UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering,
Mathematics & Computer Science

A software development project ontology

Marc J. F. Jacobs

M.Sc. Thesis
August 2022

Supervisors:
dr. W. Corbo Ugulino
dr. J.L. Rebelo Moreira
dr. ir. E.J.A. Folmer

ABSTRACT

The purpose of this research was to explore the possibility to recommend programming languages and frameworks to be used in a Software Development Project, given a certain set of expected quality attributes of the project and a set of information about the developers' team, specifically while using ontology reasoning. The artifact developed in this research is the ontology itself. This research is performed in collaboration with the Judicial Information Service (Justid). The method used for this research is the Design Science Methodology defined by Wieringa. The development of the ontology is done following the SABiO Development Process. To develop the ontology three tools have been used, (1) Visual Paradigm, (2) OntoUML, and (3) Protégé, to evaluate the ontology three tools were used as well, (1) the Protégé Reasoner, (2) Query testing, and (3) OOPS!. Next to this two interviews were performed, one with the department manager and a software developer of Justid, and one with an objective ontology expert.

This research found that it is possible to recommend programming languages and frameworks using ontology reasoning. The main stakeholder, Justid, found that in general the listed requirements were met sufficiently. In both interviews several improvement points were noted, of which the main improvement noted in both interviews is to extend the level of detail in the developed ontology. This is one of the future works, as well as for example using this ontology to recommend a development team when there is a restriction on the programming language being used.

Keywords — Ontology, Recommender system, SABiO

CONTENTS

Abstract **2**

1 Introduction **8**

1.1 Context: Justid 8

1.2 Relevance and Motivation 8

1.3 Overview: Research Question, Hypothesis & Solution, Evaluation 9

1.4 Methodology 10

1.5 Outline 11

2 Theoretical foundations **13**

2.1 Methods of Recommender Systems 13

2.2 Challenges of Recommender Systems 14

2.3 Evaluation of Recommender Systems 16

2.4 Ontology 16

2.5 Ontology-Based Recommender Systems 17

3 Systematic Review of Literature **18**

3.1 The review protocol 18

3.2 Overview 19

3.3 Discussion 23

4 Solution **25**

4.1 SABiO: Ontology Development 25

4.2 SDP-Ontology development according to SABiO 26

4.2.1 SABiO step 1 26

4.2.2 SABiO step 2 28

4.2.3 SABiO steps 3 & 4 35

4.2.4 SABiO step 5 37

5 Evaluation **43**

5.1 Justid Interview 43

5.1.1 Results 43
5.2 Ontology expert 46
5.3 Discussion 47

6 Conclusion **48**
6.1 The research questions 49
6.2 Limitations 50
6.3 Future Works 50

References **51**

A Justid interview transcript **57**

List of Figures

1.1 Overview of the engineering cycle [1] 10
1.2 Overview of the RQs and their corresponding Sections 12

2.1 Overview of the different types of Recommender Systems 14

3.1 Overview of the paper selection process 19
3.2 The number of papers published per year 20
3.3 The number of papers published per country 20
3.4 A heat map of the published papers per country 21
3.5 The distribution of the different evaluation types 21
3.6 The number of papers for each domain 23

4.1 Overview of SABiO's Processes [2] 26
4.2 Graphical representation of the conceptual model for the sub-ontology Programming Languages and Frameworks 28
4.3 Graphical representation of the conceptual model for the sub-ontology Development Team 29
4.4 Graphical representation of the conceptual model for the sub-ontology Software Project . 33
4.5 Graphical representation of the functional model for the sub-ontology Programming Languages and Frameworks 36
4.6 Graphical representation of the functional model for the sub-ontology Development Team 36
4.7 Graphical representation of the functional model for the sub-ontology Software Project . . 37
4.8 The classes included in the model as shown in Protégé [3] 38
4.9 Overview of the OOPS! results [4] 42

List of Tables

3.1 Search strings for each of the used databases 19

4.1 The definitions of the terms used in the sub-ontology ’Programming Languages and Frameworks’ 29

4.2 The definitions of the terms used in the sub-ontology ’Development Team’ 30

4.3 The definitions of the terms used in the sub-ontology ’Software Project’ 31

LIST OF ACRONYMS

DSS	Decision support system
OB-RS	Ontology-based recommender system
RS	recommender systemrecommender systems
DSM	Design science methodology
SRL	Systematic review of literature
CF	Collaborative Filtering
CB	Content-Based Filtering
KB	Knowledge-based Filtering
MAE	Mean Absolute Error
MSE	Mean Square Error
RMSE	Root Mean Square Error
EC	Exclusion Criteria
CQ	competency question
VP	Visual Paradigm

1 INTRODUCTION

The process of a software development project has a lot of steps. An example, among others, is determining the software requirements and other project requirements. Another step that occurs early in the development process is choosing the programming language and/or framework for the software development project. The decision to use a certain programming language is currently based on human techniques such as expert opinion, if it is based on a technique at all. This way of deciding which programming language to use is prone to errors. Errors that might occur are, among others, bias towards or against a certain language as well as inconsistently picking a language. To improve the process of deciding which programming language/framework should be used, a Decision support system (DSS) can be used.

DSSs have been used for decades to improve decision-making processes [5]. DSSs have been used for supporting the decision-making in for example human relations, (university) budget planning, and marketing strategy. DSSs based on *inductive reasoning*, like those powered by machine learning predictive models, require the use of large datasets for the model training. When a company decides which programming language should be used for a new project, it usually takes into account its particular context, which includes the available staff, their experience in comparison with the complexity of the project, infrastructure for deploying software developed with a certain language, and so on. This contextual nature makes it difficult to produce datasets big enough to allow for statistical learning and the development of predictive models (based on data) for supporting this decision. In this paper, we take this into consideration, which led us to consider a solution for a recommendation system that leverages the benefits of *deductive reasoning*, which does not depend that much on data from previous decisions.

1.1 Context: Justid

This research will be conducted in collaboration with the organization Justid[1]. Justid is short for ”Justitiële Informatiedienst”, which translates to Judicial Information Service. This organization is part of the Ministry of Justice and Safety. Justid is responsible for the provision of information using its IT infrastructure. The information they provide has a high privacy level, an example of the provided information is a person’s criminal record. Justid often needs to develop new software, or additional components to already existing software packages. Therefore, Justid benefits from the proposed recommender system (RS).

1.2 Relevance and Motivation

In this research, we acknowledge the relevance of having a DSS to help Software Project Managers make decisions about which programming language and frameworks to use for a project. To that end, we analyzed the most popular methods for developing RS (Chapter 2), including those powered by machine learning, ontology reasoning, and hybrid ones.

During an open conversation with Justid, the need for such a RS was voiced. During this conversation, several reasons are given by Justid for this need. Firstly, they expected such a system would reduce the time needed when selecting a language. Secondly, there would be less bias towards (or away

[1] `https://www.justid.nl/`

from) certain languages/frameworks. Imagine for example that one person really likes JavaScript, and constantly points towards using this language for all the projects. Using a RS (based on an ontology) this bias would be reduced. Another reason is the fact that humans are creatures of habit, which consequently could lead to using a language because it is always used. Then there is the argument that such a system could help estimate what kind of knowledge is needed for coming projects. This in turn could help educate employees in advance or hire new people with the right qualifications.

In this research, we start looking into the use of Ontology-based recommender systems (OB-RSs) for this problem. Partly because one limitation to using machine learning methods is that, to the best of our knowledge and to the extent of our systematic review of literature, there are no datasets with enough examples of projects, including their quality attributes and the languages and frameworks used. Producing such a dataset would require a lengthy time span and would make this research infeasible. Finally, the common setbacks of using machine learning-based recommendations are: the cold start problem and the lack of diversity in the recommendation. Both issues can usually be solved by an OB-RS.

A question that might arise is why we do not use simple business rules, e.g. if the project is a front-end mobile application then Java should be used to develop it. The answer to this question lies in the lack of serendipity business rules provide. When recommending a language and/or framework using business rules, a new/another language will never be recommended for a similar case unless the rule is changed. However, when this rule is changed the ’old’ language will never be recommended. Add to this that the level of complexity will rise with each new language, project attribute, and team composition, the decision is made to research the usage of OB-RSs.

1.3 Overview: Research Question, Hypothesis & Solution, Evaluation

As previously stated, Justid wants to reduce the time needed for selecting a programming language/framework as well as reduce the bias when choosing a language/framework. Therefore, this research’s main goal is to investigate the **feasibility** of making recommendations based on ontology reasoning (deductive approach). This decision was taken based on the results of our Systematic Review of Literature that showed that ontology-based recommendations have been used to solve/reduce the occurrence of common issues in the development of RSs, like the cold-start problem, the lack of diversity in the recommendation and the need for a dataset (in case of machine learning-based recommendation) (Section 2).

Ontology reasoning requires a well-defined ontology that captures the software development domain to, somehow, make it possible to emulate the reasoning humans do. Therefore, the artifact being developed and investigated in this research is the ontology itself and the main research question being investigated is:

Research Question: How to make recommendations of programming languages and/or frameworks to be used in a Software Development Project, given a certain set of expected quality attributes of the project and set of information about the developers’ team?

Hypothesis: It is possible to obtain recommendations of programming languages or frameworks by using ontology reasoning based on the artifact (the ontology) produced as part of this research. The **solution** investigated in this research is the Ontology itself.

Evaluation: The proposed ontology was first tested with a mix of fake and real data, aiming at verifying its feasibility. The real data we provided is regarding the team of collaborators. Finally, the results of the ontology reasoning were analyzed by the project manager and a software developer of Justid and we interviewed them to capture their opinion on the possible insights (or the lack of). Secondly, an objective ontology expert also evaluated the ontology. The evaluation of this proof of concept ontology reasoning and the findings from the interview with the project managers are described in Chapter 5.

Additionally, following up on the aforementioned Research Question, this research includes sub-questions to support the process of answering the main research question:

RQ1 What are the fundamental concepts with respect to recommender systems?

RQ2 How are ontologies used in a recommender system?

RQ3 Does the ontology needed to execute a recommendation already exist? And if not, are there parts of ontologies that can be re-used when developing the ontology?

RQ4 How should an ontology be developed?

RQ5 How are the programming languages and frameworks represented in the ontology?

RQ6 How is the development team represented in the ontology?

RQ7 How is the development project represented in the ontology?

RQ8 To what extent does the ontology fulfill the needs of the stakeholder?

1.4 Methodology

This Section discusses the procedures taken for the conduction of this research. This research was planned and conducted based on a Design science methodology (DSM). Traditionally, sciences such as physics deal with natural things and are called *Natural Sciences*. Often, questions regarding these sciences are regarding ”how things work.” On the other hand, the science of engineering has to do with *artificial things* and how to make these things with the desired properties [6]. Design Science is an appropriate method for such a scenario. In this research, the DSM is applied according to Wieringa [1].

As Wieringa proposes, the engineering cycle is comprised of five phases, as shown in Figure 1.1. (1) Problem investigation, (2) Treatment design, (3) Treatment validation, (4) Treatment implementation, and (5) Implementation evaluation. We will discuss each of these five phases shortly.

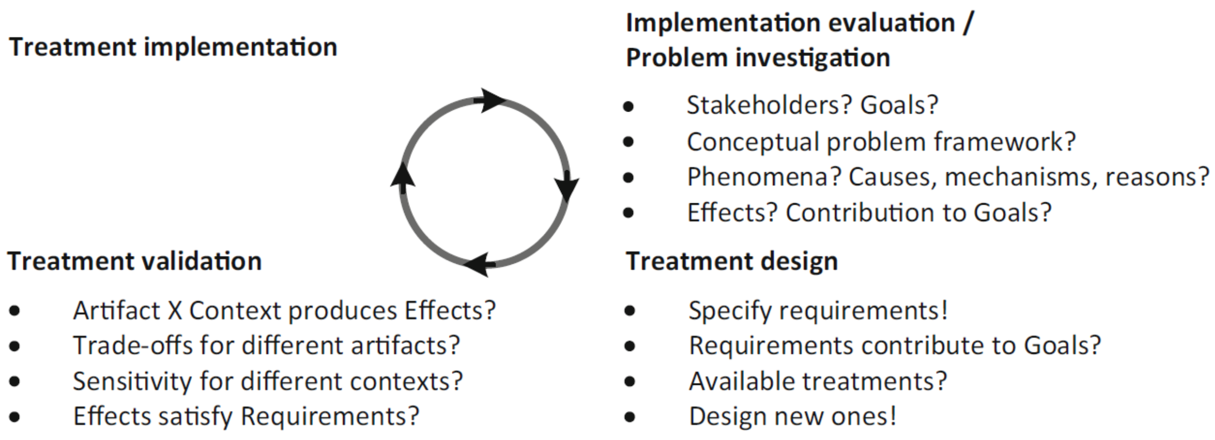

Treatment implementation
Implementation evaluation / Problem investigation
Stakeholders? Goals?
Conceptual problem framework?
Phenomena? Causes, mechanisms, reasons?
Effects? Contribution to Goals?
Treatment validation
Artifact X Context produces Effects?
Trade-offs for different artifacts?
Sensitivity for different contexts?
Effects satisfy Requirements?
Treatment design
Specify requirements!
Requirements contribute to Goals?
Available treatments?
Design new ones!

Figure 1.1: Overview of the engineering cycle [1]

The first phase is concerned with the question ”What phenomena must be improved, and why?”. Additionally, in this phase, the stakeholders and goals are defined as well as the problem context. The next phase is concerned with designing one or more artifacts that could treat the problem. In this phase, the requirements of (a) possible treatment(s) are defined, and said treatment(s) are designed/developed. The third step concerns validation; this phase deals with the question: ”Would these design(s) treat the problem?” We will investigate the developed treatment(s) in a controlled environment in this phase. During this investigation, we will discuss the treatment’s contribution to the stakeholder’s goals from the stakeholder’s point of view (through interviews) and the researcher’s (technical inspection and analysis). In the fourth phase, we will treat the problem with one designed artifact. The treatment will be used in the real world in this phase and thus not in a controlled environment. The last step is the evaluation, where the question ”How successful has the treatment been?” is answered. This is done using the problem context defined in phase one.

The first three phases and the fifth phase of the engineering cycle make up the design cycle. The design cycle is the method used in this paper. This means that we will not use the developed artifact in a real-world, uncontrolled environment yet.

1.5 Outline

The present work aims to support the development of RSs and, therefore, should take into account the same classical threats that affect all RSs. We made a review of the literature to capture the main concepts and explain the main threats that may affect RSs. This review can be found in Section 2.

The literature review continues with a Systematic review of literature (SRL) on OB-RSs, discussed in Section 3. This review aimed at obtaining an overview of the field and to understand how ontologies have been used in the context of RSs and what are the gaps to be filled. The first gap we found is the absence of an inferential machine that could recommend a certain programming language given certain contextual information. The second gap we found is the need for a conceptual model of contextual information that is aligned with the needs of companies and that we can use in the inference machine. The discussion about the related work can be found in Chapter 3.

The SRL is followed by Section 4. In this Section, the artifact (the ontology) and its development are discussed. First, the method used to develop the ontology is discussed. This method is the SABiO Development Process [2]. Then the development is shown following the steps proposed by the SABiO method.

The evaluation in Section 5 is the next chapter. As said, the results of the interview with the project managers and the ontology expert, and the findings of this interview will be discussed. Lastly, the conclusion, limitations, and future works will be presented in Section 6.

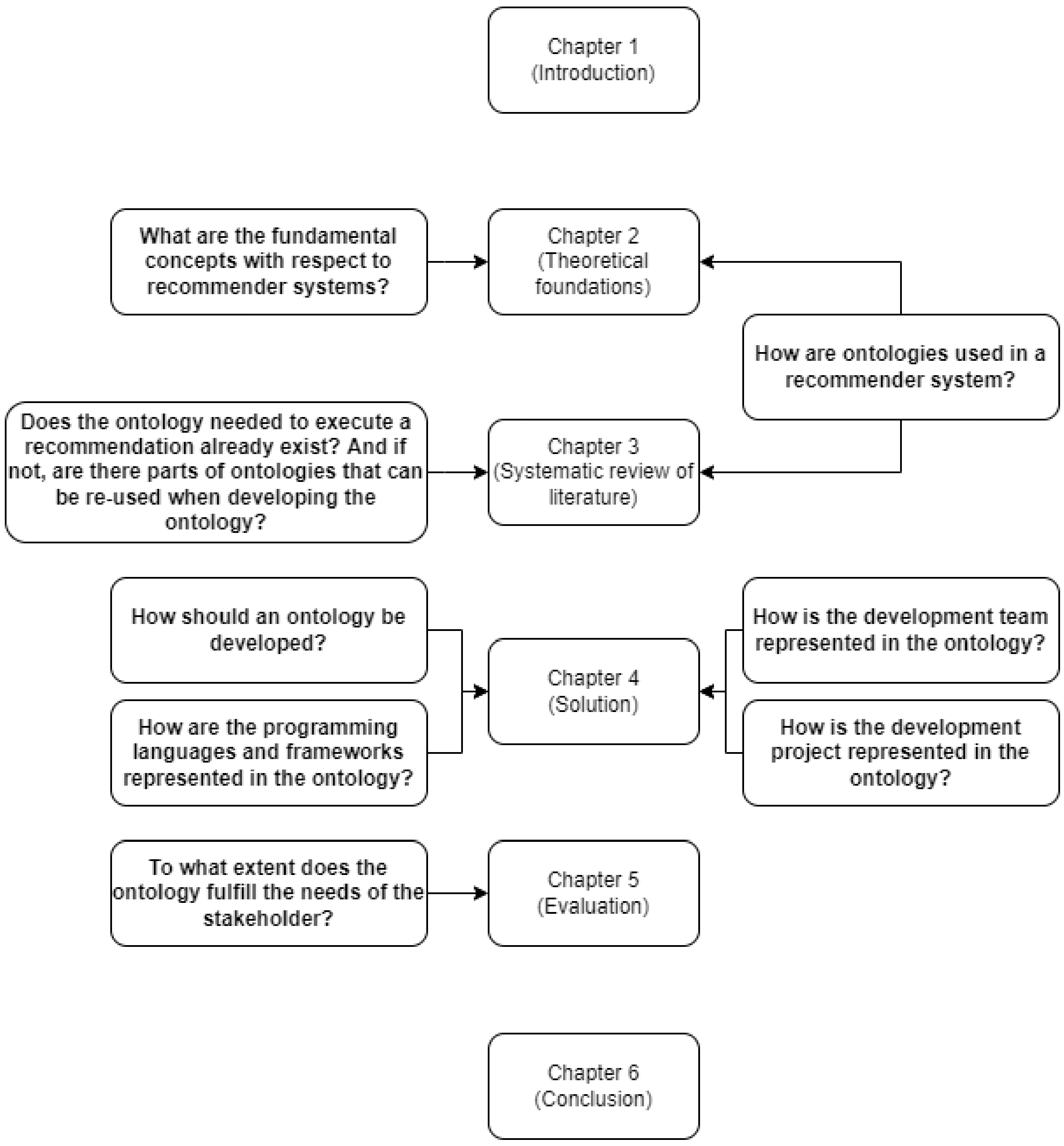

Chapter 1 (Introduction)
What are the fundamental concepts with respect to recommender systems?
Chapter 2 (Theoretical foundations)
How are ontologies used in a recommender system?
Does the ontology needed to execute a recommendation already exist? And if not, are there parts of ontologies that can be re-used when developing the ontology?
Chapter 3 (Systematic review of literature)
How should an ontology be developed?
How are the programming languages and frameworks represented in the ontology?
Chapter 4 (Solution)
How is the development team represented in the ontology?
How is the development project represented in the ontology?
To what extent does the ontology fulfill the needs of the stakeholder?
Chapter 5 (Evaluation)
Chapter 6 (Conclusion)

Figure 1.2: Overview of the RQs and their corresponding Sections

2 THEORETICAL FOUNDATIONS

RSs are software tools and techniques providing suggestions for items to be of use to a user [7]. These suggestions are designed to be a key part of the user's decision-making process, but not the decision itself. An example is when Netflix recommends a movie or series you might like or when Amazon tells you that, based on your selection, you might be interested also in other products. Both the movies and the additional products are recommendations done by these systems. To make a recommendation, these systems use known methods alone or in combination. The Section 2.1 defines the methods used by RSs when making a recommendation to their users. The design of such systems entails some classical challenges shared by almost any RS. These challenges are discussed in the Section 2.2. Section 2.3 provides a discussion on the challenges for the evaluation of such systems. Next, the subject of ontology will be discussed in Section 2.4. Finally, Section 2.5 will be about OB-RSs.

2.1 Methods of Recommender Systems

RSs often are based on one of these three central approaches: *Collaborative Filtering (CF)*, *Content-Based Filtering (CB)*, and *Knowledge-based Filtering (KB)* [8]. These central approaches can be broken down into more specific kinds. A common approach used by many RSs is to combine different approaches into a hybrid one.

In *collaborative filtering*, the RSs compare the characteristics and preferences of a user with that of other users; Whenever it finds users with similar preferences, it recommends to the former the items that were tried and *approved* by the latter, given that they share profile characteristics and/or preferences. Take for example the similarity in the preference for movies: if a group of people liked movies $a, b, c, d,$ and e and user x liked movies $a, b, c,$ and e (in that case, the users share similar preferences for movies, it is inferred from the rating history). It is likely that user x will also like movie d [7, 9, 10, 11].

