

Using Human Computation for Ontology evaluation

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Software Engineering und Internet Computing

eingereicht von

Miruna Orsa, BSc

Matrikelnummer 01227248

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Projektass.(FWF) Reka Marta Sabou, MSc PhD

Mitwirkung: Projektass.in Dipl.-Ing.in Stefani Tsaneva, BSc

Wien, 1. Oktober 2023



Miruna Orsa



Reka Marta Sabou

Using Human Computation for Ontology evaluation

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Software Engineering and Internet Computing

by

Miruna Orsa, BSc

Registration Number 01227248

to the Faculty of Informatics

at the TU Wien

Advisor: Projektass.(FWF) Reka Marta Sabou, MSc PhD

Assistance: Projektass.in Dipl.-Ing.in Stefani Tsaneva, BSc

Vienna, 1st October, 2023



Miruna Orsa




Reka Marta Sabou

Erklärung zur Verfassung der Arbeit

Miruna Orsa, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Oktober 2023


Miruna Orsa

Acknowledgements

First of all I want to express my gratitude to my supervisor Marta Sabou and her colleague Stefani Tsaneva for providing me information, help, support and feedback through the whole process of writing my thesis. I learned a lot through their guidance and am proud to have chosen this field of study and research team to complete my Masters Degree.

A big thank you to all my friends and colleagues that I met during my study and work, which were always supportive and there for me with a good word.

I want to thank my family, too, especially my father Gratiean, who always believed in me and hold me up even when I thought I wouldn't get through with it. I owe him a big part of what I achieve today. A last thank you goes to Lukas, which was there for me with every step I took through the studying journey.

Kurzfassung

Anwendungen benutzen Ontologien als Wissensbasis. Da falsch dargestellte Informationen zu falschen Ergebnissen führen können, kann die Qualität einer Ontologie ein entscheidender Faktor für den Erfolg der Anwendung sein. Weil der Entwicklungsprozess einer Ontologie fehleranfällig ist, ist eine Ontologie-Evaluierung notwendig. Es gibt zwar automatisierte Ansätze dafür, jedoch gibt es Fehler, die Hintergrundinformationen und menschliches Wissen erfordern. Hier können Human Computation und Crowdsourcing eingesetzt werden, sodass die Wissensbasis für die Ontologie erweitert und validiert werden kann.

In der Literatur werden häufige Fehler vorgestellt, die den Entwicklungsprozess betreffen. Jedoch fehlt eine solche Analyse bezüglich Ontologien, die von Anfängern entwickelt werden. Darüber hinaus besteht der Bedarf an einer Human Computation Methodik zur Lösung von Entwicklungsfehler. Außerdem ist es unklar, inwieweit die Gesamtleistung der Mitarbeiter beeinflusst wird, wenn mehrere Fehlertypen in einem Human Computation Task vorkommen und wenn die Vielfalt der Fehler von drei auf fünf ansteigt.

Diese Arbeit sammelt zunächst Informationen über häufige Fehler durch eine *Literaturrecherche* und nutzt einen *Datenvergleich* zwischen Anfängerontologien, um praktische Informationen herauszufinden. Um eine Human Computation Methodik zur Lösung von Problemen bei der Entwicklung von Ontologien zu finden, wurde der bestehende Ansatz *VeriCoM* um einen Ontolgien-Anwendungsfall erweitert. Schließlich wird in dieser Arbeit gezeigt, wie sich die Leistung der Mitarbeiter bei der Verifizierung von Fehlern verändert, wenn die Fehlertypen und die Vielfalt der Fehler in einem Human Computation Task zunehmen. Dies erfolgt durch ein Experiment, das den Prinzipien von *Designing the experimental process* folgt.

Die häufigsten Fehler in Anfängerontologien betreffen: Lesbarkeit, Unvereinbarkeit von Klassen, Nichtdeklaration inverser Beziehungen und Verwechslung von logischem "und" und "oder" (i). Darüber hinaus zeigt der VeriCoM-Ansatz, dass die durchschnittliche Leistung der Mitarbeiter bei 78% liegt, während die durchschnittliche Geschwindigkeit bei der Erledigung einer Aufgabe bei 55 Sekunden liegt. Dies beweist, dass der Ansatz eine hohe Leistung bei der Verifizierung spezifischer Fehler in Ontologien erbringt (ii). Bezüglich des Experiments, bei einer Erhöhung der Fehlertypen und der Anzahl an Fehlern pro Task, im Vergleich zu einem bestehen Experiment, sinkt die tatsächliche Leistung von 92,58% auf 78%, während die Antwortzeit im Durchschnitt bei etwas

weniger als einer Minute blieb. Trotzdem ist die Gesamtleistung hoch und hat sich nicht drastisch verändert (nicht mehr als 30%). Dies zeigt, dass Human Computation immer noch zuverlässig ist, selbst wenn Ontologien unterschiedliche Fehler enthalten, und dass es sich um einen praktikablen Ansatz für die Evaluierung der Ontologien im Allgemein handelt (iii).

Abstract

Applications rely on ontologies as knowledge bases and as wrongly represented information can lead to false outcomes, the quality of an ontology can be a deciding factor for the success of the system using it. Because the process of ontology engineering is liable to errors, the need for ontology evaluation arises. While automated approaches exist, there are still errors which need background information and human knowledge. For such cases, Human Computation and Crowdsourcing can be applied, such that with the help of crowds, the knowledge base for the ontology can be enhanced and the existing one validated.

While common errors are introduced in the literature, an analysis of typical engineering errors in beginners' ontologies in practice is missing. Moreover, the need for a methodology to solve problems regarding ontology engineering with Human Computation arises and it is not clear, while considering error classification, to what extent is the overall performance influenced, when having multiple error types in one Human Computation task and when the variety of errors increases from three to five.

This thesis firstly collects information regarding common errors through a *literature research* and uses *data comparison* between beginners' ontologies to find out practical information. Concerning finding a methodology to solve problems regarding ontology engineering with Human Computation, an existing approach called *VeriCoM* was extended to an Ontology Engineering use case. Lastly, the thesis shows how the performance of workers regarding verification of ontology engineering errors changes, when increasing the error types and the variety of errors in an Human Computation task, through an experiment that follows principles of *designing the experimental process*.

Based on the results, we conclude that: most common errors in beginners' ontologies are: readability, disjointness of classes, not declaring inverse relationships and the confusion between logical "and" and "or" (i). Moreover, the VeriCoM approach shows that the average performance of the workers is at 78% while the average speed of completing a task is at 55 seconds, proving that the approach is able to achieve high performance regarding the verification of specific defects in ontologies (ii). Comparing a previous similar experiment with this thesis experiment, that has an increased number of error types of five, but also tasks with multiple errors, respectively, the actual performance decreased from 92.58% to 78%, while the response time remains on average at slightly less than one minute. Even though the performance reduced by 14.58%, the overall

performance is still high and did not changed drastically. This goes to show that Human Computation is still reliable, even when the ontologies to be verified contain multiple and different errors, and is a viable approach for Ontology verification in general (iii).

Contents

Kurzfassung	ix
Abstract	xi
Contents	xiii
1 Introduction	1
1.1 Problem Statement	2
1.2 Research Questions	3
1.3 Methodological approach	3
1.4 Thesis Structure	5
2 Background and related work	7
2.1 Ontology evaluation	7
2.2 Common errors in Ontology Engineering	8
2.3 Human Computation and Crowdsourcing	9
3 Practical comparison of beginners' ontologies	13
3.1 Collection of data	13
3.2 Methodology and results	14
4 Using Human Computation for Ontology evaluation: The VeriCoM Approach	21
4.1 Addressed ontology modelling errors	21
4.2 The VeriCoM Approach	28
4.3 Applying the VeriCoM Approach on an Ontology Engineering Use Case	30
4.4 VeriCoM Summary	32
5 Ontology evaluation experiment	37
5.1 Experiment scoping	38
5.2 Experiment planning	39
5.3 Experiment operation	46
5.4 Experiment analysis and interpretation	49
5.5 Experiment presentation and package	53
	xiii

5.6 Experiment summary	53
6 Conclusion	55
6.1 Research Questions Discussion	56
6.2 Limitations and Future work	57
List of Figures	59
List of Tables	61
Bibliography	63
Appendices	67
Appendix A: Self Assessment Test	68
Appendix B: Qualification Test	72
Appendix C: Post-study Questionnaire	78

CHAPTER 1

Introduction

The concept of *ontology* in Computer Science emerged from the Knowledge Engineering field [12]. There are a lot of definitions for ontologies, but the original and most cited one in the literature belongs to Gruber that defined it as an "*explicit specification of a conceptualization*" [12]. Ontologies are therefore used as conceptual models and knowledge representations and are key concepts in the Semantic Web [5].

Since a lot of applications rely on ontologies as knowledge bases, one of the most important requirement for them is the quality. Many ontologies reuse large-scale existing ones, because of resources issues such as time, costs, or knowledge [6]. Therefore ensuring that quality errors don't propagate in them is crucial. As wrongly represented information can lead to false outcomes of applications, the quality of an ontology can be a deciding factor for the success of the system using it.

From the literature, it is known that ontology engineering is error-prone and that ontologies typically exhibit a number of quality issues. Several authors [19], [23], [28] have identified common errors and pitfalls regarding the quality of ontologies. Some of the most frequent ones are missing annotations, missing domain or range properties, missing disjointness or usage of recursive definitions, but also incorrect use of universal and existential restrictions or confusion between logical and linguistic "and".

Because the process of ontology engineering is liable to errors, the need for *ontology evaluation* arises, which deals with checking the technical quality of an ontology against a frame of reference.

In order to evaluate ontologies and detect the possible errors, automated approaches exist. For example, the OOPS tool developed in [19] can detect up to 33 pitfalls regarding ontology engineering out of a catalogue of 41. For instance, the tool covers the detection of missing annotations, relationships not explicitly declared, missing domain or range properties, etc. Nevertheless, there are still errors which cannot be identified

automatically, because they need background information and human knowledge: wrong data types, missing requirements, irrelevant information, overspecialisation etc.

For cases where the automated approaches are not sufficient, *Human Computation* and *Crowdsourcing* can be applied. Human Computation is constituted by "the problems that fit the general paradigm of computation, and as such might someday be solvable by computers" together with "the human participation, directed by the computational system or process" [22]. While Human Computation does not speak about the number of humans needed to solve a problem, Crowdsourcing is an idea made popular by Jeff Howe where a high number of people can solve tasks that cannot yet be fully automated, online [1]. The connection between the Semantic Web and the field of Human Computation has been synthesized and discussed in [24], where it is explained that the two fields are often used together, even though it is stressed that much more research is done in showing how Human Computation is used for Ontology Engineering and Verification, and not so much the other way around. With the help of crowds, the knowledge base for the ontology can be enhanced, but also existing knowledge can be validated. The increasing of knowledge depends on the task and workflow design, on the genre of the crowd, but also on the usage of hybrid workflows that combine algorithmic and human computation. Therefore one can look into specific ontology engineering problems and how they can be resolved through Human Computation and Crowdsourcing.

1.1 Problem Statement

In the literature there is a variety of errors that can occur when engineering an ontology. While most of them result from empirical analysis, there is still a need to identify what are the most common ones when considering beginners' ontologies, with limited experience in the field, in order to be able to concentrate on these types of errors when needed.

Performing expert evaluations to verify the quality of ontologies is expensive, time-consuming, and has limited scalability. Human Computing and Crowdsourcing have already been successfully utilized for verification tasks in the software engineering domain [24], but there is not yet a defined and established methodology to solve problems regarding ontology engineering with Human Computation.

As for the ontology engineering problems, the following errors regarding incorrect modelling are identified:

- D1: missing existential restriction
- D2: universal restriction instead of existential restriction
- D3: missing universal restriction
- D4: missing disjointness
- D5: confusion between linguistic and logical "and" (intersection versus union)

While the first three engineering problems have already been tackled using Human Computation in [27], an experimental investigation is still missing of ways how Human Computation can be used to address D4 and D5. Moreover, it is not clear how the enhancement of the set of problems will influence the performance of the crowd.

1.2 Research Questions

In order to contribute to the Human Computation and Ontology Evaluation fields of research, this thesis aims to answer the following research questions:

1. **RQ1:** What are typical engineering errors in beginners' ontologies in practise, compared to the common errors introduced in the literature?

While the literature describes a variety of engineering errors regarding the ontology, beginners' ontologies are evaluated in a comparison, in order to acquire information about common novices' ontology engineering errors in practice.

2. **RQ2:** What is a good methodology to solve problems regarding ontology engineering with Human Computation?

One aim of this thesis is to investigate how incorrect modelling problems can be solved with Human Computation. For this, a proper approach has to be chosen. In [25], the authors develop an approach that would solve the problem of verifying conceptual domain models with Human Computation. The generic approach that is developed is called Verifying Conceptual Models (VeriCoM) and it was applied in a software engineering use case, focusing on the verification of an EER diagram, based on the system specification document. In this thesis the approach will be adapted to ontology engineering.

3. **RQ3:** Considering error classification, to what extent is the overall performance influenced, when having multiple error types in one Human Computation task and when the variety of errors increases from three to five?

The last part of the thesis focuses on the design of a Human Computation task for solving the ontology defects D1-D5. Building on top of [27], the variety of errors increases from three to five and thus the Human Computation task should support the detection of multiple error types at once. This thesis aims to investigate the influence on the performance of the crowd when the data item to be verified includes a single defect against the case when it includes multiple defects.

