve their workflow. Yeah, I think it will impact in a positive way, the research workflow. Like, not only like I would say, mainly, the speed, I believe the researchers can perform their work faster using your tool.

Interviewer: Okay. Okay.

Participant C: I mean, It's just that's what I believe, but I don't have like, concrete data to prove it.

Interviewer: To back that up?

Participant C: Yeah.

Interviewer: Okay. And you have seen that the system consists of some subsystems like the crawling, filtering, the repository manager and transformation. Do you think that these systems are sufficient in covering the steps in model analytics? So, what additional steps would you like to be automated? Or what steps are missing?

Participant C: Yeah, well, this question is really difficult for me. Can you specify the kind of analysis that you want to perform?

Interviewer: Okay, I will give an example.

Participant C: No, no. Using your tool. Can you specify the kind of analysis that you want to perform? Let's say, can you detect clones? Can you measure the clone detection?

Interviewer: This tool is only for input. You only get input for your analysis, for your, so let's say I want to do clone detection. I have this tool separately, somewhere outside of the system. And this system's responsibilities are to only collect these models and prepare them for your tool to conduct analysis.

Participant C: Okay, yeah. So, well, in that case, yeah, maybe the question should be rephrased.

Interviewer: Okay. Yeah.

Participant C: Because, you ask if those steps are sufficient in covering the steps in the workflow?

Interviewer: Yeah.

Participant C: I believed it goes until the analysis, right?

Interviewer: Yeah.

Participant C: Okay. So, excluding the analysis, let's consider only the input. In that case, I don't think anything that can be included is missing. At least not from the top of my head, but yeah.

Interviewer: So yeah, so it has the basic things included, you think?

Participant C: Yeah, I believe so. At least regarding the input, I think it's fine. Okay, I will mention this problem. Just because. But like your tool has, like a problem that it doesn't. Let's say you stop the crawling.

Interviewer: The functionality is there. But not yet in the interface. That's going to be added later. Yes, this question also came up in the other interviews. You have to stop the tool from the command line. But yeah, this function will be in there in the final version.

Participant C: So maybe, okay, what steps are still missing? This one?

Interviewer: Yeah, like stopping the crawler.

Participant C: Yes, okay. What additional steps would you like to be automated by the system? Do you think that the analysis of the, you know, the final output of the analysis should be included in the system as well? Let's say I used your system to collect all the data, etc. And then, I performed my analysis. And do you think that the results of my analysis should also be included? Or no?

Interviewer: No, this is outside of the system.

Participant C: It's outside? So, yeah.

Interviewer: Yeah, we are only concerned with collection and preparation, that's it.

Participant C: Yeah. Well, in that case, I think your tool is fine. I mean, at least I don't see any additional steps. That could be automated.

Interviewer: Yeah, because this tool is built on the Arrowhead Framework, composed of several systems. But maybe in the near future, you want to automate another step, then you can just write another system and use it in the Arrowhead Framework.

Participant C: Yes.

Interviewer: So then, you can use the systems to automate more steps for you.

Participant C: Okay, maybe one additional step that could be automated, I mean, it's not really an additional step, but maybe a change. Because in the crawler, we have to use our own credentials, right? We have to put our GitHub user, and then maybe your tool could provide, let's say, a default user.

Interviewer: Yeah, we have a default user. But if you use the tool on an unknown device, GitHub sends a verification code to the email. And my crawler cannot handle that. So yeah, I don't know. So that's why you need to give your own account credentials. But, what I'm mainly asking is there are other steps in model analytics, like feature engineering, feature selection. There are many different steps in data analytics, model analytics that's what I meant actually with, with the steps.

Participant C: Not really, I mean, the thing that you can do on these maybe. Yeah, but it can be transferable to what you're asking. But then, in any case, it can be. Because let's say the definition of a domain that you have is really generic, then maybe you could use like a specific engineering domain. And how you are going to implement this, I have no idea. But, like, only download models from software engineering. So to say, let's say only UML, or whatever.