One special kind of CF is the *demographic filtering* [7, 9, 10, 12]. It consists in defining similarity (to recommend items) based on the profile information, instead of the user ratings, mentioned in the movie's example. Information such as age, gender, language, etc. can be used to build such a profile. Similar to demographic filtering is the *context-based filtering*, in which information about the context of the user is taken into account, such as GPS information, time of the day, weather conditions, etc. These conditions might be used to recommend a certain tourist attraction or a place to eat, for instance [9, 10, 13]. *Community-based recommendation systems* is another special kind of CF. In this case, this method takes into account data about the social relationships a user has. From these relationships, the preferences are taken together with the ratings they already gave. Based on this data a recommendation is given to the user [7, 14].

In *content-based filtering*, the RS will compare the characteristics of different items instead of comparing the users. Here the idea is that when a user likes a certain item with a set of characteristics, this user is likely to like other items with similar characteristics. When looking at the example of movies, a movie has a genre, a duration, can be categorized as fiction or non-fiction, etc. If a user liked a non-fictional action movie that was between 90 and 105 min. long, then this user might like another movie with similar characteristics.

Another type of RSs is known as *knowledge-based filtering*. A RS is knowledge-based when it makes recommendations based not on a user's rating history, but on the specific requirements made by the user applied to *a priori* domain knowledge in the system [10]. This works well in situations where the domain is complex and there is not much data about this kind of decision, like rating history. An example of a complex domain is the real estate domain (many particular reasons may be key to the decision of user a, while another set may be decisive for user b), like the support for buying or renting a house. KB can be divided into two types, *constraint-based filtering* and *case-based filtering*. Constraint-based filtering uses constraints, like preferred color and size, to generate recommendations. Case-based filtering, on the other hand, makes use of examples given to the system, for instance, the user gives an example of a house that is not for sale in order to get recommendations for houses that are similar and are for sale. The biggest advantage of using KB is that they avoid the classic **cold start** problem of RSs.

Previously discussed methods are so-called personalized methods. They need user-specific information to make a recommendation. There exist RSs which are *non-personalized*. The recommendations resulting from this method are the same for all users. For this method to make a recommendation, it uses data of the community, either the ratings of the community or the actions of the community. For example, the best-liked movie of the week or the most sold book of the week respectively [15].

As previously mentioned, these methods can be combined in a so-called *hybrid* RS. Hybrid RSs can be created using different methods as defined by [16]. When results of several algorithms are used together it is called the *mixed method*. When the *switching method* is used, the system switches between several techniques based on a certain situation or criteria. A third approach is to combine recommendation techniques by giving the results from each of the algorithms in the system their own weight, this is known as the *weighted method*. The type of hybrid RS where the information of a CF technique is used as an extra feature over which a CB technique is run is called *Feature combination*. The hybrid recommendation system method *cascade* uses different RS methods in a specific order. The first RS method will make a recommendation of the items, over these recommendations a subsequent recommendation method is run to refine the result by solving possible equally ranked recommendations. Another hybrid RS method where an order of the different methods is important is the *feature augmentation method*. This method uses its first RS technique to generate a classification, over this classification a second RS is applied. The difference between feature augmentation and cascade is the fact that in feature augmentation the second RS system uses the output of the first system. The last method is called *meta-level*. This method has a higher level in the sense it uses the model learned by the first RS as input for the second RS [7, 9, 10].

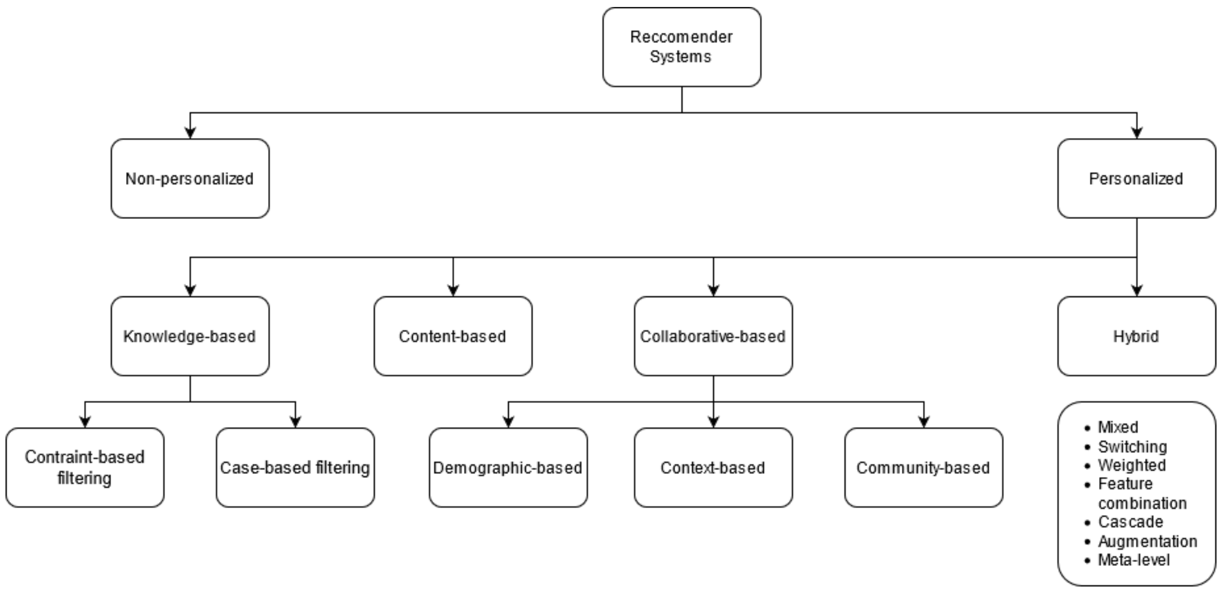


Figure 2.1: Overview of the different types of Recommender Systems

2.2 Challenges of Recommender Systems

All the previous methods have their challenges. The most common challenges that might occur when working with RSs are (a) the cold start problem, (b) latency problems, (c) shilling attacks, (d) privacy issues, (e) limited content analysis, (f) the grey sheep problem, (g) sparsity, (h) scalability, (i) the occurrence of synonyms, (j) evaluation problems[17, 18].

One of the most common challenges in CF is the *cold start problem* [19, 11]. The cold start problem is the problem where a recommendation has to be made for a new user or item. It is possible to differentiate between three types of cold start problems. The problem where a recommendation has to be made for a new user, the one where a recommendation has to be made for a new item (also known as the latency problem), and the combination of both, a recommendation for a new user about a new item. Solutions for the new user problem as mentioned by [11] are to ask users to rate some items, to give non-personalized recommendations, to ask users to list preferences, or to use demographic information given by the user.

For a product to be considered for recommendation using a CF method, it first needs ratings. Since newly added products do not have ratings yet they are not considered by the RS, this is known as the *latency problem* [20]. Clustering techniques might reduce the latency problem. Another possible solution is to use CB methods in combination with CF methods, or a category-based approach using stereotype cases [21].

Shilling attacks are concerned with purposely giving false information by malicious users [22]. These attacks are mainly used to gain an economical advantage by (in/de)creasing a certain product's rating. There are several types of shilling attacks, such as random, average, and bandwagon attacks, which are discussed in more depth by [22]. In order to reduce these attacks, they have to be detected. This can be done using statistical techniques such as anomaly detection or probabilistic Bayesian network models. Other ways to detect shilling attacks are to use supervised classification techniques, unsupervised clustering techniques, and variable selection.

Since data about users is used, there is also a concern about the *privacy* of the user [23]. To counter the privacy issues developers of the RSs can do a couple of things, such as anonymizing the data. Another option is to alter the data a bit by adding some uncertainty, this is called randomization. But external solutions can also play a role. Creating more awareness among users of what kind of data they are giving, as well as putting laws and regulations in place concerning the users' privacy helps to reduce the privacy problem.

The *limited content analysis problem* exists because there is not enough available content to analyze. All content-based techniques need feature information about the items and users, which is gathered using information retrieval techniques. The lack of available content to analyze can be explained because the content is either scarce or hard to obtain in a fixed format. This problem may lead to overspecialization, which in turn causes no novel items to be recommended. This can be solved by adding randomness.

A *grey sheep* is a user that does not match with any other group of users, therefore no recommendation based on the CF techniques can be given [24]. To counter this, a hybrid version of CF and CB can be used. Another approach to solve this is using clustering techniques such as K-means algorithms as shown in [24].

Because most of the databases used by RSs are big, the problem of *sparsity* exists [25]. There are millions of products on sites like Amazon.com, and for most products, there are no or very few reviews. If a user has liked a couple of items it is impossible to calculate the correlation between two users because there probably is no one that has a couple of the same items reviewed. The percentage the RS can give recommendations about is called coverage. The coverage on sites like Amazon.com is often very low. This problem occurs when using the CF technique. To reduce this problem hybrid CF methods such as a content-boosted CF method are proposed. But also solutions such as Singular value decomposition (SVD) techniques and demographic filtering can be used [26, 17].

Another common challenge is the *scalability* problem. For simple memory-based recommendation methods, giving recommendations based on a small dataset of items and users is no problem. When this dataset grows to have millions of products and users, all this data cannot be analyzed anymore and thus a scalability problem occurs [27]. Model-based RSs might be a solution but they have their own challenges. However, a combination of both is proposed by [27]. They use clustering techniques, so not all of the dataset has to be searched. Other solutions are to reduce the dimensionality using SVD techniques, or by pre-processing that combines cluster and content analysis [17].

The existence of *synonyms* in the data can also be a challenge [26]. For example the difference between an 'action movie' or an 'action film'. With both terms the same is meant, however, the methods will treat it as a different characteristic. SVD techniques can help reduce this problem.

Lastly, there is the *problem of evaluation*. There are nearly no benchmark datasets for specific domains. If they exist this would encourage the research and development of RSs. When using one of the few benchmark datasets that exist, it is important to choose the right metrics to evaluate the RS with the purpose of the system in mind.

2.3 Evaluation of Recommender Systems

The evaluation of the performance of a RS can be divided into three types of evaluation, offline analytics, user study, and online experiments [28]. In *offline analytics* statistical measures such as the accuracy are calculated. This type of evaluation does not need any human interaction with the RS, it only needs a dataset. The *user study* is a study where the RS is used in a controlled environment. Here the users act as testers, they have to perform certain tasks and answer questions about the RS. The last form of evaluation is *online experiments*, here the RS is used by real users in an uncontrolled environment.

One of the most important measures is the prediction accuracy of a RS. This can be measured using three metrics: (1) the Mean Absolute Error (MAE), (2) Mean Square Error (MSE), and (3) Root Mean Square Error (RMSE). All these metrics look at the difference between the rating prediction created by the RS and the actual rating given by the user. The difference is that MAE only looks at the difference, but does not take into account the direction of the error (i.e. if the prediction was higher or lower than the actual value). Because the difference between the predicted and actual value is squared the penalty is higher when using MSE, however, this gives a non-intuitive result. To make the result more intuitive the root of the result is taken in the RMSE metric.

There are other performance measures such as precision and recall. For precision, the number of recommended items that the user liked is compared to the total number of items recommended to the user. For example, if the user liked 6 out of the 10 recommended items, the precision is $\frac{6}{10} = .6$. Recall is measured by comparing the items the user liked from the recommended items and the total number of items the user prefers. So if the user liked 6 of the recommended items and in total there are 12 items the user would like, the recall is $\frac{6}{12} = .5$. In other words, recall is the proportion of the items the user likes that are not missed by the recommendation.

Other performance evaluation measures to take into account are not statistical calculations. Examples of these measures are the diversity of the items that are recommended or the level of trust a user has in the recommendations as well as the novelty and serendipity of recommendations made by the system.

2.4 Ontology

The term *Ontology* can be split into two parts *onto* and *logy*. *Onto* in Greek means 'existence', *logy* has the Greek meaning a branch of learning'. Putting this together *Ontology* can be translated to *the study of beings*. Ontology can also be defined as *a branch of metaphysics concerned with the nature and relations of beings* [29]. Ontology is among others concerned with asking questions about identity, classification, and causality. Examples of ontological questions are: "What kind of entities exist?", "Are things bundles of properties?", and "Is change possible without a changing thing?" [29].

However, a different meaning can be given to the word Ontology when it is used in a different community [30]. For instance, in the Computer Science field, Ontology is defined by [31] as: *an Ontology is an explicit representation of a conceptualization*. To understand this definition of Ontology, a definition of conceptualization has to be given. Conceptualization is defined by [32] as: *an abstract, simplified view of the world that we wish to represent for some purpose*.

An Ontology can be modeled in several ways. Each of these ways divides the ontology into several components. The four ways of modeling an ontology mentioned by [33] are *frames and first-order logic*, *description logic*, *software engineering techniques*, and *database techniques*. Because of the scope of this paper, only the frames and first-order logic approach will be discussed. The frames and first-order logic approach divides the ontology into five components: classes, relations, functions, formal axioms, and instances. *Classes* are used to represent concepts in the broadest sense of the word. Examples of classes are locations (city, village, etc.) or types of real estate (residential, commercial, industrial, etc). The second component is *Relations*, which represents a type of association between concepts of the ontology. An example of a relation type is the subclass-of relation. Looking at the real-estate example, a ’semi-detached house’ is a subclass of ’residential’, and ’residential’ in its turn is a subclass of ’real-estate’. *Functions* is the third component to model the ontology. A function applies information about a certain element onto another element. For example, we could have a function called ’Pay’ that takes the element ’hotelroom’ and the element ’discount’ to come to an element called ’finalPrice’. The fourth component is the so-called *formal axioms*, which are statements that will always be true. For example, ”a semi-detached house has exactly one other house adjacent to itself”. The last component is Instance. An instance represents an individual or element, for example, the house on the address ’ontologystreet 123 located in Ontologycity’.

Ontology can be evaluated using several criteria. An ontology can be verified and validated as well as being evaluated on accuracy, adaptability, clarity, completeness, computational efficiency, conciseness, consistency, and organizational fitness as mentioned by [34]. When an *ontology verification* is performed, it is checked if the ontology is built in the right way. When an *ontology validation* is performed, an answer is given to the question of whether the right ontology is built. The *accuracy* of an ontology is based on the question of whether the ontology correctly represents the aspects of the real world. The *adaptability* has everything to do with whether or not an ontology can handle a range of tasks and how it reacts to minor changes. The level of *clarity* is based on how effectively the meaning of the defined terms in an ontology is communicated. In other words, is this ontology understandable? The ontology’s *completeness* depends on how well the domain of interest is covered, i.e., does it include all relevant concepts and their lexical representations? The level of *computational efficiency* of an ontology is determined by the measure of how easily and successfully a reasoner can process the ontology. The *conciseness* of an ontology depends on whether there are axioms in the ontology that are not relevant in the domain the ontology is built for. Consistency of an ontology is based on the questions such as: ”when using these axioms, do they lead to contradictions?” and ”Does the documentation match the given specifications?”. The *organizational fitness* criteria are determined by how easy the ontology can be deployed within an organization. It is important to note that these criteria are hard to measure, if not impossible. However, they can be used as a guide when developing an ontology.

2.5 Ontology-Based Recommender Systems

OB-RSs is a type of knowledge-based RS [35]. As was previously mentioned, an ontology is a representation of a simplified world (in a certain domain). This representation is applied as the knowledge used by the knowledge-based RS. This representation could contain information about users, products, etc. in combination with the relationships between them.

Because OB-RSs are a type of knowledge-based RS, most of the challenges discussed in Section 2.2 will not apply. The cold start problem, rating sparsity, and overspecialization challenges, for example, will not be experienced since OB-RSs uses the ontology for the recommendation it makes, not ratings. However, it is important to note that developing an ontology is difficult and time-consuming [36]. It has been shown multiple times that the use of OB-RSs can improve the quality of a RS [36, 35, 37].

3 SYSTEMATIC REVIEW OF LITERATURE

In this section, the related work regarding OB-RSs will be discussed. We try to answer the following question: *What are the research projects conducted regarding the application of ontology-based recommender systems?*. First, the review protocol is discussed in section 3.1, which describes the way papers from different databases are selected. Next, the findings of the selected papers will be shown in Sections 3.2 and 3.3.

3.1 The review protocol

The databases in which the searches are performed are Scopus[1], Web of Science[2], IEEE[3], and ACM[4]. Scopus and Web of Science are used because of their large coverage. IEEE and ACM are used because they are specialized in engineering and computer science respectively. In each database, a search was performed on the data fields ”title”, ”keywords”, and ”abstract”. In these data fields a search was performed on the terms *Ontology-based Recommendation System(s)* and *Ontology-based Recommender system(s)*. The exact search string for each of the databases can be found in Table 3.1. When this search was performed (on 22 July 2021) there were a total of 103 hits.

Next, the selection criteria have to be defined. We made use of seven Exclusion Criteria (EC). When at least one of the EC is applicable to a paper, this paper will be excluded. The seven EC are:

EC 1 The paper is duplicate.

EC 2 The paper is not available.

EC 3 The paper is not peer-reviewed.

EC 4 The paper is not in English.

EC 5 The paper is not a domain application of an OB-RS.

EC 6 The paper is not mainly focused on an OB-RS.

EC 7 The paper is not an extended abstract.

After applying EC 1 to the set of papers, and thus removing all duplicates, 59 papers were left. The next step in the procedure is to read all the abstracts and titles of the remaining papers. If it is clear from the abstract and/or the title that the paper would have at least one of the EC, it would be removed from the set. In order to make the selection more reliable two people read all the abstracts and titles and give a verdict on whether the paper should be included. In case of a tie, i.e. one wants to remove the paper and the other one wants to keep it in, a third person is consulted to make the decision. After this step, 43 papers were left.

The next step is to read the remaining papers entirely and make a verdict based on the EC on whether the papers should be included. This, again, is done with two people reading every paper and a possible

[1] `https://www.scopus.com/search/form.uri?display=basic#basic`

[2] `https://www.webofscience.com/wos/woscc/basic-search`

[3] `https://ieeexplore.ieee.org/Xplore/home.jsp`

[4] `https://dl.acm.org/`

Scopus	TITLE-ABS-KEY (”Ontology-based Recommendation System*” OR ”Ontology-based Recommender System*”)
WoS	((TI=(”Ontology-based Recommendation System*” OR ”Ontology-based Recommender System*”)) OR AB=(”Ontology-based Recommendation System*” OR ”Ontology-based Recommender System*”)) OR KP=(”Ontology-based Recommendation System*” OR ”Ontology-based Recommender System*”)
IEEE	”Publication Title”:”Ontology-based recommender system*” OR ”Ontology-based recommendation system*” OR ”Abstract”:”Ontology-based recommender system*” OR ”Ontology-based recommendation system*” OR ”Author Keywords”:”Ontology-based recommender system*” OR ”Ontology-based recommendation system*”
ACM	Title:(”Ontology-based recommender system*” OR ”Ontology-based recommendation system*”) OR Abstract:(”Ontology-based recommender system*” OR ”Ontology-based recommendation system*”) OR Keyword:(”Ontology-based recommender system*” OR ”Ontology-based recommendation system*”)

Table 3.1: Search strings for each of the used databases

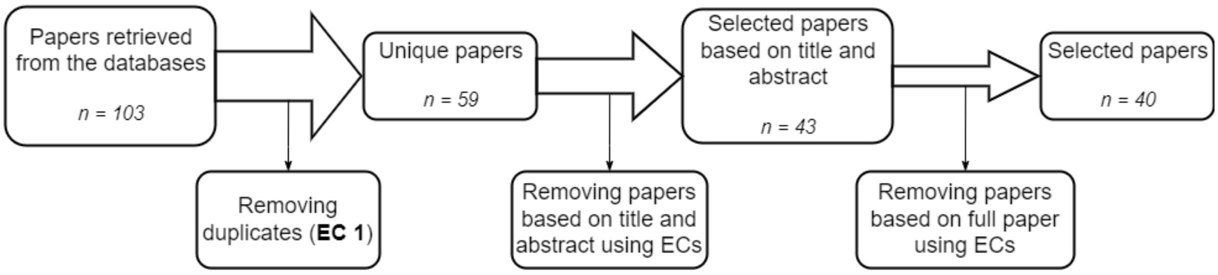

Papers retrieved from the databases
n = 103
Unique papers
n = 59
Selected papers based on title and abstract
n = 43
Selected papers
n = 40
Removing duplicates (EC 1)
Removing papers based on title and abstract using ECs
Removing papers based on full paper using ECs

Figure 3.1: Overview of the paper selection process

third person in case of a tie. One notion should be made, two papers [38, 39] found in the search were from the same author about the same topic. The latest published paper [39] seems to be an extension of the first published paper [38]. In order not to have misleading data representation these two papers will be regarded as one paper, used [39] as reference since this is the most developed paper. After this step, 40 papers were left. These papers were fully reviewed and analyzed. In the analysis, we started by collecting metadata of each paper, then we proceeded to an overview of all papers (geographic and temporal), which included a clusterization of these papers among two axes (type of evaluation and domain of application), to allow more in-depth analysis. This overview is discussed in Subsection 3.2. Based on the overview and clusters, we proceeded to an in-depth discussion, as detailed in Subsection 3.3.