1.3 Methodological approach

Firstly, as it can be observed in Figure 1.1, this thesis shows what typical engineering errors in beginners' ontologies are, considering practical data. For gathering the necessary information about the topics of Ontology engineering and evaluation, Human Computation

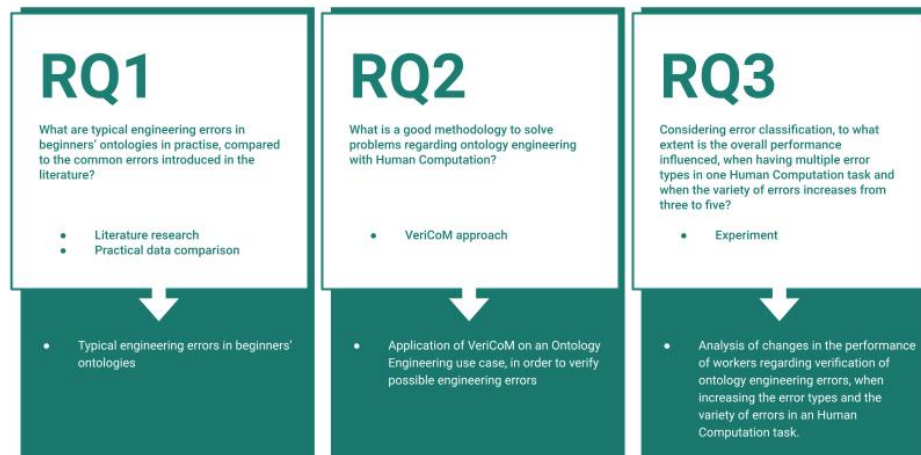


Figure 1.1: Research questions, methodologies used and contributions

and Crowdsourcing, **literature research** is the first methodological approach. In order to find out which of the common errors of the literature are present in practice, a **data comparison** between beginner's ontologies is going to be performed to answer RQ1.

The second part of Figure 1.1 concerns RQ2. While initial research was done in this field, the thesis follows up with implementing a Human Computation and Crowdsourcing approach called **VeriCoM** on an Ontology Engineering use case, in order to verify possible engineering errors.

Lastly, the thesis shows how the performance of workers regarding verification of ontology engineering errors changes, when increasing the error types and the variety of errors in an Human Computation task, as per Figure 1.1.

Therefore an experiment is conducted where Crowdsourcing is applied. This experiment is an optional part of the Vienna University of Technology (TU Wien) course 188.399 Introduction to Semantic Systems. For preparing the experiment, one of the important methodologies is **designing the experimental process**. This will follow the recommended steps in [29], as shown in Figure 5.1:

- *Scoping*: deciding on the goals and purpose of the study, settling the performance of the subject as the studied effect and deciding on the context of running the experiment as part of the Vienna University of Technology (TU Wien) course 188.399 Introduction to Semantic Systems.

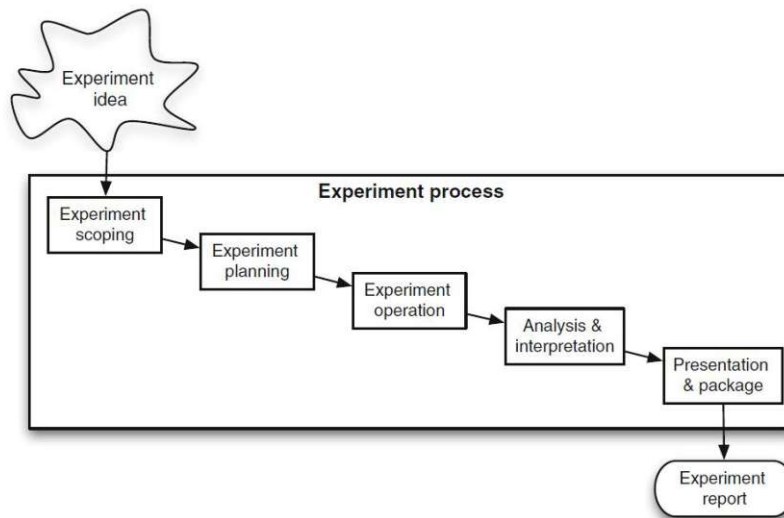


Figure 1.2: Overview of the experiment process [29]

- *Planning*: determining which ontology is used as initial data: either a well-known, correct and easy for beginners ontology, where the errors that are going to be considered in the experiment have to be manually planted into, or a beginner's ontology which already has all the considered errors, but for which a golden standard has to be made. The tools and platforms for the Human Computation tasks are chosen and the tasks are designed. The participants are chosen between a layman crowd and a group of internal students with a level of knowledge in the domain.
- *Operation*: the execution of the experiment, supported by a strategy of communication with the students.
- *Analysis and interpretation*: the collected data is going to be further analyzed in order to answer RQ3, by using qualitative and quantitative methods to synthesise the results. The goal of the qualitative evaluation is to analyse the experimental process while the quantitative methods focus on the results of the students taking part in the experiment.
- *Presentation and package*: the results are presented in this thesis.

1.4 Thesis Structure

The thesis is structured in the following chapters:

- Chapter 2 Background and related work focuses on the state of the art in the two main research areas this thesis concentrates on: Ontology Evaluation and errors

that appear in the process of ontology engineering, and Human Computation and Crowdsourcing.

- Chapter 3 [Practical comparison of beginners' ontologies](#) presents the comparison of beginners' ontologies, in order to acquire information about common novices' ontology engineering errors in practice.
- Chapter 4 [Using Human Computation for Ontology evaluation: The VeriCoM Approach](#) describes the VeriCoM Approach, used for solving the problem of verifying conceptual domain models, and the adaptation of it on an ontology, in order to verify it.
- Chapter 5 [Ontology evaluation experiment](#) introduces the second practical part of the thesis, the Ontology evaluation experiment.
- Chapter 6 [Conclusion](#) summarizes the findings of this work and states further aspects for possible future research in the field.

Background and related work

This thesis concentrates on two main research areas and this chapter covers them in more detail. First of all, the section [2.1 Ontology evaluation](#) describes the need for Ontology evaluation, as well as different approaches to achieve this. Afterwards the section [2.2 Common errors in Ontology Engineering](#) presents discussions in literature regarding which errors are known and commonly noticed. In order to address some of these errors, the thesis uses Human Computation and Crowdsourcing, covered in section [2.3 Human Computation and Crowdsourcing](#).

2.1 Ontology evaluation

While different definitions for ontologies exist, the original and most cited one in the literature belongs to Gruber and defines an ontology as an "*explicit specification of a conceptualization*" [\[12\]](#). Ontologies are therefore used as conceptual models and knowledge representations. Not only are they key concepts in the Semantic Web [\[5\]](#), but they are also used for reasoning and information retrieval [\[16\]](#).

Ontology evaluation presumes checking the technical quality of an ontology against a reference system. The need for evaluation is given not only because it might help with the engineering process [\[17\]](#), but also because many ontology reuse large-scale existing ones, because of resources issues [\[6\]](#), so ensuring the quality of these ontologies play a crucial role. The evaluation can be achieved through different approaches:

- Topology based: [\[3\]](#), [\[10\]](#), [\[26\]](#) - based on the internal structure of the ontology
- Application based: [\[11\]](#), [\[18\]](#) - based on how a task can be accomplished in an application that uses the ontology
- Data driven: [\[3\]](#), [\[7\]](#) and Gold standard: [\[8\]](#), [\[15\]](#) - comparison against an unstructured or structured resource representing the problem domain

- Expert knowledge: [14], [13] - humans assess the quality of the ontology

2.2 Common errors in Ontology Engineering

Concerning the common errors typically introduced during ontology engineering, the main papers that are used to synthesise them are: [19], [23] and [28].

In [19] two contributions are made. The first one refers to a catalogue of 41 pitfalls that one can fall into when developing an ontology and the second one regards OOPS!, a web-based tool for diagnosing potential problems of ontologies, based on a checklist of common errors. The pitfalls in the catalogue are classified regarding their perspective: structural, functional or usability-profiling, but also regarding their importance levels: critical, important and minor. It is important to specify that the levels have no clear boundaries and can depend on the project they are used on. For the classification of the importance, both the opinion of experts and OOPS! users was considered. Regarding the OOPS! tool, it provides mechanisms to diagnose 33 of the 41 pitfalls described in the catalogue. Those ones that require an external reference framework or human intervention are not yet automated. For the detection of the problems, three methods were used: structural pattern matching, lexical content analysis and specific characteristic search. For some pitfalls, these detection methods might not cover all the possible situations where a problem could occur, but rather a subset of them. Still, the advantages of OOPS! are enlarging the list of detectable errors, being independent of an ontology development environment, working with the main browsers and being able to configure the detection method.

The authors of the second paper, [23], tried to find the most common difficulties of the OWL description language, especially from the perspective of newcomers. The most common errors found regard, among others, disjointness, the open world assumption and domain and range constraints. The disjointness problem appears because users would naturally assume that two defined concepts are different, unless they had an explicit common child, but in OWL classes are overlapping until disjointness between them is added. The next problem has to do with the fact that almost all systems used by newcomers use a closed world reasoning, so understanding the open world reasoning is crucial for OWL. In OWL, something is false only if it can be proved to contradict other information of the ontology. The next major problems are the difficulties regarding domain and range constraint, because in OWL they are axioms and are used in reasoning, with potential unexpected effects. Another important topic discussed in the thesis is the logical problem of understanding the difference between "only" and "some", and that the first does not imply the other ("Universal restrictions can be satisfied trivially"), the linguistic versus the logical use of "and" and "or", but also the confusion of "some not" and "not some". In order to avoid as many errors as possible, a guideline for engineering ontologies was created. Moreover, the paper proves the idea of understanding expressions by paraphrasing them in explicit language, so a table of OWL definitions and their paraphrases was created.

The third work, [28] regards knowledge representation languages, used by both computer scientists and domain experts, in particular Description Logics. Firstly, the difficulties experienced with using DLs are discussed. These include both understanding and reasoning with DLs. The second large topic of the paper is an overview of different theories of reasoning developed by psychologists, like the rule-based or the model-based logics. Also the ambiguity of natural language is pointed out. Lastly, three different studies were conducted, where the participants were computer scientists or domain experts familiar with ontologies. The first two studies concentrated on identifying difficulties experienced with DLs, while the last study checked the possibility of using alternative syntactic constructs. This work showed that some of the constructs are ambiguous and the results of the last study show that using different or additional keywords could affect the performance of understanding and reasoning with DLs. It is also important to note that one way of understanding the problems that users might have with knowledge representation languages is getting insights from cognitive psychology and language studies.

These three works are used as the base of the selection of ontology engineering errors to be considered in this thesis' ontology use case:

- D1: missing existential restriction
- D2: universal restriction instead of existential restriction
- D3: missing universal restriction
- D4: missing disjointness
- D5: confusion between linguistic and logical "and" (intersection versus union)

While these findings have a rather theoretical approach, the practical data comparison in [3] will determine how often the selected errors occur in the ontology engineering process.

2.3 Human Computation and Crowdsourcing

With reference to Human Computation and Crowdsourcing, an extensive work was done by [21]. Firstly a literature study was made, in order to summarize the definition of the important terms, but also the factors that influence crowdsourcing projects:

- the human factor (requesters and workers);
- the crowdsourcing process (from different perspectives: the workers, the requesters, the tasks):
 - 1. Define the problem
 - 2. Collect data requirements

- 3. Design the task
 - 4. Launch the task online via a crowdsourcing platform
 - 5. Analyse the result
 - 6. Send rewards to workers
- the types of crowdsourcing tasks;
 - but also other factors: user interface, training questions, length of task, ordering of data in the task.

Afterwards some crowdsourcing experiments were conducted. One of the result was that the performance of the workers is better when ordering the tasks in unbalanced label situations. For balanced classes, the performance was better when the relevant documents were presented first. For unbalanced batches, increasing the number of documents was find to improve performance, while for balanced batches it lowers it. Also, in absence of training questions, agreement is higher when relevant documents are shown at the beginning of the batch. Another outcome is that the results of crowdsourcing projects are reliable (high level of agreement between workers and experts), repeatable (consistent over weeks time on the same platform), generally not reproducible (inconsistent on different platforms) - but they can be if the expectations on payment and rejection rate across different platforms are equally set. The last important result is that, generally, the motivation for crowdsource workers is the reward, followed by interest, time required to complete the task and the difficulty level. Even though requesters agree on a higher payment, because of channels in between (studied on F8), the workers get less amount of money for the completed tasks. The demographics and gender influence the tasks and the requesters should pay attention to this when choosing the channels where the tasks are available.

While [21] conducted the experiment regarding the ordering of the tasks with relevance judgements, another paper worked with tasks in the writing domain, [9] focus on the performance of the workers when ordering the tasks regarding three characteristics: continuity - same complexity level of the coming microtasks, regarding operations and content; transitions - transitioning to another complexity; ease-in - starting with simpler ones to ease into more complex tasks. The experiments showed that regarding continuity, the workers perform better in low-complexity chains when the same operations are coming, and better in high-complexity chains when same content is following. For the transitioning characteristic, they found out that workers perform better when transitioning to microtasks of the same complexity. Regarding ease-in, the authors state that the workers were more motivated and felt more "mentally warmed up" if they first completed microtasks of low complexity before a more complex microtask. These findings show that the ordering of the tasks is important and can influence the performance of the workers. The design of this thesis' experiment was based on the findings and conclusions of [21] and [9].

The authors of [25] developed an approach that solves the problem of verifying conceptual domain models with Human Computation. While they applied the new approach called Verifying Conceptual Models (VeriCoM) in a software engineering use case, this thesis will extend the approach on ontology verification. In order to evaluate the performance of this new approach, an experiment using Human Computation was conducted. The participants in the experiment were students of the Vienna University of Technology (TU Wien), having already knowledge in the software engineering field. After a preparation stage and a tutorial, the participants had to verify the given model by identifying defects as part of the Human Computation task. The experiment showed a precision of 73% when identifying defects and 63% when recalling from a Gold Standard defect set.

The usage of Human Computation for Ontology verification is rather in its preliminary state. In [20] the author proposes a two-step hybrid human-machine verification process. Firstly defect candidates are found automatically, afterwards these defects are verified by crowds using Human computation. The author concentrates on the following 4 types of defects: Missing disjointness axiom, Confusion between logical and linguistic "and", Trivially satisfiable allValuesFrom, Missing closure axiom. Considering the fact that these defects cannot be found automatically, because of missing human knowledge, the automated detection of defect candidates uses a heuristic approach and has its limitations: the findings are only possible errors which have to be validated by crowds in a second step.

A large literature study and a review of a list of papers regarding human-centric ontology evaluation were made by [27], by investigating the evaluation tasks, their characteristics and the solution approaches used. Most of the quality issues that require human knowledge can be classified in the following categories:

- **incorrect information** - wrong data types, relations, taxonomic structures, translations
- **missing information** - missing requirements, relevant concepts, type and direction of relationships, either in the resulting ontology or in the requirements themselves
- **cognitive defects** - irrelevant, controversial or contradicting information, polysemous elements
- **incorrect modeling** - overspecialisation, duplication or wrong usage of restrictions, "some not" and "not some", classes

Afterwards a list of ontology evaluation tasks that have not yet been approached using Human Computation techniques was collected. Out of this list, the author selected the task of ontology restriction verification and proposed for it a Human Computation approach. This approach was examined in an experiment using the Crowdsourcing technique with students, similarly to the method that will be used in this thesis. By building on top of this work, not only will the number of singular error tasks types

2. BACKGROUND AND RELATED WORK

increase from three to five, but also the tasks get up to three different errors together, in order to check the performance of workers to identify these errors.