Interviewer: Okay.

Participant C: Or maybe, let's say, class diagrams. And because there are already tools that can identify when a model is a class diagram or not. But then, as far as I know, there is no tool that can download them. Like, let's say like, I mean, in your tool, maybe you could, I'm just thinking, you could like, combine your tool with this other tool that can detect if the model is a class diagram or not. And then download them.

Interviewer: Yeah. That's what I meant with the other feature, you can build a system that is responsible for what you just said, for the UML check for a class diagram. And then you can just use it as another step for your system.

Participant C: Yeah, but yeah, but I mean, but then you ask me what additional steps you would like to automate. I really I mean, yeah, I don't know, we can spend like, one hour.

Interviewer: Yeah, we can go on.

Participant C: Yeah.

Interviewer: But, these steps that are already in there are sufficient at the moment?

Participant C: Yeah, yeah. I think I they're fine.

Interviewer: Okay, okay.

Participant C: Yeah. I mean, you can improve the kind of models that you want to download, but by additionally, another system that can improve the filtering. But again, this is just extracting that. It's not really necessary.

Interviewer: Okay. Just some minor things. Okay. Yeah, good. Let's move on. This is another tough question. So how much time do you think you saved using the system instead of having to use a manual approach?

Participant C: Oh, it's. Yeah. I mean, it's really difficult to estimate. But I believe this is just what I think that I saved a lot of time using your tool instead of doing all the processes manually. But I cannot estimate how much.

Interviewer: Okay, just a lot of time?

Participant C: Yeah, I believe it was. At least 50% faster.

Interviewer: Okay. Okay. That's a good indication.

Participant C: But again, this is just what I believe I don't have the complete data to.

Interviewer: Yeah. Because you can think about it, like maybe going to a GitHub repository, download the model, and then do the steps manually compared to using the tool. Yeah. And if you do it for 1000s of models manually, and it will. Yeah, it will take some time.

Participant C: Yes.

Interviewer: Yeah.

Participant C: Yeah. That's why I said at least 50%.

Interviewer: Okay. I think the last main question is, yeah. Do you think that the system will reduce the effort for researchers to prepare models?

Participant C: Yes, yes. Just like I mentioned in the beginning that, just because I think it is faster, it will already reduce the effort needed to, you know, to prepare the documents and everything because having like one tool that can already provide you all the data that you need. Without this tool, you would have to collect all the data by yourself. And then it would take a lot of time. Yeah, that's what I believe.

Interviewer: Yeah. Okay. Good answer. Is there any additional feedback you might have for the system?

Participant C: Not really, not with me, of course, there are things here and there that could be improved, but in general, you know, I mean, just to summarize the things that could be improved. The front end, of course, I mean, the user interface, but again.

Interviewer: Yeah, the usability can be better.

Participant C: I mean, it is, again, really a minor issue. Also, you could add some extra filtering options, let's say a date when the model was created, and by whom, these kind of things. Yeah, but I would say that's all.

Interviewer: Okay. Yeah, I think that's it for the interview. Let me stop recording.

Bibliography

[1] *About the Business Process Model And Notation Specification Version 2.0.* URL: https://www.omg.org/spec/BPMN/2.0/ (visited on 06/21/2021).

[2] Kerstin Altmanninger et al. "AMOR–towards adaptable model versioning". In: *1st International Workshop on Model Co-Evolution and Consistency Management, in conjunction with MODELS.* Vol. 8. 2008, pp. 4–50.

[3] *Arrowhead Framework.* https://github.com/arrowhead-f/.

[4] *Axellience: GenMyModel.* https://www.genmymodel.com/.

[5] Aaron Bangor et al. "Determining what individual SUS scores mean: adding an adjective rating scale". In: *Journal of usability studies* 4.3 (2009), pp. 114–123. ISSN: 1931-3357.