3.2 Overview

In Figure 3.2, we illustrate the number of papers published per year. In absolute numbers, the chart in Figure 3.2 shows a sharp increase in numbers, although irregular. It’s worth noticing that the Coronavirus pandemic started in 2020, which may have had some influence on the number of publications. In addition to that, it’s worthwhile to mention that this review was written in 2021. Therefore, new papers may get indexed during the year, which will require this review to be updated accordingly.

Figure 3.3 illustrates the geographic distribution of the papers reviewed in this research. The geographic location is defined by the affiliation of the first author, which is usually where the research was conducted (regardless of the nationality of the authors). Alternatively, Figure 3.4, illustrates the same information as a heat map of the world. From these two figures, it can be concluded that India is the country that published the most papers regarding OB-RSs. Given that scientific production in Europe tends to be fragmented, it is also convenient to see an analysis per continent. In this case, Asia published the most

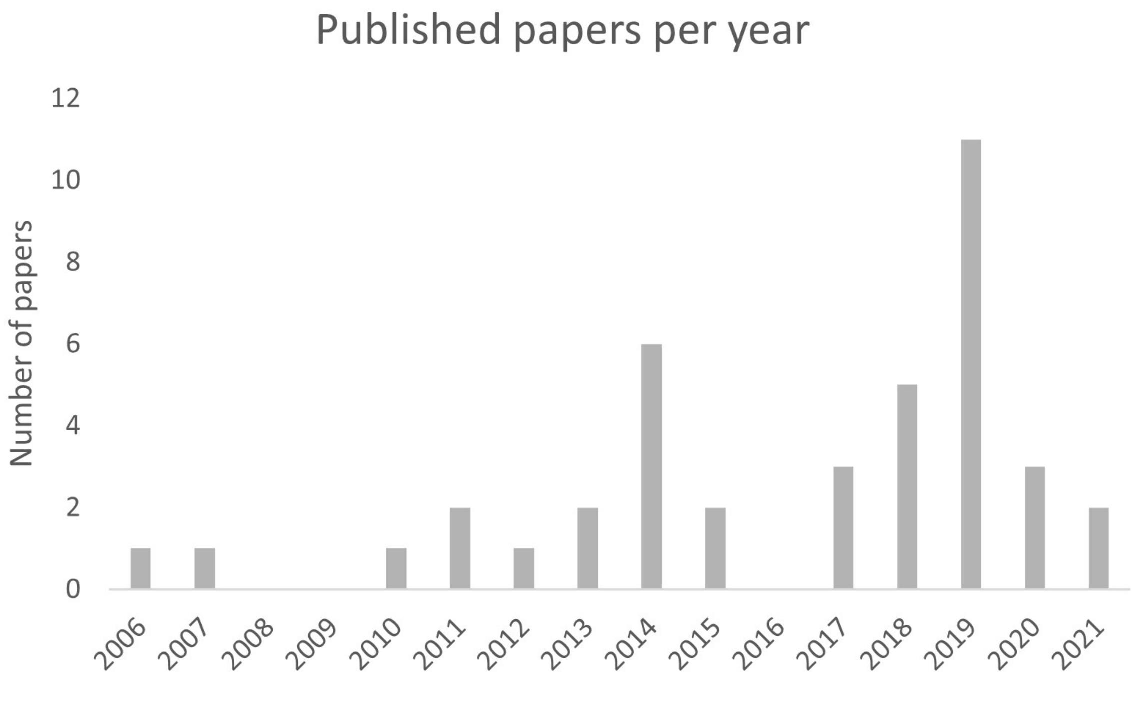


Figure 3.2: The number of papers published per year

papers (18 papers), followed by Europe (17 papers).

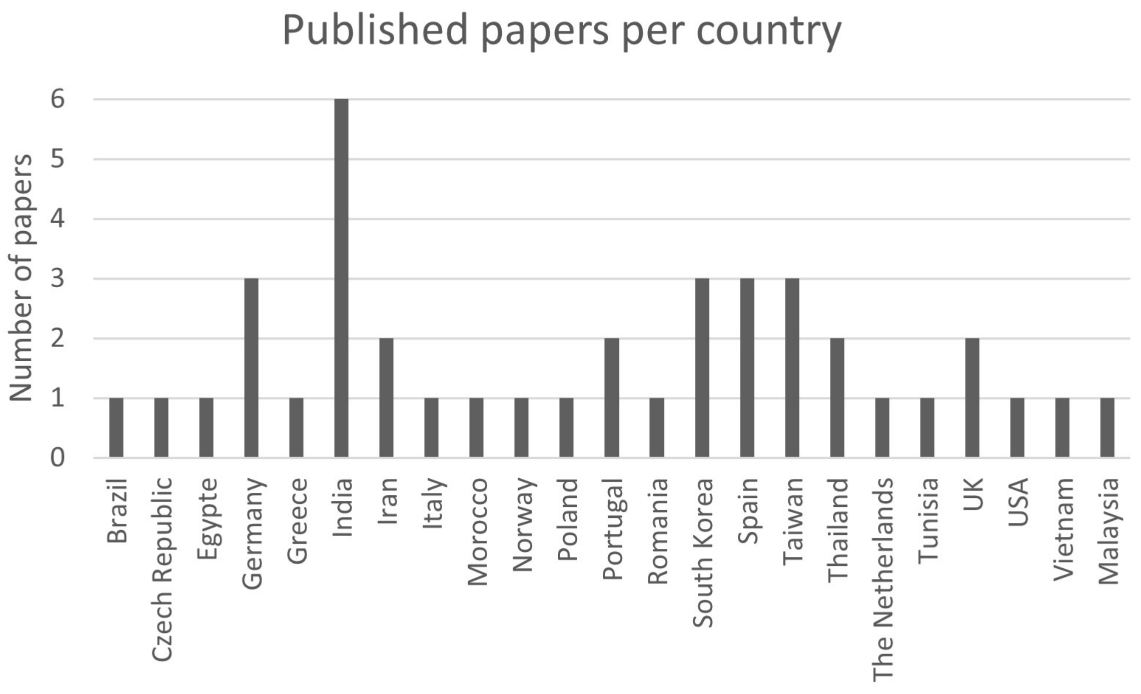


Figure 3.3: The number of papers published per country

From the 40 papers we reviewed, 28 papers had an evaluation of some sort, this could be a simulation/lab-based evaluation and/or an user study. In Figure 3.5 the distribution of the type of evaluation can be found, the number of papers for each type followed by the percentage with respect to the total papers

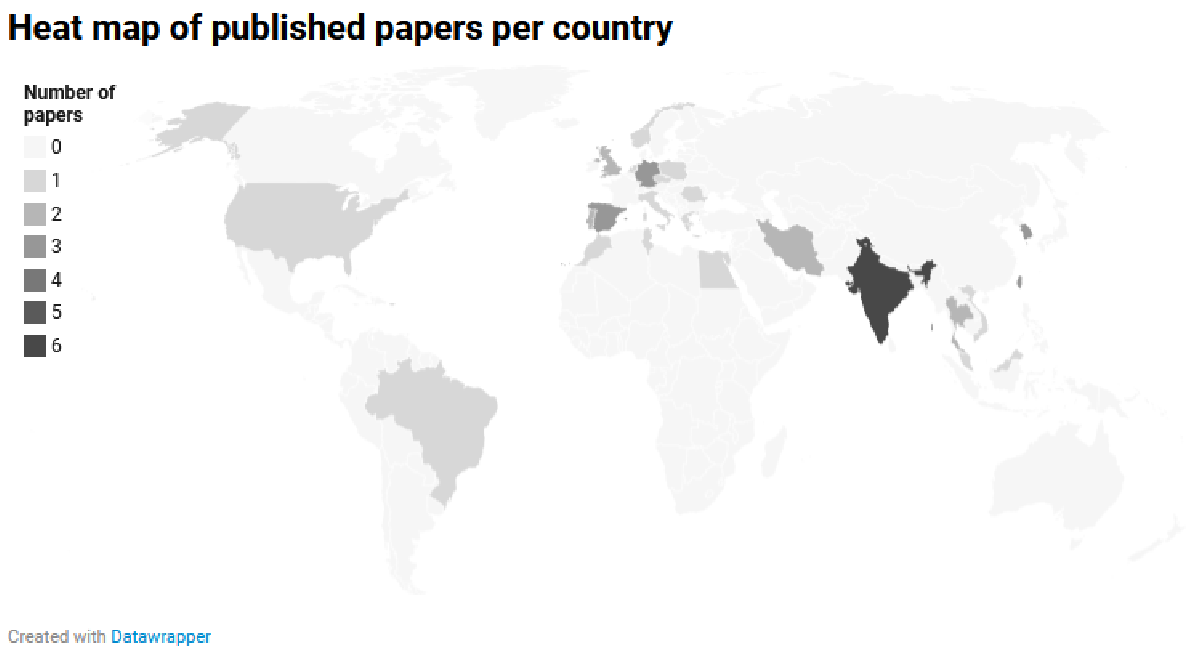


Figure 3.4: A heat map of the published papers per country

used is shown. Worth noticing is the fact that if an evaluation was performed this was mostly a lab-based evaluation where measures such as accuracy and recall are measured, as discussed in 2.3.

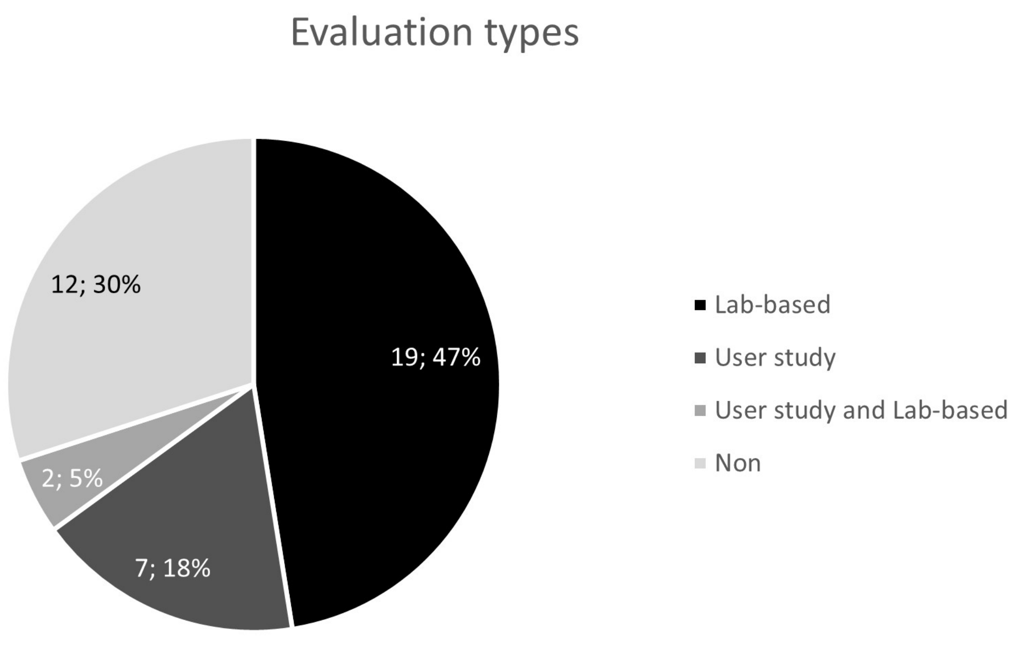


Figure 3.5: The distribution of the different evaluation types

Considering the maturity of the research presented in the papers of this review, we can highlight that 30% (12 papers) presented no evaluation at all. These papers contribute mostly to their domain, while they (slightly) contribute to the RS field by discussing the technique used to develop such systems in a specific domain. However, these side contributions become particularly relevant when a specific technique proves suitable for a given domain. These 12 works are distributed mainly among the domains of Healthcare (4 papers: [40, 41, 42, 43]), E-commerce (2 papers: [44, 45]), and Web 2.0 (2 papers:

[46, 47]). Among the papers with no evaluation, the less frequent domains are Knowledge Management [48], Transport Industry [49], Tourism [50], and Learning object [51]. Even though they didn't present an evaluation, the relevance of the aforementioned works is the novelty they bring to their respective fields. For instance, in Healthcare, the amount of information available on the web is massive, and users often lack confidence in the source and quality of information. Therefore, a well-designed RS is the first step to provide filtered content that matches some quality criteria defined by the users. This is further discussed in Subsection 3.3.

Almost half of the reviewed papers (47%) used lab-based evaluation. Of the total of 19 papers using lab-based evaluation, the Entertainment domain has four papers using it (4 papers: [52, 53, 54, 55]), which makes it the domain that uses this method the most. Three domains have three papers each using lab-based evaluation. The first one is Healthcare (3 papers: [56, 57, 58]), followed by the domain E-commerce (3 papers: [59, 35, 60]), and the domain of the RS field itself (3 papers: [61, 62, 63]). Worth noticing is the fact that the papers used in this review in both the domains Entertainment and the RS field solely used lab-based evaluation. Regarding the Healthcare domain, with the exception of [64] that presents both user-study and lab-based evaluation, the most recent papers are the ones presenting an evaluation (lab-based), while the 3 oldest papers presented no evaluation. This tendency indicates a maturity evolution in the Healthcare domain: more than presenting just a novel idea to approach information retrieval, recent papers in this domain must show some evaluation and a solid contribution to the field. The domain Agriculture also solely uses lab-based evaluation, this domain is represented by two papers in this review [65, 66]. Other papers using lab-based evaluation with their respective domains are: Books [39], Web 2.0 [67], Software [68], and Transport [69]. All papers, in this review, using lab-based evaluation have been published in 2014 or later.

The user-study method as an evaluation has been used by 7 (18%) papers. There is not one domain predominately using this evaluation method. For each of the following domains, one paper was found in this review that used an user-study as an evaluation method: Books [70], E-commerce [71], Software [72], Transport [73], Catering industry [74], Assistive Technology [75], and Learning object [76]. It should be noted the design of user studies differs greatly, e.g., [74] had users giving feedback without a standard questionnaire or any guidelines, whereas other papers used standard instruments, like the User Acceptance Test (UAT) to assess the system to evaluate Usefulness and Ease of use [76], among other common Human-Computer Interaction measures. The number of respondents also differed across the different papers found in this review. The last two papers (5%) found in this review used both a lab-based and user-study evaluation. These papers were found in the domains of Healthcare [64] and Books [77].

In Figure 3.6 the different domains of the papers are displayed. It can be seen that there are 10 domains with two or more papers. The group ”Others” contains the domains that are represented by only one paper in this research. These domains are: Assistive Technology, Catering Industry, Process plant design, and Tourism. Furthermore, Figure 3.6 shows that Healthcare and E-commerce are the furthest researched domains with 8 and 6 papers respectively in this research.

Topics discussed within the Healthcare domain are: Healthcare information management, Physical activities, Nurses' education and training, Biochemical emergencies, Drugs and nutrition, and Herbal medicine. It should be noted that the topic of nurses' training and education could also be part of the Learning object domain, however, the Healthcare domain is deemed a better classification. Most of the papers within the healthcare domain in this research are published in Europe, starting with the first publication in 2012 and the last paper in 2019.

When looking at the E-commerce domain all papers are focused on improving product recommendations. The difference between the papers lies in the approach they take to improve the recommendations. Some papers use a different combination of RS types, whereas other papers lay their focus on a certain type of information such as the user interest over time. The oldest paper regarding E-commerce in the scope of this systematic review was published in 2007, while the most recent publication we reviewed in this domain is from 2021.

The approach used to develop RSs in most of the papers is hybrid, where one of the used techniques was ontology-based (the scope of this SRL). Because of this filter, for all the papers included in this review, the approach is either hybrid or solely ontology-based.

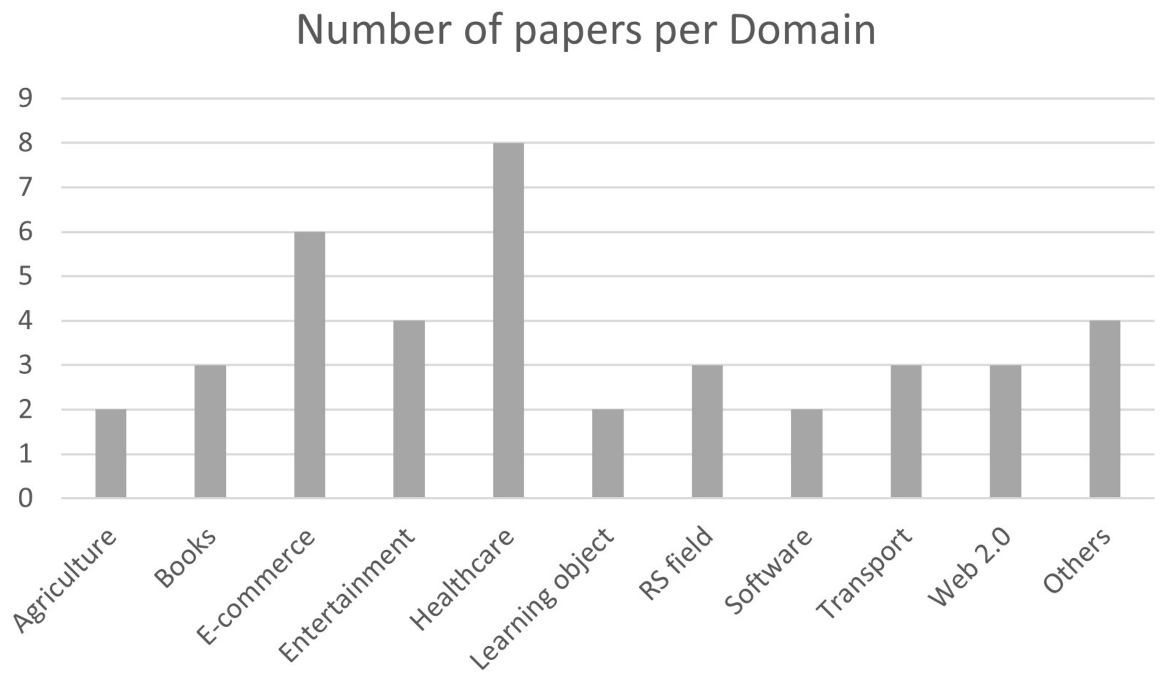


Figure 3.6: The number of papers for each domain

3.3 Discussion

As mentioned in Section 2.5, the cold start problem has been frequently addressed with the use of Ontologies. Four papers reviewed in this research explicitly cited the decision for an OB-RS aiming to mitigate the cold start problem [44, 45, 52, 55]. These papers indicate that using ontologies has been an emerging approach in our community to minimize the cold start problem. Another emerging use of ontologies in RSs is the attempt to increase diversity (serendipity). This review included three papers using OB-RS aiming at increasing diversity [35, 59, 60].

In this review, one trend that stands out is the use of an ontology to improve the recommendations in Healthcare and Entertainment. The evaluation approach used by the authors of these papers is lab-based (no user tests found). This might be because, in both domains, there is plenty of data available, which makes lab-based evaluations convenient. Another reason for the Healthcare domain is that the opinions of non-experts can be pretty dangerous, especially when these sources try to prove wrong assumptions regarding public health policies (like the COVID-19 protective measures). In these cases, although the ontology helps to filter, the contribution of experts to assert the quality of the information can be crucial. Regarding the Entertainment domain, the contribution of an expert is not as necessary as it is in the Healthcare domain. However, the large datasets available in the Entertainment domain (and its sub-domains, TV, and Music) usually include the rating given by users, making it convenient to perform lab-based evaluations.

Regarding the subject of Computer Science, and specifically Programming Languages, only one paper was found in this research [76]. This paper is included in the domain of Learning Objects recommendation. In this research, the goal was to reduce the time needed to find the most suitable options for learning objects for a specific learner of programming. However, this paper did not go into detail regarding the ”use of programming languages” (like productivity, suitability for dealing with certain problems/domains, etc.), rather they discussed the relationship between the topics of the language and how the learner should approach them, for instance, ”variables”, ”classes”, and ”SQL queries”. Between these concepts, a logic order has been proposed, together with an initial knowledge level test for each of these concepts. Using this information, a personalized learning path can be suggested. One of the goals of our research is to investigate existing ontologies regarding the use of programming languages that can help us choose one language given a certain project to be developed by a certain group. Since we didn't find such a paper, this remains an open topic that must be addressed in future steps of any research aiming at using ontologies to help recommend programming languages. For this kind of recommendation, the expected axes may include concepts like productivity, popularity, maintainability, and costs.

4 SOLUTION

The RS for which the proposed ontology in this research will be used is based on deductive reasoning. Therefore, it is less dependent on datasets and more robust to fight the cold-start problem than those based on inductive reasoning. However, although deductive reasoning systems don't require huge datasets to learn from past experiences, they may still need to be fed with some instances of data from previous experiences and, therefore (although to a lesser extent) they may also suffer from said problem. Because we can reliably test ontology-based systems with a few instances of data from previous experiences, the chosen approach makes it feasible to collect data and feed the system with instances to test it. In addition to the robustness against the cold-start problem, as discussed in Chapter 3.3, ontology-base systems can also be designed to mitigate other classic problems like: the latency problem, shilling attacks, lack of diversity (serendipity), sparsity, low coverage, and other data related issues. Because of the aforementioned reasons, we see the Ontology-Based recommendation as the most convenient approach to be investigated in this research. Hence, the decision for the use of such kind of reasoning. In this chapter, we describe the development of the ontology following the SABiO methodology [2].