In summary, the work of this thesis builds upon prior research in two domains: Ontology engineering and Verification, especially regarding common errors that can occur while developing an ontology, and using Human Computation and Crowdsourcing to try to resolve some of the ontology engineering problems. Firstly, this thesis shows what typical engineering errors in beginners' ontologies are, considering practical data. Secondly, while initial research was done in this field, the thesis follows up with implementing a Human Computation and Crowdsourcing approach on an Ontology Engineering use case, in order to verify possible engineering errors. Lastly, the thesis shows how the performance of workers regarding verification of ontology engineering errors changes, when increasing the error types and the variety of errors in an Human Computation task.

Practical comparison of beginners' ontologies

While the previous chapter concentrates on the common errors found in the literature regarding Ontology Engineering, this chapter presents a comparison using data extracted from beginners' ontologies, in order to answer RQ1.

RQ1: What are typical engineering errors in beginners' ontologies in practise, compared to the common errors introduced in the literature?

The chapter contains firstly information about the collection of the data and afterwards explains the methodologies and tools used for the comparison, together with the results of the comparison for each of them, respectively.

3.1 Collection of data

In order to exercise their skills regarding modelling an ontology, students of the Vienna University of Technology (TU Wien) course 188.399 Introduction to Semantic Systems were given a graded assignment to suggest a semantic application that uses knowledge graphs and to create an ontology that models the domain for this application. The application idea could be built upon different domains as Music, Movies, University, Retail, Mobility etc., as long as there were stated also 5 to 10 competency questions that the application would be able to answer. While creating the ontology that represents the selected domain, the students had to make sure that at least 15 concepts and 20 data/object properties, as well as relationships between the concepts are considered.

An example for a solution of the assignment, uploaded by one of the students, is an application for the administration of a research and working domain in a specific institute of a university. Therefore, information about the persons working at the institute are

considered, as well as research projects, activities, courses carried by the institute. Some competency questions could be "What lecturers are work colleagues?", "Which books were written by workers of the institute?" or "Which courses are offered by the institute?".

The total of 68 submitted ontologies were anonymised and the level of knowledge of the students is assumed, because of the field of study and the novice level of the course.

3.2 Methodology and results

For the comparison of the ontologies, two tools were used: the OOPS! tool developed by [19] and the tool for finding defect candidates developed by [20].

3.2.1 OOPS! Tool

The OOPS! tool was developed in order to automate the finding of ontology engineering pitfalls. While the authors specify that only pitfalls that do not require an external reference framework or human intervention could be automated, and also the detection might not cover all the possible situations where a problem could occur [19], the results of the comparison prove what the literature stated already, that ontology engineering is error prone.

The following pitfalls, identified and explained in [19], were detected on the 68 beginners' ontologies:

- *P02 Creating synonyms as classes*: synonyms identifiers show duplicated classes. Identified in: 2/68 ontologies.
- *P03 Creating the relationship "is" instead of using "rdfs:subClassOf", "rdf:type" or "owl:sameAs"*: OWL primitives should be used. Identified in: 3/68 ontologies.
- *P04 Creating unconnected ontology elements*: elements with no relation to the rest of the ontology. Identified in: 17/68 ontologies.
- *P05 Defining wrong inverse relationships*: relationships which are not necessarily inverse are defined as such. Identified in: 4/68 ontologies.
- *P07 Merging different concepts in the same class*: defining one class with an identifier composed of more concepts. Identified in: 2/68 ontologies.
- *P08 Missing annotations*: lack of human readable annotations such as `rdfs:label` or `rdfs:comment`. Identified in: 54/68 ontologies.
- *P10 Missing disjointness*: lack of disjoint axioms between classes or properties. Identified in: 23/68 ontologies.
- *P11 Missing domain or range in properties*: relationships and/or attributes without domain or range are defined. Identified in: 45/68 ontologies.

- *P13 Inverse relationships not explicitly declared*: any relationship, besides symmetric ones, does not have an inverse relationship defined. Identified in: 64/68 ontologies.
- *P19 Defining multiple domains or ranges in properties*: multiple domains or ranges represent the intersection instead of union, respectively. Identified in: 35/68 ontologies.
- *P20 Misusing ontology annotations*: as for example, an explaining sentence in "rdf:label" and a word in "rdf:comment". Identified in: 3/68 ontologies.
- *P21 Using a miscellaneous class*: using a class for classifying instances that do not belong to any of its sibling classes. Identified in: 4/68 ontologies.
- *P22 Using different naming conventions in the ontology*: using different naming conventions in the same ontology. Identified in: 43/68 ontologies.
- *P24 Using recursive definitions*: a concept is used in its own definition. Identified in: 16/68 ontologies.
- *P25 Defining a relationship as inverse to itself*: using inverse instead of symmetric. Identified in: 1/68 ontologies.
- *P26 Defining inverse relationships for a symmetric one*: a symmetric relationship is inverse to self, so cannot be inverse to another relationship. Identified in: 2/68 ontologies.
- *P27 Defining wrong equivalent properties*: properties defined as equivalent even though they have different semantics. Identified in: 1/68 ontologies.
- *P28 Defining wrong symmetric relationships*: relationships which are not necessarily symmetric are defined as such. Identified in: 3/68 ontologies.
- *P29 Defining wrong transitive relationships*: relationships which are not necessarily transitive are defined as such. Identified in: 8/68 ontologies.
- *P30 Equivalent classes not explicitly declared*: classes which are equivalent should be defined as such. Identified in: 9/68 ontologies.
- *P32 Several classes with the same label*: same content in the "rdfs:label" annotation. Identified in: 1/68 ontologies.
- *P34 Untyped class*: usage of a class without declaring it as such. Identified in: 1/68 ontologies.
- *P35 Untyped property*: usage of a property without declaring it as such. Identified in: 1/68 ontologies.

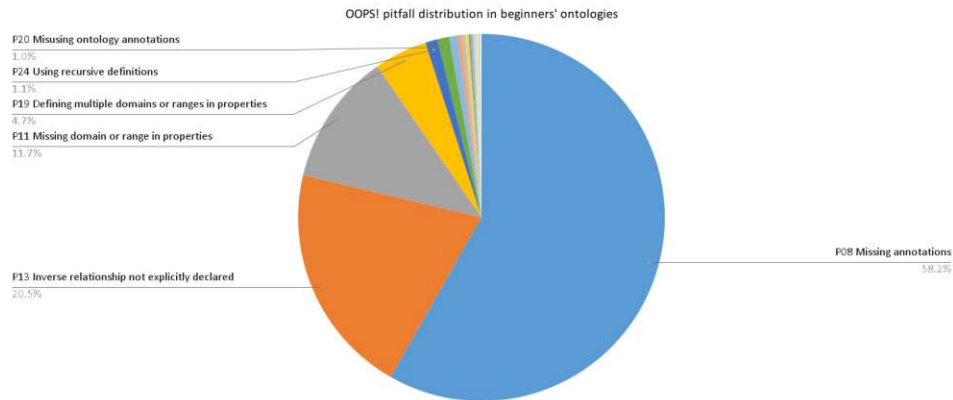


Figure 3.1: OOPS! pitfall distribution in beginners' ontologies

The two pitfalls P10 and P22, that regard the specific ontology as a whole, were found in 33.8% and 63% of the ontologies, respectively.

For the other type of pitfalls that apply to single classes and/or relationships, Figure 3.1 shows the pitfall distribution in beginners' ontologies: P08, P13, P11, P19, P24 and P20 have an average of 27, 9.5, 5.5, 2 and 1 occurrences per ontology respectively. The other pitfalls listed above have an average of occurrences per ontology below 1 and are omitted in Figure 3.1.

To be more concrete, Figure 3.2 shows the histogram of each of the 4 pitfalls with a median value over 1. It can be observed that most of the ontologies have between 0 and 5 errors regarding missing annotations, an almost equal distribution of errors between 0 and 20 regarding inverse relationships not explicitly declared, between 0 and 2 errors regarding missing or defining multiple domain or range in properties.

3.2.2 Automatic Defect Candidate Detection

In the thesis "Hybrid Human-Machine Ontology Verification" [20], the author developed a two step methodology for detecting errors in ontologies. This methodology uses firstly an automatic defect candidate detection which is afterwards verified through Human Computation and Crowdsourcing. The detection of defect candidates is made through a heuristic approach, since the thesis only concentrates on 4 defects which cannot be found automatically, because of missing human knowledge. As well as the previous tool, this one also has as limitation the fact that the findings should be considered as "defect candidates" or possible errors, since they must be validated by crowds in a second step, because of the need of human knowledge to identify such errors. However, the tool improves the scalability of the ontology verification process by facilitating the search for possible error prone locations in the ontology.

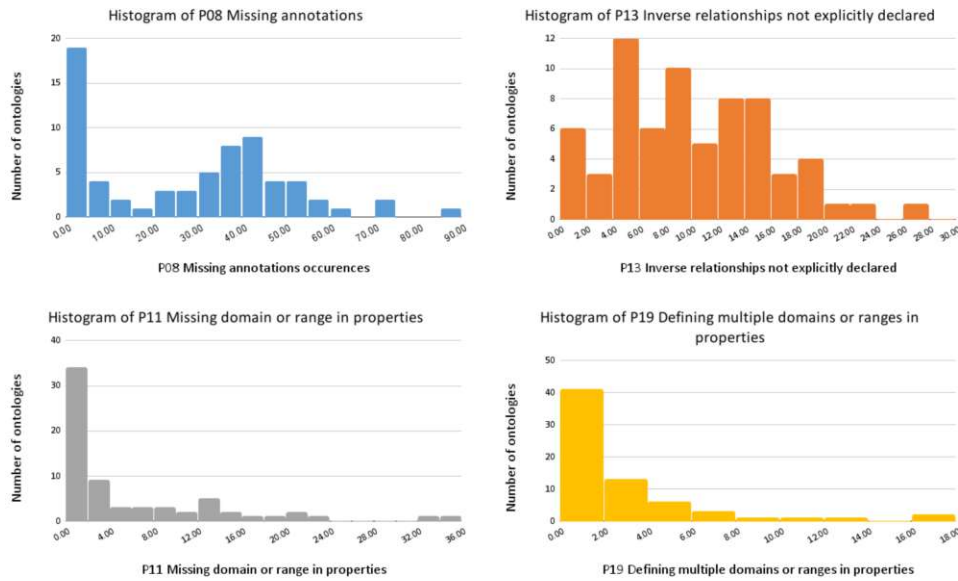


Figure 3.2: Histogram of each of P08, P13, P11 and P19

The following defect candidates, as explained in [20], were detected on the 68 beginners' ontologies:

- *Missing disjointness axiom*: same as P10 from Subsection 3.2.1, lack of disjoint axioms between classes or properties. Identified in: 59/68 ontologies.
- *Trivially satisfiable allValuesFrom*: using the universal restriction without considering that is trivially satisfiable, falsely implying an existential restriction too. Identified in: 15/68 ontologies.
- *Missing closure axiom*: using the existential restriction without considering a universal restriction too, falsely implying a Close World Assumption. Identified in: 27/68 ontologies.

Figure 3.3 shows the error distribution in beginners' ontologies: Missing disjointness axiom, Trivially satisfiable allValuesFrom and Missing closure axiom have an average of 29, 1.5, and 1 occurrences per ontology respectively. To be more concrete, Figure 3.4 shows the histogram of the Missing disjointness axiom error, the only error found by the tool having a median value over 1. It can be observed that most of the ontologies have between 0 and 40 locations where the missing disjointness error could be a problem, while a few of them seem to not have considered such axioms at all.

The practical comparison presented in this chapter shows that beginners' ontologies are error prone and many of the errors can be detected using automated tools. With the

3. PRACTICAL COMPARISON OF BEGINNERS' ONTOLOGIES

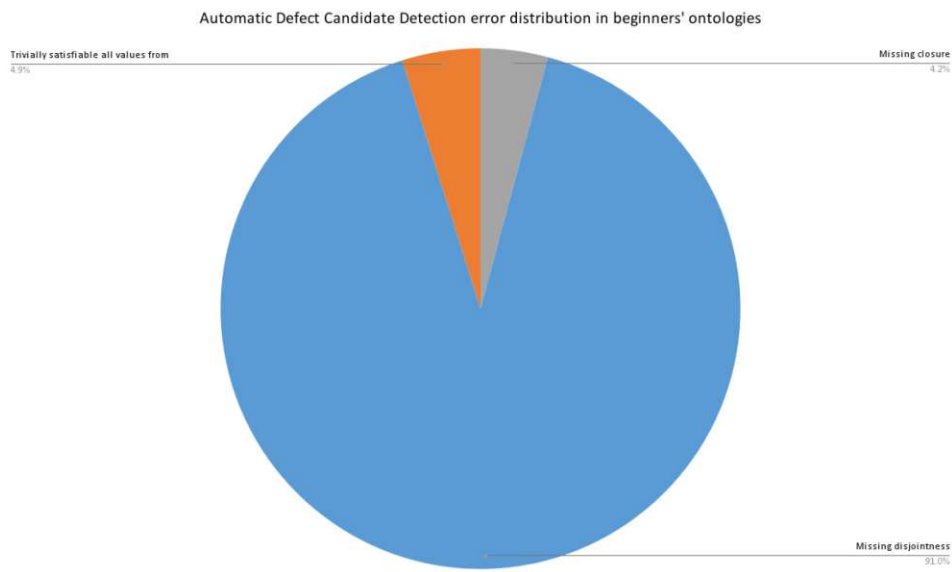


Figure 3.3: Automatic Defect Candidate Detection error distribution in beginners' ontologies

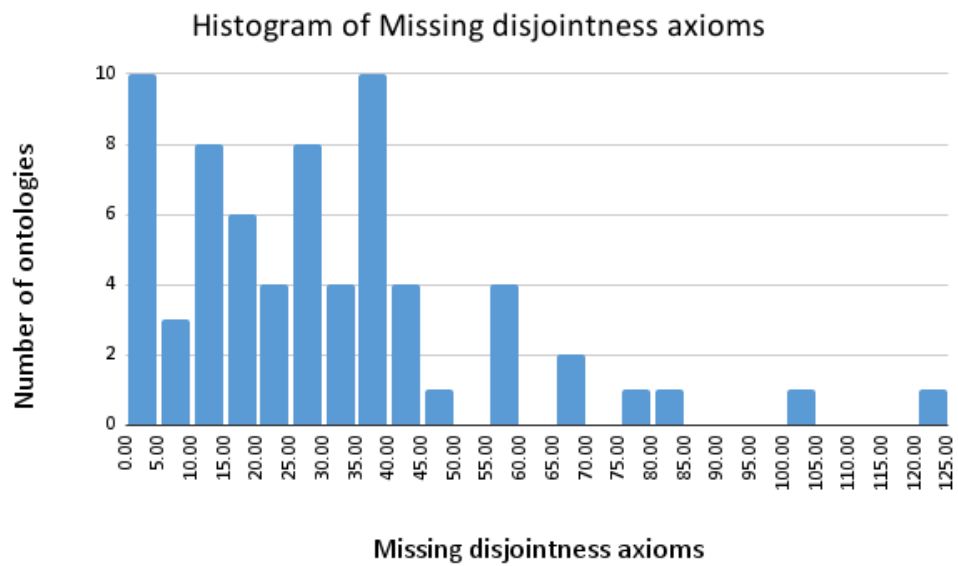


Figure 3.4: Histogram of Missing disjointness axioms

help of the OOPS! tool [19], one can say that the main errors that novices make when engineering ontologies regard readability (P22 and P08), disjointness of classes (P10), not declaring inverse relationships (P13) and the confusion between logical "and" and "or" (P19). The Automatic Defect Candidate Detection [20] heuristic shows that even if some specific errors cannot be identified automatically, locations in ontologies can be found, where such an error could be present. Based on the result of this tool, it seems that the main problem of beginners' ontologies could constitute the missing of disjointness axioms, which supports the finding regarding P10 of the first comparison tool.

