Tampere University

Panu Kortelainen

MANAGE YOUR WORKFLOWS

A Classification Framework and Technology Review of Workflow Management Systems

Faculty of Information Technology and Communication Sciences (ITC)
Master of Science Thesis
April 2021

ABSTRACT

Panu Kortelainen: Manage your Workflows: A Classification Framework and Technology Review of Workflow Management Systems
Master of Science Thesis
Tampere University
Master's Programme in Information Technology
April 2021

Managing workflows and complex asynchronous operation flows is a common problem that needs to be solved in a variety of software products. Workflow management systems are used to provide solutions that implement those features and orchestrate their execution. The infrastructure and data models in those products vary significantly and the amount of them can make the choice of a single workflow management system a tedious task.

In this thesis, we try to make this task easier by providing a common classification framework that can be used to compare different workflow management systems with each other. By using the classification framework we can distinguish the project and technical viewpoints from each other and provide a more objective baseline for the comparison of different workflow management systems. A systematic mapping study is used as a method to derive an initial classification framework for the workflow management systems.

In the scope of this study we focus on cloud-native and open source products to get a clear view on the freely available modern solutions on the field. A set of the most popular products with those characteristics is chosen by using a variety of different popularity metrics. As a result we have ten different workflow management systems that meet our popularity and study scope requirements and that can be reviewed against the classification framework.

The initial classification framework is refined with the results of a documentation analysis done on the selected workflow management systems. After that a full technology review is conducted on them using the classification framework. The steps and results of this technology review are documented in the thesis.

Finally, the learnings of that process are gathered into a set of guidelines for selecting a workflow management system. Those guidelines can be used to recreate the study with a new set of systems and iterate through them until a final choice can be made. By offering a classification framework, guidelines for its usage and an example of the review we believe that the work can be extended on any set of workflow management systems and used to perform a review on them against each other.

Keywords: workflow, workflow management, workflow management system, workflow engine, classification, comparison, cloud-native, open source, technology review

TIIVISTELMÄ

Panu Kortelainen: Hallitse työnkulkusi: luokitteluviitekehys ja teknologiakatsaus työnkulun hallintajärjestelmille
Diplomityö
Tampereen yliopisto
Tietotekniikan DI-ohjelma
Huhtikuu 2021

Työnkulun ja monimutkaisten asynkronisten operaatioiden hallinta on yleinen ongelma johon erilaisissa ohjelmistoprojekteissa täytyy löytää ratkaisu. Työnkulun hallintajärjestelmän (workflow management system) käyttö on yksi tapa ratkaista tähän teemaan liittyvät ongelmat. Eri järjestelmiä on lukuisia ja niiden välillä on suuria eroja niin infrastruktuurissa kuin tietomalleissakin, joten oikean järjestelmän valinta voi osoittautua todella vaikeaksi haasteeksi.

Tässä diplomityössä pyritään helpottamaan valintaprosessia luomalla yleinen luokitteluviitekehys, jonka avulla erilaisia työnkulun hallintajärjestelmiä voidaan verrata systemaattisesti toisiinsa. Käyttämällä luokitteluviitekehyksen kategorisointia voidaan projektikohtaiset ja tekniset näkökulmat helpommin erottaa toisistaan ja luoda objektiivisempi lähtökohta eri järjestelmien vertailulle. Luokitteluviitekehyksen pohja on luotu tekemällä systemaattinen kartoitustutkimus alan vastaavaan kirjallisuuteen.

Työn rajaamiseksi keskityttiin järjestelmiin jotka ovat pilvinatiiveja (cloud-native) sekä avoimen lähdekoodin (open source) tuotteita. Näin saatiin selkeä rajaus, johon sisältyvät järjestelmät ovat ilmaiseksi saatavilla ja sopivat hyvin moderniin ohjelmistoprojektiin. Edellä mainittuja reunaehtoja ja erilaisia käyttöaktiivisuuteen liittyviä metriikoita hyödyntäen valittiin kymmenen suosituinta järjestelmää tarkempaa arviointia varten.

Kartoitustutkimuksen avulla luodun viitekehyksen kehittämiseksi ja varmentamiseksi valituille järjestelmille tehtiin ensin dokumentaatioanalyysi. Sen jälkeen viitekehys otettiin käyttöön tekemällä järjestelmien välinen laaja teknologiakatsaus, jossa kaikki luokittelut otettiin huomioon. Katsauksen vaiheet ja tulokset on dokumentoitu osana tätä työtä.

Lopuksi työn aikana opituista asioista ja tuloksista kerättiin ohjeet työnkulun hallintajärjestelmän valintaan. Ohjeiden avulla voidaan valintapäätökseen liittyvä prosessi käydä läpi alusta loppuun työssä käytettyä työtapaa mukaillen. Voidaan todeta, että luokitteluviitekehyksen, valintaohjeistuksen ja esimerkinomaisen teknologiakatsauksen myötä on tutkimuksen laajentaminen helposti toteutettavissa erilaisille uusille järjestelmille ja projekteille.

Avainsanat: workflow, työnkulku, työnkulun hallintajärjestelmä, luokittelu, vertailu, avoin lähdekoodi, pilvipalvelu

PREFACE

The journey has been long but not exhausting. As years go by, you tend to get attached to places, their customs and especially the people around you. I have an unsurpassed gratitude towards all of you who have, in a way or other, been with me during this time, sharing the voyage. The amount of things I have learned from you is more than what can be deciphered from the diploma or the pages of this thesis.

My life has been enriched by multiple different organizations and groups while studying in the Tampere University, and its predecessor Tampere University of Technology. Those groups have given me a sense of time and permanence, in a way that I feel being a part of a longer line of similar stories.

I must thank the Student Unions, new and old, for providing a thriving community to do and learn. I am thankful to TiTe, my guild, for growing me up from a freshman into an active teekkari. I am thankful to the sporrrrts team NMKSV and the friends from TEA-club for good times and a lifelong companionship. I am more than grateful to each of the small and the big boards, projects and positions of trust that I was privileged to be a part of. Also, special kudos for the *rocks who write* for providing peer support and joy to my days of research.

I am also forever grateful to my family for giving me a safe environment to grow and learn. I wouldn't be here without your support.

I might not be here either without Nokia Corporation understanding that it's not about the destination but the journey, and reminding me that sometimes you also need to arrive somewhere. In the end, big thanks to Davide Taibi and David Hästbacka for providing continuous support and guidance during the thesis writing process. I wish all the best for your future studies.

Now it is finally the time to give a closure for this era and continue towards a new, intriguing future.

Tampere, 7th April 2021

Panu Kortelainen

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

API	Application programming interface
BPMN	Business Process Model and Notation
CI/CD	Continuous integration and continuous delivery
CLI	Command-line interface
CNA	Cloud-native application
CNCF	Cloud Native Computing Foundation
CWL	Common Workflow Language
DAG	Directed Asyclic Graph
GUI	Graphical user interface
IDE	Integrated development environment
JSON	JavaScript Object Notation
PDL	Process Definition Language
REST	Representational state transfer standard
RQ	Research Question
WAPI	Workflow application programming interface
WDL	Workflow Description Language
WfMC	Workflow Management Coalition
WfMS	Workflow Management System
WS-BPEL	Web Services Business Process Execution Language
XPDL	XML Process Definition Language
YAML	YAML Ain't Markup Language
YAWL	Yet Another Workflow Language

1 INTRODUCTION

Managing workflows and complex asynchronous operation flows has been a common problem in software systems for decades. [1] The solutions range from utility tools that automate high-level business objectives via graphical user interface to programmable solutions that handle and followup complex, dynamically generated execution graphs and rules in scalable cloud-native environments.

These workflow management systems (WfMS) provide an infrastructure and models for executing, monitoring and evolving different kinds of workflow processes and tasks.

However, the solutions base is huge and the most common denominator is the theoretical model behind workflow management systems. Therefore, the technical and project characteristics vary significantly between different workflow tools. This makes selecting the most suitable tooling a complex task with multiple dimensions from high-level project requirements balanced against low-level functional requirements.

The goal of this thesis is to provide a workflow management system classification framework to make it easier to choose what kind of a system would suit the needs of a specific organization or a software project. The classification framework comprises of two different viewpoints.

The *project view* summarizes high-level needs of e.g. a project manager or a software architect who need to narrow down what kind of a systems are valid to consider. The second viewpoint is named *technical view*, and it has a more detailed set of technical requirements, which can be used to make the final choice between tools that meet the project and business requirements.

By applying the workflow management system classification framework we can review our alternatives objectively and get a more thorough view on the strengths and weaknesses of the system that has been chosen for our purpose.

To narrow down the study and to focus on the modern approaches of the field we chose to limit our study to cloud-native workflow management systems and their characteristics. By this meaning software systems that are designed to be run in cloud environments. We also decided to include only freely available open source products to ensure a more comparable set of items to be analyzed. These decisions allowed us to get a good view on the current options that are available for any project to use, thus enabling the creation of a classification framework that works with modern day workflow management systems.

The initial classification framework was formed by conducting a systematic mapping study on available scientific literature related to comparison and classification of workflow management products. The results were then refined to align with the context of this study by a documentation analysis done on the most popular open source and cloud-native workflow management systems.

The classification framework was also used to perform a workflow management system technology review, which classifies ten popular workflow management systems to give a good understanding on the current tools available for use. The results and lessons learned during this review were also gathered together into selection guidelines which can be used to apply the classification framework to a new set of suitable tools to consider for project specific needs.

The contribution of this work is threefold:

(i) A workflow management systems classification framework,

(ii) A workflow management system technology review and

(iii) Workflow management system selection guidelines.

The structure of this paper is organised in the following way. Section 2 introduces and describes the research questions of the study. Section 3 consists of background and theory about workflow management systems and related concepts. Section 4 goes through the research methodology used to achieve the results of this paper. Section 5 introduces the *Workflow management system classification framework* and its views. Section 6 contains the *Workflow management system technology review*, which is used to create the *Workflow management system selection guidelines* in Section 7. In Section 8 the validity of this study is pondered and Section 9 holds the discussion and implications of the study. The Section 10 consists of future work and conclusions for the topic.

2 RESEARCH QUESTIONS

This section discusses the research questions and provides background for them. The research questions are as follows:

RQ1 What are the main classification features to distinguish workflow management systems from each other?

The problem in making a choice about what would be the best workflow management system for a certain software project is that we should first be able to identify the differences in those systems. Before being able to do so we need to find the right things to measure and combine those into a meaningful set of data. This research question combines the literature review, documentation analysis and forming of the classification framework itself.

RQ2 Which are the most popular open source, cloud-native and generic workflow management systems? What are the main differences between these systems?

For gaining an understanding on what are the most popular open source, cloud-native and generic workflow management systems we can make sure that the data we are looking at is relevant and corresponds to the actual usage of the systems. This research question is used to filter out the most interesting workflow management systems in the scope of this study. After that a technology review is done to evaluate the differences between the systems.

RQ3 How to select an open source Workflow management system for a generic cloud-native environment?

One of the main motivations of this thesis was to find a way to help choosing a suitable workflow management system for a certain project. To fully utilize the classification framework and the results of the technology review we aggregate the results into selection guidelines. That can then be used to make the decision-making easier when adopting a new workflow management system.

3 BACKGROUND

To get a good understanding on the concepts discussed in this study we describe the theoretical background behind the subject in this section. The study focuses on concepts around cloud-native workflow management systems and their evaluation. Therefore, we discuss workflow management topics, such as workflow management systems, workflow definition languages and common concepts in workflow management. We also look into cloud-native applications, their features and implementation concepts.

3.1 Workflow management

The concept of workflows has developed from the need to define processes in industrial manufacturing and the surrounding activities such as office practises. From the separation of those work activities into tasks, procedures, roles and rules, the efficiency could be controlled and bottlenecks more easily defined. The main rationale for embracing the new way of defining processes was efficiency, and splitting the work into smaller pieces made it possible to reorganize and tune the overall process. Even though the first implementations of workflows were performed by humans, the shift into information technology based solutions was natural as the rules and tasks were just now partly done by software. [2]

The management of those workflows became an important step in how the actual execution itself is eventually delivered as the requirements change, the workflows grow more complicated and the amount of data increases. Monitoring, controlling and modifying the complex workflows requires a system that can answer the needs of the changing landscape. For this purpose, workflow management systems (WfMSs) were developed. The WfMSs provide a core component in service-oriented systems that require complex orchestration or are tightly coupled with business processes.