4.1 SABiO: Ontology Development

SABiO stands for Systematic Approach for Building Ontologies The SABiO Development Process consists of five main steps, as illustrated in Figure 4.1.

The five prescribed steps are (1) Ontology Purpose Identification and Requirements Elicitation, (2) Ontology Capture and Formalization, (3) Ontology Design, (4) Ontology Implementation, and (5) Ontology Testing. We will briefly explain these steps next. More detailed information, such as the support processes on the right side in Figure 4.1, can be found in [2].

Step 1 in the SABiO Development Process is the Ontology Purpose Identification and Requirements Elicitation. In this step, we will perform four iterative activities. These are (a) Purpose & Intended Uses Identification, (b) Requirements Elicitation, (c) Competency Questions Identification, and (d) Ontology Modularization.

To perform the steps mentioned above, we should start by identifying the intended use needs. When this is done, we can define the requirements. When defining the requirements, there are two different types. Non-functional and functional requirements. The non-functional requirements can also be divided into three separate types. Non-functional requirements are the requirements concerned with (1) how the ontology will be/is intended to be used, (2) which characteristics/quality attributes it should have, and (3) how it will be applied to the project it is part of. Examples for each of the types respectively are, the use of standards for the terminology, the usability of the ontology, and time-to-market. Functional requirements are defined by the so-called competency questions (CQs). There are two types of CQs, low-level CQs, and high-level CQs. The low-level CQs are questions that the ontology should be able to answer relatively easily; the high-level CQs are more complex and abstract. Both types serve their own purpose. The low-level CQs are being used to derive test cases, while the high-level CQs guide the ontology modularization and general development direction. The fourth step, step 1d, is the Ontology Modularization. In this step, sub-ontologies will be defined if necessary. Finally, the complexity of the domain determines the need for sub-ontologies.

Step 2 of the development process is the Ontology Capture and Formalization. In short, this step entails capturing the domain conceptualization based on the CQ. To accomplish this, there are four iterative

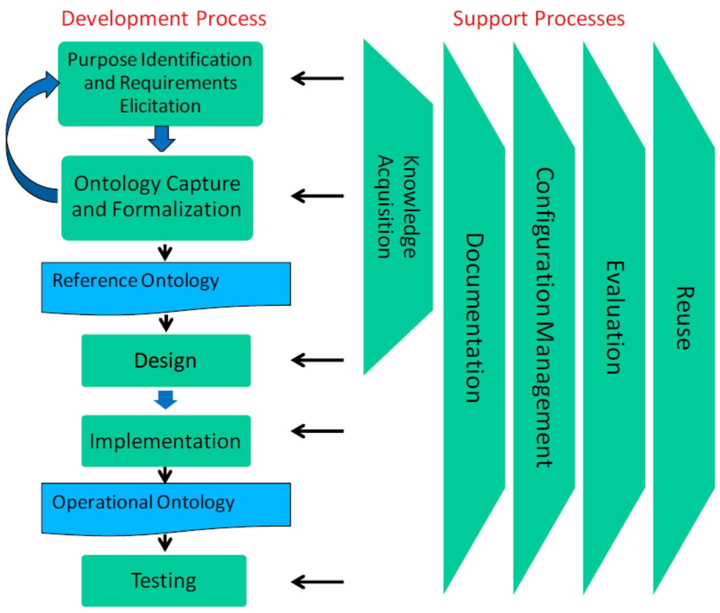


Figure 4.1: Overview of SABiO's Processes [2]

steps. First, starting at (a) Conceptual Modeling, next is (b) Dictionary of Terms Definition, followed by (c) Informal Axioms Definition, and lastly (d) Formal Axioms Definition.

The Ontology Capture and Formalization step generally make a lot of use of the support process "knowledge acquisition." In order to properly perform the capturing and formalization step, a representation language such as UML needs to be used to represent the ontology. The first step, step 2a, is concerned with identifying the main concepts and relations. The concepts and relations are the basis of any ontology. These concepts are accompanied by a dictionary that explains the definition of each term. This dictionary is the second step. Following this dictionary, axioms need to be defined. This will first be done informally, i.e., using a natural language. When the previously defined informal axioms are agreed upon, the axioms have to be made formal using formal language. Both ways of representation serve their purpose. The informal axioms are easier to understand since they are written in a language such as English. On the other hand, formal axioms are more precise and correct, making them easier to evaluate. After executing the first two steps, a so-called reference ontology is obtained.

Step 3 in the development process is Ontology Design. In this step, the previously created reference ontology will be made ready to be implemented. SABiO defines the goal of this step as follows: *In the design phase, the conceptual specification of the reference ontology should be transformed into a design specification by taking into account a number of issues ranging from architectural issues and technological nonfunctional requirements, to target a particular implementation environment.* [2]

Step 4 & 5, the Ontology Implementation and Ontology Testing follow. The Ontology Implementation entails the implementation of the ontology in a chosen operational language, such as OWL. The testing of the ontology will be done using test cases. These cases are an implementation of the CQs defined in the earlier steps.

4.2 SDP-Ontology development according to SABiO

4.2.1 SABiO step 1

As said the first step is to identify the purpose and the intended use of the ontology. The purpose of the ontology to be developed, is to be able to reason over it and recommended a programming language/framework for a certain software project. Therefore, the intended use is to use it as a representation of domain knowledge in a RS.

Next, the functional requirements needed to be defined in the form of CQs, as well as the non-functional requirements. The higher level CQ for this project is as follows: **Which programming environment should be used for the given project?**
Next, a list of the low-level CQs will be shown. These CQs typically are focused on one part of the ontology, and can be re-written to represent different use cases, by changing, for example, a certain quality attribute or programming language.

1. Which programming language and programming framework is used in the Programming environment?
2. What is the quality level of a given quality attribute for a given programming language?
3. What are the proficiency levels of an employee?
4. Which role(s) does a member of the team have?
5. What is the general experience of a certain employee?
6. What is the availability of a certain employee?
7. Which employees are in the team?
8. Which development method is being used?
9. For which type(s) of platform(s) is the software being built?
10. What is the type of development project?
11. Which architectural style belongs to the project?
12. What is the level of the project requirements?
13. What is the expected effort needed for the project?

What follows are the non-functional requirements. These are divisible into three groups. For each of these groups the requirements are listed.

Quality attributes:

1. The ontology is understandable/readable
2. The ontology can be changed/adapted
3. The ontology remains to be use able in the future

Project requirements:

1. The ontology should be implemented using standard languages and tools
2. Documentation of the ontology
3. The ontology is use able in the standards of the organization

Intended uses-related requirements:

1. It is intended to be used as the knowledge representation of the Software development domain for a, to be developed, recommender system.

The last part of the Purpose Identification and Requirements Elicitation step is to identify the modules/sub-ontologies. During the discovery period, it was found that three sub-ontologies would be needed. These are: (1) Programming Languages and Frameworks, (2) Development Team, and (3) Software Project.

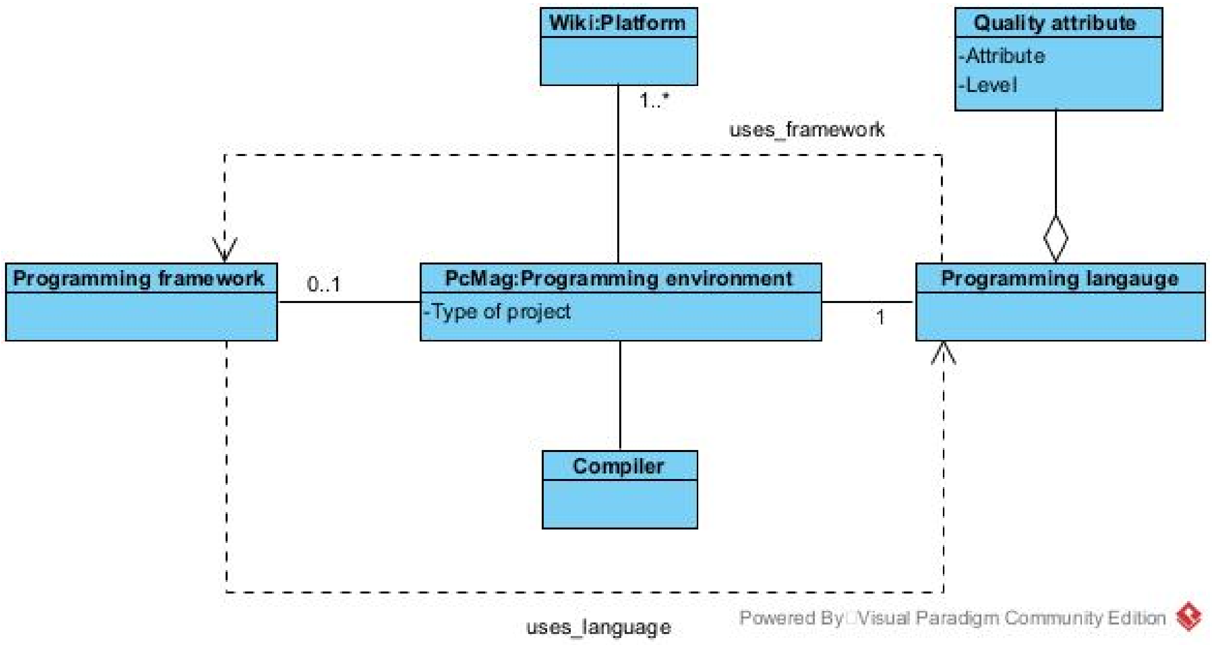


Figure 4.2: Graphical representation of the conceptual model for the sub-ontology Programming Languages and Frameworks

4.2.2 SABiO step 2

For the conceptual modeling part of this step, it is advised to use a graphical model. In order to do this Visual Paradigm (VP) [78] is used. After cycling through the iterative steps of step 2, the final sub-ontologies will be discussed, followed by the Dictionary of Terms Definition, and the axioms.

The conceptual models

The first sub-ontology is the Programming Languages and Frameworks ontology, which can be found in Figure 4.2. Here the 'Programming environment' is the central term. This environment contains a programming language, and possibly a framework. The environment makes use of a compiler, this class is added for completeness. Each of the programming languages has quality attributes. Each of these attributes in its turn has a certain quality level. Some environments can only be used for a certain type (e.g. Back-end) and a certain platform (e.g. Web).

The second sub-ontology is the Development Team ontology, which can be found in Figure 4.3. In this ontology, the central term is the Team itself. A team consists of a number of members who each have their own role(s) within the team. The decision is made to make a difference between a member and an employee. This is because a (team) member does not exist anymore when the team seizes to exist. However, The employee still does exist. In other words, an employee can be a member of multiple teams. When one of the teams this employee is part of is canceled, this employee is not a member of that team, however, the employee still exists. The consequence that follows is that a member can fulfill a business role, but an employee has a proficiency level (about a programming language), general experience level, and availability.

The third and last sub-ontology is the Software Project ontology, which can be found in Figure 4.4. In this ontology, the central term is the software development project. This project has a given effort, a certain type, an architectural style, and a development method. This development project is part of a Program, a collection of several projects. One specialized project is the Program increment which consists of Epics, which in its turn exists of Stories. Each Program increment is a collection of 3 Sprints. Next to the previously mentioned attributes, a project also has (a) given Platform(s) for which the project is intended to be used. As well as non-functional requirements, these requirements have an importance level. Lastly,

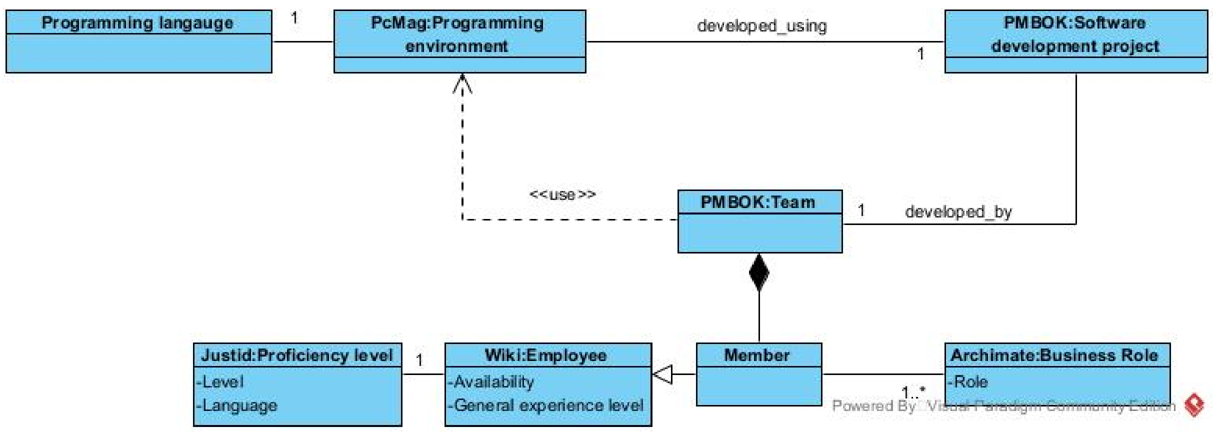


Figure 4.3: Graphical representation of the conceptual model for the sub-ontology Development Team

a project needs resources, of which the Team and the Programming environment are specializations.

Dictionary of Terms Definition

In this section the tables with all the terms and their corresponding definitions will be shown. The table will consist of three columns. Column one contains the terms as shown in the previously discussed models. Column two has its definitions. Column three will have the source used to define the definition when this is applicable. It can be possible for this column to not contain anything, in this case, the definition shown is decided upon after discussion with all parties involved. There will be three tables corresponding to the three models.

Table 4.1: The definitions of the terms used in the sub-ontology 'Programming Languages and Frameworks'

Programming Languages and Frameworks

Term	Definition	Source
Programming environment	A combination of language and frameworks that are used for software development	
Compiler	A computer program that translates instructions into machine language	[79]
Programming framework	A framework in programming is a tool that provides ready-made components or solutions that are customized in order to speed up development	[80]
Programming language	Any of the various high-level languages used for computer programs	[81]
Quality attributes	A characteristic that the developed software has	
Security	The degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.	[82]
Complexity	The state of having many parts that usually leads to being difficult to understand or find an answer to	[83]
Ease of use	The degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.	[82]
Maintainability	This characteristic represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it, or adapt it to changes in the environment, and requirements.	[82]

Programming Languages and Frameworks

Term	Definition	Source
Testability	Degree of effectiveness and efficiency with which test criteria can be established for a system, product, or component, and tests can be performed to determine whether those criteria have been met	[82]
Reliability	The degree to which a system, product, or component performs specified functions under specified conditions for a specified period of time.	[82]
Quality level	The degree to which a set of inherent characteristics of a product, service, or result fulfills the requirements. In agile, the definition of Done helps in maintaining a certain Quality level (link)	[84]

Table 4.2: The definitions of the terms used in the sub-ontology 'Development Team'

Development Team

Term	Definition	Source
Team	A set of individuals performing the work of the project to achieve its objectives.	[84]
Employee	An employee is a person who is hired to provide services to a company on a regular basis in exchange for compensation and who does not provide these services as part of an independent business	[85]
Availability	The number of hours an employee works per week (0-40h)	
General experience level	A measure of the number of experiences a person had with a certain Software Development Technology	[86]
Junior	Less advanced 0-2 years of experience	[87]
Intermediate	Having reached a fairly good level of skill or knowledge 2-4 years of experience	[88]
Senior	More experienced than the other members of a team 4+ years of experience	[89]
Proficiency level	The level of understanding of or information about a subject that you get by experience or study, either known by one person or by people generally	[90]
Aware	There is a basic knowledge of the topic, but not enough skill to apply it to a project	Justid
Beginner	Skills can be applied in a project, but guidance might still be needed	Justid
Professional	Experience with using the skill in a project, can work independently. Can solve issues on their own, and can guide more junior employees with related tasks.	Justid
Expert	Experience with using the skill in a project and can coach other employees. Follows the trends of the skill and shares knowledge with other employees	Justid
Leading expert	Understands the big picture and keeps the skill up-to-date. Advises and proactively shares information with other employees.	Justid
Business Role	A business role represents the responsibility for performing specific behavior, to which an actor can be assigned, or the part an actor plays in a particular action or event.	[91]
Scrum Master	Servant leaders and coaches for a team. They help educate the team in Scrum, Extreme Programming (XP), Kanban, and SAFe, ensuring that the agreed Agile process is being followed. They also help remove impediments and foster an environment for high-performing team dynamics, continuous flow, and relentless improvement	[92]
Software engineer	Someone whose job is to create computer programs	[93]
Tester	A person or a machine that tests the developed software	[94]
Product owner	Is a member of the team responsible for defining Stories and prioritizing the team Backlog to streamline the execution of program priorities while maintaining the conceptual and technical integrity of the Features or components for the team. The PO is the person most involved with the other stakeholders and is at the end responsible for the defined Stories (other team members might write stories as well).	[95]
Technical administrator	A role basis-oriented role in which the admin is, among others, concerned with installing and updating different stacks	[96]

Development Team

Term	Definition	Source
Functional administrator	A role focused on the, among others, implementation side, business process definition, and integration of systems	[96]

Table 4.3: The definitions of the terms used in the sub-ontology 'Software Project'

Software Project

Term	Definition	Source
Software development project	A temporary endeavor undertaken to create a unique product, service, or result. The temporary nature of projects indicates a beginning and an end to the project work or a phase of the project work. Projects can stand alone or be part of a program or portfolio. E.g. one Software development project can be the development of a mobile platform for some application while another project is the development of a web platform for the same application.	[84]
Program	A group of related projects, subprograms, and program activities that are managed in a coordinated way to obtain benefits not available from managing them individually.	[84]
Effort	A combination of time and people necessary to develop or maintain software. It is measured by Function Points and/or Story Points. Agile uses the term Velocity to measure the units of work done within a given timeframe (link)	[97] [98]
Architectural style	A 'software architectural style' is a specific method of construction, characterized by the features that make it notable"	[99]
SOA	An enterprise-wide approach to software development of application components that takes advantage of reusable software components, or services. In SOA software architecture, each service is comprised of the code and data integrations required to execute a specific business function — for example, checking a customer's credit, signing into a website, or processing a mortgage application. Each feature functions on its own.	[100]
Monolith	A monolithic application describes a single-tiered software application in which the user interface and data access code are combined into a single program from a single platform	[101]
Development Method	A process of dividing software development work into smaller, parallel, or sequential steps or sub-processes to improve design, and product management.	[102]
Agile	An iterative approach to delivering a project, which focuses on continuous releases that incorporate customer feedback. The ability to adjust during each iteration promotes velocity and adaptability. This approach is different from a linear, waterfall project management approach, which follows a set path with limited deviation.	[103]
Waterfall	The waterfall project management approach entails a clearly defined sequence of execution with project phases that do not advance until a phase receives final approval. Once a phase is completed, it can be difficult and costly to revisit a previous stage. Agile teams may follow a similar sequence yet do so in smaller increments with regular feedback loops.	[103]
The type of project	The type of the development project	
Front-end	Piece of software, or website that the user sees and interacts with to perform his work/tasks/business.	[104]
Back-end	Piece of software, etc., where data is stored or processed rather than the parts that are seen and directly used by the user. This includes the databases as well as middleware.	[105]
Full-stack	The combination of both front- and back-end.	
Platform	A computing platform or digital platform is an environment in which a piece of software is executed	[106]
Mobile	Used to describe a service available on a mobile phone, small computer	[107]
Web	Used to describe a service available on an internet browser	

Software Project

Term	Definition	Source
Desktop	Used to describe a service available on PCs and Laptops running a certain OS	
Non-interface	Used to describe a service available that doesn't have an UI interface	
Program increment	A software development project with a period of three sprints of three weeks, so each program increment (PI) is 9 weeks	
Epic	An epic is a large body of work that can be broken down into a number of smaller stories	[108]
Story	A user story is the smallest unit of work in an agile framework. It's an end goal, not a feature, expressed from the software user's perspective	[109]
Sprint	A given period, 3 weeks, in which a number of features will be developed	
Non-functional Requirement	A requirement represents a statement of need defining a property that applies to a specific system as described by the architecture or stakeholders.	
Importance	The importance level of the requirement	
Security level	The degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.	[82]
Time to Finish	The time in which the project needs to be finished	
Usability level	The degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use. Low ->Requires only acceptance test before deploying Medium ->Requires Expert inspection of UI before acceptance test High ->Requires Expert inspection of UI before acceptance test followed by a User Testing Report.	[82]
Resources	Things used during or to support the execution of activities	[110]
Software resources	Any software product that is used in the accomplishment of an activity, such as a network management software or a database management system	[110]
Hardware resources	Any hardware equipment required to perform an activity, such as computers and printers	[110]
Human resources	The roles that human agents are required to perform in an activity, such as requirement analyst, project manager, client, and so on	[110]

Axioms

The next phase in step 2 is the definition of the informal and formal axioms. In this section the informal axioms and their corresponding formal axioms will be shown. Since all the axioms are of the form ”if x, then y” the logical equivalent of ”y or (not x) ” is used to represent the axioms.