Using Human Computation for Ontology evaluation: The VeriCoM Approach

This chapter answers RQ2 by investigating how incorrect modelling problems can be solved with Human Computation, adapting the VeriCoM Approach on ontology engineering.

RQ2: What is a good methodology to solve problems regarding ontology engineering with Human Computation?

While the importance of Ontology Evaluation (Section 2.1) and the presence of Common errors in Ontology engineering (Section 2.2) have been discussed previously, because the usage of Human Computation to solve such issues is still in a preliminary state, as introduced in Section 2.3, there are nevertheless missing ways how it can be used to address ontology engineering problems.

First of all, the errors regarding incorrect modeling of ontologies that are going to be considered are defined and explained in Section 4.1. Afterwards, Section 4.2 contains a break down of the previously introduced VeriCoM Approach into stages and an extensive description of each one of them. Finally, the methodology described by VeriCoM is going to be applied in Section 4.3 for specific problems of Ontology Engineering.

4.1 Addressed ontology modelling errors

Based on their appearance in the literature [19], [23], [28], [27], but also on their importance when engineering ontologies and presence in the beginners' ontologies practical comparison regarding errors (Section 3), following errors regarding incorrect modelling are identified and addressed:

- D1: missing existential restriction
- D2: universal restriction instead of existential restriction
- D3: missing universal restriction
- D4: missing disjointness
- D5: confusion between linguistic and logical "and" (intersection versus union)

In the following, each error will be explained and then exemplified making use of the well-known Pizza ontology [23], developed for educational purposes, which will also be used in the practical part of this thesis. For better visualisation of the ontology, the Visual Notation for OWL Ontologies (VOWL) [2] formalism is used. In order to focus on the addressed errors, the models are simplified, as it can be seen in Figure 4.1, by omitting different characteristics as for example the *Subclass of* relationship of pizzas to the class *Pizza*. In this case, the modelled components of an ontology are: a pizza and its ingredients as *classes*, the *hasTopping* relationship between them, the *disjoint* relationship between ingredients, *universal and existential restrictions* as well as the *union or intersection* of classes. For the example of a Hawaii pizza, it is a pizza that has Mozzarella, Pineapple, Tomato and Ham as toppings.

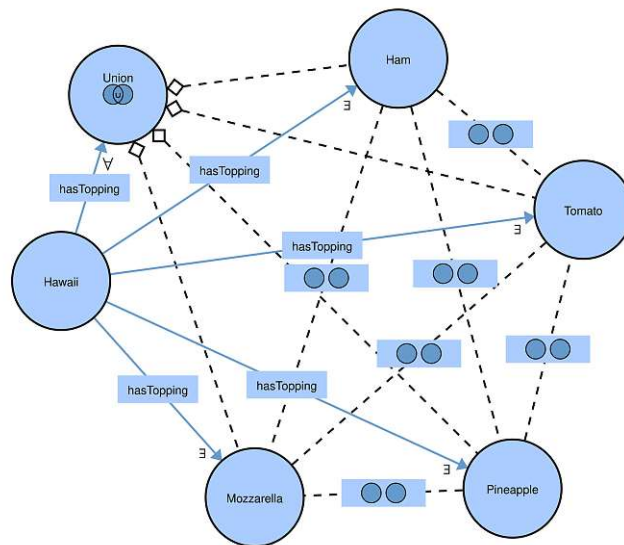


Figure 4.1: Correct modelling of Hawaii pizza

In order to understand the addressed errors, following terms are to be clarified:

- The **existential restriction** indicates that there must be a property of the specified type, but other types are not restricted. As seen in Figure 4.2, every instance of *PetLoverTypeB* has a *Cat* pet and may also have other pets.

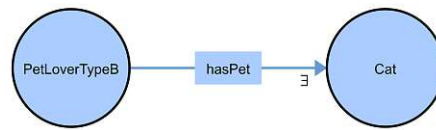


Figure 4.2: Existential restriction example

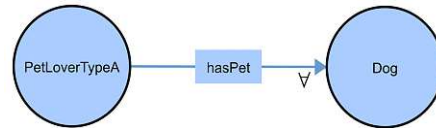


Figure 4.3: Universal restriction example

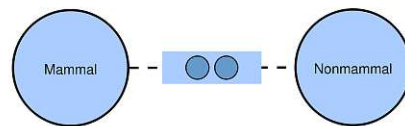


Figure 4.4: Disjointness example

- **Universal restrictions** signal that all property values from the specified property must be of a certain type. All values must be of that type, but a property value must not necessarily need to exist. The universal restriction does not imply the existential restriction. Figure 4.3 shows that if instances of *PetLoverTypeA* have pets, the pets are always *Dog* pets. Instances of *PetLoverTypeA*, however, may not have any pets at all.
- **Disjointness** of two (or more, mutually) classes means that they do not share any instances: there is no instance that belongs to both (all) classes. Disjoint classes also cannot have any common subclass, because it would be unsatisfiable. As in Figure 4.4, the animals can be either of type *Mammals* or *Nonmammals*.
- The **union** between two classes is a class that contains instances belonging to any of the two classes. *Animals* are the union of *Mammals* and *Nonmammals*, as it can be seen in Figure 4.5.
- The **intersection** between two classes is a class of instances that belong to both classes (i.e., shared instances). In the example of Figure 4.6, *WorkStudents* are both *Workers* and *Students* at the same time.

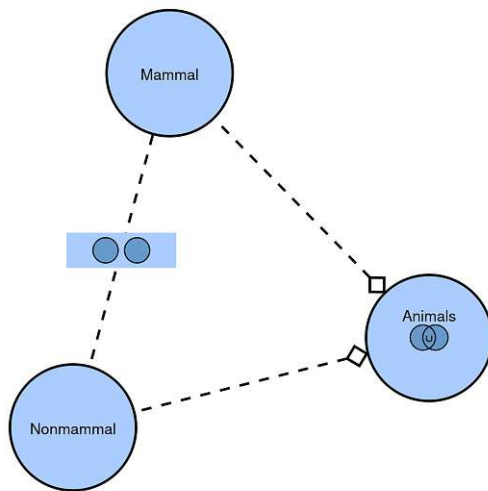


Figure 4.5: Union example

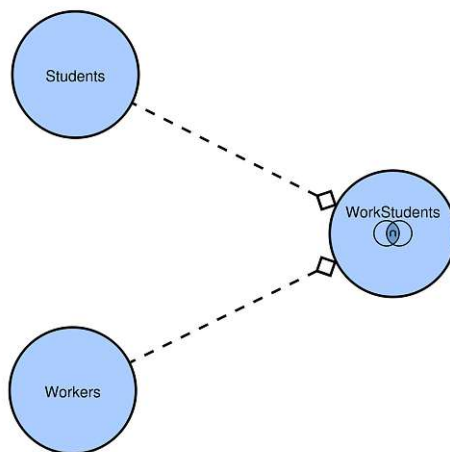


Figure 4.6: Intersection example

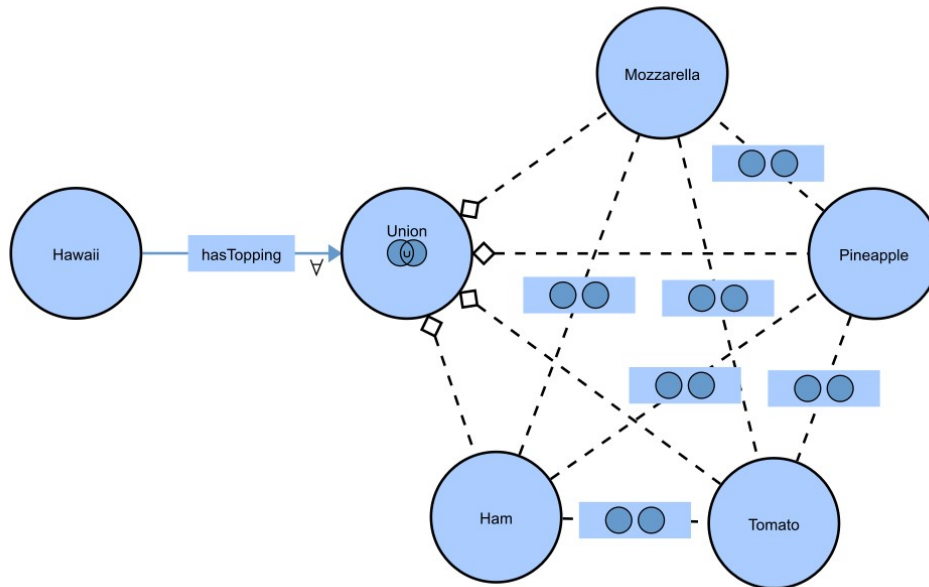


Figure 4.7: D1: missing existential restriction on Pizza Hawaii

4.1.1 D1: missing existential restriction, D2: universal restriction instead of existential restriction

Both the error of missing existential restriction as well as using the universal restriction instead are caused by assuming that the universal restriction implies the existential one [23]. Because of the trivial satisfaction of the universal restriction, unwanted effects can occur.

In the context of the Pizza ontology, Figure 4.7 shows the missing existential restriction error and, because of the trivial satisfaction of the universal restriction, it allows for the Hawaii pizza to have no ingredients at all. The similar case of using the universal restriction instead of the existential one can be seen in Figure 4.8, where as well the Hawaii pizza could have no ingredients at all.

4.1.2 D3: missing universal restriction

The error of missing universal restriction, also called the "Closure restriction", is caused by the Open World Assumption (OWA), present in the field of Ontology engineering. Opposite to other areas like constraint languages, databases, programming languages, which follow a Closed World Assumption (CWA) - everything that is not described is considered incorrect - the OWA leads to the fact that the information that it's not explicitly modeled is not incorrect, but rather cannot be inferred [23] [28].

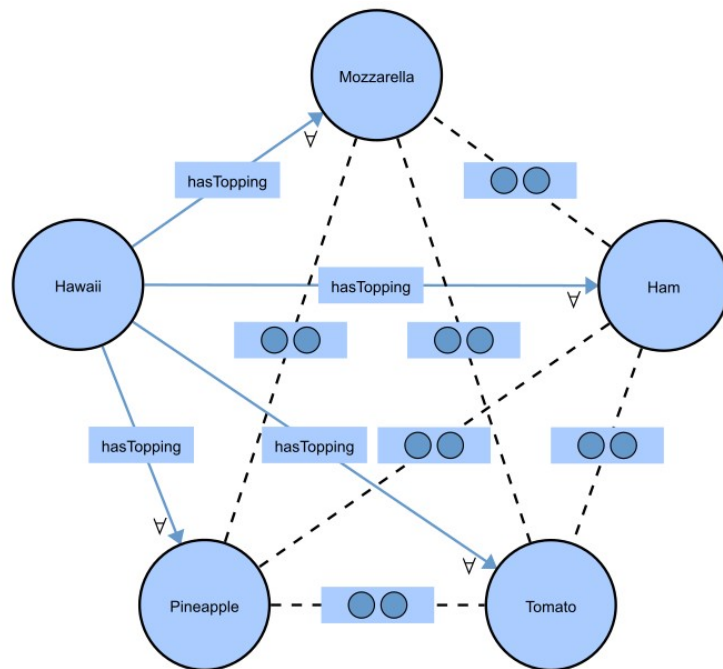


Figure 4.8: D2: universal restriction instead of existential restriction on Pizza Hawaii

Figure 4.9 exemplifies this defect, by allowing the Hawaii pizza to also contain other ingredients, for example Mushrooms, since only the existential restriction does not assure a closure.

4.1.3 D4: missing disjointness

While naturally one can assume that two defined concepts are different, in Ontology engineering classes are overlapping unless disjointness between them is explicitly declared [23]. This is because of the concept of Open World Assumption explained above. If the disjointness is not stated, like for example in Figure 4.10, then the Hawaii pizza could omit the Cheese, since Mozzarella could be a subclass of Ham.

4.1.4 D5: confusion between linguistic and logical "and" (intersection versus union)

"The ambiguity of natural language" and the fact that "and" and "or" can be interchanged in the linguistic field - contrary to their logical values - leads to the confusion between intersection and union [23]. While the union widens the possibilities, the intersection of classes restricts the range of the instances.