[6] Francesco Basciani et al. "MDEForge: An extensible Web-based modeling platform". In: *CEUR Workshop Proceedings* 1242.619583 (2014), pp. 66–75. ISSN: 16130073.

[7] Fredrik Blomstedt et al. "The arrowhead approach for SOA application development and documentation". In: *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society.* 2014, pp. 2631–2637. DOI: 10.1109/IECON.2014.7048877.

[8] *Card sorting.* URL: https://github.com/ds4se/chapters/blob/master/zimmermann/card-sorting.md (visited on 07/30/2021).

[9] Ozren Dabic, Emad Aghajani, and Gabriele Bavota. "Sampling Projects in GitHub for MSR Studies". In: (2021). arXiv: 2103.04682. URL: http://arxiv.org/abs/2103.04682.

[10] Fred D. Davis. "Perceived usefulness, perceived ease of use, and user acceptance of information technology". In: *MIS Quarterly: Management Information Systems* 13.3 (1989), pp. 319–339. ISSN: 02767783. DOI: 10.2307/249008.

[11] Jerker T A T T Delsing. *IoT automation : arrowhead framework LK.* Boca Raton, 2017.

[12] Juri Di Rocco et al. "Collaborative repositories in model-driven engineering". In: *IEEE Software* 32.3 (2015), pp. 28–34.

[13] Steve Easterbrook et al. “Selecting Empirical Methods for Software Engineering Research”. In: *Guide to Advanced Empirical Software Engineering*. Ed. by Forrest Shull, Janice Singer, and Dag I K Sjøberg. London: Springer London, 2008, pp. 285–311. ISBN: 978-1-84800-044-5. DOI: 10.1007/978-1-84800-044-5_11. URL: https://doi.org/10.1007/978-1-84800-044-5_11.

[14] *Elasticsearch*. URL: https://github.com/elastic/elasticsearch (visited on 06/21/2021).

[15] Mturi Elias and Paul Johannesson. “A survey of process model reuse repositories”. In: *Communications in Computer and Information Science*. Vol. 285 CCIS. 2012, pp. 64–76. ISBN: 9783642291654.

[16] Erich Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1st ed. Addison-Wesley Professional, 1994. ISBN: 0201633612. URL: http://www.amazon.com/Design-Patterns-Elements-Reusable-Object-Oriented/dp/0201633612/ref=ntt_at_ep_dpi_1.

[17] Georgios Gousios. “The GHTorrent dataset and tool suite”. In: *Proceedings of the 10th Working Conference on Mining Software Repositories*. MSR ’13. San Francisco, CA, USA: IEEE Press, 2013, pp. 233–236. ISBN: 978-1-4673-2936-1. URL: http://dl.acm.org/citation.cfm?id=2487085.2487132.

[18] Martin Haeusler et al. “ChronoSphere: a graph-based EMF model repository for IT landscape models”. In: *Software and Systems Modeling* 18.6 (2019), pp. 3487–3526.

[19] Regina Hebig et al. “The quest for open source projects that use UML: Mining GitHub”. In: *Proceedings - 19th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2016*. 2016, pp. 173–183. ISBN: 9781450343213. DOI: 10.1145/2976767.2976778.

[20] Christian Hein, Tom Ritter, and Michael Wagner. “Model-driven tool integration with ModelBus”. In: *Future Trends of Model-Driven Development - Proceedings of the 1st International Workshop on Future Trends of Model-Driven Development - FTMDD 2009 In Conjunction with ICEIS 2009*. 2009, pp. 35–39.

[21] Thomas S. Heinze, Viktor Stefanko, and Wolfram Amme. “Mining BPMN processes on GitHub for tool validation and development”. In: *Lecture Notes in Business Information Processing*. Vol. 387 LNBIP. Springer, 2020, pp. 193–208. ISBN: 9783030494179. DOI: 10.1007/978-3-030-49418-6_13.

[22] *HtmlUnit*. https://htmlunit.sourceforge.io/.