Workflow or a process definition language (PDL) is a crucial part in the process of implementing or choosing a workflow management system. As the language chosen affects the way how the workflows are defined and how the applications used by the WfMS are developed it has a heavy impact on the whole system developed around it. If the language proves to be a wrong choice for the business requirements or if it has a limited transferability between different workflow management systems, it can easily lead into a vendor lock-in and difficulties in implementing the required use cases. [3]

For tackling this problem, there has been a significant effort in the standardization of the workflow concepts. One of the biggest contributors on this field has been the Workflow

Management Coalition (WfMC), which is a global organization bringing together organizations, developers and research groups engaged in the topic. They have created XML Process Definition Language (XPDL), which is the leading process definition format used to store and exchange process models, the Business Process Simulation standard (BPSim), which defines parametrization and interchange of data allowing analysis for optimization of the process models and the Business Process Analytics Format (BPAF), which provides efficiency and effectiveness insight for organizational processes. [1]

The XPDL format is especially interesting in the scope of this study as it can be used to describe the process definition, workflow, itself. The XPDL is also used as the serialization format for BPMN, which is a visual process notation standard that is widely adopted in the industry. However, the standardization has not been a complete success and BPMN standard has been criticized for containing multiple ambiguities and underspecifications of its concepts. [3] This can cause different stakeholders to implement the constructs differently and by that significantly reduce the benefit of having a standard in the first place.

As for the recent years of development in the field of cloud computing, the workflow management systems have also moved towards being designed in a scalable, cloud-native way. This development has created multiple new workflow management systems in the field to challenge the design principles and architectural choices made in the older systems.

The recent advancements in the field of workflow management have also moved towards new ways of defining workflows. The standard definitions are not used as widely as before and their place has been taken by defining the workflows in less strict JavaScript Object Notation (JSON) or YAML Ain't Markup Language (YAML) formats. Some of the WfMSs have completely removed the need for a common language by providing only APIs that can be used by a programming language making the whole workflow definition a software development process. These approaches have their advantages and disadvantages, which can be discussed in the following sections.

3.1.1 Workflow concepts

Georgakopoulos et al. [2] define workflow as "a collection of *tasks* organized to accomplish some business process". These tasks are described to be performed either by software systems, humans or as a combination of them both. The definition from WfMC Terminology and Glossary [4] leans to the same way as they describe workflows as "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules".

From these definitions we can see that workflow consists of **tasks** or *work items* that conduct work, pass information or make decisions. These tasks can be defined as something that is done by the computer, or something done by an external source, e.g. a human pressing a button. This form of work can also be described as an **activity**. Also **rules** are mentioned as part of a workflow definition. For example, they can be post- or pre-conditions that are used to determine whether a process can be started or completed. [4]

For handling the connections between different tasks Aalst and Hee [5] use the term **routing**. It is described to determine "which tasks need to be performed and in which order". According to the source, routing can determine if tasks are run *sequentially*, *in parallel*, *selectively* or *iteratively*.

Sequential routing means that the *tasks* are executed one by one in a sequence where each of the tasks depends on the previous one. The tasks in sequence often pass their results as an input to the next task. The opposite of sequential routing would be *parallel routing* where the tasks are executed simultaneously. In this case the result of each of those tasks cannot affect the others and they have no dependency on each other. Parallel routing is also often referred to as a *parallel split* or a *fork*. [6] Parallel tasks are initiated by an AND-**split** and ended by a similar AND-**join** operation that is a synchronization pattern for converging the branches of execution.

Selective or *conditional routing* is a concept where there are multiple optional tasks that could be run, but based on a set of *rules* or properties a choice is made on which of the tasks are to be initiated. This behaviour could be described as a kind of a switch-case situation where the case is a condition or a set of conditions. If only one route can be taken this kind of an operation is an exclusive **choice** or a XOR-split and the end of it a simple **merge** or a XOR-join. If multiple choices are available the pattern is called a multi-choice branch or an OR-split. This kind of a workflow pattern can be merged by a *structured synchronizing merge*, *multi-merge* or a *structured discriminator* pattern. [6]

The last form of routing mentioned was *iterative routing* or an *arbitrary cycle*, in which one task in a workflow is repeated multiple times. It can be described as kind of a while-loop where the condition needs to be met before the loop ends. This could mean running a certain task for ten times in a row or repeating a task execution until a certain other condition is met.

3.1.2 Workflow definition language

The workflow patterns described in previous section are a subset of different control mechanisms that are recognized as a part of workflows. All or some of these patterns are somehow implemented in different workflow or process definition languages. There are multiple different solutions to describe how the workflow is formed. The XPDL format standardized by WfMC [1], as mentioned before, is the most prominent one in the field. On top of that there are multiple other standardization efforts such as Web Services Business Process Execution Language (WS-BPEL) [7], Yet Another Workflow Language (YAWL) [8], Common Workflow Language (CWL) [9] and Workflow Description Language (WDL). [10]

During recent years of development the definition of workflows has also gone into a direction where the definition follows no standard at all. The strict standards can seen as an overhead to the development work and the implementation of the standard is not guaranteed to be equivalent between different workflow management systems. [3] Workflows are defined in formats like JSON or YAML where the contract is only between the workflow management system and the definition language it has. This creates a risk of vendor locking when taking a new system into use.

Some of the workflow management systems have completely removed the need to have a separate workflow definition language. Instead, they have implemented a programmable API that can be used to write the workflows as code. This, however, makes workflows more a technical solution not providing easy way to contribute without programming knowledge.

To summarize the workflow definition languages mentioned previously, they have been collected below to give a good idea on the possibilities in the field.

XPDL (XML Process Definition Language) is a format ratified by WfMC (2002) that can be used as a interchange process model between different tools. It describes the process definition, network of different activities and their relationships together with criteria for starting or terminating processes. XPDL is also a serialization format for **BPMN**, which is a visual process notation standard that is widely adopted in the industry. [11]

Whereas the first definition of BPMN was purely graphical, the second version of it also introduced execution semantics and serialization syntax. For modern workflow management systems, **BPMN 2.0** specification is one of the most used definition languages, allowing all the key stakeholders to participate and understand in the definition process. The latest version for the specification, BPMN 2.0.2, was published in 2014. [12]

WS-BPEL (Web Services Business Process Execution Language) or simply BPEL is also an XML-based executable language published for Web Services and Service Oriented Architecture (SOA) in 2003. It tries to standardize the business process orchestration mechanics such as sequencing parallel and asynchronous operations, fault handling and long running transactions. Because BPEL is designed with web service approach it models web services as processes in its definition. [7]

YAWL (Yet Another Workflow Language) is a graphical workflow definition language based on Petri nets and extended to facilitate complex workflow features on top of that. The design of YAWL is based on the workflow patterns introduced by Van der Aalst et al. [13] and it tries to tackle the problems of other workflow languages inability to provide solutions to the patterns introduced. [8]

CWL (Common Workflow Language) standard specifies a data and execution model for workflows implemented on top of a variety of platforms. The target segment for the language is data-intensive science meaning fields such as bioinformatics, chemistry or physics. The notation used is written in JSON or YAML format. CWL is one of the newer workflow definition languages as its version 1.0 dates to year 2016. [9] **WDL** (Workflow

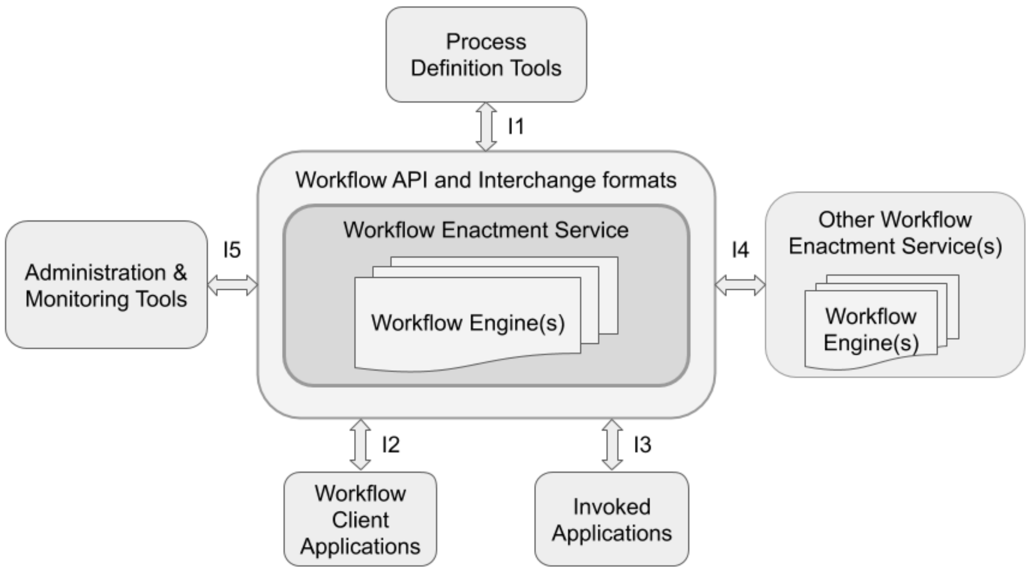
Process Definition Tools
I1
Workflow API and Interchange formats
Workflow Enactment Service
Workflow Engine(s)
I5
Administration & Monitoring Tools
I4
Other Workflow Enactment Service(s)
Workflow Engine(s)
I2
I3
Workflow Client Applications
Invoked Applications

Figure 3.1. *Workflow Reference Model by WfMC [14]*

Description Language) is a language that tries to specify data processing workflows in a human-understandable format. The main users of this community driven standard are also mainly in the field of science like in CWL.

Workflows as **Code** have no other language expression than a software interface that can be used to perform the workflow operations. The interface then takes care of the workflow features such as state handling, scalability or fault handling. The approach of using code instead of an external language makes the workflow definition a lot more versatile as it contains all the features the programming language would. It, however, makes the workflow management system not available for other stakeholders than programmers as defining workflows cannot be done straight in a language that could be used without programming experience.

3.1.3 Workflow management systems

Workflow management system is a software system designed to define, control and manage workflow execution. A generic workflow application architecture has been specified by WfMC in a *Workflow Reference Model*. [14] In the model the interfaces and components of a workflow management system are defined together into six different sections as displayed in Figure 3.1.

The *workflow enactment service* is the run-time environment for the workflow process that executes the workflow instances in one or more workflow engines. It is also responsible for interpreting the process definition and communicating with external interfaces if needed. The process and activity control logic is separated in the model. Also, the enactment service does not necessarily need to be a single entity; it may be either functionally distributed or centralized. For interacting with the enactment service there is a workflow application programming interface (WAPI) between the interfaces marked from I1 to I5. [14]

A *workflow engine* described in the reference model is the actual execution environment of the workflow instance. The level of responsibilities between the engine and the whole enactment service, however, varies. The workflow engine typically interprets the process definition, controls the execution and handles the data required in specific process instance.

The interfaces described in the picture use the WAPI to access the workflow enactment service. The Workflow API is used to interchange data and workflow definitions, to control the activities done and to monitor the whole service. The interfaces as defined in the workflow reference model are described below. [14]

The first interface (I1) is used to exchange process definition information between the counterparts. It may be used to import or export the definition from the runtime environment to the workflow definition environment. This makes it possible to differentiate the process modeling from the actual work done in code level. It also allows the user to choose their modeling tool as long as it produces definition that can be read by the workflow enactment service.

The second interface (I2) towards the workflow client applications. The client application can be, for example, an application that records user inputs from a machine towards the workflow enactment service. The API then uses a predefined interface to map the inputs to specific workflow definitions. The client applications could also control the process flow or follow the status of certain process in stances.

The third interface (I3) described in the picture is towards invoked applications which are external from the workflow management system like system process calls or transactions. These interfaces may be triggered by the workflow engine and they may respond notifications or events back to the system.

The fourth interface (I4) is an interface that is used to communicate between other workflow enactment services. That interface can be used to transfer activities or data between other workflow services.

The last interface (I5) is used to monitor and administer the system. This interface handles user and role management, auditing and other supervisory functions. The use of this interface allows multiple different solutions to be managed and monitored by a single management system.

The ideas of the workflow model described above are still visible in many modern day workflow management systems. Even though paradigms change and certain architectures gain more space in the field, the generic aspects still remain valid.

3.2 Cloud-native workflow management

The cloud-native applications have been reshaping the field of software engineering since the first general purpose public cloud service was introduced by Amazon Web Services in 2006. [15] The first release started a trend of providing software, infrastructure or platforms as a service in the cloud. This kind of a product family is often described as being cloud-native.