1. If the functional requirement ”Security Level” is high then the Quality attribute ”Security” of the chosen language should be high

```
(ProgrammingLanguage and (QualityAttribute and ((Qlevel value High)
and (type value Security))))
or (not (SoftwareDevelopmentProject and (NonFunctionalRequirement and
(importance value High_Importance) and (requirement value SecurityLevel))))
```

2. If the functional requirement ”Security Level” is medium then the Quality attribute ”Security” of the chosen language should at least be medium

```
(ProgrammingLanguage and (QualityAttribute and (((Qlevel value High) or
(Qlevel value Medium)) and (type value Security))))
or (not (SoftwareDevelopmentProject and (NonFunctionalRequirement and
(importance value Medium_Importance) and (requirement value SecurityLevel))))
```

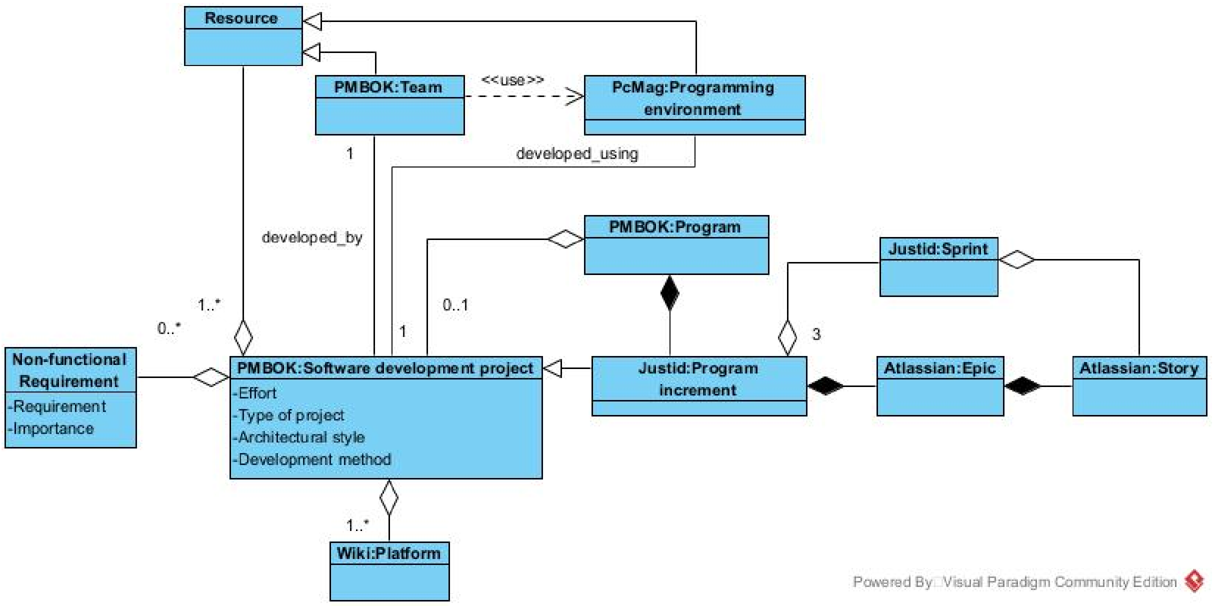


Figure 4.4: Graphical representation of the conceptual model for the sub-ontology Software Project

3. If the functional requirement ”Time to finish” is high then the Quality attribute ”Ease of use” of the chosen language should be high

```
(ProgrammingLanguage and (QualityAttribute and ((Qlevel value High)
and (type value EaseOfUse))))
or (not (SoftwareDevelopmentProject and (NonFunctionalRequirement and
(importance value High_Importance) and (requirement value TimeToFinish))))
```

4. If the functional requirement ”Time to finish” is medium then the Quality attribute ”Ease of use” of the chosen language should at least be medium

```
(ProgrammingLanguage and (QualityAttribute and (((Qlevel value High) or
(Qlevel value Medium)) and (type value EaseOfUse))))
or (not (SoftwareDevelopmentProject and (NonFunctionalRequirement and
(importance value Medium_Importance) and (requirement value TimeToFinish))))
```

5. If the functional requirement ”Time to finish” is high then the Quality attribute ”Complexity” of the chosen language should be low

```
(ProgrammingLanguage and (QualityAttribute and ((Qlevel value Low)
and (type value Complexity))))
or (not (SoftwareDevelopmentProject and (NonFunctionalRequirement and
(importance value High_Importance) and (requirement value TimeToFinish))))
```

6. If the functional requirement ”Time to finish” is medium then the Quality attribute ”Complexity” of the chosen language should at most be medium

```
(ProgrammingLanguage and (QualityAttribute and (((Qlevel value Low) or
(Qlevel value Medium)) and (type value Complexity))))
or (not (SoftwareDevelopmentProject and (NonFunctionalRequirement and
(importance value Medium_Importance) and (requirement value TimeToFinish))))
```

7. If the proficiency level of a member of the team is aware or beginner and the general experience level is junior, the complexity level of the language must be low

```
(ProgrammingLanguage and (QualityAttribute and ((Qlevel value Low) and
(type value Complexity))))
or (not (Employee and (ProficiencyLevel and
(((Plevel value Aware) or (Plevel value Beginner)) and
(language some ProgrammingLanguage))) and (GeneralExperienceLevel value Junior)))
```

8. If the proficiency level of a member of the team is aware or beginner the complexity level of the language must be low, when the functional requirement ”Time to finish” is high

```
(ProgrammingLanguage and (QualityAttribute and ((Qlevel value Low)
and (type value Complexity))))
or (SoftwareDevelopmentProject and (NonFunctionalRequirement and
(importance value High_Importance) and (requirement value TimeToFinish))
and (not (Employee and (ProficiencyLevel and (((Plevel value Aware) or
(Plevel value Beginner)) and (language some ProgrammingLanguage))))))
```

9. If the type of the project is “Front-end” then the type of the programming environment is ”Front-end” or ”Full-stack”

```
(ProgrammingEnvironment and ((typeOfProject_1 value FrontEnd) or (typeOfProject_1
value FullStack)))
or (not (SoftwareDevelopmentProject and (typeOfProject value FrontEnd)))
```

10. If the type of the project is “Back-end” then the type of the programming environment is ”Back-end” or ”Full-stack”

```
(ProgrammingEnvironment and ((typeOfProject_1 value BackEnd) or (typeOfProject_1
value FullStack))) or
(not (SoftwareDevelopmentProject and (typeOfProject value BackEnd)))
```

11. If the type of the project is “Full-stack” then the type of the programming environment is ”Full-stack”

```
(ProgrammingEnvironment and (typeOfProject_1 value FullStack)) or
(not (SoftwareDevelopmentProject and (typeOfProject value FullStack)))
```

12. The programming environment must be able to be used for the platform for which the project is developed.
    Since there are four platforms there are four formal axioms as well.

    Desktop:

```
(ProgrammingEnvironment
and (can_be_used_for value DesktopPlatform)) or (not (SoftwareDevelopmentProject
and (softwareDevelopmentProjectHasPlatform value DesktopPlatform)))
```

    Mobile:

```
(ProgrammingEnvironment
and (can_be_used_for value MobilePlatform)) or (not (SoftwareDevelopmentProject
and (softwareDevelopmentProjectHasPlatform value MobilePlatform)))
```

Web:

```
(ProgrammingEnvironment
and (can_be_used_for value WebPlatform)) or (not (SoftwareDevelopmentProject
and (softwareDevelopmentProjectHasPlatform value WebPlatform)))
```

Non-interface:

```
(ProgrammingEnvironment
and (can_be_used_for value Non-interfacePlatform))
or (not (SoftwareDevelopmentProject and
(softwareDevelopmentProjectHasPlatform value Non-interfacePlatform)))
```

4.2.3 SABiO steps 3 & 4

The next step in the SABiO process is the design phase followed by the implementation. To transfer the reference ontology into an operational ontology two tools are used. The first one is a plugin for VP [78] called OntoUML [111]. For this research OntoUML version 0.5.3 is used. This tool has several features, among others it enables OntoUML stereotypes to be defined in VP, as well as verifying the created models. The stereotypes of each of the elements can be found between the double-angled brackets, for example «kind». Each of the previously created models in step 2, and shown in Figures 4.2-4.4, are extended with the mentioned stereotypes and are specified with instances. Each of the new models will now be discussed in the same order as step 2.

In the model for the programming languages and frameworks, Figure 4.5, it can be seen that each of the programming languages has six attributes, (1) Security, (2) Complexity, (3) Ease of Use, (4) Maintainability, (5) Testability, and (6) Reliability. These attributes are picked from ISO/IEC 25010 [82] in consultation with Justid. Each of these attributes has one of three quality levels, (1) low, (2) medium, or (3) high. Since the assignment of these quality levels is depended on the organization we used arbitrary quality levels while testing the ontology. In the model programming languages and frameworks can be added as instances, although low-code platforms such as Mendix [112] are not included. Each of the added languages, e.g. Java and JavaScript, will have a value for each of the quality attributes. These quality attributes and their values, next to the project type and the platform, can be used to match a programming environment to the given project and team.

For the second model, the sub-ontology representing the Development team, found in Figure 4.6. A change with regard to the model made in step 2 can be found. The change is regarding the representation of an employee and a team member. In the model shown in step 2 (Figure 4.3), ”Member” was a specialization of ”Employee”. However, when modeling this sub-ontology using OntoUML, the relator pattern was recommended to represent this kind of relationship. Through this pattern, an Employee and a Business Role are connected through the relation of Membership. At the same time both Employee and Business Role are connected to the Team using the same relator.

The business roles used at this moment are in consultation with Justid, as well as the division of Experience levels. Each of the Employees has exactly one Experience level, either (1) Junior, (2) Intermediate, or (3) Senior. However, an Employee can have different Business Roles, within or across different teams. The proficiency levels per language are based on a matrix from Justid. For each of the programming languages, an Employee has exactly one level, from least to most proficient these are: (1) Aware, (2) Beginner, (3) Professional, (4) Expert, or (5) Leading Expert. The precise definition of each of the levels and roles can be found in Chapter 4.2.2-Dictionary of Terms Definition.

For the sub-ontology representing the Software Project, shown in Figure 4.7, some instances are created as well. For the non-functional requirements, the requirements chosen, in consultation with Justid, are (1) Security level, (2) Time to Finish, and (3) Usability. Each of these requirements also has an importance level ranging from Low to High.

A project can be made for several Platforms, namely (1) Mobile, (2) Web, (3) Desktop, and/or (4) Non-Interface. A Project has a certain architectural style and development method. Respectively these are

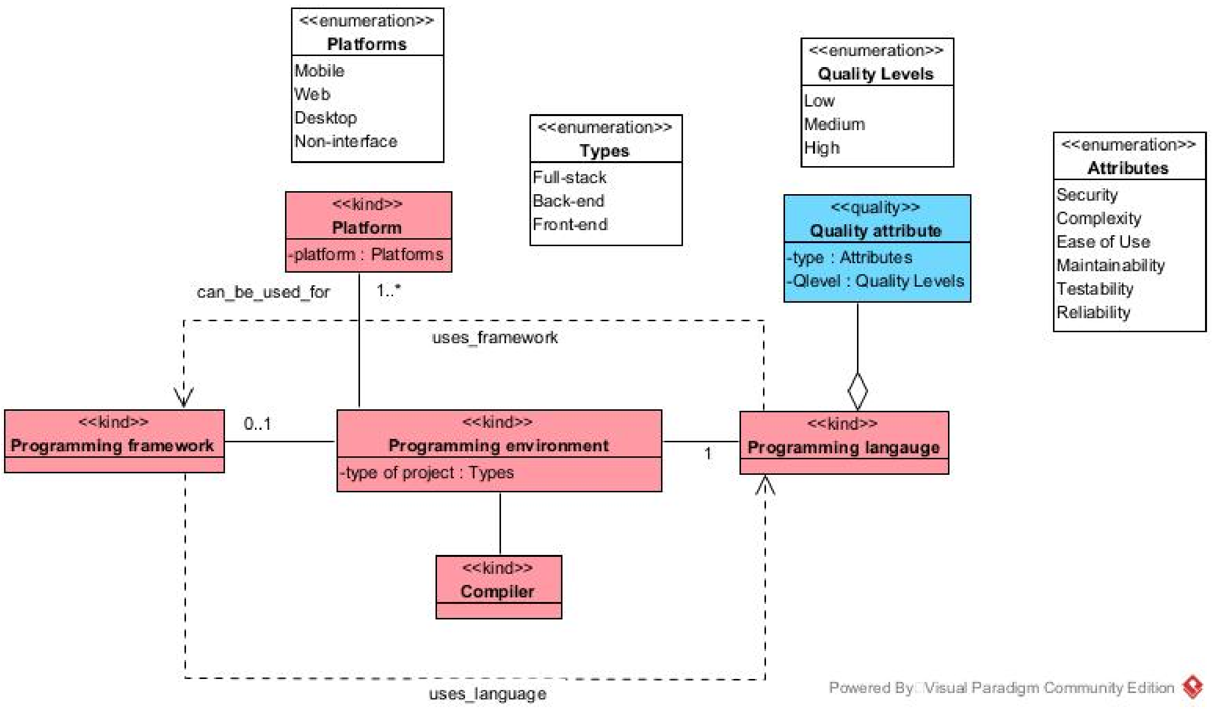


Figure 4.5: Graphical representation of the functional model for the sub-ontology Programming Languages and Frameworks

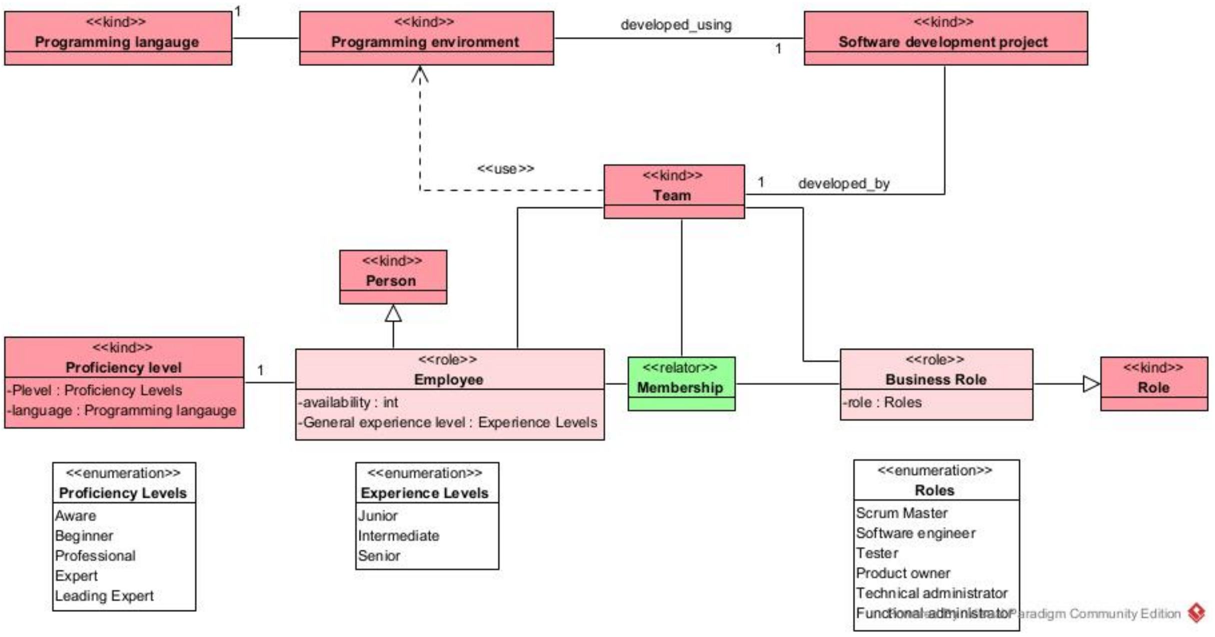
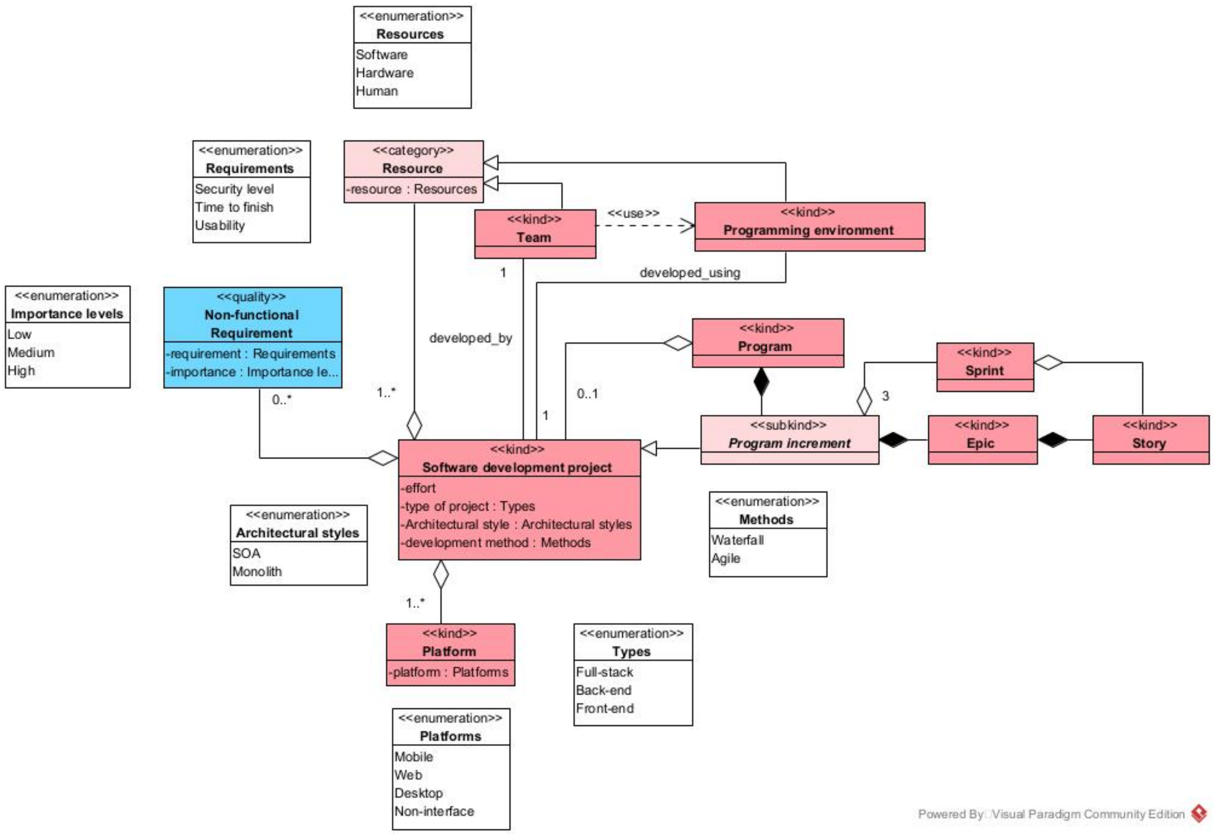


Figure 4.6: Graphical representation of the functional model for the sub-ontology Development Team



<<enumeration>> Resources
Software
Hardware
Human
<<enumeration>> Requirements
Security level
Time to finish
Usability
<<category>> Resource
-resource : Resources
<<kind>> Team
<<use>>
<<kind>> Programming environment
developed_using
<<enumeration>> Importance levels
Low
Medium
High
<<quality>> Non-functional Requirement
-requirement : Requirements
-importance : Importance le...
developed_by
<<kind>> Program
<<kind>> Sprint
<<subkind>> Program increment
<<kind>> Epic
<<kind>> Story
<<kind>> Software development project
-effort
-type of project : Types
-Architectural style : Architectural styles
-development method : Methods
<<enumeration>> Architectural styles
SOA
Monolith
<<enumeration>> Methods
Waterfall
Agile
<<kind>> Platform
-platform : Platforms
<<enumeration>> Types
Full-stack
Back-end
Front-end
<<enumeration>> Platforms
Mobile
Web
Desktop
Non-interface
Powered By Visual Paradigm Community Edition

Figure 4.7: Graphical representation of the functional model for the sub-ontology Software Project

(1) SOA or (2) Monolith, and (1) Waterfall or (2) Agile. The possible instances of the type of project are (1) Back-end, (2) Front-end, or the combination of both (3) Full-stack.

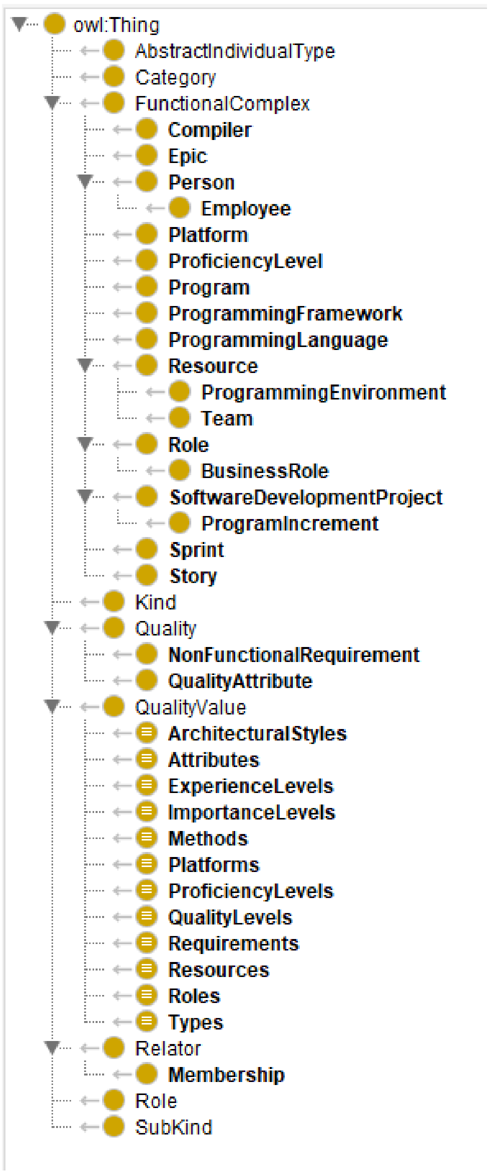
Lastly, a project has several Resources which can be subdivided into three categories, (1) Software, (2) Hardware, and (3) Human. The element Team is a specialization of a Human Resource, whereas the Programming environment is a specialization of a Software Resource.