[23] Mo Jamshidi. *Systems of systems engineering: principles and applications*. CRC press, 2017, pp. 2–3.

[24] *jsoup: Java HTML Parser*. https://jsoup.org/.

[25] Zador Daniel Kelemen et al. “Selecting a Process Modeling Language for Process Based Unification of Multiple Standards and Models”. In: June 2014 (2013), pp. 1–14.

[26] Ahsanun Naseh Khudori and Tri Astoto Kurniawan. “Transforming EPC Aris Markup Language into BPMN Metadata”. In: *Proceedings of 2019 4th International Conference on Sustainable Information Engineering and Technology, SIET 2019* (2019), pp. 358–363. DOI: 10.1109/SIET48054.2019.8986075.

[27] Maximilian Koegel and Jonas Helming. “EMFStore - A model repository for EMF models”. In: *Proceedings - International Conference on Software Engineering* 2 (2010), pp. 307–308.

[28] Oleksii Kononenko et al. “Mining modern repositories with Elasticsearch”. In: *11th Working Conference on Mining Software Repositories, MSR 2014 - Proceedings*. Association for Computing Machinery, Inc, 2014, pp. 328–331. ISBN: 9781450328630. DOI: 10.1145/2597073.2597091. URL: http://dx.doi.org/10.1145/2597073.2597091.

[29] Vladimir Kotsev, Ivan Stanev, and Katalina Grigorova. “BPMN-EPC-BPMN Converter BPMN-EPC-BPMN Converter”. In: January 2016 (2011).

[30] Daniel Kozma, Pal Varga, and Kristof Szabo. “Achieving Flexible Digital Production with the Arrowhead Workflow Choreographer”. In: *IECON Proceedings (Industrial Electronics Conference)* 2020-October.October (2020), pp. 4503–4510. DOI: 10.1109/IECON43393.2020.9254404.

[31] Marcello La Rosa et al. “APROMORE: An advanced process model repository”. In: *Expert Systems with Applications* 38.6 (2011), pp. 7029–7040.

[32] Urška Lah, James R Lewis, and Boštjan Šumak. “Perceived Usability and the Modified Technology Acceptance Model”. In: *International Journal of Human-Computer Interaction* 36.13 (2020), pp. 1216–1230. ISSN: 15327590. DOI: 10.1080/10447318.2020.1727262. URL: https://www.tandfonline.com/action/journalInformation?journalCode=hihc20.

[33] Steve LaValle et al. “Big Data, Analytics and the Path From Insights to Value”. English. In: *MIT Sloan Management Review* 52.2 (2011). Copyright - Copyright © Massachusetts Institute of Technology, 2011. All rights reserved; Document feature - Charts; Tables; Diagrams; Last updated - 2020-11-17; CODEN - SMRVAO, pp. 21–32.

[34] Ruopeng Lu and Shazia Sadiq. “A survey of comparative business process modeling approaches”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 4439 LNCS. Springer Verlag, 2007, pp. 82–94. ISBN: 9783540720348. DOI: 10.1007/978-3-540-72035-5_7.

[35] Miklós Maróti et al. “Next generation (Meta)modeling: Web- and cloud-based collaborative tool infrastructure”. In: *CEUR Workshop Proceedings*. Vol. 1237. 2014, pp. 41–60.

[36] Robin J.P. Mennens. “The Philips Remote AI Streaming (PRAIS) platform”. English. PdEng Thesis. PhD thesis. Oct. 2020.

[37] Cecilia Titiek Murniati and Ridwan Sanjaya. “Learning technologies in education: Issues and trends”. In: (2017).

[38] Pete Chapman Ncr et al. “Step-by-step data mining guide”. In: *SPSS inc* 78 (2000), pp. 1–78. URL: http://www.crisp-dm.org/CRISPWP-0800.pdf.

[39] Object Management Group. “BPMN Core Elements”. In: *httpbpmnorgSamplesElementsCoreBPMNElementshtm* (2010). URL: https://www.omg.org/bpmn/Samples/Elements/Core_BPMN_Elements.htmhttp://www.bpmn.org/Samples/Elements/Core_BPMN_Elements.htm.