According to the research done by Kratzke and Quint [16] the term cloud-native is commonly understood to mean applications or services that are designed using **cloud-native application** (CNA) **principles**, built with certain **architectures**, containing certain **properties** and utilizing a set of **methods** accompanied with the design patterns. All of these high-level topics are affecting each other and contributing towards a cloud-native approach.

The study mentioned in the previous section proposes a cloud-native application to be defined as being "distributed, elastic and horizontal scalable system composed of (micro)services which isolates state in a minimum of stateful components". [16] For traditional workflow management systems, the change towards being a CNA requires a careful design on the *stateful components* as most of the implementations rely on a traditional database that cannot be scaled horizontally. Also scalability of the workload in an elastic manner is a problem with systems that are not designed with that in mind.

The rest of the definition states that "the application and each self-contained deployment unit of that application is designed according to cloud-focused design patterns and operated on a self-service elastic platform". [16] This underlines the need for carefully designing the components in a cloud-native manner. It can be seen in many established applications that are moving towards a cloud deployment that the first steps of design only containerize the old components into a set of deployables that might not actually be designed for that purpose. This might lead into deploying a set of monolithic applications inside a container.

For cloud-native workflow management systems there are a few key aspects that can be risen as examples to be taken under a more detailed inspection when determining the capabilities of running efficiently in a cloud environment. The concept of *stateful components* for storing the workflow data was mentioned in the previous sections and it is definitely one of the most critical aspect to be taken into consideration. A truly cloud-native application cannot be designed as being highly available and scalable without also handling the state in aforementioned fashion.

Additionally, the design must not rely on any components that act as a single point of failure, for example a workflow software might rely on a single instance scheduler that dispatches the work orders. If this kind of a component fails, it ensures a downtime for the whole system relying on it.

3.2.1 Virtualisation: Docker

The concept of virtualisation is tightly coupled with cloud-native architectures, especially when specifying the container solutions of it. In principle, virtualisation means wrapping a piece of software infrastructure as an entity that emulates a whole software system. A virtual machine is one way to implement virtualisation; in it the whole operating system is emulated on top of a hypervisor system. For container based virtualisation, the whole encapsulated software works on top of the kernel of the host. This allows a smaller overhead and further empowers customization of the contents.

A common way to deploy cloud-native applications is using a tool named Docker. Docker is a container engine that provides management for containerized applications. A containerized application is an application that is deployed using lightweight virtualisation technology wrapping the software run into an executable package that includes everything needed for running the application. The containers deployed this way are often used to form a set of software components that communicate with each other to provide the needed functionality. [17]

4 RESEARCH METHODOLOGY

In this section we present the research method applied. We adopted a systematic mapping study process, following the guidelines proposed by Petersen et al. [18] together with the snowballing techniques proposed by Wohlin [19]. A simplification of the whole process is shown in Figure 4.1.

The whole process of our study consists of eleven distinguishable phases. In first three sections, we defined the conditions for study search, the search strategy and described the selection criteria for the studies. After that we used the studies found during the search phase to apply a snowballing technique to them in order to find studies that we had previously missed. After that the study data was assessed and extracted into a set of predefined data fields.

From aforementioned data we derived an initial classification framework for the research. After that, we did a systematic search for workflow management systems, applied our exclusion criteria on them and conducted a popularity study on the remainders. With those results we selected ten workflow management systems for further analysis. Those were used to refine the initial classification framework using the data we had discovered. This framework was then used to do a thorough review on the engines. The following sections describe the research method's used in detail.

4.1 Study search

In this stage, we specify the search scope to identify the most applicable bibliographic sources to support this mapping study. The rationale for the scope decisions is described in the following section.

4.1.1 Search scope

The scope of the search should be defined carefully as it has a big impact on the effort required and the coverage acquired from the primary studies related to our researched topic. For this mapping study we chose to define our scope by specifying a time period of relevant publications and to describe which kind of electronic databases we use for searching them.

The *time period* to be used to limit our search was chosen to start from 1993 when the

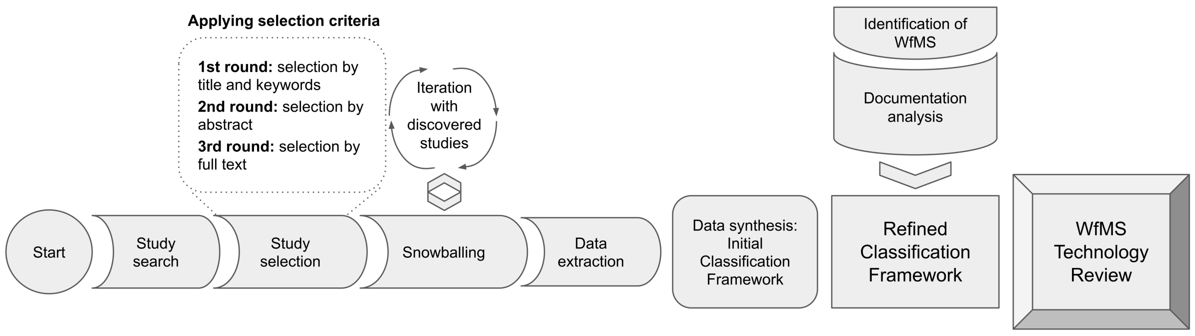
Applying selection criteria
1st round: selection by title and keywords
2nd round: selection by abstract
3rd round: selection by full text
Iteration with discovered studies
Identification of WfMS
Documentation analysis
Start
Study search
Study selection
Snowballing
Data extraction
Data synthesis: Initial Classification Framework
Refined Classification Framework
WfMS Technology Review

Figure 4.1. *The research methodology and procedure*

Workflow Management Coalition was established and their workflow standardization work began. [1] The search period end was chosen to be 2020 as that is when this study was started.

The *electronic databases* considered from the search results in this study were chosen to be the ten main databases discussed by Chen et al. [20], including: Google Scholar, IEEEXplore, ACM Digital Library, EI Compendex & Inspec, ISI Web of Science, CiteSeer, Science Direct, SpringerLink, Wiley InterScience, SCOPUS and Kluwer Online.

4.2 Search strategy

The search strategy that we followed for this study comprised of two parts:

(i) The search string was analyzed and adjusted to return meaningful results from the search engine. The initial search was done with ("workflow management system") AND (comparison OR evaluation). We also included alternative term "workflow engine" and "classification framework" into the search string. During the evaluation of the search string we noticed that our results were more accurate when applying a plural form to the first keyword pair. The search phrase was finally changed to ("workflow engines" OR "workflow management systems") AND (comparison OR evaluation OR "classification framework")

(ii) The automatic searches were performed on Google Scholar and the results were collected until no relevant papers were emerging in consequent ten pages of twenty articles in the search results.

4.3 Study selection

In this section we describe the *selection criteria* and the *selection process* that was followed in the study.

4.3.1 Selection criteria

Papers to be included (I) or excluded (E) in the search were identified and chosen with the following criteria:

- I1: The paper evaluates and compares existing workflow management systems or workflow engines
- I2: The paper is peer-reviewed and published in a journal, a conference proceeding, a workshop proceeding or a chapter of a book.
- E1: The paper focuses solely on certain aspects of the system, e.g. performance benchmarks
- E2: The paper is not available as full text
- E3: The paper is published in a form of a tutorial, an abstract or a talk, which is not considered to contain enough information for the study and thus cannot be included in the results.

4.3.2 Selection process

The selection process of the study by the defined inclusion and exclusion criteria is described in Table 4.1. A more thorough explanation on how the process itself was conducted is following:

- (i) The first round of study selection: a researcher went through the studies based on the title and the returned matching keywords shown on the Google Scholar search while applying the inclusion criteria I1 and I2 and exclusion criteria E1 to the contents. The exclusion criteria E2 and E3 were not used in this step as the availability of a full text or the form of the paper cannot be reliably known without opening the relevant database link. Every study with any doubt about its inclusion was included for the second round of the selection process.
- (ii) The second round of study selection: a researcher went through the abstracts of the studies selected in the first round while applying the inclusion criteria I1 and exclusion criteria E1 and E3 to them. Because inclusion criteria I2 was already identified fully in the first round of selection it was not necessary to apply it anymore in this round. Any doubts in the inclusion of a study resulted in the study being included in the third round of study selection.
- (iii) The third round of study selection: a researcher did a full text reading of the studies included in this section while applying the inclusion criteria I1 and exclusion criteria

E1 and E2 on the selected studies. As the exclusion criteria E3 was already identified fully in the second round of selection it was not necessary to apply it anymore in this round. Any doubts were discussed with the other researchers until issues were resolved and the final results were mutually accepted.

Table 4.1. *Study selection: the use of selection criteria*

Selection round	Criteria used
1st round: selection by title and matching keyword results	I1, I2, E1
2nd round: selection by abstract	I1, E1, E3
3rd round: selection by full text	I1, E1, E2

4.4 Snowballing

On top of the selection process we used the snowballing technique [19] to identify a more thorough set of relevant publications. In practise, we performed a forward snowballing by analyzing all the research papers with citations to any of the publications selected earlier, and after that performed a backwards snowballing recursively by analyzing the research papers that were cited by the initial set of publications. Each of these checks follow the three steps mentioned in the selection process and each of the newly selected papers were snowballed themselves.

By doing this we ensured that a minimal amount of relevant studies were left out in the study selection phase. In the end, all of the results were combined together and used in the following steps of the study.

4.5 Data extraction

To start formulating answers for the research questions in Section 2, the selected sources were used to provide the information portrayed in Table 4.2. This data was recorded and then used to form the initial classification framework.

Before starting the final data extraction, the data items to be collected were tested with a pilot data extraction on three studies. The findings were collected and the extracted fields were updated if ambiguities still existed.

4.6 Data synthesis: derive initial classification framework

Data synthesis is used to gather the extracted data together to get closer into answering the research questions by forming a classification framework for workflow management systems. For creating an initial classification framework to begin with, a technique called keywording [21] was used. The overall procedure was the following:

Table 4.2. *Data extraction: the data collected from each study*

#	Data item	Description
D1	Year	The year when the study was published
D2	Venue	The name of the venue where the publication was published
D3	Type	Journal, conference paper, workshop proceeding or a book chapter
D4	Evaluation categories	High level categories on which the evaluation was separated in the study
D5	Evaluated attributes	The attributes used to evaluate the WfMSs in each corresponding category

1) One author analyzed the data collected of each publication for identifying common concepts and keywords matching them, the second author then verified the correctness, and in case of disagreements, discussed with inconsistencies with the first author.
2) The keywords found were discussed and gathered together into clusters to form the initial classification framework.

After going through the process, we had a set of categories that could be used in the classification framework, e.g. Development, Licensing, Release maturity, Architecture and Code reuse, each of them containing more detailed criteria and details. As an example, Development contains the workflow and workflow task definition categories, which listed the supported languages and definition styles for the workflow management systems.

4.7 Identification of the relevant workflow management systems

In this step we selected ten relevant workflow management systems for subsequent refinement of the initial classification framework. The workflow management systems analyzed and collected in this section will also answer the RQ2 as we collect a diverse set of popularity data from different sources.

We defined that a workflow management system must comply with the following requirements to be included in further documentation analysis and review:

- The workflow management system must be *general-purpose* in a way that it is not designed only for certain operations or scenarios (e.g. employee processes in a company or data transformation workflows)
- The workflow management system must be *cloud runnable* in Docker or as a serverless distribution.
- The workflow management system must be *open source* and usable in production

without commercial licenses.

- The workflow management system must be *actively maintained* and *established*, meaning that there must be a sufficient number of stakeholders contributing towards new releases for the system, and that the popularity meets certain thresholds.
- The workflow management system documentation must be available in English.

We conducted the search and selection of relevant workflow management systems by applying these requirements and following a process described in the following sections.

Table 4.3. *Initial set of workflow management systems and reasons for exclusion in this research*

Exclusion reason	Name
N/A - Included	Activiti
	Apache Airflow
	Argo Workflows
	Brigade
	Cadence
	Camunda BPM
	Conductor
	Digdag
	Flyte
	Kogito
	N8n
	Wexflow
	Zeebe
Commercial license	Flowable
	Prefect
	Processmaker
	Rundeck
Not generic-purpose	Azkaban
	Luigi
Not maintained	Lyra

4.7.1 Phase 1: WfMS Search

For searching the workflow management systems, we used both gray and white literature sources. The systems mentioned in the studies found during the systematic mapping study were considered together with the results of a search done on predefined search keywords on Google. As a verification step on finding a comprehensive list of results we compared them to an unofficially maintained list of open source workflow management systems [22]. The initial list of items used for the research was limited by applying an effort bounded stopping criteria [23] on selecting twenty different workflow management systems for further analysis of the search results.