For all the sub-ontologies it holds that when needed/desired the enumerations can be extended. For example, if another Quality attribute is found to be important to be included as well, this can be added.

Using the OntoUML plugin for VP a file could be created that could be imported into the second tool that was used to create a functional model. This second tool is called Protégé [3]. Protégé is an open-source ontology editor. For this research version 5.5.0 of Protégé is used. A screenshot of the classes can be found in Figure 4.8. The whole ontology can be found here.

4.2.4 SABiO step 5

The last step in the SABiO process is testing. The first step in testing is the reasoner of Protégé. The reasoner checks the consistency of the statements and definitions of the ontology. As previously mentioned the CQs are also a way of testing the ontology. Lastly, OOPS! [4] will be used to evaluate the ontology. OOPS! stands for 'OntOlogy Pitfall Scanner!' which is an online tool that helps detect some of the most common pitfalls. After the reasoner successfully ran, the CQ queries were executed, followed by the OOPS! machine.



- owl:Thing
  - AbstractIndividualType
  - Category
  - FunctionalComplex
    - Compiler
    - Epic
    - Person
      - Employee
    - Platform
    - ProficiencyLevel
    - Program
    - ProgrammingFramework
    - ProgrammingLanguage
    - Resource
      - ProgrammingEnvironment
      - Team
    - Role
      - BusinessRole
    - SoftwareDevelopmentProject
      - ProgramIncrement
    - Sprint
    - Story
  - Kind
  - Quality
    - NonFunctionalRequirement
    - QualityAttribute
  - QualityValue
    - ArchitecturalStyles
    - Attributes
    - ExperienceLevels
    - ImportanceLevels
    - Methods
    - Platforms
    - ProficiencyLevels
    - QualityLevels
    - Requirements
    - Resources
    - Roles
    - Types
  - Relator
    - Membership
  - Role
  - SubKind

Figure 4.8: The classes included in the model as shown in Protégé [3]

CQ queries

The queries are used to test whether the correct answer is given to the query. To run the queries the 'SPARQL Query' module of Protégé is used. Next, all the CQs, including a context, with their corresponding queries will be shown. Each of these queries will begin with the following header:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX so: <http://SDP-Ontology.com#>
```

1. **CQ:** Which programming language and programming framework is used in the Programming environment?
   Context: When there are a variety of possible programming environments, which language and framework are included in the possibly recommended environment is one of the key pieces of information.

   ```
   SELECT ?language ?framework
       WHERE {
       so:JavaOracleEnv so:programmingEnvironmentHasProgrammingLangauge ?language.
       so:JavaOracleEnv so:programmingEnvironmentHasProgrammingFramework ?framework.
       }
   ```

2. **CQ:** What is the quality level of a given quality attribute for a given programming language?
   Context: When a language is being selected the question arises which quality level each of the attributes corresponding to the selected language has. A question that could arise is what the quality level of the quality attribute Security for the programming language JavaScript is.

   ```
   SELECT ?type ?qlevel
       WHERE {
       so:Javascript so:programmingLangaugeHasQualityAttribute ?QualityAttribute .
       ?QualityAttribute so:type ?type.
       ?QualityAttribute so:Qlevel ?qlevel.
       FILTER(?type = so:Security)
       }
   ```

3. **CQ:** What are the proficiency levels of an employee?
   Context: When a team is known for the proposed project it is needed to know what the proficiency level for each of the language is for each member in a team in order to select a programming environment. It is also one of the pieces of information on which the selection of the environment is based. The example shown answers the question: "What is the proficiency level for the programming language Java of an employee?"

   ```
   SELECT ?language ?plevel
       WHERE {
       so:Employee1 so:EmployeeHasproficiencyLevel ?fullprof.
       ?fullprof so:language ?language.
       ?fullprof so:Plevel ?plevel
       FILTER(?language = so:Java)
       }
   ```

4. **CQ:** Which role(s) does a member of the team have?
   Context: When a team is selected, the team's composition regarding the roles is useful information, especially when an extra member needs to be added.

   ```
   SELECT ?employee ?role
       WHERE {
       ?membership so:membershipHasTeam so:Team1.
   ```

```
    ?membership so:membershipHasEmployee ?employee.
    ?membership so:membershipHasBusinessRole ?definedRole.
    ?definedRole so:role ?role
    }
```

5. **CQ:** What is the general experience of a certain employee?
   Context: When a team is selected, the team's composition regarding the general experience level is useful information, especially when an extra member needs to be added. It is also one of the pieces of information on which the selection of the environment is based.

   ```
   SELECT ?level
       WHERE {
       so:Employee1 so:GeneralExperienceLevel ?level.
       }
   ```

6. **CQ:** What is the availability of a certain employee?
   Context: When a team is selected, the team's composition regarding the availability is useful information, especially when an extra member needs to be added.

   ```
   SELECT ?availability
       WHERE {
       so:Employee1 so:availability ?availability.
       }
   ```

7. **CQ:** Which employees are in the team?
   Context: When a team is selected, each employee's knowledge level, both general and language-specific, is needed to select a fitting programming language.

   ```
   SELECT ?employee
       WHERE {
       so:Team1 so:teamHasEmployee ?employee.
       }
   ```

8. **CQ:** Which development method is being used?
   Context: Having an overview of the development method used, possibly for different projects, can help better decide what the appropriate method for the next project should be.

   ```
   SELECT ?method
       WHERE {
       so:Project1 so:developmentMethod ?method.
       }
   ```

9. **CQ:** For which type(s) of platform(s) is the software being built?
   Context: The project's intended used platforms is information on which certain languages/frameworks can be in- or excluded, as it might not be suitable for the desired platform.

   ```
   SELECT ?platform
       WHERE {
       so:Project1 so:softwareDevelopmentProjectHasPlatform ?temp.
       ?temp so:platform ?platform.
       }
   ```

10. **CQ:** What is the type of development project?
    Context: The project's intended type of development project is information on which certain languages/frameworks can be in- or excluded, as it might not be suitable for the desired type of development project.

    ```
    SELECT ?type
        WHERE {
        so:Project1 so:typeOfProject_1 ?type.
        }
    ```

11. **CQ:** Which architectural style belongs to the project?
    Context: Each project uses an architectural style. Knowing what style is used in each of the projects might be beneficial in choosing the next style for a new project.

```
SELECT ?style
    WHERE {
    so:Project1 so:ArchitecturalStyle ?style.
    }
```

12. **CQ:** What is the level of the project requirements?
    Context: When a project has a certain non-functional requirement that has a certain value, this means that for example a certain employee can't be part of the team, or a certain language can't be used in order to successfully complete the project. For example, the requirement 'Usability', as shown in the example.

```
SELECT ?req ?importance
    WHERE {
    so:Project1 so:softwareDevelopmentProjectHasNonFunctionalRequirement ?fullreq.
    ?fullreq so:requirement ?req.
    ?fullreq so:importance ?importance.
    FILTER(?req = so:Usability)
    }
```

13. **CQ:** What is the expected effort needed for the project?
    Context: When the project is finished, this information is needed to check whether the expected effort was correctly estimated, and when making a new estimation this can be used as a reference.