[40] Saheed Popoola, Jeffrey Carver, and Jeff Gray. "Modeling as a service: A survey of existing tools". In: *CEUR Workshop Proceedings*. Vol. 2019. 2017, pp. 360–367. URL: `https://azure.microsoft.com`.

[41] Qualtrics. *Net Promoter Score (NPS®) - The Ultimate Guide | Qualtrics*. 2021. URL: `https://www.qualtrics.com/uk/experience-management/customer/net-promoter-score/?rid=ip&prevsite=en&newsite=uk&geo=GB&geomatch=ukhttps://www.qualtrics.com/uk/experience-management/customer/net-promoter-score/` (visited on 07/31/2021).

[42] Khalid Raza. *APPLICATION OF DATA MINING IN BIOINFORMATICS*. Tech. rep. 2. 2010, pp. 114–118. URL: `http://multalin.toulouse.inra.fr/multalin/multalin.html`.

[43] Gregorio Robles et al. "An extensive dataset of UML models in GitHub". In: *IEEE International Working Conference on Mining Software Repositories* May (2017), pp. 519–522. ISSN: 21601860. DOI: `10.1109/MSR.2017.48`.

[44] *SPARX Systems: Enterprise Architect*. `https://sparxsystems.com/`.

[45] A.S. Tanenbaum M. van Steen. *Distributed Systems. 3rd ed.* distributed-systems.net, 2017.

[46] "Strategies for Qualitative Interviews". In: *Harward University* (2017), pp. 1–4. URL: `http://sociology.fas.harvard.edu/files/sociology/files/interview_strategies.pdf`.

[47] Bedir Tekinerdogan et al. "Introduction to model management and analytics". In: *Model Management and Analytics for Large Scale Systems*. 2020, pp. 3–11. ISBN: 9780128166499. DOI: `10.1016/b978-0-12-816649-9.00009-0`. URL: `https://doi.org/10.1016/B978-0-12-816649-9.00009-0`.

[48] *The CDO Model Repository*. `http://www.eclipse.org/cdo`.

[49] Willi Tscheschner. *Transformation from EPC to BPMN*. Tech. rep. 2006.

[50] Pal Varga et al. "Making system of systems interoperable – The core components of the arrowhead framework". In: *Journal of Network and Computer Applications* 81 (2017), pp. 85–95. ISSN: 1084-8045. DOI: `https://doi.org/10.1016/j.jnca.2016.08.028`. URL: `https://www.sciencedirect.com/science/article/pii/S1084804516301965`.

[51] *What is New in BPMN 2.0? - BPMN-Guide*. URL: `https://bpmn.gitbook.io/bpmn-guide/what-is-bpmn/what-is-new-in-bpmn-2.0` (visited on 06/21/2021).

[52] Stephen A White. "Introduction to BPMN". In: *Ibm Cooperation* 2.0 (2004), p. 0.

[53] Roel J. Wieringa. *Design science methodology: For information systems and software engineering*. 2014, pp. 1–332. ISBN: 9783662438398. DOI: `10.1007/978-3-662-43839-8`.

[54] Claes Wohlin et al. *Experimentation in software engineering*. Vol. 9783642290442. 2012, pp. 1–236. ISBN: 9783642290442. DOI: `10.1007/978-3-642-29044-2`.

[55] Zhiqiang Yan, Remco Dijkman, and Paul Grefen. "Business process model repositories - Framework and survey". In: *Information and Software Technology* 54.4 (2012), pp. 380–395. ISSN: 09505849.

[56] Annie TT Ying et al. "Predicting source code changes by mining change history". In: *IEEE transactions on Software Engineering* 30.9 (2004), pp. 574–586.

[57] Anass Zioual. *Model Analytics Automation System*. Aug. 2021. URL: `https://github.com/nszioual/MAAS`.