We went through the initial list of workflow management systems found at this stage and checked them against the criteria defined earlier. This resulted in the exclusion of seven systems on the list. For example, we excluded Lyra, because it was not actively maintained by its authors as they stopped working on the project. Also, four of the systems stated that they would require a commercial license to be used in production and two of them were considered to be tailored for a too specific use case. The initial list of workflow management systems and their reasons of exclusion, if applicable, are listed in Table 4.3.

It should also be noted that Activiti and Camunda BPM are two considerably similar systems as Camunda BPM is a development fork of Activiti, which split off the project in 2013. Anyhow, the development has been diverged onto different branches for so long that we could not leave either one out of the scope of this study for that reason. [24]

4.7.2 Phase 2: WfMS Selection

To find the actively maintained and widely used workflow management systems that have their source code available and that can be used in a production environment without commercial obligation, we chose popularity as one of our main criteria in the selection. This way we could get a set of workflow management systems that represents the real use of them in the field. We used a set of popularity quantifiers for the systems like the amount of search engine hits, GitHub repository statistics and amount of Stack Overflow [25] questions posted on the topic.

The amount of search engine hits was counted by applying the workflow management system name as the search string (e.g., “Apache Airflow”, “Zeebe”, “Argo Workflows”) or by adding keyword "workflow" in the end of the search on names that gave multiple unrelated results (e.g. "Cadence workflow"). We approximated the overall popularity and activity of the source code in GitHub by measuring the amount of releases, contributors (including anonymous) and stars given by GitHub users. GitHub statistics were fetched using the GitHub API. In Table 4.4 we describe the search and code activity thresholds that were used to choose the final tools.

These thresholds were used to cut out part of the initially found workflow management systems by having a requirement of matching at least three of these thresholds.

Table 4.4. *Popularity: Search and code activity thresholds*

Platform (measure)		Threshold (amount)
Google	hits	50k
Stack Overflow	questions	50
Github	stars	500
	contributors	50
	releases	20

Table 4.5. *Popularity: Media and scientific literature platforms*

Platform (measure)		Threshold (amount)
Media Hits	Reddit	1000
	Medium	1000
	DZone	50
Scientific Search Hits	Google Scholar	100
	ResearchGate	50
	Scopus	10
LinkedIn Group	Members	200
Google Group	Amount	1
	Posts	500
Community Page	Exists	Yes

This resulted in the following list of tools: Apache Airflow, Activiti, Argo Workflows, Brigade, Cadence, Camunda BPM, Conductor, Kogito, N8n and Zeebe.

To make a more thorough popularity assessment of these tools we also did an analysis on their appearances in different online media platforms and scientific literature. We also added thresholds for those to be able to quantify which workflow management systems are the most popular in the field. These thresholds and sources analyzed can be found from Table 4.5.

Table 4.6. *Full popularity data: workflow management systems chosen for documentation analysis*

Platform	Measure	Apache Airflow	Camunda BPM	Conductor	Activiti	Cadence	Brigade	Zeebe	Argo Workflows	N8n	Kogito
Google	hits	350k	150k	1000k	340k	778k	133k	169k	10k	39k	86k
GitHub	stars	17000	1600	2700	7100	3900	2000	1300	5800	8100	129
	contributors	1454	216	153	269	68	87	55	233	68	276
	releases	144	91	282	494	51	31	59	88	331	20
Stack Overflow	questions	500	500	500	500	500	267	33	472	10	23
Media hits	Reddit	2720	120	4009	44	2370	713	43	1260	51	14
	Medium	8830	780	1230	139	1250	489	403	1130	701	192
	DZone	198	177	53	35	19	153	2	557	0	34
Scientific search hits	Google Scholar	4510	714	2700	1280	206	25	100	2150	8	12
	ResearchGate	158	1250	636	8680	873	17	0	59	0	22
	Scopus	35	10	2	29	0	0	0	10	0	0
LinkedIn group	Members	637	791	-	886	-	-	16	-	-	-
Google group	Amount	1	2	-	1	-	-	-	-	-	2
	Posts	37	2266	2913	432	572	1	612	11	0	1009
Community Page	Exists	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 4.7. *List of selected workflow management systems, popularity scores and documentation sources*

Workflow management system	Popularity Score	Documentation Source
Apache Airflow	14	https://airflow.apache.org/docs/
Camunda BPM	13	https://docs.camunda.org/manual/
Conductor	12	https://netflix.github.io/conductor/
Activiti	11	https://www.activiti.org/documentation
Argo Workflows	11	https://argoproj.github.io/argo/
Cadence	11	https://cadenceworkflow.io/docs/cadence/
Brigade	7	https://docs.brigade.sh
Zeebe	7	https://docs.zeebe.io
Kogito	6	https://docs.jboss.org/kogito/release/latest
N8n	4	https://docs.n8n.io/reference/reference.html

Table 4.7 contains the final workflow management systems selected for the documentation analysis and the popularity score calculated by the amount of popularity thresholds passed (in Table 4.4 and Table 4.5) and the respective documentation sources we considered. Data used to calculate the popularity score based on predefined thresholds can be found from Table 4.6. The data was fetched on 3.7.2020.

4.8 Analyze workflow management system documentation

The documentation analysis was done by going through the official documentation sources of the workflow management systems listed in Table 4.7. While doing the documentation analysis it was reflected against the initial classification framework and the missing parts were notioned. After conducting the full documentation analysis the results were discussed between the authors to reflect the needed changes for the initial classification framework.

4.9 Refining the classification framework

After conducting the documentation analysis and doing the modifications based on the notions that were found there, the whole classification framework was analyzed as a whole by the authors. The results of the refinement and the final workflow management system classification framework is described in Section 5 (Workflow management system classification framework).

4.10 Conduct workflow management system technology review

The final step to fulfil the goals of this study was to do a thorough technology review for the workflow management systems identified in Section 4.7. The review was done by going through the classification framework categories for each of the workflow management systems and collecting the results down on a spreadsheet.

The results of the technology review were collected by the first author. After that, the verification of the results was done by the other author. For resolving conflicts the issues were discussed and final results were recorded only after all the parties were satisfied with them. The full technology review and the results are presented in Section 6 (Workflow management system technology review).

5 A WORKFLOW MANAGEMENT SYSTEM CLASSIFICATION FRAMEWORK

To make a justified choice on what kind of a workflow management system to use in a project, many things need to be taken into consideration. The mapping study described in previous sections was used to supply us with relevant data to form the classification framework. We conducted a study search, which resulted in a total of twelve studies to pass our selection criteria described in Table 4.1. The studies are as follows: [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36] and [37]. Those studies were also snowballed, which resulted in four more studies: [38], [39], [40] and [41] to be included in this study. The total amount of studies used to derive the initial classification framework was sixteen. The results of studies filtered out and selected by study search and snowballing are described in Table 5.1.

Table 5.1. *Study selection: Results of study search and snowballing*

	Selection round	Studies passed
Study search	Passed 1st round	43
	Passed 2nd round	35
	Passed 3rd round	**12**
Snowballing	Passed 1st round	6
	Passed 2nd round	5
	Passed 3rd round	**4**
Total		**16**

The distribution of the results over publication types were split as described in Figure 5.1. From the results it is clear that most of the academic discussion over the topic is done in conference proceedings, which totaled thirteen different studies. For journal entries we had two matches and only one book chapter was used as a source in the study.

When looking into the yearly distribution of the publications shown in Figure 5.2 it can be seen that the studies range evenly from the beginning until the end of the time period chosen. This implies that the topic has had a considerably constant amount of interest over the years in the community.

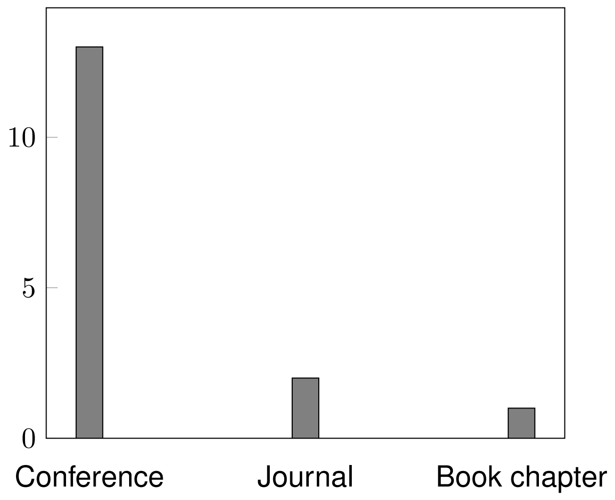

Figure 5.1. *Distribution of selected studies over publication types*

The publications were analyzed to form an initial classification framework, which was then analyzed against the official documentation provided by the selected modern day workflow management systems shown on Table 4.7. This analysis helped to refine the classification framework towards answering the relevant questions for modern workflow management systems. Especially the cloud-native aspects were included more strongly through the documentation analysis.

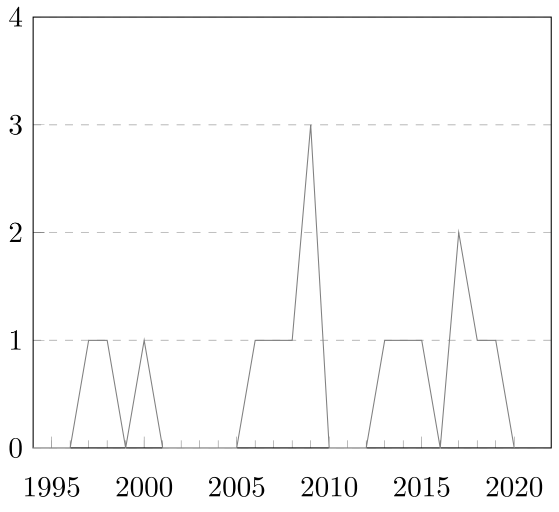

Figure 5.2. *Distribution of selected studies over year published*

As a result of conducting the mapping study and the workflow management system documentation analysis we identified two different levels of concerns and used those to define a *project view* and a *technical view* on the classification of workflow management systems. The views can be used to split the work required in making the choice between project management and technical experts.

The *project view* comprises of high-level requirements that can be used to narrow down the search and make a decision on whether a system applies for a more thorough analysis. This can mean, for example, checking if the license or the platform requirements

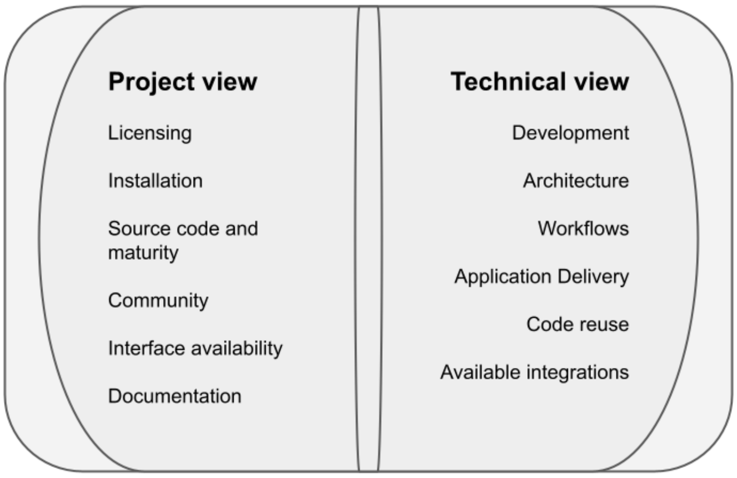

Figure 5.3. *The classification framework: categories for project view and technical view*

of the system are suitable for the project under development. These requirements also comprise a set that can be checked quite fast compared to the more technical properties.

On the other hand, the *technical view* focuses on the details that can be used to determine which of the systems actually provides the best solution for the technical problem being solved. For example, the availability of access management features or workflow definition languages might prove to be the differentiating factor between two otherwise equally classified workflow management systems.

Using these two views we expect to be able to distinguish the workflow management systems from each other and provide a good set of features to be checked before making a choice. The requirements and categories used to define the *project view* are explained in the next subsection followed by the *technical view* described in the subsection after it. Figure 5.3 summarizes the contents of both of the views.

5.1 A project view on workflow management systems

The project view on the workflow management system classification framework has categories and requirements that are of interest to project managers or architects identifying which system would comply with the generic project requirements. The project view consists of the following categories:

- Licensing,
- Installation,
- Source code and release maturity,
- Community,
- Interface availability and

- Documentation.

These categories are explained in detail in the following sections.

5.1.1 Licensing

The licensing category of the classification framework gives us insight on what licenses are used in the products and helps us categorize them. All the products chosen in this study for technology review have their source available but there can be limitations on how, and what parts of, the code can be used.