Figure 4.11 shows that the Hawaii pizza cannot exist, since in the context of using disjointness, the intersection of two disjoint classes is an empty set, corresponding to

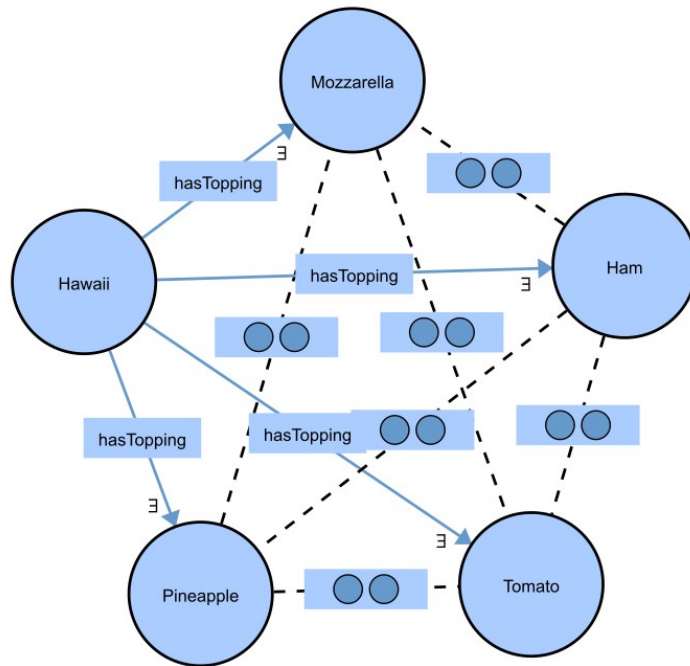


Figure 4.9: D3: missing universal restriction on Pizza Hawaii

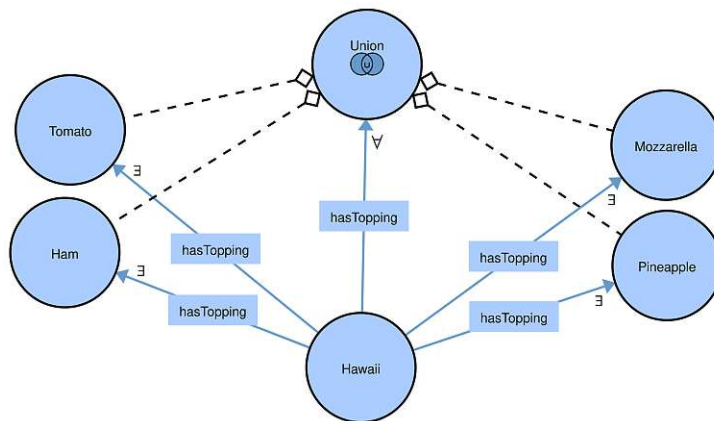


Figure 4.10: D4: missing disjointness on Pizza Hawaii

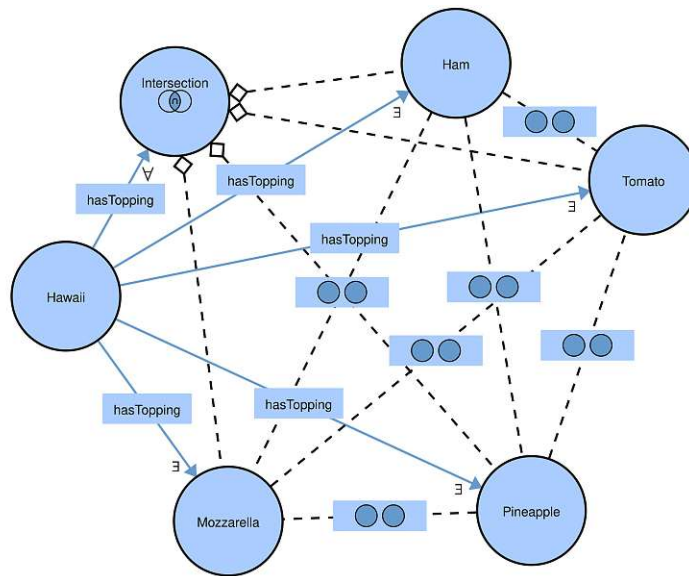


Figure 4.11: D5: confusion between linguistic and logical "and" (intersection versus union) on Pizza Hawaii

an unsatisfiable class. Even when omitting the disjointness, the correct definition of a Hawaii pizza would have all the four ingredients on, and not the intersection between Ham, Tomato, Pineapple and Mozzarella which would naturally be expected to be empty.

In order to identify the addressed errors in ontologies, expert evaluations could be carry out. Because such an evaluation is expensive, time-consuming, and has limited scalability, a Human Computing and Crowdsourcing methodology to solve such problems regarding ontology engineering has to be defined. In the following section, the VeriCoM Approach [25] is presented.

4.2 The VeriCoM Approach

To archive the goal of a more broad approach for evaluating conceptual models with human computation, the authors of [25] first had to **formalize** the problem at a generic level. Therefore, they used a function (γ), that applied to the model (M) and the frame of reference (FR), leads to a set of defects (D).

$$\gamma(M, FR) \rightarrow D$$

The model M is the union of at least two model element sets, ME_C - the concepts in a domain, and ME_R - the relations between them. This is a minimal requirement for M , that can also contain other model elements, like concept attributes, relation attributes or instances. The frame of reference FR can have different forms, like for example specific documents or common sense knowledge, but the authors concentrate

on textual specifications (*Spec*). An expected model element (*eme*) appears in the *Spec* and is expected in the model *M*, such that the group of all these elements (*EME*) is what has to be verified. The relevant evidence for an *eme* is a representative text chunk from *Spec* where the *eme* is mentioned and described, leading to the collection of evidence for all the *emes* of *EME*, $EV_{EME,Spec}$. The function of assigning each *eme* to a specific evidence from *Spec* is defined by the authors as:

$$\gamma(EME, Spec) \rightarrow EV_{EME,Spec}$$

Because the model *M* can contain elements that are not present in *Spec*, the intersection of the expected and the actual modelled elements contain the *emes* that have a equivalent or synonym (\approx) in the model.

$$EME \cap ME = \{eme | \exists me \in ME \wedge eme \approx me\}$$

The result of the function $\gamma(M, FR)$ is a set of defects (*D*) in the model text *M* with respect to the *Spec*. Each defect $d \in D$ refers to an *eme* and it can be either of type *Missing*, if it appears in the *Spec* but not in *M*, or of type *Superfluous*, if it is modelled in *M* without being mentioned in *Spec*. The first type is represented by the set $EME \setminus ME$, while the second by the set $ME \setminus EME$. Other domain specific defect types can be identified.

Having a formal problem, the authors of [25] developed a generic approach for Verifying Conceptual Models (VeriCoM) regarding the textual specification, using Human Computation. In order for the approach to be applied, the following stages have to be completed:

- Data Preparation Stage
- Task Design and Execution Stage
- Aggregation Stage
- Evaluation Stage

Data Preparation supposes first identifying the model *M* and the specification *Spec* used for the verification. Afterwards, the model element types that are going to be verified are set in order to be able to identify the *EME*, the elements expected to appear in the model. As stated in the formalization, each *eme* needs a representative evidence in the *Spec*, so the set of evidences $EV_{EME,Spec}$ is defined such that it covers all the *emes*, it is small enough to be used in human computation tasks and is enough to explain the context to the workers. The last step in this stage is deciding over the defect types, if other defects than *Missing* and *Superfluous* are to be considered.

Having the initial stage completed, the next one to follow is the **Task Design and Execution** Stage. The task design has to enable the detection of the considered defects. The authors exemplify this by a design that guides a worker to decide about three characteristics of an *eme*:

- Relevance: is the *eme* relevant?
- Representation: is the relevant *eme* missing? A *missing* defect is encountered; Is a irrelevant *eme* represented? A *superfluous* defect is encountered.
- Correctness and Interpretation: is the relevant *eme* or a synonym of it represented correctly?

The result of the execution of the tasks is a collection of individual defect reports (*DR*), that connects an *eme* and its evidence to a defect type, based on the judgment of a worker (W_x): $DR(eme, Ev_{(eme, Spec)}, W_x, D_{type})$.

For the **Aggregation**, the individual defects are aggregated in order to identify a final defect type, using an output agreement strategy. Each individual defect of an *eme* receives an agreement coefficient (*ACoeff*) and the one with the highest or, as the case may be, above a threshold value, is selected as the final defect type for the *eme*. In the case of a tie, the defect is labeled as *Undecided* and, if the possibility of free-text defects is given, they must be aggregated manually. The result of the aggregation, Aggregated Defect Reports, is defined as follows: $ADR(eme, Ev_{(eme, Spec)}, ACoeff, W_{1..n}, D_{type})$.

The last stage contains the **Evaluation** of the gathered data and subsequent of the quality of the process. In the best case scenario, the collected defects can be mapped to a list of true (known) defects (*TD*) and recall and precision metrics could be computed. Otherwise, the defects could be evaluated manually, perhaps under specific conditions (e.g defects with high *ACoeff*). The output of this last stage are verified defect reports, possibly aligned to a golden standard (TD_k): $VDR(eme, Ev_{(eme, Spec)}, ACoeff, TD_k, W_{1..n}, D_{type})$.

In [25] the authors exemplified the VeriCoM Approach on a Software Engineering Use Case. The upcoming section extends the Approach in the field of Ontology Engineering Verification.

4.3 Applying the VeriCoM Approach on an Ontology Engineering Use Case

This part of the thesis covers an ontology verification example by using the VeriCoM Approach explained in Section 4.2. In order to show the adaptation of VeriCoM to an Ontology Engineering Use Case, the same Pizza Ontology [23] is used, as mentioned in Section 4.1. For an easier understanding of the use case, the Pizza Ontology is split into small ontologies, each modelling one type of pizza.

4.3.1 Data Preparation

The first element in the Data Preparation is the *Spec*. In this specific case, it is a textual requirements in form of pizza menu item, described in natural language (e.g. "Hawaii

Pizza contains ham, tomato, pineapple, mozzarella"). The requirement is in the English language and is considered to be correct, representing the reference document.

The conceptual model M that has to be verified is an ontology describing one type of pizza, as for example the correct model for Hawaii pizza in Figure 4.1. Formally, $M = M_C \cup M_R \cup M_{RR}$:

- M_C is the set of all classes, e.g. *Hawaii(Pizza)*, *Ham*, *Tomato*, *Pineapple*, *Mozzarella*, *Union*, *Intersection*;
- M_R is the set of relations, e.g. *hasTopping*, *disjoint*;
- M_{RR} is the set of relation restrictions, e.g. *existential restriction* \exists , *universal restriction* \forall .

The goal of the presented use case is to verify the modelling of *Union* and *Intersection* from M_C , the *disjoint* from M_R and all of M_{RR} such that their correct modelling constitutes the *emes*. The evidence $EV_{EME,Spec}$ is exactly the pizza menu item, which is small enough for a human computation task and also gives enough context to the workers, especially because it is regarded as common knowledge. Regarding the definition of defect types, in order to guide the workers to identify the expected defects and ease the aggregation process, the defects described in Section 4.1 are addressed: D1: missing existential restriction 4.1.1, D2: universal restriction instead of existential restriction 4.1.1, D3: missing universal restriction 4.1.2, D4: missing disjointness 4.1.3, D5: confusion between linguistic and logical "and" (intersection versus union) 4.1.4.

4.3.2 Task Design and Execution

The tasks are designed in such a way, that the workers see a pizza menu item in natural language and the ontology representing this pizza in a visual formalism (VOWL). These two elements are variables, in order to be able to show different pizzas and models in different tasks.

The question for the workers focuses on whether the menu item is represented correctly or not by the ontology. If the answer is no, the worker has to choose out of a multiple choice list of defects the ones that appear in the model. For the sake of completeness and symmetry, but also to filter out spam answers, for the defects D2, D4 and D5, also their opposite was added as an answer possibility (D22, D42, D52):

- D1: "One or more existential (\exists) restrictions need to be added."
- D2: "One or more universal (\forall) restrictions need to be replaced by existential (\exists) restrictions."
- D22: "One or more existential (\exists) restrictions need to be replaced by universal (\forall) restrictions."

- D3: "One or more universal (\forall) restrictions need to be added."
- D4: "Disjointness between entities needs to be added."
- D42: "Disjointness between entities needs to be removed."
- D5: "The set operator intersection (\cap) needs to be replaced by the set operator union (\cup)."
- D52: "The set operator union (\cup) needs to be replaced by the set operator intersection (\cap)."

A free-text field was also present in the task, for eventual comments or other identified defects.

Figure 4.12 shows an example of a modelled task. The red borders delimit the elements of the task. Number 1 represents the textual requirements, the menu item, while number 2 is the model that has to be verified, if it depicts correctly the requirement. As it can be seen in the red square number 3, the worker is initially asked if the pizza item is represented correctly. If so, then the list of errors is not shown. If, as in the figure, the item is not represented correctly, then the multiple selection of errors appears. Afterwards, a comment section is available. In the same view of the task, an instruction panel is offered - number 4, and the "Submit" button (number 5) finishes the task while submitting the answer. The instruction panel offers three tabs, as it can be observed in Figures 4.13, 4.14 and 4.15. The summary 4.13 gives the worker a short tutorial through an overview and the steps to be executed. Under "Detailed Instructions" 4.14, the worker gets familiarised with all the concepts present in the model, as well as tips regarding them. Figure 4.15 shows the last tab of the Instruction, that offers an example of correct modelling of a pizza item, as well as examples for incorrect modelling for each defect respectively.

4.3.3 Aggregation

The aggregation of the data is done for each *eme* separately following the VeriCoM approach by using the majority voting strategy. The comments gathered are analysed manually to identify additional defects or general feedback.

4.3.4 Evaluation

For the evaluation, a golden standard was available, meaning that all the models that contained defects had a list of defects, respectively. The accuracy of the results and the response time are evaluated.