```
SELECT ?effort
    WHERE {
    so:Project1 so:effort ?effort.
    }
```

All the queries ran without an error and gave the expected output. This means that the proof by query is successful.

OOPS! results

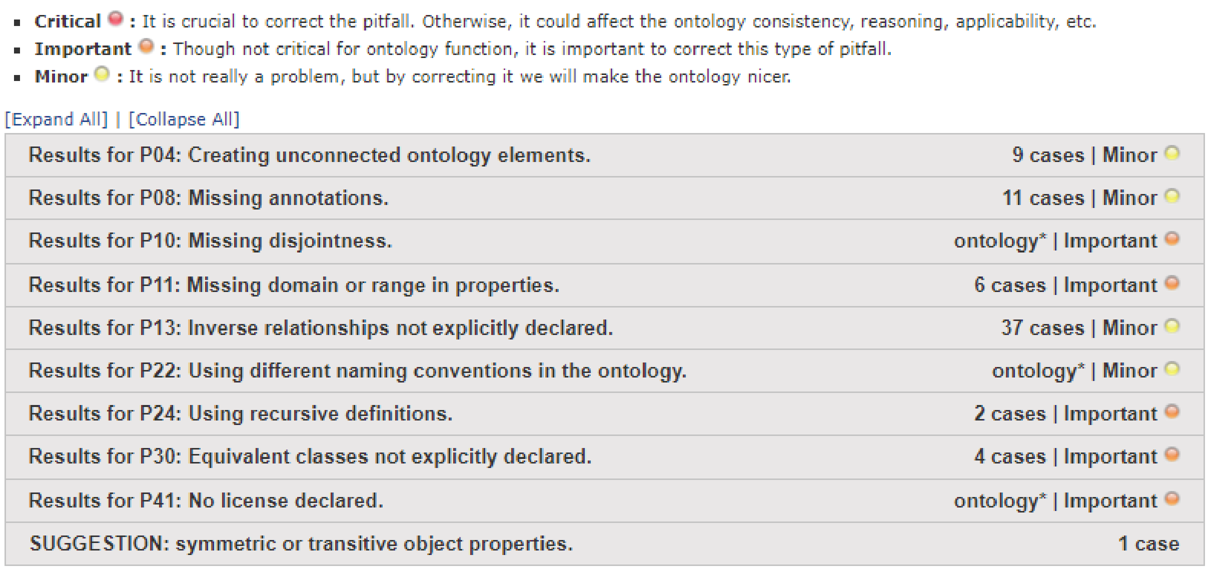
The CQ queries are followed by the results of the OOPS! tool. As can be seen in Figure 4.9, there are no critical pitfalls found within the proposed ontology. There are some other pitfalls, of which five are of the category 'Important', and four are 'Minor', as well as a suggestion. These pitfalls will be discussed next in the order shown in Figure 4.9.

The first pitfall is 'P04: Creating unconnected ontology elements', this means that there are elements that are not connected with any other element. However, if we look at the elements shown where this pitfall occurs, it can be seen that all these elements are created by the OntoUML plugin for VP, for example, the element 'Kind' is shown.

The next pitfall is 'P08: Missing annotations', which means that there is no human readable annotation attached, however, the mentioned elements are also created by the OntoUML plugin, with the means to group them together as one stereotype, hence an annotation is not really necessary.

'P10: Missing disjointness' is the next pitfall, which is concerned with elements/classes not having a specified disjoint with other classes. For this ontology to work this is not necessarily needed.

'P11: Missing domain or range in properties' is the pitfall where a domain or range is not specified for a certain data or object property. However, when looking into this pitfall, four of the six elements noted are inverses of another element where the domain and range are defined. This means that when the reasoner is run, these domains and ranges will be defined for these four elements. The other two elements are, again, created by the OntoUML plugin and therefore don't have or need a domain and range.



- **Critical** ● : It is crucial to correct the pitfall. Otherwise, it could affect the ontology consistency, reasoning, applicability, etc.
- **Important** ● : Though not critical for ontology function, it is important to correct this type of pitfall.
- **Minor** ● : It is not really a problem, but by correcting it we will make the ontology nicer.

[Expand All] | [Collapse All]

Results for P04: Creating unconnected ontology elements.	9 cases | Minor ●
Results for P08: Missing annotations.	11 cases | Minor ●
Results for P10: Missing disjointness.	ontology* | Important ●
Results for P11: Missing domain or range in properties.	6 cases | Important ●
Results for P13: Inverse relationships not explicitly declared.	37 cases | Minor ●
Results for P22: Using different naming conventions in the ontology.	ontology* | Minor ●
Results for P24: Using recursive definitions.	2 cases | Important ●
Results for P30: Equivalent classes not explicitly declared.	4 cases | Important ●
Results for P41: No license declared.	ontology* | Important ●
SUGGESTION: symmetric or transitive object properties.	1 case

Figure 4.9: Overview of the OOPS! results [4]

The following pitfall is 'P13: Inverse relationships not explicitly declared'. However, the suggested elements don't have an inverse relationship. For example, the element 'platform' is given as having this pitfall, but there isn't any element that is the inverse of 'platform'.

'P22: Using different naming conventions in the ontology' is noted next, which is concerned with for example using CamelCase and delimiters('_') in one ontology. Although this is the case, it doesn't influence the logical use of the ontology.

'P24: Using recursive definitions', is the pitfall where you use the element in its own definition. However, this was necessary to define certain axioms.

'P30: Equivalent classes not explicitly declared' is noted. The elements that are flagged as possible equivalents are actually the class itself and the enumeration class.

'P41: No license declared', An usage license is not declared, however, for this research this is not necessary.

'SUGGESTION: symmetric or transitive object properties'

5 EVALUATION

According to the Design science methodology (DSM), one way of performing the treatment validation step is by conducting an interview. Another possible approach to validate the artifact is having an evaluation by an expert. In this research, we performed both types of evaluation. The interview included key participants of Justid, while the expert evaluation was performed by an external ontology expert, that wasn't a participant in the development process.

5.1 Justid Interview

The interview was conducted with the Department Manager and one of the Software Developers of Justid[1]. The Department Manager is the person that requested such a research. The developer chosen for the interview is the person who was selected to support our research and followed the research since its first steps. We consider these two participants key to the evaluation because they understand the problem and they are capable of saying whether the produced ontology meets their requirements. The transcript of the interview can be found in Appendix A. Note that the interview was performed in Dutch and transcribed and translated to English.

Regarding the profile of the interviewees, the department manager takes an active role in the decision-making processes regarding IT projects. This employee has worked for 7 years within the organization Justid, in different roles of which 3 years in the role of department manager. Before working for Justid this employee had other working experiences as a software tester and team manager for IT teams. The other employee is a software developer who has worked for Justid for two years. Before this work, he did a Masters, and while studying he had a part-time job as a developer for a couple of companies. In total, this employee has about 7 years of experience, of which 2 years are full-time.

The interview script was prepared in a semi-structured approach comprising five topics: (1) Background, (2) Quality attributes, (3) Project requirements, (4) Intended-use related requirements, and (5) The future. Topics 2-4 relate to non-functional requirements, as listed in Section 4.2.1. In the interview, these requirements were discussed and the participants checked to what extent they are met. Topic 1 is included to gather information about the background of both employees. Topic 5 has as goal to discuss possible future directions and possible usages of the developed ontology.

The interview took place on Friday the 8th of July. It started with a presentation of the developed ontology to both employees. After the presentation, participants were interviewed at the same time (as in a focus group approach). This format was chosen to allow a more in-depth discussion in which both employees could add to each other's observations.

5.1.1 Results

In general, both employees found that the listed requirements were met sufficiently. However, some notions were made with regard to the ontology, as well as to possible further development paths. All the important findings, both positive and negative, will be discussed below. Starting with some general findings across the interview, followed by findings related to a specific topic. After this, the findings of

[1]more contextual information about Justid can be found in Section 1.1

the ontology expert will be discussed. In the final section of this chapter, all findings will be combined in a discussion.

General Findings

With regards to the Quality attributes both participants agreed that the ontology is understandable/readable and reflects the way the organization executes its work. As said by the department manager:

> I certainly agree with the statement "The ontology is understandable/readable"... ...at its core, all three models are very understandable and reflect the way we work.

They also agreed that the ontology is maintainable and remains use-able in the future, partly because of the usage of enumerations.

> At most, we are not familiar with the tooling, but the way it is modeled with the flexibility of different sub-models, objects, and building blocks, I certainly think it is maintainable, adaptable, and usable for us as an organization.

This gives the possibility to change the ontology specifically toward their needs, for example when it is found that there is no need for an instance of the mobile platform, they can remove this. As can be read they did note that they were not familiar with the tools used to develop the ontology and see this as a possible challenge. To overcome this challenge, it is proposed to make a guide on how to use the tools.

The participant representing the developers was very positive about the representation of the team, specifically the fact that an employee can have different roles within and across teams. The participant noted that this represents the actual situation very well.

> What I think is well represented is that the employees and the roles are separated. This is also how we work, one day I can have another role than the other day...

One of the improvements noted by this participant is the possibility to differentiate between a proficiency level in language and in the framework. This could be done by adding another class similar to the proficiency level class.

> I do want to note that we also look quite often at the experience level of an employee in a certain framework, and we now represent the general experience level and the proficiency in a certain language.

One of the project requirements is that the ontology should be use able in the standards of the organization. Both participants agreed that this is successfully done. For example with the use of proficiency levels, which are based on their own definitions. It was also noted that a large part of the organization leans on the principles of Agile, and these principles are found to be reflected well in the ontology.

> This organization, for a large part, is based on the principles of the agile world.
> So, you partly have the context of Justid, very well, and is well represented, also from an HR perspective, but is also good to see that the agile world is well represented, this is a good combination.

Although it was noted by the software developer that some terms in, for example, the roles could be made more general. For example, the role 'Scrum Master' does exist when using the Scrum method, but when using the Kanban method there is a similar role called 'Agile coach'.

> Within the roles, there is for example the Scrum Master, and this is within the method we use very commonly, but when looking at other agile methods, like Kanban, then this is not a Scrum Master but an Agile coach.

As possible future works, the department manager noted that making the ontology generally applicable and validating the ontology in a broad range of cases, by using information from other organizations, could be beneficial.

> I think it would be of added value to enrich this model with insights other than Justid.

Another suggestion for possible future work, is a system where some data has to be uploaded and a recommendation will be given.

> I think in the end the goal is to create a system in which all variables can be put in and a recommendation, given the context, will be given from this system.

After proposing the possible usage of recommending a development team instead of the language/framework, when for example a customer demands the software being developed in a certain language. Although some pitfalls noted by the participants have to be addressed, the participants saw this usage of the ontology as a possibility with the notion that it shouldn't become a competency management tool.

> It is possible, but then the risk rises that the nice goal for which this is built will be degraded to a sort of competence management tool. That only a list will be given of just some employees that could do this. This might miss the purpose of the tool.

The developer added as possible future work the possibility to connect the system to the sprint backlog. This could give the possibility to automatically update the skill levels of the given employee.

> ...the possibility to connect the backlog of the sprint onto the system so that experience levels can automatically be updated given the completed tasks.

Project management side

As said, both the developer and the project manager agreed that given the current goal, representing the software development process, the ontology reflects the way of working well. When answering the question "Does the ontology reflect the way the company sees Software Project Management?", The department manager noted that the project management side isn't included completely, however since this was not the goal it is not a downside, but intended as possible future work, as can be seen in the quote.

> But if I look at it, for the major part the development part is well represented but the project management part is not that represented. Take for example the project manager self, the HR-like parts, maybe something from PRINCE2, that is not in there yet. But I do not mind it at all, because this should be focused on the software development process.

When discussing the queries the department manager again pointed toward an extra business value when adding the project management side of a development project to the ontology. This addition of this knowledge being represented could open the possibility for more interesting queries and, therefore, more insights.

> ...when including the project management side and corresponding variables, more interesting queries could be built giving it even more business value than it already has.

Business understanding

The developer noted that some of the skill matrices they use now could be replaced by this ontology. This could be an extra usage of the ontology next to the ones already discussed.

> We work with skill matrices, and this could replace the skill matrix. This could give us more insight into the capabilities of the team and what the project entails, and the possible mismatch.

The developer acknowledged the usages of the ontology for recommending a programming language/framework but also saw the potential to use the ontology for existing projects to make adjustments based on the information represented in the ontology. From his point of view, there are two approaches/usages for the ontology. The first one is for ongoing projects where improvements and adjustments can be made using the ontology. The second one is where the ontology will be used for a new project and help decide what programming language/framework should be used. Both approaches use the context of the team and the project, to help their respective goals.

> So, there are two approaches I see. The first one is in a current project, you can look into what is the team composition, what is needed, what kind of project is it, which techniques are being used, and if needed where can we improve, need training, and/or share knowledge. The second one is for a new project, to look at what the team composition is and what the project entails and use this to recommend a programming language.

The department manager agreed with these two usages but wanted to emphasize that the ontology can really help make an objective recommendation. The department manager, therefore, sees more value in the ontology being used when starting a new project, whereas the software developer sees more value in using it for the ongoing projects since it is more relevant for the software developer.

> I agree with the viewpoint of SD, but I think it can really contribute to the decision-making for new IT projects within Justid. ... This ontology can help us make a more objective choice when starting a new IT project.

5.2 Ontology expert

The ontology expert had three main points of feedback. The first point of feedback is the level of detail. As noted by the expert the level of detail could be higher, e.g. next to the programming languages, the descriptors, the application services, and the development environment could be taken into account as well. Although this might make the ontology more complete, given the goal of Justid and the competency questions (CQs) this was not within the scope of this research. Therefore, the choice was made to not make it as specific. This could, however, be used as possible future work.

The second point of feedback is the level of depth in the CQs. Some of the CQs were too specific. Although the general case could be deduced from the specific CQs, the better approach would have been to have a more general CQ and possibly prove the CQ with a specific use case. Therefore the CQs that were too specific were changed and the CQs in Section 4.2.1 are the rewritten CQs. The CQs that were previously made are now used as an use case as can be found in Section 4.2.4.

The third point is the type of development method noted. In the ontology two development methods are listed, Waterfall and Agile. There are more development methods than these two. Since a software development project doesn't use, for example, only Waterfall the expert noted that it would be beneficial to add all these methods as well. However, some of these methods can be represented within the current ontology already. For example, the whole development project uses the Waterfall method, but all the steps within the Waterfall method use an Agile approach. This can be represented since the whole project is the Program which has as attribute the development method Waterfall, however, the program exists of several program increments which are projects on their own (albeit part of the larger program). These program increment projects have their own development method, which can be the Agile method. Hence such a development method for the whole project can be represented as well.

5.3 Discussion

The developed ontology is not yet complete, as was noted in both interviews. However, given the goal of Justid, the ontology can sufficiently represent the software development process within Justid. Continuation lies in adding more detail to the ontology and adding additional knowledge such as the project management side of a software development project.

An ontology in general can add business value [113]. An example can be found in [114]. As the department manager said in the interview, this ontology can also be used to add business value to Justid. So although the level of detail in the ontology could be more in-depth, the ontology at this point can already have added business value when a recommender system (RS) is developed to generate recommendations using this ontology. The addition of more information will, of course, add more business value.

It is interesting to see that there are more usages of the ontology. As noted in the interview the two most prominent ones are when starting a new project and when making adjustments/improvements to a current project. As noted in several researches, for example [115, 116], the team composition of a software development team is an important factor in the success of the project. When changing the ontology a bit, a third application for the developed ontology is possible as well in the form of recommending a development team. Although the department manager was cautious about it not becoming a competence tool.

6 CONCLUSION

In this research, we started with a literature review on the main concepts related to RSs (RSs). Following this narrative review, our research included an Systematic review of literature (SRL) that aimed to obtain the state-of-the-art in the development of RSs. These two literature reviews, combined, helped us answer RQ1, RQ2, and RQ3. After the literature review, this research investigated the development of an ontology for this domain. The goal of this second part is to answer RQ4 to RQ7. It is worth noticing that, for the development of our ontology, we followed the SABiO development process.

Regarding RQ3, we found that there was not yet an ontology developed that represented the software development domain including a representation of programming languages and frameworks, the development team, and the development project itself (RQ3). We acknowledge that this originality adds considerable value to the relevance of our research. In fact, to the extent of our review, only one paper was found regarding the topic 'Programming languages [76]. The goal of this paper was to reduce the time needed to find the most suitable options for learning objects for a specific learner of programming (not the same goal). Furthermore, given the difference in the goals, it wasn't possible to reuse any part of said ontology.

Another finding that is made in the SRL, is a confirmation that Ontology-based recommender systems (OB-RSs) can be used to mitigate the cold-start problem and increase serendipity. Another finding that stood out is the fact that the domains 'Healthcare' and 'Entertainment', which were well represented, only used lab-based evaluations.

The SRL is followed by the development of the ontology. The ontology is evaluated in multiple ways, objective (testing phase of SABiO methodology) and subjective (obtained through interviews with the potential user, project leader, and ontology expert). For an objective analysis, the Protégé reasoner and SPARQL Query modules were used to check its correctness. The reasoner ran successfully and the Queries gave the expected results. Next, to further substantiate the objective part of this research, we used the online tool *OOPS!* to check for the vulnerability for some common pitfalls. Although some pitfalls were indicated, there were no critical ones found. Secondly, some of the pitfalls were due to the usage of the OntoUML plugin for Visual Paradigm (VP).

The next step in the evaluation of our ontology included the evaluation of the ontology by an external expert (RQ8). During this evaluation, it was found that the competency questions (CQs) should have been less specifically defined, where a specific case could be used as a use-case to prove it can be queried. Next to this, the expert noted that a more in-depth representation of the domain would make the ontology more complete, however, given the scope of the research it was chosen not to do this and see this as possible future work

Finally, an interview with Justid occurred where the manager and one of the developers were asked to give their opinion on the ontology (RQ8). In this interview, it was found that in general both employees found that the listed requirements were met sufficiently. They noted that the representation of the employee-team member relation is as it is in practice. Next to this, they noted that the standards used in the organization were well represented and that the representation of the software development process was complete as well. As possible improvements, they noted that a manual for the usage of the tools used to create the ontology would be useful, as well as an added class for to represent the proficiency level in a certain framework. For possible future works, the idea to connect the system to the sprint backlog was introduced, as this could help automatically update the skill levels of an employee. Another possibility given is the extension of the project management side of the whole process.

Regarding the main research question[1], our approach consisted in investigating its feasibility with the use of ontology reasoning (since no dataset was found and that it's very difficult to obtain such a dataset). Our investigation showed that it's possible to develop such an ontology to be at *the heart* of a (future) RS for programming languages. Our ontology allows its users to consider a certain set of expected quality attributes of the project and the developers' team. Our results show that it's possible to use the ontology developed in this research as the knowledge representation of this particular domain in an OB-RS. Furthermore, during the course of this investigation, we realized the problem presented by Justid could be seen from different angles. For instance, if the Project Manager faces any restriction to the Programming Language to be used in a certain project, then the same ontology can be used by the RS to recommend the best team composition, given the project and programming quality attributes. We consider this feature an exciting Future Work of this research.

6.1 The research questions

In Chapter 1 eight RQs are stated. In this section, each of the RQs will be answered. For convenience the RQs will be listed in this section as well, accompanied by the chapters in which these RQs are answered in more depth.

RQ1 What are the fundamental concepts with respect to recommender systems? (Chapter 2)

To address this research question, we conducted a narrative literature review on the fundamental concepts regarding RSs. This first step clarified the relevance of ontology reasoning to solve some classic problems of RSs. As part of the fundamental concepts, we highlight two main approaches for the development of RSs, Collaborative Filtering (CF), or Content-Based Filtering (CB). An in-depth investigation of literature presented more (sub)types and, highlighted the popularity of hybrid approaches to develop RSs. Each approach has strengths and weaknesses. However, among all of them some common challenges stand out, like the cold-start problem and the lack of diversity in the produced recommendations, both problems being more relevant in approaches that are more dependent on data.

RQ2 How are ontologies used in a recommender system? (Chapter 2 and 3)

Ontology reasoning lies at the core of a specific type of knowledge-based RS, called OB-RSs. In this kind of RSs, the ontology functions as the knowledge base that represents specific domain knowledge. There are several examples of OB-RSs that successfully mitigated the cold-start problem and the lack of diversity, i.e. increasing serendipity.

RQ3 Does the ontology needed to execute a recommendation already exist? And if not, are there parts of ontologies that can be re-used when developing the ontology? (Chapter 3)

In order to answer this question, a Systematic Review of Literature was performed. In this SRL, we didn't find an ontology that could be used for the purpose of recommending a programming language/framework. Specifically, only one paper was found regarding the domain Programming, however, this paper didn't match the goal of our research in any way, and therefore had no reusable parts.

RQ4 How does an ontology should be developed? (Chapter 4)

For the development of an ontology, a specific process has to be chosen. In this research, the SABiO development process [2] is chosen. This method consists of five steps starting from the initial development step to the testing of the developed ontology. It is worth highlighting that, in the testing step of this research, besides performing several SPARQL queries and software tests for correctness, an expert evaluated the ontology as well.

[1]For convenience, we repeat here our main Research Question: *How to make recommendations of programming languages and/or frameworks to be used in a Software Development Project, given a certain set of expected quality attributes of the project and set of information about the developers' team?*

RQ5-RQ7 How are the programming languages and frameworks represented in the ontology? (Chapter 4)
How is the development team represented in the ontology? (Chapter 4)
How is the development project represented in the ontology? (Chapter 4)

For each of the three sub-ontologies, Programming Languages and Frameworks, Development Team, and Software Project, a conceptual model was created with the use of VP [78]. Following this conceptual model, a functional model was created using a VP plugin named OntoUML [111] and the tool Protégé [3]. Each of these sub-ontology's consists of several elements. These elements all have a definition based on sources such as PMBOK [84] and ISO/IEC 25010 [82], but also based on definitions provided by Justid.

RQ8 To what extent does the ontology fulfill the needs of the stakeholder? (Chapter 5)

The main stakeholder for this research is Justid, as it is the organization where the research is performed. To represent Justid and test to what extent the requirements were full filled, an interview with the department manager and one of the developers took place. From this interview, a general approval was given, although some possible improvement points were noted as well. In total Justid is happy with the result created from this research.

6.2 Limitations

Although part of the research is partly based on the input of one organization, Justid, e.g. the element 'proficiency levels', these definitions can be generally used. I.e. although the source of some information might be Justid, the definitions are still generically applicable. However, the ontology is only validated in one case and should be validated in a broad range of cases.

This ontology is filled with some instances of, for example, programming languages with the purpose of testing. However, to have a better overview of, for example, all programming languages, more instances should be added, such as 'Python' and 'C++'.

Lastly, this ontology doesn't have a high level of detail. For the purpose of this research, the level of detail was deemed enough by the stakeholder. When further using the ontology the level of detail could be higher in order to more completely represent the full domain.

6.3 Future Works

Despite the limitations, this research has laid a foundation for possible future work which will be discussed next. As discussed earlier, a possible future work is to use the proposed ontology to recommend the formation of a development team given a project and a certain language. Since organizations typically use a certain subset of languages, it might be beneficial to recommend a development team instead of a programming language.

Another future work we envision is the development of an online environment where the ontology can be used for the recommendation process. This would entail, for example, an environment where an user can input some data about the project and about the development team (or the language of the previously mentioned future work is done) and get a recommendation for a language and/or framework. And when extending this, even give a couple of options with pro's and con's. The advantage of providing a central place for such recommendations is that it can reach a bigger number of users and potentially help us capture some feedback from the users that will help us further improve our ontology.

As noted by the department manager of Justid, the ontology doesn't completely represent the software project management side of the whole process. Therefore, future work could be the extension of the ontology with software project management.

Lastly, as previously mentioned, the validation across broad ranges of cases is another possible future work. As this would test the general applicability of the developed ontology.

REFERENCES

[1] R. J. Wieringa, *Design science methodology for information systems and software engineering*. Springer, 2014.

[2] R. de Almeida Falbo, “Sabio: Systematic approach for building ontologies.” in *ONTO. COM/ODISE@ FOIS*, 2014.

[3] M. A. Musen, “The protégé project: a look back and a look forward,” *AI Matters*, vol. 1, no. 4, pp. 4–12, 2015. [Online]. Available: https://doi.org/10.1145/2757001.2757003

[4] M. Poveda-Villalón, A. Gómez-Pérez, and M. C. Suárez-Figueroa, “OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation,” *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 10, no. 2, pp. 7–34, 2014.

[5] R. Sharda, S. H. Barr, and J. C. McDonnell, “Decision support system effectiveness: a review and an empirical test,” *Management science*, vol. 34, no. 2, pp. 139–159, 1988.

[6] H. A. Simon, *The sciences of the artificial*. MIT press, 1996.

[7] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*, 1st ed. Berlin, Heidelberg: Springer-Verlag, 2010.

[8] A. Felfernig, S. Polat-Erdeniz, C. Uran, S. Reiterer, M. Atas, T. N. T. Tran, P. Azzoni, C. Kiraly, and K. Dolui, “An overview of recommender systems in the internet of things,” *Journal of Intelligent Information Systems*, vol. 52, no. 2, pp. 285–309, 2019.

[9] G. van Capelleveen, C. Amrit, D. M. Yazan, and H. Zijm, “The recommender canvas: a model for developing and documenting recommender system design,” *Expert systems with applications*, vol. 129, pp. 97–117, 2019.

[10] C. A. Charu, *Recommender Systems: The Textbook*, 2016.

[11] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, “Collaborative filtering recommender systems,” in *The adaptive web*. Springer, 2007, pp. 291–324.

[12] Y. Wang, S. C.-F. Chan, and G. Ngai, “Applicability of demographic recommender system to tourist attractions: a case study on trip advisor,” in *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, vol. 3. IEEE, 2012, pp. 97–101.

[13] G. Adomavicius and A. Tuzhilin, “Context-aware recommender systems,” in *Recommender systems handbook*. Springer, 2011, pp. 217–253.

[14] G. Groh and C. Ehmig, “Recommendations in taste related domains: collaborative filtering vs. social filtering,” in *Proceedings of the 2007 international ACM conference on Supporting group work*, 2007, pp. 127–136.

[15] A. Poriya, T. Bhagat, N. Patel, and R. Sharma, “Non-personalized recommender systems and user-based collaborative recommender systems,” *Int. J. Appl. Inf. Syst*, vol. 6, no. 9, pp. 22–27, 2014.

[16] R. Burke, “Hybrid recommender systems: Survey and experiments,” *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[17] S. Khusro, Z. Ali, and I. Ullah, “Recommender systems: issues, challenges, and research opportunities,” in *Information Science and Applications (ICISA) 2016*. Springer, 2016, pp. 1179–1189.