5.1.2 Installation

The installation category defines the platforms in which the workflow management system can be installed and also shows the dependencies needed to be installed with it. We also look up the availability of an official docker image for the system. This category can be a strict limiting factor if the project being developed has strict platform requirements or knowledge.

5.1.3 Source code and release maturity

The source code availability and languages used are checked in this category. We also classify the release maturity of the workflow management systems. The statuses are defined by following the states of maturity discussed in a book about continuous delivery by Humble and Farley [42] or by using the definitions applied by CNCF organization [43], if applicable. To give perspective on the life-cycle of the products, we also included the first release date available on GitHub for the products.

5.1.4 Community

The activity of different community platforms is analyzed under this category. We look into different media and collaboration platforms to analyse the impact of the product across the field. The popularity was already assessed when including or excluding workflow management systems for the technology review but this aspect should also be considered in the classification of the systems.

5.1.5 Interface availability

In the interface category we assess the availability of different interfaces to access or control the workflow management systems. Different types of interfaces are divided into command-line interface (CLI), application programming interface (API) and graphical user interface (GUI). This category gives a view on the structure and principles on how the products are designed and how they are supposed to be used. It also provides valuable information on the capabilities of the systems.

5.1.6 Documentation

The documentation category analyses the availability of different types of documentation for the systems. These types of documentation being evaluated are workflow development, production deployment of the system, architecture and further platform development. There categories are classified with the following values: having a documentation, documentation lacking, no documentation.

5.2 A technical view on workflow management systems

The technical view on the workflow management system classification framework has categories and requirements that are of interest to software development and operations specialists identifying which platform would comply with low level, technical requirements. The technical view consists of following categories:

- Development,
- Architecture details,
- Workflow features and control mechanisms,
- Application Delivery and
- Code reuse and external integrations.

These categories are explained in detail in the following sections.

5.2.1 Development

The development category consists of details on how the workflow management systems implement their workflows and workflow tasks. We also look into available integrated development environment (IDE) extensions and plugins available to find out which of the systems provide that kind of support for the development work.

In the workflow definition part of the category we look into the way how the high-level workflows are being created and defined in the system. This can vary from pure code solution to strict definition languages. We also analyse the implementation details for the lower level of abstraction inside workflows called workflow tasks.

5.2.2 Architecture details

In the architecture details category we look into architectural principles and details for the workflow management systems. This consists of support for asynchronous messaging, high availability, scalability, state persistency.

5.2.3 Workflow features and control mechanisms

The workflow features and control mechanisms category consists of the capabilities of the workflows themselves in the system. The basic functionality in the workflow management systems to be checked is the availability of features like wait for signal and branching of tasks. A more complex feature considered is forming a dynamically advancing graphs of execution. This means that the execution flow can be determined runtime and will produce a different outcome on different circumstances.

On top of that the capability to schedule workflows, limit the rate of workflow tasks and enforce global semaphores on shared resources are taken into consideration. Also, access management support on the workflow management system is determined in this category.

The contents of this category contain a limited set of functionality as the amount of different workflow specific features is very large. If the requirements of a software project are relying on certain workflow features, those need to be checked separately. There are good scientific studies about feature compliance especially for workflow management systems designed to follow some specific workflow management standard.

5.2.4 Application delivery

The application delivery category contains features like deployment automation support and utility tool availability to be used in continuous integration and delivery (CI/CD) processes. The data collected in this section will give a view of how much work is done and still needed when taking the product into use by a CI/CD pipeline. The capability to support deployment and integration procedures is an important factor when evaluating the maintenance costs of using and developing on top of the product.

5.2.5 Code reuse and external integrations

The code reuse and external integrations category is used to evaluate the amount of work done beforehand to support different use cases for the system. In this category we look into the availability of code samples and example workflows. We also look into external service integrations provided for the system to create a view on the extensibility it has. It is a clear advantage if the workflow management system already supports some tools that are otherwise used in the project.

6 A WORKFLOW MANAGEMENT SYSTEM TECHNOLOGY REVIEW

In this section, the workflow management systems that passed our popularity thresholds, listed in Table 4.7, are evaluated against our classification framework. The first part of this section focuses on the *project view* and the second part on the *technical view* of the framework.

6.1 A project view on workflow management systems

The categories introduced in Section 5 will provide a foundation for comparing and classifying the workflow management systems from the *project view* level. This level of classification will provide means for an early analysis of inclusion from high-level requirements point of view. This helps in the workflow management system choice process as certain tools can be immediately ruled out without starting a more thorough *technical view* analysis.

6.1.1 Licensing

Detailed information about the licences used in our systems under review are listed in Table 6.1. As can be seen from the table, majority of the workflow management systems are under permissive Apache 2.0 or MIT licenses. However, two of the systems use a more limiting license that, in both cases, limits the use of the software in certain business cases.

Zeebe Community License allows free usage as long as the customers cannot freely create and define the services that are orchestrated by the workflows they have designed. This means that the service provider can create back-end services, which the customer can then use to define custom workflows. However, the use is not allowed if the service provider offers the capability to create the back-end services and the workflows. This can be considered as selling Zeebe-as-a-service.

N8n uses Apache 2.0 with Commons Clause, which limits the usage if the software that utilizes N8n produces revenue to the company. This revenue is limited to 30 000 USD per annum and after that a separate license needs to be agreed on. This makes N8n fully free-to-use only for smaller software products or internal systems. The authors call the

Table 6.1. *Project view: Licensing*

	License
Apache Airflow	Apache 2.0
Camunda BPM	Apache 2.0
Conductor	Apache 2.0
Activiti	Apache 2.0
Cadence	MIT
Brigade	Apache 2.0
Zeebe	Zeebe Community License
Argo Workflows	Apache 2.0
N8n	Apache 2.0 with Commons Clause (fair-code)
Kogito	Apache 2.0

license "fair-code". [44]

6.1.2 Installation

Table 6.2 shows what are the platforms and dependencies that are used for production deployment of the systems under study. We also collected information about the availability of official Docker images for the systems. As can be seen all of the systems run in containers as we searched specifically for cloud-native solutions. There are, however, notable differences in the limitations on what platforms and with which other dependencies the systems are supposed to be run with.

Most notably, Brigade, Argo Workflows and Kogito are developed to be used on a certain platform only, this is a clear limiting factor on the choice as it enforces the deployment platform for the whole infrastructure that is being run with the workflow management system. Also, a link between the licensing and installation platform can be seen on Zeebe and N8n, as they have an as-a-service installation and specifically restricted the license, so that it cannot be provided by any other service provider than the developers of the platform.

The dependencies introduced also vary between the workflow management systems as some of them enforce the use of specifically named persistent storages and others offer support for multiple solutions. For example, Conductor relies on another Netflix product Dynomite to provide a high availability production setup, while a solution like Activiti supports multiple different database solutions. Also, Kubernetes specific installations rely on the internals of the platform and therefore need less external dependencies.

It should also be taken into consideration that each of the Docker deployable tools are

Table 6.2. *Project view: Installation platforms and dependencies*

	Platforms	Dependencies	Official Docker Image
Apache Airflow	Kubernetes, Docker	MySQL or PostgreSQL database	Exists
Camunda BPM	Docker, Kubernetes, as-a-service	PostgreSQL, Oracle, MySQL, MariaDB, IBM DB2, Microsoft SQL or H2 database	Exists
Conductor	Docker	Dynomite, Elasticsearch	Only build instructions
Activiti	Docker, Kubernetes, as-a-service	H2, MySQL, Oracle, PostgreSQL, DB2 or MSSQL database and Elasticsearch	Exists
Cadence	Docker	Apache Cassandra, MySQL[1]database or Elasticsearch	Exists
Brigade	Kubernetes	-	Exists
Zeebe	Kubernetes, Docker, as-a-service[2]	Elasticsearch	Exists
Argo Workflows	Kubernetes	-	Exists
N8n	Docker, as-a-service[2]	Postgres, MySQL, MariaDB or MongoDB[1] database	Exists
Kogito	OpenShift	Infinispan	Exists

[1] not recommended for production
[2] hosted as-a-service installation in early access or beta stage

usable on most of the cloud environments available. It just might require extra work from the team taking it into use if no instructions are provided by the platform itself.

6.1.3 Source code and release maturity

The source code information and release maturity of the systems under review are shown on Table 6.3. As all of the considered workflow management systems are open source their codebase is relatively similar. Each of the systems has its source code available in GitHub and the code is fully available.

The main difference on the source code level is the programming language used to provide the functionality. It can be seen that Java and Golang are the languages used for

Table 6.3. *Project view: Source code and release maturity*

	Programming language	Maturity	First release
Apache Airflow	Python	Production	Jan 2015
Camunda BPM	Java	Production	Mar 2013
Conductor	Java	Production	Dec 2016
Activiti	Java	Production	Jun 2010
Cadence	Golang	Production	Jun 2017
Brigade	Golang	CNCF Sandbox	Oct 2017
Zeebe	Java	Ready to market	Jul 2017
Argo Workflows	Golang	CNCF Incubating	Nov 2017
N8n	TypeScript	Ready to market	Jun 2019
Kogito	Java	Development preview	Jun 2019

majority of the products. Golang has been growing as a language of choice on many cloud-native products as major platforms like Kubernetes and Docker have proven to be powerful tools made with the language. [45] It is also natural choice for some of the systems under our review as they are developed by extending existing Kubernetes functionality.

The maturity of the workflow management systems is clearly varying a lot as some of the products have seen over five years of releases while others have not reached their second year yet. The ones with production label have already seen vast use under big loads whereas some of the platforms are still looking for major production usage.

6.1.4 Community

For analysis of community activity, we already did a comprehensive study on the workflow management system popularity assessment part of the study, where we decided what systems to choose for the documentation review. From those it can be said that Apache Airflow, Camunda BPM, Conductor, Activiti, Argo Workflows and Cadence hold the highest score among the systems under study whereas other products have considerably less activity around them.

The results of the popularity assessment can be seen from Tables 4.7 and 4.6.

6.1.5 Interface availability

Table 6.4 shows the available interfaces for the workflow management systems. Most of the studied systems have some kind of a command-line interface that can be used to control the system by e.g. creating or starting new workflows. The API can be used for dynamically querying data or performing operations. Graphical user interface, on the other hand, is in the workflow management system context usually, by minimum, used to view the execution graphs of the workflows. However, there are four outliers in the data and those are Conductor, Cadence BPM, Zeebe and N8n.

Table 6.4. *Project view: Interface availability*

	CLI	API	GUI
Apache Airflow	Yes	Yes	Yes
Camunda BPM	**No**	Yes	**Yes** [1]
Conductor	**No**	Yes	Yes
Activiti	**No**	Yes	Yes
Cadence	Yes	Yes	Yes
Brigade	Yes	Yes	Yes
Zeebe	Yes	Yes	**Yes** [2]
Argo Workflows	Yes	Yes	Yes
N8n	Yes	**No**	Yes
Kogito	Yes	Yes	Yes

[1] limited features for free
[2] free only for non-production usage

Conductor, Camunda BPM and Activiti are not having a command-line interface available. For Camunda BPM and Activiti the main way to run the workflow and BPM methods is through their Java or REST API. They both also always require an XML to be generated for a workflow before it can be run.

Conductor heavily relies on using its REST API to handle all the interactions with the system. These, for example, vary from creating workflow definitions to starting and following workflows. This emphasises their ability to create new workflows on the flow as any user of the workflow management system could be a client for the system.

Zeebe and Camunda BPM, on the other hand, provide all the interfaces we looked into, but they had limitations on the GUI usage based on their revenue model. For Zeebe, it is stated in the documentation that it is allowed to use the GUI for free as long as you use it in non-production environments only. This is expected as their license is not entirely permissive and they are also providing a hosted version of the system in the near future. Camunda BPM is slightly different in this sense as it provides a more extensive feature-set on the enterprise license of the software.

For N8n, the difference compared to other workflow management systems is the lack of a documented API towards the system. The whole concept relies heavily on defining the workflows on the GUI with multiple predefined integrations to various different external systems. While a possibility for extension by code is provided, it can be fully used through the graphical interface.

6.1.6 Documentation

The categories studied under documentation for workflow management systems are shown in Table 6.5. Workflow development documentation is not included in the table as it was sufficient for all of the systems. It can be seen that most of the documentation categories are available, however the quality varies a lot. Main issues within the documentation are with production deployment which for the limited services only contains a quickstart command-line example or a vague list of configurations.