4.4 VeriCoM Summary

This chapter firstly addressed specific ontology modelling errors that occur in the field of Ontology Engineering: missing existential restriction, universal restriction instead

Figure 4.12: HIT: Mushroom Pizza with defects D2 and D5

Figure 4.13: HIT: Instructions


of existential restriction, missing universal restriction, missing disjointness, confusion between linguistic and logical "and" (intersection versus union). Because of the occurrence of these errors, the problem of ontology verification arises. The VeriCoM methodology offers a way for evaluating conceptual models with Human Computation, by formalizing the problem and developing a generic approach. While the VeriCoM approach was exemplified on a Software Engineering Use Case by its authors, in this chapter it was adapted for an Ontology Engineering one. All the presented stages of VeriCoM can be of course customised for different Ontology Engineering problems, the example focusing only on the presented defects applied on a specific, well-known ontology. In order to show that the proposed VeriCoM solution is a good methodology for solving problems regarding ontology engineering with Human Computation and to answer **RQ2**, an experiment was conducted, which is broadly described in Chapter **5**.

Summary **Detailed Instructions** Examples

Rules & Tips


Rules :

- Existential restrictions indicate that there must be a property of the specified type. Other types are not restricted.




- Every instance of PetLoverTypeB has a Cat pet and may also have other pets.

- Universal restrictions indicate that all property values for the specified property must be of a certain type. All values must be of that type but a property value must not necessarily need to exist.



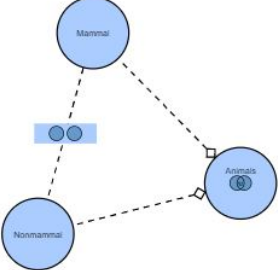
- If instances of PetLoverTypeA have pets, the pets are always dogs. Instances of PetLoverTypeA, however, may not have any pets.

- Two (or more, mutually) disjoint classes do not share any instances: there is no instance that belongs to both (all) classes. Disjoint classes also cannot have any common subclass (this class would be unsatisfiable).



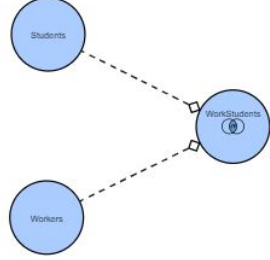
- Instances of animals can be either mammals or nonmammals.

- Union between two classes is a class that contains instances belonging to any of the two classes.



- Animals are the union of mammals and nonmammals.

- Intersection between two classes is a class of instances that belong to both classes (i.e., shared instances).



- Working students are the intersection between workers and students.

Tips:

- The universal restriction does not imply the existential restriction.
- If some information is not included in the model, the missing information is not false unless it contradicts the model.
- While naturally one can assume that two defined concepts are different, in OWL classes are overlapping unless disjointness between them is explicitly declared.
- If two classes are disjoint, their intersection is an empty set (corresponding to an unsatisfiable class).

Figure 4.14: HIT: Detailed Instructions

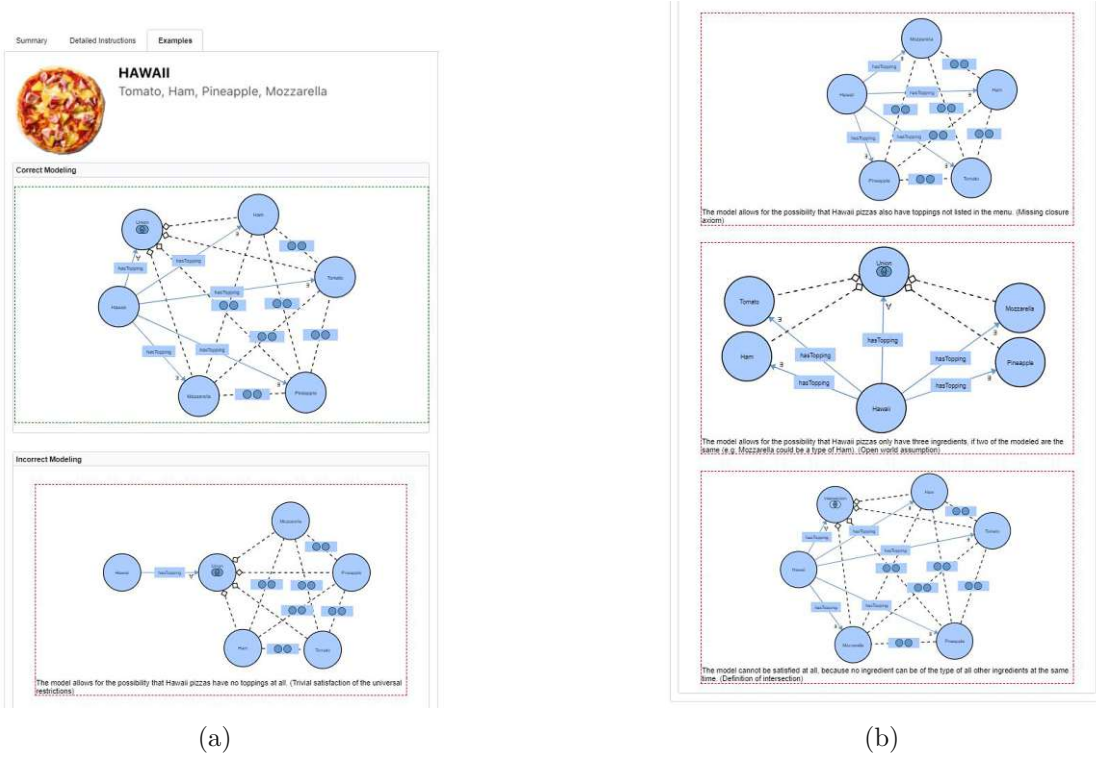


Figure 4.15: HIT: Examples

Ontology evaluation experiment

This chapter verifies the VeriCoM approach proposed in Chapter 4 for solving problems regarding ontology engineering with Human Computation and answering RQ2. The evaluation of the experiment answers RQ3.

RQ3: Considering error classification, to what extent is the overall performance influenced, when having multiple error types in one Human Computation task and when the variety of errors increases from three to five?

In order to develop the experiment, designing the experimental process follows the recommended steps in [29], as shown in Figure 5.1, and discussed in the rest of this

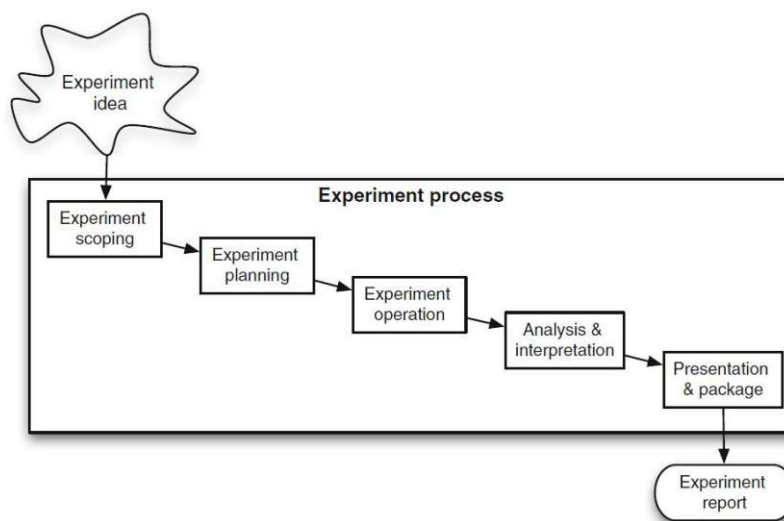


Figure 5.1: Overview of the experiment process [29]

chapter:

- *Scoping*: defining the problem and the aims
- *Planning*: defining the design and the instrumentation
- *Operation*: following the planned design of the experiment
- *Analysis and interpretation*: evaluating the measurements of the operation step
- *Presentation and package*: showing the final results of the experiment

5.1 Experiment scoping

The scoping of the experiment focuses on defining the goals, before the planning and the execution take place, so that the experiment follows the defined path, can be used in the intended study and the rework is reduced to a minimum [29]. In order to manage this, one has to define the following concepts [4]:

- Object of study (what is studied?)
- Purpose (what is the intention?)
- Quality focus (which effect is studied?)
- Perspective (whose view?)
- Context (where is the study conducted?).

The Object of study is the VeriCoM Approach applied on an Ontology Engineering use case. The Purpose is evaluating if it is a good methodology to solve problems regarding ontology engineering with Human Computation. The main Quality focus is the performance of the subjects in verifying the ontology and the Perspective is the researchers'. The Context defines the subjects of the experiment - students of the Vienna University of Technology (TU Wien) course 188.399 Introduction to Semantic Systems, with a basic level of knowledge in the domain - as well as the objects of the experiment. In this case, the objects of the experiment are the errors regarding incorrect modelling of ontologies discussed in Chapter 4, represented by small ontologies. The experiment context classification is then a blocked subject-object study [29], since more subjects are evaluating more than one defect/ontology.

According to the goal template of [4], this experiments' goal outline is:

*Analyze the VeriCoM Approach applied on an Ontology Engineering use case
for the purpose of evaluation*

with respect to *the performance of the subjects*

from the point of view of *the researcher*

in the context of *students evaluating specific defects in individual ontologies*.

5.2 Experiment planning

This phase determines how the experiment will be conducted, by getting from the goal definition to an experiment design through an iterative process and it is a crucial phase in designing an experiment. As it can be seen in Figure 5.2, the process contains the following seven steps [29]:

- Context selection: defining the personnel and the environment
- Hypothesis formulation: formally stating the hypothesis, including a null and an alternative one
- Variables selection: defining the independent variables (inputs) and the dependent variables (outputs), including the value ranges and measurement scales
- Selection of subjects: defining the criteria of selection for the subjects
- Choice of design type: deciding on using design principles and standard design types
- Instrumentation: preparing the objects or artifacts needed, for instance: examples, guidelines, measurement systems etc.
- Validity evaluation: deciding over the validity of the study considering internal, external, construct and conclusion validity

5.2.1 Context selection

Since the experiment is part of the Vienna University of Technology (TU Wien) course 188.399 Introduction to Semantic Systems, the participants are a group of internal students with a level of knowledge in the domain. Other *context selection* characteristics of the experiment, as described by [29], are being off-line, since the project is not a real, large, software project, being of toy size, because of costs, time and understanding matters and, even though the used ontology is specific, it can be easily generalized to different ontologies.

As part of the context selection, the decision on the *platform* on which the experiment is executed fell on Amazon Mechanical Turk (mTurk)¹. This is one of the most widely used

¹<https://www.mturk.com/>

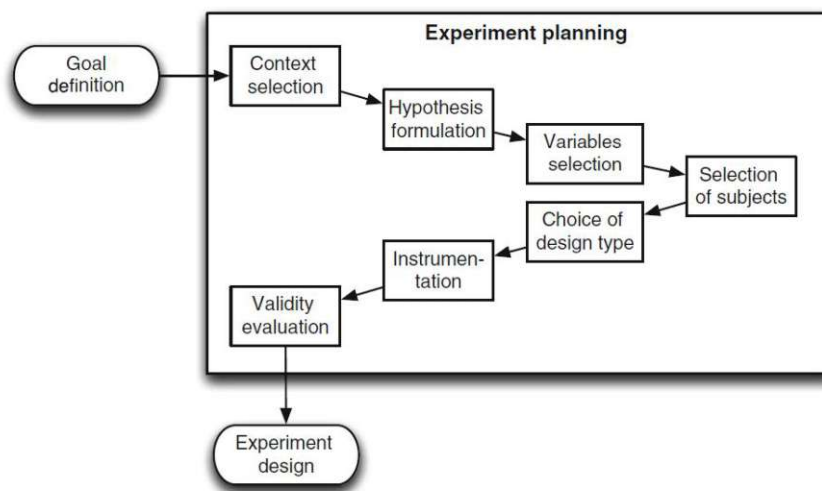


Figure 5.2: Overview of the experiment planning phase [29]

crowdsourcing platform, that allows for distributed workers to complete tasks virtually. Requestors have to build their work package in HITS (Human Intelligence Tasks) as part of a Job. These tasks are independent from one another. mTurk offers requestors the flexibility of implementing individual user interfaces through HTML, so that the tasks are fitting as well as possible. The platform allows for a monetary reward for workers that complete tasks or jobs, as well as a restriction of possible workers based on different criteria as demographics, language, specific qualifications, etc. mTurk offers a free of charge Sandbox where HIITs can be developed and tested before they go live and the experiment will run in such an environment.

5.2.2 Hypothesis formulation

The hypothesis is the basis of the study and it has to be proven or refuted. The null hypothesis is the one that has to be formulated and after collecting the data, rejected. By that means, conclusions can be drawn. The alternative hypothesis is the one in favor of which the null hypothesis is rejected. [29]

For this experiment, which answers two research questions, there are two null hypothesis to be refuted:

H_{0_1} : Using the VeriCoM approach to solve errors regarding Ontology Engineering with Human Computation leads to low performance (under 70%) of the contributors (or, in other words, VeriCoM is not a good methodology for these type of problems).

H_{0_2} : Considering error classification, the overall performance changes drastically (over 30%), when having multiple error types in one Human Computation task and when the variety of errors increases from three to five.

Therefore, there are two alternative hypothesis, answering the RQ2 and RQ3 respectively:

H_1 : The VeriCoM approach is a good methodology to solve problems regarding ontology engineering with Human Computation, because of the high performance (over 70%) of the workers.

H_2 : Considering error classification, the overall performance does not change drastically (over 30%), when having multiple error types in one Human Computation task and when the variety of errors increases from three to five.

There are two types of risks when doing hypotheses testing: type-I-error which proves the null hypothesis to be true, and type-II-error which consists of not being able to refute the null hypothesis. These risks have to be considered when planning an experiment. [29]

5.2.3 Variables selection

Independent variables can be changed and controlled in the experiment and they have effects on the dependent variable. The dependent variable is derived directly from the hypothesis, and while choosing the dependent variable, the measurement scale and range are also determined. [29]

As independent variable for this experiment, the *number of errors in each task* was considered. This means that the models to be verified by the workers in a task had different number of errors, respectively. The dependent variable is the *accuracy of the results*, or more precise the *percentage of correct responses*, as well as the *speed of the verification*.

5.2.4 Selection of subjects

According to [29], the selection must be representative for that population and the sampling can be either a probability or a non-probability sample. The difference is that for the first one, the probability is known, while for the second one unknown. The size of the sample also impacts the generalization of the conclusions.

As stated above, the *students* of the Vienna University of Technology (TU Wien) course 188.399 Introduction to Semantic Systems were the subjects of the experiment. One of the reasons for choosing internal students over a layman crowd is because a basic knowledge of ontologies and possible errors is required, in order to fulfill the tasks. Not only would this be hard to require from laymen workers on a crowdsourcing platform, but it would also be hard to filter out the spammers. The students of the course had to choose if they attend the experiment or not, and were given extra points in the above-stated course, depending on the participation and the quality of work (over 50% score) resulting from the experiment.