[18] M. H. Mohamed, M. H. Khafagy, and M. H. Ibrahim, “Recommender systems challenges and solutions survey,” in *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*. IEEE, 2019, pp. 149–155.

[19] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, “Facing the cold start problem in recommender systems,” *Expert Systems with Applications*, vol. 41, no. 4, Part 2, pp. 2065–2073, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417413007240

[20] I. Esslimani, A. Brun, and A. Boyer, “Detecting leaders to alleviate latency in recommender systems,” in *International Conference on Electronic Commerce and Web Technologies*. Springer, 2010, pp. 229–240.

[21] M. Sollenborn and P. Funk, “Category-based filtering and user stereotype cases to reduce the latency problem in recommender systems,” in *Advances in Case-Based Reasoning*, S. Craw and A. Preece, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 395–405.

[22] I. Gunes, C. Kaleli, A. Bilge, and H. Polat, “Shilling attacks against recommender systems: a comprehensive survey,” *Artificial Intelligence Review*, vol. 42, no. 4, pp. 767–799, 2014.

[23] A. J. P. Jeckmans, M. Beye, Z. Erkin, P. Hartel, R. L. Lagendijk, and Q. Tang, *Privacy in Recommender Systems*. London: Springer London, 2013, pp. 263–281. [Online]. Available: https://doi.org/10.1007/978-1-4471-4555-4_12

[24] M. A. Ghazanfar and A. Prügel-Bennett, “Leveraging clustering approaches to solve the gray-sheep users problem in recommender systems,” *Expert Systems with Applications*, vol. 41, no. 7, pp. 3261–3275, 2014.

[25] A. Bergholz, “Coping with sparsity in a recommender system,” in *International Workshop on Mining Web Data for Discovering Usage Patterns and Profiles*. Springer, 2002, pp. 86–99.

[26] X. Su and T. M. Khoshgoftaar, “A survey of collaborative filtering techniques,” *Advances in artificial intelligence*, vol. 2009, 2009.

[27] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen, “Scalable collaborative filtering using cluster-based smoothing,” in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, 2005, pp. 114–121.

[28] M. Chen and P. Liu, “Performance evaluation of recommender systems.” *International Journal of Performability Engineering*, vol. 13, no. 8, 2017.

[29] G. Guizzardi, “Ontological foundations for structural conceptual models,” 2005.

[30] N. Guarino, D. Oberle, and S. Staab, *What Is an Ontology?* Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–17. [Online]. Available: https://doi.org/10.1007/978-3-540-92673-3_0

[31] T. R. Gruber, “Toward principles for the design of ontologies used for knowledge sharing?” *International journal of human-computer studies*, vol. 43, no. 5-6, pp. 907–928, 1995.

[32] M. R. Genesereth and N. J. Nilsson, *Logical foundations of artificial intelligence*. Morgan Kaufmann, 2012.

[33] A. Gómez-Pérez, M. Fernández-López, and O. Corcho, *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Science & Business Media, 2006.

[34] D. Vrandečić, *Ontology Evaluation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 293–313. [Online]. Available: https://doi.org/10.1007/978-3-540-92673-3_13

[35] M. Guia, R. R. Silva, and J. Bernardino, “A hybrid ontology-based recommendation system in e-commerce,” *Algorithms*, vol. 12, no. 11, p. 239, 2019.

[36] J. K. Tarus, Z. Niu, and G. Mustafa, “Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning,” *Artificial intelligence review*, vol. 50, no. 1, pp. 21–48, 2018.

[37] S. E. Middleton, D. De Roure, and N. R. Shadbolt, “Ontology-based recommender systems,” in *Handbook on ontologies*. Springer, 2004, pp. 477–498.

[38] L.-C. Chen, P.-J. Kuo, I.-E. Liao, and J.-Y. Huang, “Scaling out recommender system for digital libraries with mapreduce,” in *International Conference on Grid and Pervasive Computing*. Springer, 2013, pp. 40–47.

[39] L.-C. Chen, P.-J. Kuo, and I.-E. Liao, “Ontology-based library recommender system using mapreduce,” *Cluster Computing*, vol. 18, no. 1, pp. 113–121, 2015.

[40] M. Khobreh, F. Ansari, M. Dornhöfer, and M. Fathi, “An ontology-based recommender system to support nursing education and training.” in *Lwa*, 2013, pp. 237–244.

[41] M. H. Nassabi, H. op den Akker, and M. Vollenbroek-Hutten, “An ontology-based recommender system to promote physical activity for pre-frail elderly,” in *Mensch & Computer 2014–Workshopband*. De Gruyter Oldenbourg, 2014, pp. 181–184.

[42] M. Husáková and P. Cech, “Exploitation of ontology-based recommendation system with multi-agent simulations.” in *KEOD*, 2011, pp. 433–436.

[43] V. Espín, M. V. Hurtado, M. Noguera, and K. Benghazi, “Semantic-based recommendation of nutrition diets for the elderly from agroalimentary thesauri,” in *International Conference on Flexible Query Answering Systems*. Springer, 2013, pp. 471–482.

[44] K. Juszczyszyn, P. Kazienko, and K. Musiał, “Personalized ontology-based recommender systems for multimedia objects,” in *Agent and Multi-agent Technology for Internet and Enterprise Systems*. Springer, 2010, pp. 275–292.

[45] K. Makwana, J. Patel, and P. Shah, “An ontology based recommender system to mitigate the cold start problem in personalized web search,” in *International Conference on Information and Communication Technology for Intelligent Systems*. Springer, 2017, pp. 120–127.

[46] A. Hejja, R. A. Buchmann, and A. Szekely, “Integration of association rule detection with rule-based ontological support for product recommendation,” in *2011 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. IEEE, 2011, pp. 137–144.

[47] N. Tilipakis and C. Douligeris, “Efficient product choice through ontology-based recommender systems,” in *E-Services*. Springer, 2007, pp. 111–132.

[48] M. Mehrpoor, A. Gjærde, and O. I. Sivertsen, “Intelligent services: A semantic recommender system for knowledge representation in industry,” in *2014 International Conference on Engineering, Technology and Innovation (ICE)*. IEEE, 2014, pp. 1–6.

[49] H.-H. Huang, H.-C. Yang, and Y. Yu, “An ontology-based recommendation system for adas design,” in *International Conference on Genetic and Evolutionary Computing*. Springer, 2018, pp. 679–685.

[50] S. S. Keng, C. H. Su, G. L. Yu, and F. C. Fang, “Ak tourism: a property graph ontology-based tourism recommender system,” 2018.

[51] O. EL AISSAOUI and L. OUGHDIR, “A learning style-based ontology matching to enhance learning resources recommendation,” in *2020 1st International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*. IEEE, 2020, pp. 1–7.

[52] P. Sheridan, M. Onsjö, C. Becerra, S. Jimenez, and G. Dueñas, “An ontology-based recommender system with an application to the star trek television franchise,” *Future Internet*, vol. 11, no. 9, p. 182, 2019.

[53] Q. Nguyen, L. N. Huynh, T. P. Le, and T. Chung, “Ontology-based recommender system for sport events,” in *International Conference on Ubiquitous Information Management and Communication*. Springer, 2019, pp. 870–885.

[54] J.-M. Kim, H.-D. Yang, and H.-S. Chung, “Ontology-based recommender system of tv programmes for personalisation service in smart tv,” *International Journal of Web and Grid Services*, vol. 11, no. 3, pp. 283–302, 2015.

[55] T. H. Ly, S. T. Do, and T. T. S. Nguyen, “Ontology-based recommender system for the million song dataset challenge,” in *2018 10th International Conference on Knowledge and Systems Engineering (KSE)*. IEEE, 2018, pp. 236–241.

[56] N. S. Selvan, S. Vairavasundaram, and L. Ravi, “Fuzzy ontology-based personalized recommendation for internet of medical things with linked open data,” *Journal of Intelligent & Fuzzy Systems*, vol. 36, no. 5, pp. 4065–4075, 2019.

[57] Y. Terziev, M. Benner-Wickner, T. Brückmann, and V. Gruhn, “Ontology-based recommender system for information support in knowledge-intensive processes,” in *Proceedings of the 15th International Conference on Knowledge Technologies and Data-driven Business*, 2015, pp. 1–8.

[58] A.-n. Promkot, S. Arch-int, and N. Arch-int, “The personalized traditional medicine recommendation system using ontology and rule inference approach,” in *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*. IEEE, 2019, pp. 96–104.

[59] R. Karthik and S. Ganapathy, “A fuzzy recommendation system for predicting the customers interests using sentiment analysis and ontology in e-commerce,” *Applied Soft Computing*, vol. 108, p. 107396, 2021.

[60] S. Shafna and V. V. Rajendran, “Fuzzy ontology based recommender system with diversification mechanism,” in *2017 International Conference on Intelligent Computing and Control (I2C2)*. IEEE, 2017, pp. 1–6.

[61] M. Martínez-García, A. Valls, and A. Moreno, “Inferring preferences in ontology-based recommender systems using wowa,” *Journal of Intelligent Information Systems*, vol. 52, no. 2, pp. 393–423, 2019.

[62] F. Osborne and E. Motta, “Inferring semantic relations by user feedback,” in *International Conference on Knowledge Engineering and Knowledge Management*. Springer, 2014, pp. 339–355.

[63] P. Selvaraj, V. K. Burugari, D. Sumathi, R. K. Nayak, and R. Tripathy, “Ontology based recommendation system for domain specific seekers,” in *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE, 2019, pp. 341–345.

[64] F. P. Romero, M. Ferreira-Satler, J. A. Olivas, and J. Serrano-Guerrero, “An ontology-based recommender system for health information management,” in *Proceedings on the International Conference on Artificial Intelligence (ICAI)*. The Steering Committee of The World Congress in Computer Science, Computer …, 2012, p. 1.

[65] A. Chougule, V. K. Jha, and D. Mukhopadhyay, “Crop suitability and fertilizers recommendation using data mining techniques,” in *Progress in Advanced Computing and Intelligent Engineering*. Springer, 2019, pp. 205–213.

[66] H. Osman, N. El-Bendary, E. El Fakharany, and M. El Emam, “Ontology based recommendation system for predicting cultivation and harvesting timings using support vector regression,” in *Proceedings of the Computational Methods in Systems and Software*. Springer, 2020, pp. 813–824.

[67] M. Arafeh, P. Ceravolo, A. Mourad, E. Damiani, and E. Bellini, “Ontology based recommender system using social network data,” *Future Generation Computer Systems*, vol. 115, pp. 769–779, 2021.

[68] N. Yanes, S. B. Sassi, and H. H. B. Ghezala, “Ontology-based recommender system for cots components,” *Journal of Systems and Software*, vol. 132, pp. 283–297, 2017.

[69] F. A. de Paiva, J. A. F. Costa, and C. R. Silva, “An ontology-based recommender system architecture for semantic searches in vehicles sales portals,” in *International Conference on Hybrid Artificial Intelligence Systems*. Springer, 2014, pp. 537–548.

[70] J. Kang and J. Choi, “An ontology-based recommendation system using long-term and short-term preferences,” in *2011 International Conference on Information Science and Applications*. IEEE, 2011, pp. 1–8.

[71] N. Thotharat, “Thai local product recommendation using ontological content based filtering,” in *2017 9th International Conference on Knowledge and Smart Technology (KST)*. IEEE, 2017, pp. 45–49.

[72] R. F. Silva, P. Carvalho, S. Rito Lima, L. Álvarez Sabucedo, J. M. Santos Gago, and J. M. C. Silva, “An ontology-based recommendation system for context-aware network monitoring,” in *World Conference on Information Systems and Technologies*. Springer, 2019, pp. 373–384.

[73] J. Lehmann, M. Shamiyeh, and S. Ziemer, “Challenges of modeling and evaluating the semantics of technical content deployed in recommendation systems for industry 4.0.” in *KEOD*, 2019, pp. 359–366.

[74] Y.-H. Chen, T.-h. Huang, D. C. Hsu, and J. Y.-j. Hsu, “Colorcocktail: an ontology-based recommender system,” *Proceedings of 20th American Association for Artificial Intelligence. Menlo Park, USA: AAAI Press*, pp. 79–82, 2006.

[75] S. Shishehchi and S. Y. Banihashem, “Jrdp: a job recommender system based on ontology for disabled people,” *International Journal of Technology and Human Interaction (IJTHI)*, vol. 15, no. 1, pp. 85–99, 2019.

[76] S. Shishehchi, N. A. M. Zin, and E. A. A. Seman, “Ontology-based recommender system for a learning sequence in programming languages.” *International Journal of Emerging Technologies in Learning*, vol. 16, no. 12, 2021.

[77] T. Thanapalasingam, F. Osborne, A. Birukou, and E. Motta, “Ontology-based recommendation of editorial products,” in *International Semantic Web Conference*. Springer, 2018, pp. 341–358.

[78] V. Paradigm, “The 1 development tool suite,” 1999. [Online]. Available: https://www.visual-paradigm.com/

[79] “Compiler.” [Online]. Available: https://dictionary.cambridge.org/dictionary/english/compiler

[80] R. Ranjan, “What is a framework in programming amp; why you should use one,” Feb 2022. [Online]. Available: https://www.netsolutions.com/insights/what-is-a-framework-in-programming/#:~:text=A%20framework%20in%20programming%20is,inversion%20of%20control%20(IoC).

[81] “Programming language definition amp; meaning.” [Online]. Available: https://www.merriam-webster.com/dictionary/programming%20language

[82] “Iso 25000 portal.” [Online]. Available: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

[83] “Complexity.” [Online]. Available: https://dictionary.cambridge.org/dictionary/english/complexity?q=Complexity

[84] *The standard for Project Management and A guide to the Project Management Body of Knowledge (PMBOK guide)*. Project Management Institute, Inc., 2021.

[85] “Employment,” Jul 2022. [Online]. Available: https://en.wikipedia.org/wiki/Employment

[86] “Experience.” [Online]. Available: https://dictionary.cambridge.org/dictionary/english/experience

[87] “Junior.” [Online]. Available: https://dictionary.cambridge.org/dictionary/english/junior

[88] “Intermediate.” [Online]. Available: https://dictionary.cambridge.org/dictionary/english/intermediate

[89] “Senior.” [Online]. Available: https://dictionary.cambridge.org/dictionary/english/senior

[90] “Knowledge.” [Online]. Available: https://dictionary.cambridge.org/dictionary/english/knowledge

[91] “8 business layer.” [Online]. Available: https://pubs.opengroup.org/architecture/archimate3-doc/chap08.html#_Toc10045369

[92] D. Leffingwell, “Scrum master,” Dec 2021. [Online]. Available: https://www.scaledagileframework.com/scrum-master/

[93] “Software engineer.” [Online]. Available: https://dictionary.cambridge.org/dictionary/english/software-engineer

[94] “Tester.” [Online]. Available: https://dictionary.cambridge.org/dictionary/english/tester

[95] R. Knaster, “Product owner,” Feb 2022. [Online]. Available: https://www.scaledagileframework.com/product-owner/#:~:text=The%20Product%20Owner%20(PO)%20is,or%20components%20for%20the%20team.

[96] E. Hartzstein, “Functional and technical administration roles in solution manager.” [Online]. Available: https://answers.sap.com/questions/6895218/functional-and-technical-administration-roles-in-s.html

[97] "Software development effort estimation," Jul 2022. [Online]. Available: https://en.wikipedia.org/wiki/Software_development_effort_estimation

[98] R. Lynn, "What is velocity in agile?" May 2021. [Online]. Available: https://www.planview.com/resources/articles/lkdc-velocity-agile/#:~:text=Velocity%20in%20Agile%20is%20a,iterations%2C%20sprints%2C%20or%20weeks.

[99] "Software architecture," Jul 2022. [Online]. Available: https://en.wikipedia.org/wiki/Software_architecture#:~:text=Architectural%20patterns%20are%20often%20documented,notable%22%20(architectural%20style).

[100] "Soa vs. microservices: What's the difference?" [Online]. Available: https://www.ibm.com/cloud/blog/soa-vs-microservices

[101] "Monolithic application," Feb 2022. [Online]. Available: https://en.wikipedia.org/wiki/Monolithic_application

[102] "Software development process," Jul 2022. [Online]. Available: https://en.wikipedia.org/wiki/Software_development_process

[103] D. Radigan, "Agile vs. waterfall project management." [Online]. Available: https://www.atlassian.com/agile/project-management/project-management-intro

[104] "Front end." [Online]. Available: https://dictionary.cambridge.org/dictionary/english/front-end

[105] "Back end." [Online]. Available: https://dictionary.cambridge.org/dictionary/english/back-end

[106] Wikipedia contributors, "Computing platform — Wikipedia, the free encyclopedia," 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Computing_platform&oldid=1096664884

[107] "Mobile." [Online]. Available: https://dictionary.cambridge.org/dictionary/english/mobile

[108] M. Rehkopf, "Epics." [Online]. Available: https://www.atlassian.com/agile/project-management/epics

[109] ——, "User stories: Examples and template." [Online]. Available: https://www.atlassian.com/agile/project-management/user-stories

[110] G. Guizzardi, R. de Almeida Falbo, and R. S. Guizzardi, "Grounding software domain ontologies in the unified foundational ontology (ufo): the case of the ode software process ontology." in *CIbSE*. Citeseer, 2008, pp. 127–140.

[111] C. M. Fonseca, T. P. Sales, L. Bassetti, and V. Viola, "Ontouml/ontouml-vp-plugin." [Online]. Available: https://github.com/OntoUML/ontouml-vp-plugin

[112] "Low-code application development platform," Aug 2022. [Online]. Available: https://www.mendix.com/

[113] J. Sequeda, "The business value of reasoning with ontologies," Jan 2015. [Online]. Available: https://www.dataversity.net/the-business-value-of-reasoning-with-ontologies/

[114] J. Gordijn, "E-business value modelling using the e3-value ontology," in *Value creation from e-business models*. Elsevier, 2004, pp. 98–127.

[115] A. R. Gilal, J. Jaafar, M. Omar, S. Basri, and I. A. Aziz, "Balancing the personality of programmer: Software development team composition," *Malaysian Journal of Computer Science*, vol. 29, no. 2, pp. 145–155, 2016.

[116] A. R. Gilal, J. Jaafar, L. F. Capretz, M. Omar, S. Basri, and I. A. Aziz, "Finding an effective classification technique to develop a software team composition model," *Journal of Software: Evolution and Process*, vol. 30, no. 1, p. e1920, 2018.

A JUSTID INTERVIEW TRANSCRIPT

Marc (M): The interview is going to be in Dutch. There will be a couple of themes we will go through which are largely based on the defined requirements, for quality attributes, the project requirements, and the intended use. After that, there will be some more overview questions. But I like to start by asking you to shortly address your background and your role within the company.

Department Manager (DM): As department manager of the department "Portalen en samenwerking" in Justid. As department manager, I am also included in the decision-making process regarding IT projects. Typically, with a new project, we look at a given technology stack that is familiar. We never ask ourselves the question is it accurately and objectively chosen.

M: And how long are you employed for Justid?

DM: I am working for Justid for 7 years, I began as a team leader and eventually became a department manager. The department consists of three teams that all work in an agile context with sprints of three weeks and with project increments of three sprints.

M: What is your previous work experience?

DM: Before I worked as a team leader for another company for three years, and before that in the secondment as, among others, a software tester. Before that, I completed an HBO ICT study.

M: And what is your experience up to now?

Software Developer (SD): I am working as a software developer for Justid for 2 years, before that, I completed a Masters. During my study a had multiple side jobs as programmer.

M: And for how long have you done that?

SD: 3 years for one company and about a year for another one, so in total (including working at Justid) about 6 to 7 years of experience.

M: Then we will continue with the topic of quality attributes. As can be seen, the requirements are: "The ontology is understandable/readable", "The ontology can be changed/adapted (maintainability / future improvements)", and "The ontology remains usable in the future". How well are these requirements met?

DM: I certainly agree with the statement "The ontology is understandable/readable", I also think it is good to see how it was created, as sort of a voyage of discovery. After interviewing us and gathering the information, the creation of this ontology came to maturity. I think there are some possibilities to add more detail, but at its core, all three models are very understandable and reflect the way we work. I like to hear from *SD* if this is true. At most, we are not familiar with the tooling, but the way it is modeled with the flexibility of different sub-models, objects, and building blocks, I certainly think it is maintainable, adaptable, and usable for us as an organization.

SD: I agree.

DM: With regards to the usage in the future, I think this is done by using the enumerations. So, we can change those based on the current insights. For example, if in the future it turns out we want more types of architectural styles or development methods, or maybe it turns out we do not need a mobile platform type, then we can add or remove that. So, I think that this can be adapted to modern architectural principles.

M: And from the development side what is your point of view, are there differences?

SD: What I think is well represented is that the employees and the roles are separated. This is also how we work, one day I can have another role than the other day, for example, one day I might go test something and the other day do something else.

DM: I like to add that I think it is good that you did not only look at the technical part of the software development process but also at the human part of the process. So how a team is formed and how does a team, with respect to proficiency level and experience level, function. Because when you look at a random IT project this becomes more and more relevant. How do the people feel, how can they improve themselves, what kind of things do they work on, and where can they be used for? This is really important for the success of an IT project.

SD: I do want to note that we also look quite often at the experience level of an employee in a certain framework, and we now represent the general experience level and the proficiency in a certain language.

M: That is indeed something we could have added.

M: With regards to the next topic, the project requirements. You are both not that familiar with the techniques used. But we used standard languages and tooling to create the ontology. I will shortly show you what this looks like. Files are being shown and explained

M: With regards to the documentation, this will be my research thesis, which I shortly explained before the interview. Do you have any questions with regards to the explanation?

SD: When I want to use this tool myself and there is something that I can not figure out, is there something within your thesis that explains how I should use this tool?

M: That is not in the thesis, although since it is a tool that is widely used there is a lot of documentation about it. But I could also have a look at making some documentation with some explanation and with the right pointers towards how to use it. There is for example also a complete tutorial for Protégé.

SD + DM: That would be great

M: With regards to the ontology being useable within the standards of the organization, we tried to do this by for example using your own definitions of proficiency level. How well is that requirement met, what points towards it fitting within the standards of the organization, and are there things that are not within the standard but should have been?

DM: This organization, for a large part, is based on the principles of the agile world. Take for example the roles, these are based on the agile principles but might have had a different name within our organization if this was not the case. So, you partly have the context of Justid, very well, and is well represented, also from an HR perspective, but is also good to see that the agile world is well represented, this is a good combination.

M: Oké great to hear, are there any things missing or not within the standards of the organization?

SD: Within the roles, there is for example the Scrum Master, and this is within the method we use very common, but when looking at other agile methods, like Kanban, then this is not a Scrum Master but an Agile coach.

M: Yes, these terms could indeed be added/changed.

SD: But, as said, the differences between Justid and the agile industry are not that big

M: When showing the models, you already said it yourself that the statement “the ontology reflects the way the company sees Software Project Management” is pretty much met.

DM: Yes, however, it is interesting to note that there is a difference, with regards to the way we work, between the viewpoint of the company and the viewpoint of the department manager. But if I look at it, for the major part the development part is well represented but the project management part is not that represented. Take for example the project manager self, the HR-like parts, maybe something from PRINCE2, that is not in there yet. But I do not mind it at all, because this should be focused on the software development process.

M: So to conclude, the management side is not yet complete, but the development side is.

DM: Yes.

M: And what is your point of view with regards to both the development and management side?

SD: For development, it is well done. And for the management side, I do not have any insights into this.

M: That makes sense, So the better statement would have been “the ontology reflects the way the company sees Software Development”?

DM: Yes, because if you wanted to answer the first question enumerations with regards to financials, scope, etc. could be added. They are not in there now, which has been done on purpose, because we did not want it.

M: I showed you the queries, do you think some queries were missing?

DM: I think, given the scope of this research, I do not miss a query. As the ontology expert said, it maybe is indeed very detailed. However, I think that is fine because it really shows what is possible with this model. But going back to the previous question, when including the project management side and corresponding variables, more interesting queries could be built giving it even more business value than it already has. But that is the only thing, and also that is part of a next step, I do not mind it not being in there.

M: So adding the project management side would give more business value.

M: Do you see this ontology being used? How and why?

SD: Yes, the ontology could be used. We work with skill matrixes, and this could replace the skill matrix. This could give us more insight into the capabilities of the team and what the project entails, and the possible mismatch.

So, there are two approaches I see. The first one is in a current project, you can look into what is the team composition, what is needed, what kind of project is it, which techniques are being used, and if needed where can we improve, need training, and/or share knowledge. The second one is for a new project, to look at what the team composition is and what the project entails and use this to recommend a programming language.

M: So the first one is to improve a current project and the other one is to recommend a programming environment for a new project.

SD: Yes.

M: Which of these two do you think has the most value?

SD: I think the first one since it is the most relevant for me.

M: And, next to relevant is there another reason?

SD: I think it is hard to say for me since I only worked on already existing projects.

DM: I agree with the viewpoint of *SD*, but I think it can really contribute to the decision-making for new IT projects within Justid. Currently, we just say “yes we can do this because we have these technologies and we can do this and this what that”. But there is never an objective view of it. This ontology can help us make a more objective choice when starting a new IT project.

M: So from the developers’ side, the improvement of current projects is more relevant and from the department manager’s side, the recommendation and objective view for new IT projects is more relevant?

DM: Yes indeed, and I also have a practical example. We start with a quotation process, and then the question from the customer is discussed, and then we say “we are going to make it like this”. However, this is only based on the technologies we currently use, so then the quotation is also based on these technologies. Later in the process, the architects tell us that we should have done a different approach because there are new technical insights. But because the quotation is already done, and a scope, timeframe, and cost are decided upon, nobody is going to want to add this since it will cost more. So then you are making a project that has already legacy in it. So I think this ontology, can be part of the solution to help mitigate this problem.

DM: And then another direction, HR, the experience level and proficiency level representation can also help with making a training and improvement plan.

SD: I have some question marks with this since we only use a subset of languages and because the recommendation is based on, among others, the experience and proficiency levels it might point towards the same languages since those languages will kind of fit and we have experience in them.

DM: That is a fair point, but on the other hand, I think if you use the tool right, you can see an employee is on a certain level, would it not be useful to improve the level of this employee. So not as a result of the tool, but as an aid in making the decision

M: And to address the point made by *SD*, under the assumption that a recommender system will be built around this ontology, this recommender system can be built in such a way that new items will be recommended as well, the so-called serendipity. It works the same as with Netflix, if you typically watch action movies, once in a while Netflix will recommend a comedy to check if you are interested in that as well. This also makes the development of such a recommender system a next step and outside of the scope of the current research. It can also be possible to have some sort of dashboard that says, from the languages the teams typically use this would be the most fitting one, but you could also have a look into this other language if time permits.

DM: I think that it is a good note this in your research, as it is a good addition to the use of the ontology.

M: Then the last question “Which directions do you suggest for the continuation of this research/project?”. We already discussed the recommender system but are there any others.

DM: Indeed, the recommender system is for sure one of the continuations.

SD: I think it is very nice that it is possible to add serendipity in the recommendation as previously mentioned.

DM: I think it would be of added value to enrich this model with insights other than Justid.

DM: I think in the end the goal is to create a system in which all variables can be put in and a recommendation, given the context, will be given from this system.

M: We thought of something ourselves, namely, imagine a customer that comes to you that specifically wants something done in a certain language, for example, JavaScript. We thought that if this is the case, we have information about the project and information about the language that needs to be used, then it would be possible to recommend a team, i.e., recommend a team composition. What is your view on this approach?

SD: You of course have to do with the availability of each of the members. So maybe just give a long list of possible members to add

M: We have the attribute availability added to the employee.

SD: But that is the number of hours an employee works per week, right?

M: That is true, although there is of course the possibility to add another attribute that represents the hours that an employee can still be put onto a project. And given the fact that project management is incorporated a bit more into the ontology, you could start talking about terms and time management and therefore recommend a team, although just a list is of course also possible. You can then also add axioms that, for example, look if there are not only juniors in the team but also a senior, etc. This, of course, is not yet done in this ontology because the goal of this ontology is to recommend a language.

DM: It is possible, but then the risk rises that the nice goal for which this is built will be degraded to a sort of competence management tool. That only a list will be given of just some employees that could do this. This might miss the purpose of the tool.

SD: Something we discussed early, is the possibility to connect the backlog of the sprint onto the system so that experience levels can automatically be updated given the completed tasks.

M: That is indeed something that might be possible for future work.

SD: And is there something representing the certainty of the requirements of the project?

M: How do you mean certainty?

SD: Well, when having a project of our own the requirements will be very certain, i.e. will not change, that much, over time, so the waterfall method could be used for this. But if it is for a client the requirements might change.

M: I understand. So these are not the non-functional requirements, but more what kind of features do the clients want.

SD: Yes.

M: We do not have this in the ontology, although, a project in this ontology is a program increment of 3 sprints. It is most likely that not that much will change in the feature requirements within those 3 sprints. Later than these 3 sprints feature requirements might change, but then it is classified as a new project, so this means that for each of the projects all the feature requirements are quite certain.

DM: Yes, but on the other hand, it would be nice to have it incorporated anyway.

M: I agree that could be a possible improvement, but it will not be a 'showstopper'

DM: I agree

M: Can you think of some limitations of this ontology?

DM+SD: Can not think of any right now.

M: Then I want to thank you both for your time.