Table 6.5. *Project view: Documentation*

	Production deploy-ment	Architecture	Platform develop-ment
Apache Airflow	Yes	Yes	Yes
Camunda BPM	Yes	Yes	Yes
Conductor	**Limited**	Yes	**Limited**
Activiti	Yes	Yes	Yes
Cadence	**Limited**	Yes	Yes
Brigade	Yes	Yes	Yes
Zeebe	Yes	Yes	Yes
Argo Workflows	Yes	Yes	Yes
N8n	**Limited**	**No**	Yes
Kogito	Yes	Yes	Yes

For N8n, the architecture has not been documented sufficiently as the system acts as a black box for the user and can be used without knowing much of the underlying architecture.

Participating in the open source platform development was fairly well documented, however, Conductor as being initially only an internal system, does not provide more than a small guide on creating a patch for a new feature.

6.2 A technical view on workflow management systems

The categories introduced in Section 5 will provide a foundation for comparing and classifying the workflow management systems from a *technical view* level. This level of classification will provide means for a more thorough analysis of the workflow management system capabilities.

6.2.1 Development

The results of the Development category of the *technical view* are shown in Table 6.6. The way how workflow definitions are implemented varies from strict modeling languages to pure code implementation. Task implementation, however, is more focused on programming languages because the business logic usually requires some coding. For one workflow management system, N8n, it was noticed that graphical user interface was sufficient for creating fully functional workflows.

Table 6.6. *Technical view: Development*

	Workflow Definition	Task Implementation
Apache Airflow	Python	Python
Camunda BPM	BPMN 2.0, CMMN 1.1, DMN 1.1	Java
Conductor	JSON DSL	Java, Python, *Golang*
Activiti	BPMN 2.0	Java
Cadence	Java, Golang, *Python, C#*	Same as for workflows
Brigade	Javascript	Container
Zeebe	BPMN 2.0, JSON	Java, Golang, *more*[1]
Argo Workflows	Kubernetes CRD	Container
N8n	JSON	GUI, TypeScript
Kogito	BPMN 2.0, DMN 1.2, more[2]	Java

languages marked in *cursive* are unofficial
[1] C#, Delphi, Ruby, Node.js, Rust, Python, TypeScript, GitHub Action
[2] DRL, XLS/XLSX decision tables

As we looked into the IDE extensions and plugins available, we found matches for five of the workflow management systems. For Brigade, the extension for VS Code provides project views and workflow running functionality. It is still, however, under development and not published in the VS Code Marketplace. For Argo, there was a suggestion to use a Kubernetes custom resource definition validator in the IDE. The Workflow management systems with some bigger impact on the development work were Camunda BPM, Activiti and Kogito, which offered the use of different modeling language extensions for IDEs. All of the previously mentioned workflow management systems had an extension for Eclipse, whereas Camunda BPM had also support for IntelliJ IDEA and NetBeans editors. Kogito also offered support for VS Code.

According to the data in Table 6.6, workflow definitions can be divided into three categories: workflow as standardized modeling language, workflow as configuration and workflow as code.

It is a certain advantage to BPMN 2.0 and other standardized definitions that they offer a well tested and functional set of features that have validators, editors and know-how already in the field of technology. This also allows people who are not developers a way to create workflows and define decision models. It, however, might prove to be inflexible and cumbersome in situations where the requirements are simple but end up requiring a lot of extra configuration. If the organization has existing workflow definitions and knowledge in the standardized models and wants to widen the spectrum of workflow creators, there is a certain advantage in the usage of Camunda BPM, Activiti, Zeebe and Kogito as a workflow management system. It should also be noted that Kogito and Camunda BPM offer a wider set of standardized definitions to be used in workflow creation.

Four of the workflow management systems offered workflow definitions as configuration: Conductor, Zeebe, Argo workflows and N8n. The most used format was JSON, which can be useful as it is also a very common content-type format for REST interfaces. This is used heavily in e.g. Conductor, which offers an endpoint to start workflows and all the required information can be given in the request itself. For Zeebe, it was mentioned in the previous paragraph that it uses BPMN 2.0, but also JSON definitions are supported. They are internally, however, transformed into the aforementioned notation.

N8n uses a slightly different kind of workflow definition schema as it utilizes Kubernetes Custom Resource Definition (CRD) to define the workflows. This is a natural choice for a tool that runs natively in Kubernetes environment as it can therefore be modified and controlled by the same toolset as any other resource handled in the Kubernetes cloud. The CRD definition is in yaml format which is a superset of JSON.

Compared to the more standard way of defining the workflow, the configuration approach also offers a possibility to include people without programming experience into the definition work but trades off simplicity at the cost of standardized notation.

For the workflow as code, we can see three systems that utilize it. This allows flexibility of the programming language but makes workflow definition only possible for people with programming experience. The way it is done, however, varies a lot between these systems.

Apache Airflow uses Python for everything and the workflow definition is in a form of a Directed Asyclic Graph (DAG), which needs to be defined from end-to-end before the run. These graphs are then scheduled to be run in a defined manner. The Brigade system works a bit like Apache Airflow, but the language used is Javascript and the system is based on catching different events and acting on top of those. Cadence, however, does not require a preset workflow definition as such but it relies on *durable functions* which are a concept that preserve the state of execution on function level.

6.2.2 Architecture details

The architectural details we chose to look up in this context are shown on Table 6.7. Each of the workflow management systems somehow implemented message queues so that is not visible on the table. It can be seen that most of the systems implement all of our points of interest.

Table 6.7. *Technical view: Architectural details*

	High availability	Scalability	Persistent state
Apache Airflow	**No**	Yes	Yes
Camunda BPM	Yes	**Limited**	Yes
Conductor	Yes	Yes	Yes
Activiti	Yes	**Limited**	Yes
Cadence	Yes	Yes	Yes
Brigade	Yes	Yes	**Limited**
Zeebe	Yes	Yes	Yes
Argo Workflows	Yes	Yes	Yes
N8n	**No**	**No**	**No**
Kogito	Yes	Yes	Yes

For Apache Airflow there is a single point of failure in the design as the implementation focuses on a single scheduler unit that runs the workflows. If that goes down, a downtime is ensured. There are unofficial community workarounds for the issue but it has not been fixed for the product itself. Therefore, Apache Airflow cannot be considered as a highly available service.

The Camunda BPM and Activiti both have the database as a limiting factor in scalability as they rely in a relational database that only scales up, not out. This, however, seems to be enough for most of the use cases.

The issue with Brigade is that it is designed to be more like a scripting tool for e.g. CI pipelines, deployments, packaging tasks et cetera. It is discouraged to implement long-running tasks with it. The state persistency is basically limited per task or per workflow of the system.

For N8n, the infrastructure is still lacking many of the features that other systems have. It does have a storage but that is mainly used to store the workflow definitions and finished executions. It also does not offer scalability or high availability in the same way as other systems. This limits the use to smaller projects and internal usage as it cannot be relied on in more business critical projects. It however works very flexibly on the tasks it performs and is constantly under development so it is worth following to see how the solution evolves.

A special notice should be given to Conductor, Zeebe, Cadence, Argo Workflows and Kogito which all met our architectural notions by having a design that has been delivered from the beginning with cloud capabilities in mind. The usage of scalable and fault-tolerant components is a key requirement for most of the modern day software projects and a very favourable way to design a system. For example Zeebe uses event sourcing to solve most of the problems that would be traditionally handled with a database, also allowing a much higher throughput via the decision.

6.2.3 Workflow features and control mechanisms

The distinguishing workflow features and control mechanisms we were interested in about the workflow management systems are visible in Table 6.8. Each of the systems supported basic functionality like having wait signals or branches of tasks so those are not listed in the table.

Table 6.8. *Technical view: Workflow features*

	Dynamically advancing graph	Scheduling	Access management	Global Semaphores	Rate limiting
Apache Airflow	**Limited**	**Yes**	**Yes**	**Limited**	**Yes**
Camunda BPM	**Limited**	**Yes**	**Yes**	No	No
Conductor	**Yes**	No	No	**Yes**	**Yes**
Activiti	**Limited**	**Yes**	**Yes**	No	No
Cadence	**Yes**	**Yes**	No	No	**Yes**
Brigade	**Limited**	**Yes**	**Yes**	No	No
Zeebe	**Limited**	**Yes**	**Yes**	No	No
Argo Workflows	**Yes**	**Yes**	**Yes**	No	No
N8n	No	**Yes**	**Yes**	No	No
Kogito	No	**Yes**	**Yes**	No	No

One feature that we thought would bring valuable information about the flexibility of a workflow management system is the capability to form a *dynamically advancing graph* of executions. This means that the execution flow can be determined runtime and will produce a different outcome on different circumstances. Conductor, Cadence and Argo Workflows implemented this by introducing dynamic forks, child workflows and various programmable features.

The limited capability on Apache Airflow means that the system does not support this kind of a feature very well. This kind of behavior can be achieved if the following task is predefined to loop through the dynamic list of items. This, however, doesn't allow control over failures inside the task so if one of them fails, all fails. It is also possible to provide this kind of functionality by creating multiple workflows but the functionality lacks proper implementation for this. Brigade also has a similar limitation to achieve this but it requires a new workflow to be defined to listen to the results of the previous one.

Camunda BPM, Activiti and Zeebe all have the same kind of problems as Apache Airflow and Brigade. They are also modeled in BPMN and the deployment of a new workflow requires a new specification file created and to be exported into the system. The capability to do so varies between the systems.

For scheduling there was one workflow management system, Conductor, that did not provide any scheduling capability out of the box. It, however, encouraged the usage of an external scheduler that would trigger workflows from it via REST or messaging endpoint. Access management had a similar kind of situation as scheduling, both Conductor and Cadence trusted something else in the cluster to handle that.

Global semaphore and rate limiting were considered features that are critical for certain types of environments. For Apache Airflow it was possible to add a semaphore for a certain workflow and amount of parallel tasks in it, but it was not possible to secure a certain task from being run in multiple workflows simultaneously. Conductor has a definable maximum amount of parallel tasks supported. Rate limiting was implemented for Cadence on top of the two workflow management systems mentioned earlier.

6.2.4 Application delivery

Application delivery details of the classification framework are in Table 6.9. It can be seen that the deployment automation features provided by the workflow management systems are limited for the platform agnostic tools as they can be deployed in any cloud environment. This leaves the deployment procedure automation to the deployer of the application.

For Kubernetes or Openshift platforms, the deployment procedures are built into the platform tools and defined for the service beforehand. This makes the deployment of those tools easier than for the ones that do not have that capability built in. Apache Airflow, had deployment procedures unofficially available for multiple different platforms.

The continuous integration and deployment utility tools were found for three of the applications. Brigade offers a testing tool that simulates the workflows being run without having a need for other dependencies. This is useful for creating functional tests that checks the workflow execution logic.

For Zeebe there is a configurable test container available to be used in e.g. integration tests. Automatically running that kind of a test setup is important, especially for larger

Table 6.9. *Technical view: Application delivery*

	Deployment automation	CI / CD utility tools
Apache Airflow	No [1]	No
Camunda BPM	Helm	No
Conductor	No	No
Activiti	Helm	No
Cadence	No	No
Brigade	Helm	brigtest
Zeebe	Helm	zeebe-test-container
Argo Workflows	Kubernetes	Argo CI, CD and Rollouts
N8n	No	No
Kogito	Openshift	No

[1] Unofficial deployment solutions available

projects. This kind of utility is therefore a valuable asset that removes the need to create and maintain a working test version of the workflow management system in each project.

For Argo Workflows, there are CI, CD and Rollout utility tools available in the product family. They utilize Kubernetes to provide deployment strategies and automation for software projects. If a team uses Argo Workflows and deploys it with the tooling provided, it is a fair consideration to start using the same tooling for other parts of the project too.

6.2.5 Code reuse and external integrations

Each of the workflow management systems had ready-made code samples and example workflows. Conductor and Brigade, however, had only documentation tutorials available to provide examples on how to use the features. The integration possibilities as part of code reuse classification are shown on Table 6.10.

First of all, N8n should be mentioned as an exemplary product in this field as they utilize the community to submit their workflows and integrations to be used by everyone. This allows having hundreds of different workflows and utilities available, reducing the need for code and, in many cases, allowing workflow creation via graphical interface only. This makes the product a very viable choice for a project where multiple different public platforms and interfaces need to be used together ranging from cloud provider solutions to social media services.

For other workflow management systems, the set of integrations is more limited. Most of the systems support the use of services provided by public cloud providers like Amazon, Google or Azure. Those are mostly covering the use of the storage functionality of the providers.