There were 61 subjects that accepted participating in the experiment, all attending the course 188.399 Introduction to Semantic Systems of the winter term 2021, master students of the following study programs: *Data Science, Business Informatics, Information & Knowledge Management, Medical Informatics, Software Engineering & Internet*

Computing. Even though it was expected for them to have at least basic knowledge on the subject, a self-assessment test and a qualification test were conducted as part of the pre-study, in order to ensure this basic knowledge.

The self-assessment test is a subjective way to qualify the subjects. They were asked to rate their English language skills, their experience with formal logic, model-driven engineering, ontology modelling and web-based knowledge representation languages. According to [28], previous knowledge in this field has an impact on the accuracy of the interpretation, so for measurement purposes, the same scale was used:

- 1 - novice: no knowledge
- 2 - beginner: little knowledge
- 3 - intermediate: some knowledge
- 4 - expert: expert knowledge

Appendix [Appendix A: Self Assessment Test](#) contains the entire self-assessment test, which was provided as a Google Form² to the experiment subjects. Figure [5.3](#) shows the summary of the subjects answers. While not all the participants took the self assessment test, one can say that almost all participants have some or expert knowledge of the English language, which is needed in order to understand the tasks of the experiment. Regarding the assumed basic knowledge in the discussed field, the majority of the subjects have intermediate experience with Formal logic and Model-driven Engineering, while the vast majority have at least little knowledge over Ontology Modelling and Web-based knowledge representation languages, which is both explainable due to them being master students in the stated IT programs.

In order to have also an objective view over the base knowledge of the subjects, a mandatory qualification test was conducted, depicted in Appendix [Appendix B: Qualification Test](#). The Sections 1,2 and 3 of the Quality Test test the levels beginner, intermediate and expert respectively. Section 1 briefly checked the understanding of the main model components (classes and relations), as well as the understanding of the universal and existential restriction, and the ability to recognize them in a graphical representation. Section 2 focuses on the implication of ontology axioms and restrictions, while Section 3 covers the idea of reasoning with ontology models, as well as comparing and relating models to one another. The subjects were able to score 4 points per section, and the algorithm for the classification worked as following:

- scorePerSection: the score of the subject per Section, with 1 point for each correct answer
- scoreTotal: the sum of the scores of each Section

²<https://www.google.com/forms/about/>

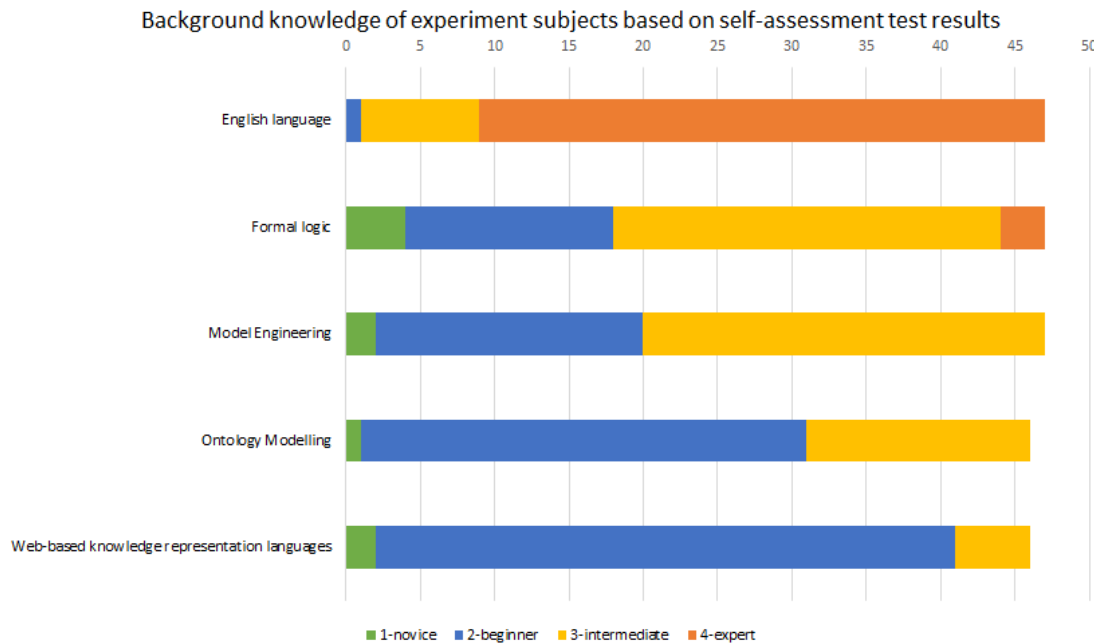


Figure 5.3: Background knowledge of experiment subjects as per self-assessment

- grading:
 - any $\text{scorePerSection} < 3$: beginner
 - scorePerSection for Section1 ≥ 3 and $\text{scoreTotal} \geq 4$: beginner
 - scorePerSection for Section2 ≥ 3 and $\text{scoreTotal} \geq 8$: intermediate
 - scorePerSection for Section3 ≥ 3 and $\text{scoreTotal} \geq 10$: expert

After analyzing the participation in the Qualification Test, 45 subjects continued the experiment. Figure 5.4 shows that most of the participants were qualified as beginners by the above explained algorithm, almost a third as expert and the rest as having some knowledge in the field, determining the assumed basic knowledge of the students, in order to be able to participate in the experiment.

5.2.5 Choice of design type

Designing the experiment plays a key role in proving or refuting the hypotheses and also influences directly the analysis of the data [29]. The existence of *general design principles* [29] eases the conception of an experiment:

- Randomization - applies for objects, subjects and the order of the tasks, in order for the observations to be independent from specific factors

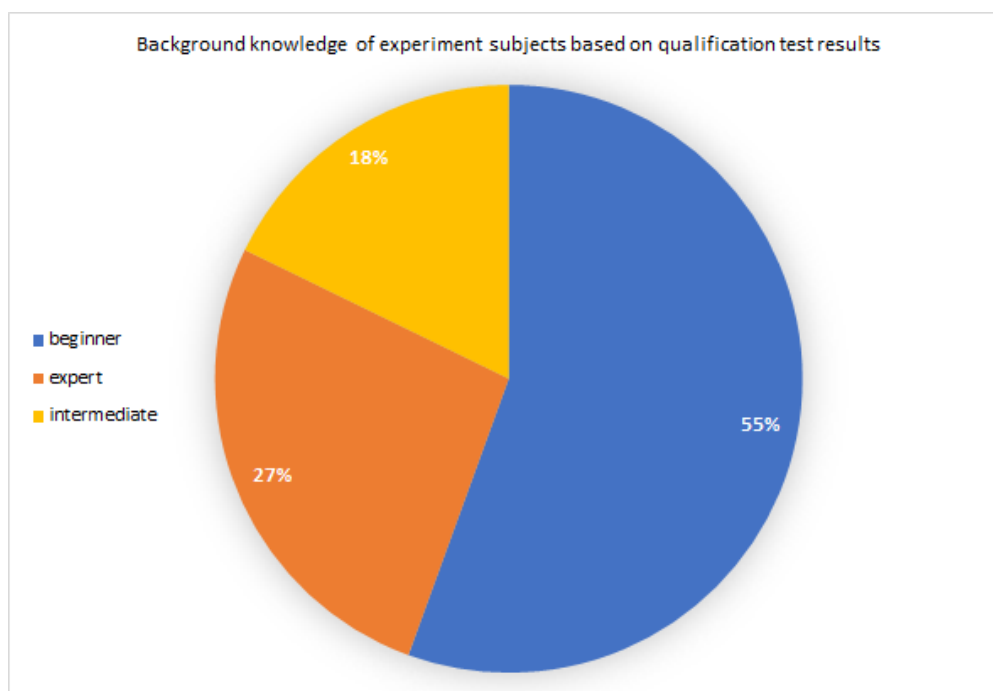


Figure 5.4: Background knowledge of experiment subjects as per qualification test

- Blocking - eliminating a specific factor that has an effect on the result in which one is not interested
- Balancing - each tasks should have an equal amount of subjects

In this experiment, the principle of randomization is used: the order of the tasks for the subjects is randomized, and as a consequence the number of errors in each task.

While there are different *standard design types*, as explained in [29], the one that fits most is one factor with two treatments: in order to extend the experiment of [27] and prove H_2 : *Considering error classification, the overall performance does not change, when having multiple error types in one Human Computation task and when the variety of errors increases from three to five*, the factor of this experiment is the number of errors per task and the treatments are the new and the old type of questions and errors. While using a paired comparison design, the precision of the experiment of [27] is increased by expanding the error types and by changing the single error tasks into mixed error tasks, such that a conclusion regarding H_2 can be drawn.

5.2.6 Instrumentation

There are three types of instruments for an experiment [29]: objects, guidelines and measurement instruments.

An object of the experiment is the ontology used as initial data. This had to be determined: either a well-known, correct and easy for beginners ontology, where the errors discussed in Chapter 4 have to be manually planted into, or a beginner's ontology which already has all the considered errors, but for which a golden standard has to be built. In order to compare the results with the results of [27], the decision fell on using the same ontology: the well-known Pizza ontology [23], developed for educational purposes. The defects D1 to D5 were manually planted into the ontology and the experiment tasks were built.

Regarding guidelines, there were a number of resources available:

- presentation slides containing information about the context and the motivation of the experiment, the topic, the tools and platforms used, the reward for attending and the timeline;
- a Google Site³ containing the presentation, extra information, prerequisites and checklists;
- a tutorial for the crowdsourcing platform, as part of the Experiment operation [5.3];

For the measurement, the answers are collected by the used platforms and are available for download, analysis and interpretation in a text file format (CSV).

5.2.7 Validity evaluation

The results of the experiment should have an "adequate validity" for the population of interest [29], so threats to the validity must be evaluated, for each step of drawing conclusions:

- Conclusion validity: there should be a statistical relationship between the treatment and the outcome
- Internal validity: the relationship between the treatment and the outcome should be causal
- Construct validity: the treatment and the outcome have to reflect the construct (the theory)
- External validity: concerns the generalization of the study

An example for a threat to the conclusion validity of the experiment could be *the number of errors in a task being too low*, but since the experiment has different tasks with different occurrences of errors, this should not affect the conclusion validity. Regarding internal validity, a threat could be that *the Pizza ontology used might be too simple, so that the number of errors in a task or the error types do not make a difference*. Since most of

³<https://workspace.google.com/products/sites/>

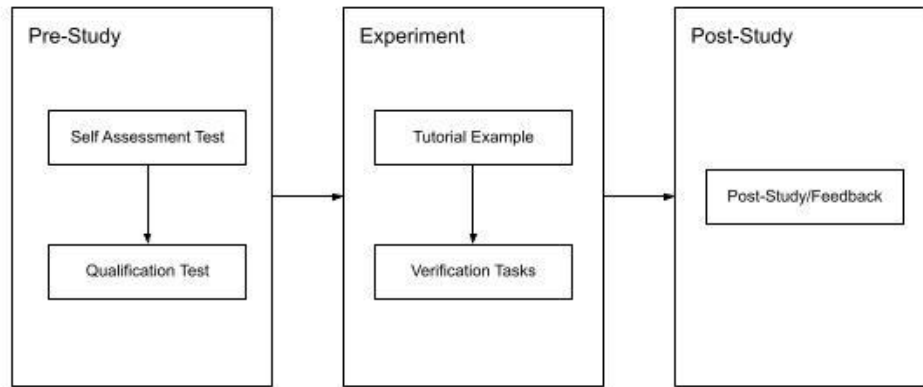


Figure 5.5: Experiment operation steps

the participants have beginner knowledge in the field, this threat should also be resolved. For construct validity, *the choice of the specific defects* might be a threat to the change of performance, but they are various and could stand as a good example for Ontologies in general. The last concern is the external validity and an example for a threat could be *inappropriate subjects*, which should also not be the case, since the subjects had to pass a self-assessment and a qualification test in order to attend the experiment.

5.3 Experiment operation

As explained by [29], there are three steps in the operation of an experiment:

- Preparation
- Execution
- Data validation

For this thesis' experiment, these steps can be seen in Figure 5.5 and are going to be discussed below.

5.3.1 Preparation

The first step was to prepare the data for each phase the subjects are going through. Firstly, the questions for the Self-assessment test were prepared, as described in Chapter 5.2.4. Since the Self-assessment is a subjective way to find out the knowledge of the subjects, a Qualification test 5.2.4 was needed and the decision fell on preparing this test

in the same platform as the experiment itself, Amazon Mechanical Turk, described in [5.2.1](#).

Afterwards, a Tutorial for the experiment was developed, in order for the subjects to get a first impression on how the tasks are designed, how the platform is working and what is to be expected of the experiment. The tasks of the Tutorial followed the designed explained in Chapter [4.3.2](#), but the ontology used was a simple Wine ontology with a few wine types and a few grape sorts.

For the experiment itself, firstly the data for the tasks was prepared. Using the Pizza ontology and the defect errors explained in Chapter [4.1](#), 30 tasks were created, according to Chapter [5.2.6](#), as following:

- 3 tasks containing D1 only
- 3 tasks containing D2 only
- 4 tasks containing D3 only
- 2 tasks containing D4 only
- 2 tasks containing D5 only
- 2 tasks containing D1 and D4
- 2 tasks containing D1 and D5
- 2 tasks containing D2 and D5
- 2 tasks containing D3 and D4
- 2 tasks containing D2, D4 and D5
- 6 tasks containing no defect (correct modelling)

For each mTurk Job: Qualification Test, Tutorial and Experiment, individual HTML files were created, in order to achieve proper user interface as explained in Chapter [4.3.2](#).

Regarding the Post-Study phase, Appendix [Appendix C: Post-study Questionnaire](#) contains the entire feedback questionnaire for the students, which was provided as a Google Form⁴. They also got feedback in form of an e-mail, in which their responses were analysed. This feedback e-mail is discussed in Chapter [5.3.3](#).