Table 6.10. *Technical view: Code reuse by integrations*

	Ready-made integrations
Apache Airflow	Amazon Web Services, Google Cloud Platform, Microsoft Azure (limited), Databricks, Qubole
Camunda BPM	Cawemo, Amazon aurora PostgreSQL, Microsoft SQL server, IBM DB2, CockroachDB, Spring
Conductor	Amazon S3, Azure Blob storage
Activiti	Spring, CDI, ActiveMQ
Cadence	Kafka
Brigade	Azure (Container Registry, Event Grid, DevOps, VSTS), GitHub, GitLab, DockerHub, BitBucket, Kubernetes events, Trello
Zeebe	GraphQL, Spring, Kafka, GitHub Action, Node-RED, Hazelcast, Event Store DB
Argo Workflows	Minio, Amazon S3, Google Cloud Storage, Alibaba Cloud OSS, Kafka
N8n	Over 90 integrations to different platforms
Kogito	Quarkus, Spring Boot, GraalVM, Knative, Kafka, Infinispan, Keycloak

7 WORKFLOW MANAGEMENT SYSTEM SELECTION GUIDELINES

The third contribution of this work (RQ3) is to provide a way to support the decision making of selecting a workflow management system. In this section, we provide guidelines and recommendations to compare any workflow management systems classified by our workflow management system classification framework with each other. The goal is to make the evaluation of different systems easier and to provide a view that gives a solid baseline for evaluation.

After conducting the workflow management system technology review in Section 6 we went through the results and combined them into groups and categories to follow. These categories follow the results already found through the work done for the classification framework in Section 5, however providing a more generic approach in this section.

By the help of the categories discussed in this section we can determine different scenarios and offer a baseline for finding the best fitting workflow management systems for specific projects. Making a choice, however, always needs thorough consideration and project-specific prioritization of the requirements to be valid.

To provide a clear process for the workflow management system selection we separated the decision making process into following steps:

1. Define requirements and motivations
2. Identify special requirements and prioritize
3. Choose WfMS to be evaluated
4. Perform a technology review
    a) The project view, priorities and early elimination
    b) The technical view and details
    c) Re-evaluate the initial requirements
5. Decision making and validation of the results

A visualisation of the steps is available as Figure 7.1. The following sections are used to describe the steps in detail.

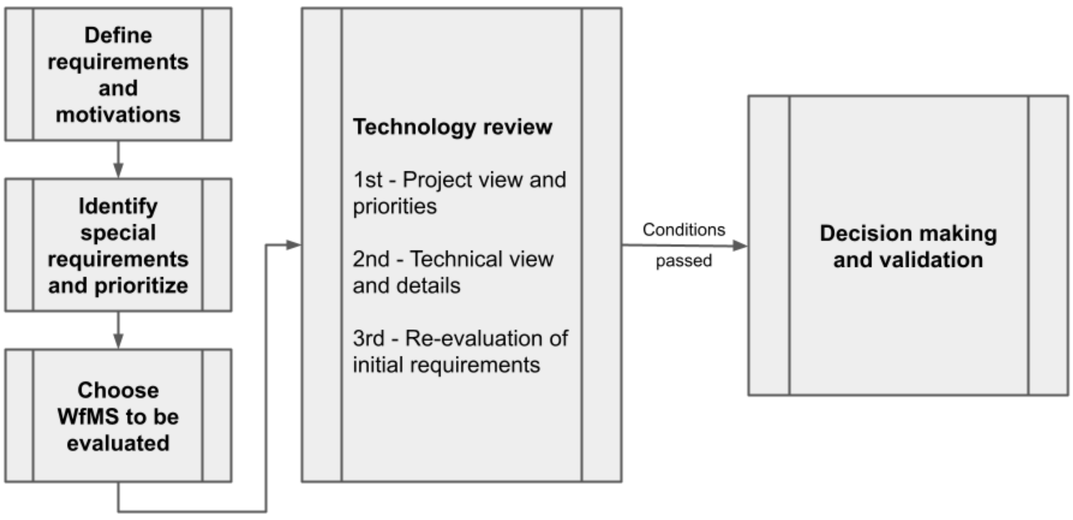
Define requirements and motivations
Identify special requirements and prioritize
Choose WfMS to be evaluated
Technology review
1st - Project view and priorities
2nd - Technical view and details
3rd - Re-evaluation of initial requirements
Conditions passed
Decision making and validation

Figure 7.1. *The WfMS Selection Process*

7.1 Define requirements and motivations

Before collecting the data or identifying the workflow management systems to be considered, we emphasize the need to do a thorough analysis on the project requirements, motivations and the environment where the workflow management system is considered to be introduced. Without understanding these base requirements the analysis may prove to result in a wrong decision, which is only perceived after using a significant amount of time in the process.

One thing to consider in the beginning of the process is that moving to use a workflow management system may impose more resource and development overhead on the system under development than the requirements actually would when developing the functionality yourself. The motivation needs to be clear and the achievable benefits large enough to consider introducing a new component to the system.

Each project also introduces their own set of requirements towards the workflow management system. It is important to identify those and find a set of requirements that satisfy the needs of the project but leave room for a comprehensive list of different workflow management systems.

7.2 Identify special requirements and prioritize

The requirements of a software project vary and prioritization is the key to find the right solutions. In Table 7.1 we introduce the generic categorization and classification framework introduced in this study to be used as a baseline for the decision makers to determine their requirements and needs. A priority does not need to be clear for each of the requirements before conducting a more thorough technology review, and a change in the prioritization of requirements can also be done during the process if needed.

On top of the requirements or classification introduced by this study, we encourage to take a look into the special needs of the project and also cover them in the evaluation if not already covered. The importance of different factors is always project specific.

7.3 Choose workflow management systems to be evaluated

This study provides a good baseline for choosing suitable open source and cloud-native workflow management systems to be used in the evaluation. However, new systems emerge all the time and a periodic check on the projects is encouraged. Also, if the conditions of inclusion end exclusion for your project are different than in this study, the set of tools might also be significantly different.

To make a choice of the systems to be evaluated, a popularity study is a good option for filtering out the relevant options. A similar process as described in Section 4.7 can be repeated to get an up-to-date view of the workflow management systems available.

7.4 Perform a technology review

After making a choice of what workflow management systems to include in the study, a technology review, similar to what was done in Section 6, needs to be done. The results of this study can be used as a benchmark and as part of the study itself if they match the set of requirements set for your project.

7.4.1 The project view, priorities and early elimination

A recommended approach is to first go through the project view and the highest priority items for your study and make an early elimination of systems not matching the requirements set at this state. This way more time is left for analysis of details and technical aspects of the workflow management system. This step can be conducted by a person without deeply technical background and should be fairly easy to do.

7.4.2 The technical view and details

As the technical view consists of more detailed requirements for the systems and might contain information that cannot be found as easily as the aspects of project view, it is done after the first step. This step might prove to be hard to conduct for certain systems if the documentation is lacking or expertise on the topics is not good enough. This step is recommended to be done by a person with technical knowledge about the domain and the technical requirements of the project.

7.4.3 Re-evaluate the initial requirements

Because the evaluation process is done around multiple different systems with different kinds of features the person doing the analysis also gets a lot of information that was not available in the beginning of the process. Requirements and priorities set in the beginning might need adjusting and new aspects found during the evaluation might bring in more requirements for the study. When realizing there are gaps in the evaluation, the new requirements must also be taken into account in the decision making process.

7.5 Decision making and validation of the results

After conducting the technology review and adjusting the requirements to get a complete set of results, a decision needs to be made. For some projects the choice might be clear, but usually there are multiple options that can be valid for the project. In this case, a further comparison of the small set of workflow management systems might provide needed information to make a choice.

However, before committing and making the final choice, we recommend a proof of concept deployment and workflow execution to be done for the system and possible alternatives. This is done to ensure a real experience-based evaluation on the ease of use and effort needed for the actual development and workflow definition work needed when using the system.

Table 7.1. *The final workflow management system classification framework*

	Category	Details
Project view	Licensing	Used license
		Limitations
	Installation	Platform
		Dependencies
		Docker image availability
	Source code and maturity	Programming language
		Stage of maturity
		Project age
	Community	Community page availability
		Github stars, contributors, releases
		Stack Overflow questions
		Google hits
	Interface availability	CLI
		API
		GUI
	Documentation	Workflow development
		Production deployment
		Architecture
		Platform development
Technical view	Development	Workflow definition
		Task implementation
		IDE extensions
	Architecture	High availability
		Scalability
		State persitency
		Asynchronous messaging
	Workflows	Feature support
		Control mechanisms
	Application delivery	Deployment automation
		CI/CD utilities
	Code reuse	Workflow examples
		Code samples
	Available integrations	Cloud environments
		External software assets

8 THREATS TO VALIDITY

Four main types of threats to the validity of a quantitative research are presented by Wohlin et al. [46] as follows: external, internal, conclusion and construct validity. In this section we can identify how those apply to our study and also discuss the mitigation actions we took to reduce their effect.

External validity means that we should be able to generalize our results to study items also outside of our current scope. This has been taken into consideration by systematically going through each of the workflow management systems and by refining our classification framework during the whole process. Also, the search process was done in a way that allows us to apply the same procedure to future studies and in a sense refresh the data currently collected about the workflow management systems.

Internal validity in this study can be described as how well our study methods resulted in the outcome without having factors which we did not take into account affecting the results. This was mitigated by having multiple revisions of the classification framework and popularity study. However, many factors taken into use in the popularity study relied on factors not totally under our control, e.g. the search result amounts could also contain hits that are not relevant for our study and about some completely different topic than what we searched for. This was mitigated by trying to define clear keywords for the search and by trying to use multiple different indicators and thresholds for popularity.

Conclusion validity focuses on how sure we can be that the method or treatment actually is related to the actual outcome of the study. In our case this was mitigated by the thorough analysis of the data, the usage of well-known sources and benchmarking other research done on the field before jumping into conclusions. A risk of having a bias during the data extraction phase and iterative documentation analysis is, however, there, as not every step of the iterative process was recorded in the study.

Construction validity means the validity of measurement against the measured target. This means that the study successfully measures what it claims to measure. In our study, the classification framework and selection guidelines are constructed using the mapping study and documentation analysis in a way that provides a view on the workflow engines themselves. The validity of this work relies on the clarity of documentation and the methods used thorough the study.

The validity can also be compromised by having only a single author doing most of the search based work and another author validating the results as it leaves room for bias in the search procedures and results. This, however, was mitigated by the clearly set goals on what to search for and how. The results are also publicly available to be re-checked after the study was initially done.

9 DISCUSSION

The amount of scientific discussion on evaluation and comparison of workflow management systems has been fairly constant during the years as can be seen in the distribution of studies found in Table 5.2. In this section we discuss our position in that timeline and the implications of this work for practitioners and researchers interested in the results. On top of that we consider the overall trends that could be deciphered from the data collected during the research process.

Determining the main classification features for distinguishing different workflow management systems from each other (RQ1) required us to perform a systematic literature review and a documentation analysis on the more recent systems.

During the literature review we analyzed 43 different academic studies, of which 12 were selected as a baseline for our study. By applying the snowballing technique we found 4 more studies. The initial classification framework was therefore constructed by using 16 academic studies. For complementing the classification framework we also did a documentation analysis on the chosen workflow management systems. By doing both of the procedures we tried to align our context towards the work done previously and tie it together with the modern trends in the field of cloud-native workflow management.

The categories of the classification framework reflect multiple aspects of the workflow management system from project and technology views. As was expected, the documentation analysis of the most popular solutions fitting our criteria brought in some details, such as *Docker* and CI / CD availability, that might have been missed with only using the results of the literature review.

For answering the question of choosing the most popular workflow management systems for documentation analysis and determining the main differences between them (RQ2) required us to perform the following steps. First we conducted an initial popularity search that resulted in 20 different workflow management systems. Then from those we chose 10 for further analysis using our popularity criteria defined in Section 4.7.

The results indicate that there are certain older, more established, workflow management systems that clearly passed our thresholds and multiple newer systems which did not reach the same level of popularity yet. It is notable, although expected, that the popularity follows closely the maturity of the products as can be seen when comparing the popularity data against the maturity data collected in the technology review.

After selecting the systems they were used to conduct the documentation analysis and the results were combined with the initial classification framework to form a final version of it. The classification framework is shown in Table 7.1. After forming the classification framework a thorough technology review was done to separate the workflow management systems by their distinguishable features and to provide answers to the latter part of the research question number two. The results and conclusions of the technology review are available in Section 6.

The differences between the workflow management systems were clear on areas such as workflow definition language or the platform used. The differences of the systems were depending mainly on the type of the system and the business goal they were solving. It was surprising that many of the newer systems did not apply any standards in the workflow definition but implemented it on their own instead. There were also a couple of licenses on the products that were unexpected: Zeebe and N8n were applying a non-orthodox license to protect their product against profitable use from software-as-a-service vendors.

One of the main learnings of this study are documented as an answer to the RQ3: "How to select an open source Workflow Management System for a generic cloud-native environment?". Section 7 describes the routine for selecting the best system for a certain use-case. The guidelines documented in that section can be used to go through the whole selection process from the beginning to an end.

The refinement of those guidelines went hand-in-hand with the study process itself as the workflow management system selection and technology review phases provided a good view on the different aspects of the selection process.

9.1 Implication for practitioners and researchers

From the point of view of a researcher this study adds up to the longer trend of evaluations by adding a viewpoint from a cloud-native perspective for the workflow management systems. The results allow other researchers to use our classification framework as a baseline and provide missing aspects in the evaluation for further inspection.

Researchers should pay attention to bringing in more specific evaluation details such as workflow definition details, usability assessment or performance comparison. These topics would bring value to the overall research on the field of workflow management.

The study also shows that new workflow management systems are being released constantly and the feature set of them keeps evolving all the time. Researchers and practitioners should keep up with the new releases and follow the trends forming around them.

For practitioners, the research gives tools to conduct a structured comparison based on their project-specific use case. The categories and guidelines in this study can be used to refine the selection process and provide a more objective look on the available workflow management systems. Also, it is important to remember that the prioritization of the categories is the most important part of making a final choice between the systems.

The selection guidelines given as one of the results of this study also require further usage and it is encouraged for practitioner and researchers alike to use them and provide feedback for possible future refinement.

9.2 Trends

The technology and literature reviews gave a good insight on the trend in which the workflow management systems have been, what the current situation is and what kind of future trends there are in the field.

One of the trends that is clearly visible in the newer workflow management systems is that the workflow definition language and task implementation methods have been moving away from the standards like BPMN and DMN, which have been very prominent in the older systems. As can be seen from Table 6.6, in which the data is recorded, the newer workflow management systems have been moving towards using their own definition formats or purely using only code as their approach to define the workflows.

This impacts the solutions base in a way that the systems are not inherently interoperable when the workflow definitions are not standardized. It also ultimately rules out some of the stakeholders that might be interested in defining their own workflows and definitions.

The use of standardized definition languages, however, has been also debated in the community. For example BPMN has been clearly critizised for having too loose definitions that leave room for implementing the same control structures in a different way. [3] This kind of problems can be seen as one reason for moving away from the standardized definition languages even though the newer versions of BPMN, for example, have been trying to resolve these problems.

Another interesting trend is the that there are multiple very well competing solutions like Conductor by Netflix and Cadence by Uber Technologies that have been mainly developed by a single company for their own use from a scratch and only later released as open source for the communities to use. This implies that the solutions already available have not met the requirements of the new emerging systems. For both of the use cases the requirements are very demanding on cloud-native requirements like scalability and high availabilty.

Both of these trends seem to imply a change in the paradigm of workflow management systems towards a more fragmented field of systems. For the future, it would be beneficial to continue following the discussion and seeing if the trend continues or if new standards take space in the field.

10 CONCLUSIONS AND FUTURE WORK

The goal of this work is to make choosing a generic cloud-native workflow management system an easier task by analyzing their popularity, defining means for classification and providing best practises for the selection process. The work around the topic evolved from the analysis of academic sources by conducting a systematic mapping study into the creation of a workflow management system classification framework, conducting a technology review and forming a set of selection guidelines for the workflow management systems.

During the study we found multiple general purpose and workflow specific categories to be used for classifying the workflow management systems against each other. The categories provided us with an overall view on the concept of workflow management systems and gave us means to compare them. The analysis and details of the classification against chosen workflow management systems are documented in Section 6 (Workflow management system technology review).

For conducting the selection process and a workflow management system technology review for a specific project we gathered a set of selection guidelines in Section 7 (Workflow management system selection guidelines) to make the process easier.

The overall discussion of the research questions and the implications of this study are documented in Section 9 (Discussion). There we highlight the results of this study and provide references to the most important parts in it.

On top of the work done in this study it is easy to add more workflow management systems under review to provide a more thorough view on the current field of technology. The amount of academic publications on the current state of workflow management systems is limited even though it would be beneficial to follow the trends and developments on this topic more closely. Therefore, we encourage to use the framework provided in this study, and to refine or extend it if seen that the existing categories are not relevant enough for specific kind of workflow management systems or use-cases under study.

BIBLIOGRAPHY

[1] Workflow Management Coalition website, Available (accessed 10.7.2020): http://www.wfmc.org/.

[2] D. Georgakopoulos, M. Hornick, A. Sheth, An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure, Distributed and Parallel Databases, Vol. 3, Apr. 1995, pp. 119–153.

[3] E. Börger, Approaches to modeling business processes: A critical analysis of BPMN, workflow patterns and YAWL, Software & Systems Modeling, Vol. 11, July 2012, pp. 1–14.

[4] WFMC, *Workflow Management Coalition Terminology and Glossary (WFMC-TC-1011)*, tech. rep., Workflow Management Coalition, United Kingdom, 1999.

[5] W. V. Aalst, K. V. Hee, Workflow Management: Models, Methods, and Systems, Cooperative information systems, MIT press, 2002.

[6] N. Russell et al., Workflow Control-Flow Patterns: A Revised View, Jan. 2006.

[7] A. Y. Manoj Das, Business Process Management and WS-BPEL 2.0: What's next for SOA Orchestration?, Oct. 2006.

[8] W. Der, W. Aalst, A. Ter, YAWL: Yet another workflow language (revised version), Jan. 2003.

[9] B. Chapman et al., Common Workflow Language, v1.0, ed. by P. Amstutz, M. Crusoe, N. Tijanić, Specification, product of the Common Workflow Language working group. http://www.commonwl.org/v1.0/, figshare, United States, July 2016.

[10] Workflow Description Language website, Available (accessed 10.12.2020): https://openwdl.org/.

[11] R. Shapiro, M. M. et al., Process Definition Interface – XML Process Definition Language: Version 2.00, Oct. 2005.

[12] About the Business Process Model and Notation specification version 2.0.2, Available (accessed 8.2.2021): https://www.omg.org/spec/BPMN/2.0.2.

[13] Workflow Patterns website, Available (accessed 10.7.2020): http://www.workflowpatterns.com/.

[14] D. Hollingsworth, The Workflow Reference Model, Jan. 1995.

[15] Simple Storage Service S3 release page, Available (accessed 15.1.2021): https://aws.amazon.com/releasenotes/release-amazon-s3-on-2006-03-13/.

[16] N. Kratzke, P.-C. Quint, Understanding Cloud-native Applications after 10 Years of Cloud Computing - A Systematic Mapping Study, Journal of Systems and Software, Vol. 126, Jan. 2017, pp. 1–16.

[17] C. Pahl et al., Cloud Container Technologies: A State-of-the-Art Review, IEEE Transactions on Cloud Computing, Vol. PP, May 2017, pp. 1–1.

[18] K. Petersen, S. Vakkalanka, L. Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: An update, Inf. Softw. Technol., Vol. 64, 2015, pp. 1–18.

[19] C. Wohlin, Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering, Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14, Association for Computing Machinery, London, England, United Kingdom, 2014, Available: https://doi.org/10.1145/2601248.2601268.

[20] L. Chen, M. Ali Babar, H. Zhang, Towards an Evidence-Based Understanding of Electronic Data Sources, 14th International Conference on Evaluation and Assessment in Software Engineering, Apr. 2010.

[21] K. Petersen et al., Systematic Mapping Studies in Software Engineering, Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, EASE'08, BCS Learning & Development Ltd., Italy, 2008, pp. 68–77.

[22] Wahnon, *awesome-workflow-engines*, https://github.com/meirwah/awesome-workflow-engines, 2020.

[23] V. Garousi, M. Felderer, M. V. Mäntylä, Guidelines for including grey literature and conducting multivocal literature reviews in software engineering, Information and Software Technology, Vol. 106, 2019, pp. 101–121, Available: http://www.sciencedirect.com/science/article/pii/S0950584918301939.

[24] D. Meyer, *Camunda engine since Activiti fork*, Oct. 2016, Available: URL: https://camunda.com/blog/2016/10/camunda-engine-since-activiti-fork/.

[25] Stack Exchange Inc, Stack Overflow, Available (accessed 10.7.2020): http://www.stackoverflow.com/.

[26] M. Berger et al., Evaluating workflow management systems, Database and Expert Systems Applications. 8th International Conference, DEXA'97. Proceedings, IEEE, 1997, pp. 412–417.

[27] M. Pérez, T. Rojas, Evaluation of Workflow-type software products: a case study, Information and Software Technology, Vol. 42, Iss. 7, 2000, pp. 489–503, Available: http://www.sciencedirect.com/science/article/pii/S0950584900000938.

[28] S. Eswaran et al., Adapting and evaluating commercial workflow engines for e-Science, 2006 Second IEEE International Conference on e-Science and Grid Computing (e-Science'06), IEEE, 2006, pp. 20–20.

[29] K. Baïna, WFESelector-A tool for comparing and selecting workflow engines, International Conference on Enterprise Information Systems, Vol. 2, SCITEPRESS, 2007, pp. 330–337.

[30] E. Bernroider, M. Bernroider, A comparative study of business process management tools based on open source software and a commercial reference, IMETI 2008 - International Multi-Conference on Engineering and Technological Innovation, Proceedings, Vol. 1, Jan. 2008, pp. 225–230.

[31] R. Garcês et al., Open Source Workflow Management Systems: A Concise Survey, in: 2009 BPM & Workflow Handbook, Jan. 2009, pp. 179–190.

[32] H. Gruber, Evaluation of workflow management systems, 2009 IEEE Conference on Commerce and Enterprise Computing, IEEE, 2009, pp. 307–311.

[33] K. Baïna, S. Baïna, User experience-based evaluation of open source workflow systems: The cases of Bonita, Activiti, jBPM, and Intalio, 2013 3rd International Symposium ISKO-Maghreb, IEEE, 2013, pp. 1–8.

[34] A. Delgado et al., A systematic approach for evaluating BPM systems: Case studies on open source and proprietary tools, IFIP International Conference on Open Source Systems, Springer, 2015, pp. 81–90.

[35] V. Ferme et al., Workflow management systems benchmarking: unfulfilled expectations and lessons learned, 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), IEEE, 2017, pp. 379–381.

[36] J. Lenhard et al., Lessons learned from evaluating workflow management systems, International Conference on Service-Oriented Computing, Springer, 2017, pp. 215–227.

[37] R. Singh et al., Evaluating Scientific Workflow Engines for Data and Compute Intensive Discoveries, 2019 IEEE International Conference on Big Data (Big Data), IEEE, 2019, pp. 4553–4560.

[38] A. Tsalgatidou, T. Panepistimiopolis, Selection criteria for tools supporting business process transformation for electronic commerce, Proceedings of EURO-MED NET, Vol. 98, 1998, pp. 244–253.

[39] P. Wohed et al., Patterns-based evaluation of open source BPM systems: The cases of jBPM, OpenWFE, and Enhydra Shark, Information and Software Technology, Vol. 51, Iss. 8, 2009, pp. 1187–1216.

[40] A. Z. Ravasan, S. Rouhani, H. Hamidi, A Practical Framework for Business Process Management Suites Selection Using Fuzzy TOPSIS Approach, International Conference on Enterprise Information Systems, Vol. 2, SCITEPRESS, 2014, pp. 295–302.

[41] A. Delgado, D. Calegari, Evaluating non-functional aspects of business process management systems, 2017 XLIII Latin American Computer Conference (CLEI), IEEE, 2017, pp. 1–10.

[42] J. Humble, D. Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, 1st, Addison-Wesley Professional, 2010.

[43] CNCF, *Cloud Native Computing Foundation projects website*, https://www.cncf.io/projects/, 2020.

[44] J. Oberhauser, K. Malac, *Fair-code website*, https://faircode.io/, 2020.

[45] P. Belagatti, *Why is Go so Damn Popular Among Developers*, https://dev.to/pavanbelagatti/why-is-go-so-damn-popular-among-developers-2d6h, 2020.

[46] C. Wohlin, Experimentation in Software Engineering: An Introduction, International Series in Engineering and Computer Science, Kluwer Academic, 2000, Available: https://books.google.fi/books?id=nG2UShV0wAEC.