The resources explained in Chapter [5.2.6](#) were prepared and the students were asked to make sure they have as prerequisites:

- a Google Account⁵: for the Self-assessment test and the Feedback Questionnaire

⁴<https://www.google.com/forms/about/>

⁵<https://accounts.google.com/signup?>

- an Amazon Account⁶ and an Amazon Mechanical Turk Sandbox Worker Account⁷ for the Qualification test, the Tutorial and the Experiment itself

5.3.2 Execution

For the execution itself, the students were split into two slots and got all the information regarding the experiment one hour before the start, per email. They were required to complete, in order, the steps presented in Figure 5.5: firstly the Self Assessment test, then the Qualification test, the Tutorial example, afterwards the Experiment tasks themselves and lastly the Post-study feedback. The time-frame for all the steps was two hours which could be split as desired. To ensure a smooth communication and support, the students could join a Zoom⁸ meeting for any questions or problems during the experiment. The data collection was done automatically by the specific tools in each step and monitored live.

5.3.3 Data validation

The automatic collection of the data during the experiment allowed a live monitoring and first validation. Nevertheless, a script was written to interpret the data, validate it and give a feedback to each student participating in the experiment, around one hour after the end of the experiment. As it can be seen in the feedback template below, each student got informed about their submission, level, correct answered questions, majority agreement and achieved points:

Dear X X,

Thank you for participating in the Ontology Verification Quiz within the Introduction to Semantic Systems Course!

We have automatically assessed your submitted verifications by comparing these with our ground truth data as well as with the judgements of your fellow students participating in this Quiz. Remarks added as comments will be analysed manually after the Quiz.

Based on this automatic assessment, your performance, per task is as follows:

Self-Assessment Test: XXXXX (submitted/not-submitted)

⁶<https://www.amazon.com/>

⁷https://requester.mturk.com/signin_options

⁸<https://zoom.us/>

Qualification Test

You answered XX/11 questions correctly. This has categorized your knowledge as XXXXX ('beginner'/'intermediate'/'expert') in Ontology Modelling.

As a reminder, your own assessment in the self-assessment test was that you consider yourself a XXXXX ('novice'/'beginner'/'intermediate'/'expert') .

Tutorial: X/7 HITS were submitted.

Quiz:

Submitted HITS: XX/30

You answered XX% correctly, which is XXXXX (above/bellow) the average performance of students in the same group.

Feedback Form: XXXXX (submitted/not-submitted)

We estimated that you verified XX ontology restriction axioms correctly, which makes your defect detection precision XX%.

For XX% of the ontology models you assessed, you were in agreement with the majority of the other students who evaluated these models.

For participating actively during the workshop, you will receive X points for participation and X points for the quality of the work.

Thank you again for participating in this workshop and supporting our research!

Best regards,

Your ISS Ontology Verification Quiz Team

In the few cases where a step of the experiment was missing, either the respective student got informed to complete the missing step while the experiment was on going, or they responded to the feedback email that something was missing. This allowed immediate correction of incomplete data.

5.4 Experiment analysis and interpretation

Following [29] in order to analyse and interpret the data, firstly it has to be described and then reduces to a data set of valid data. Afterwards the data is examined in accord with the hypothesis.

The students were informed that all the steps shown in Figure 5.5 are to be completed. In order to link the results to the students, for them to get the extra points for the course, the participation to the Qualification test was required. Also, participating in at least one more step besides the Qualification test and the experiment itself was mandatory to get participation points. These requirements filtered out subjects which were not planning to thoroughly participate in the experiment.

After reducing the data to a valid data set, there was a total of 1293 responses for 30 tasks. The table 5.1 shows the data for each task: the number of responses, the correctness per task, the average response time, as well as the accuracy of the majority, which is 100% for all the tasks.

The two dependent variables to be verified are, as described in Chapter 5.2.3, the accuracy of the results, better said the percentage of correct responses, as well as the speed of the verification. The overall accuracy was at 78% and the average response per task took around 55 seconds. Figure 5.6 shows how the accuracy remains more or less constant over all the 30 tasks of the experiment, being already at a relative high level. The time that subjects took to complete the tasks can also be observed in Figure 5.6: it was initially slightly higher, which can be explained by the subjects still getting used to the platform and tasks, but gets shorter and normalized over the time, afterwards remaining constant except for one singular point. This means that the students got better with time in identifying the defects, since the accuracy remained high, but the time to complete a task shortened.

While resolving the tasks, the subjects could write a comment regarding that specific task. Most of the comments were a justification for the chosen answer. Some of the comments concerned defect D5 and the correct answer for that: "The set operator intersection (\cap) needs to be replaced by the set operator union (\cup)." The subjects were missing an "Add an union operator" answer possibility, probably misunderstanding the above written possibility. One comment was stating that the answer possibility for defect D4: "Disjointness between entities needs to be added." was not stating if it applies for two or all the classes. Some of the models were found "confusing" or "not readable", because there were a lot of classes and relationships between them and the models looked crowded.

In order to get feedback on the experiment, a Post-study questionnaire presented in Appendix 6.2 was available. The questions covered:

- the usefulness of the tutorial, the instructions and the examples
- the awareness regarding the chosen defects
- the difficulty level of the tasks
- the understanding of the concepts (the existential and universal restriction, disjointness, union and intersection)
- general feedback

task_ID	responses	correct	time_per_resp	accuracy_majority
1	43	71.94%	44.54	1
2	43	71.39%	51.15	1
3	43	93.52%	34.17	1
4	43	98.61%	40.53	1
5	43	94.91%	48.75	1
6	43	83.52%	40.39	1
7	43	86.85%	35.70	1
8	43	82.78%	78.53	1
9	43	98.61%	55.13	1
10	43	95.28%	77.91	1
11	43	64.72%	44.16	1
12	43	66.11%	38.12	1
13	43	59.63%	100.88	1
14	44	65.83%	54.74	1
15	43	79.81%	46.18	1
16	43	71.39%	72.82	1
17	43	69.44%	69.22	1
18	43	59.17%	61.33	1
19	43	63.98%	49.52	1
20	43	54.20%	45.06	1
21	43	69.07%	42.17	1
22	43	38.70%	62.20	1
23	43	65.37%	45.04	1
24	44	61.39%	57.04	1
25	44	87.22%	49.23	1
26	43	83.15%	57.75	1
27	43	64.72%	79.33	1
28	44	63.89%	71.72	1
29	43	75.09%	49.82	1
30	42	73.03%	67.18	1

Table 5.1: Performance of subjects in verifying each of the 30 tasks

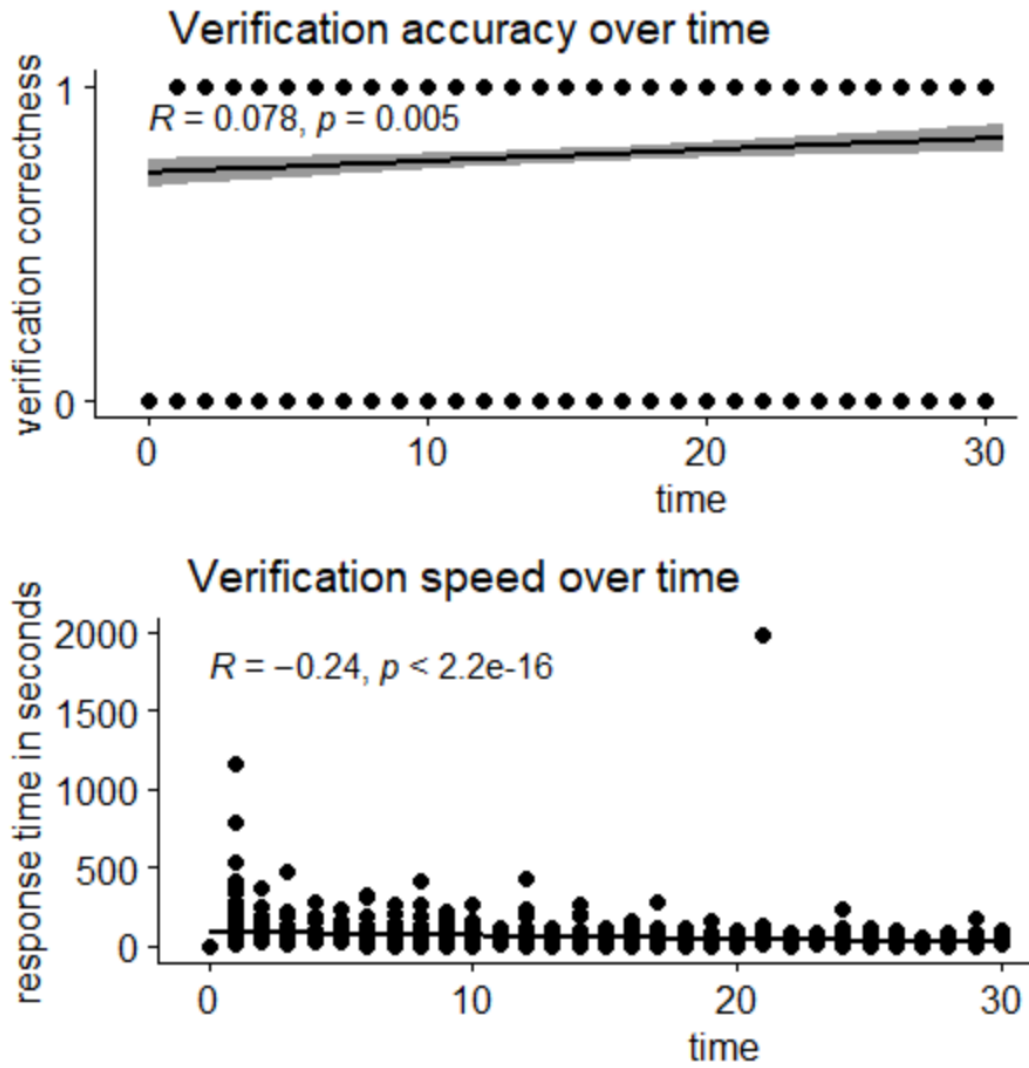


Figure 5.6: Accuracy and speed of subjects' responses over time

Almost all the students found the tutorial, the instructions and the examples useful or partially useful, while 85% stated to have used the instructions at least sometimes while resolving the tasks. Considering the awareness of the presence of the chosen defects in ontology engineering processes, 2/3 of the participants stated that they were aware or partially aware of it. Around 80% of them stated also that it was easy to recognise invalid representations of the models. The question regarding the understanding of the concepts got some different answers, but still the majority of the subjects agreed on the concepts being clear, or getting clearer with the help of examples or through practice, while advancing with the tasks. As general feedback, the subjects seemed contempt with the experiment process and the experiment itself. Most of the negative feedback regarded the tasks being repetitive and the existence of a pattern in the questions, which could be easily predicted.

5.5 Experiment presentation and package

One of the final steps of the methodology explained by [29] is the presentation and package of the experiment which, for the actual case, is done through this thesis.

5.6 Experiment summary

This chapter synthesises the experimental process developed and run, in order to test the VeriCoM approach on an Ontology use case, as explained in Chapter 4. The second purpose of the experiment was to analyse the changes in the overall performance of the subjects, when variables of the experiment change: the variety of Ontology engineering errors, as well as different occurrences of the defects per task.

This experiment was building on top of the work done in [27]. Their experiment was similar, but the author considered the defects D1 to D3 in singular error tasks. This means that each task contained a model with only one possible error. The overall performance in their experiment was at 92.58%, with a single verification taking around one minute. Compared to that data, the fact that the error types increased from three to five, but also that the tasks got multiple errors, respectively, sunk the performance of the subjects at 78%, while the response time remains on average at slightly less than one minute.

After analyzing the collected data, in the context of the hypothesis defined in Chapter 5.2.2, one can without a doubt refute the two null hypothesis as following:

H_{0_1} : Since the performance of the contributors is at an average of 78%, using the VeriCoM approach to solve errors regarding Ontology Engineering with Human Computation does not lead to low performance of the contributors.

H_{0_2} : Compared to the previous experiment of [27], where the average performance was at 92.57%, the actual performance dropped by only 14.58%, such that the overall performance did not change drastically when having

multiple error types in one Human Computation task and when the variety of errors increases from three to five.

Therefore, the two alternative hypothesis, answering the RQ2 and RQ3 respectively, are proven:

H_1 : The VeriCoM approach is a good methodology to solve problems regarding Ontology Engineering with Human Computation, because of the high performance (over 70%) of the workers.

H_2 : Considering error classification, the overall performance does not change drastically (over 30%), when having multiple error types in one Human Computation task and when the variety of errors increases from three to five.

Interpreting the comments and the feedback given by the students after the study, one could regards this aspects for possible future works:

- Having a tutorial, as well as instructions and examples that can be accessed during the tasks, is considered useful for solving problems
- Increasing the variety of the defects and questions, such that they are not repetitive and patterns cannot be easily recognised
- Choosing models or preparing data which is as small as possible, in order for the model to be readable
- Improving answering possibilities, such that they are unambiguous

While this experiment was a second of its kind, extending the work of [27], it still allows for future works, where not only the points stated above can be improved, but also the crowd or the types of defects can be extended.

Conclusion

Ontologies are used as conceptual models and knowledge representation and a lot of application rely on them, so one of the most important characteristics they have is their quality. Ensuring that quality errors do not propagate when reusing ontologies is crucial, because wrongly represented information can lead to false outcomes of application and therefore the quality of an ontology can be a deciding factor for the success of the system using it.

The literature [19], [23], [28] shows that the process of Ontology engineering is error prone and common errors and pitfalls are identified, but this thesis extends this by considering beginners' ontologies, with no experience in the field.

Because of the liability to errors, the need for Ontology evaluation arises. While automated approaches exist, there are still errors that cannot be identified through such methods, because they need background information and human knowledge. For such cases, Human Computation and Crowdsourcing can be applied. While [24] developed an approach called VeriCoM for verifying conceptual models, this has only been used for a Software Engineering use case. This thesis successfully applies the VeriCoM approach on an Ontology engineering use case, by considering the following errors regarding incorrect modelling:

- D1: missing existential restriction
- D2: universal restriction instead of existential restriction
- D3: missing universal restriction
- D4: missing disjointness
- D5: confusion between linguistic and logical "and" (intersection versus union)

An experiment is conducted, in order to check if the VeriCoM approach is fitting for the explained Ontology engineering use case, but also to compare the performance of the crowd in the field of Ontology verification with the work of [27], when changing parameters as error types or the variety of errors in one Human Computation task.

The following sections provide a discussion of the research questions and the main contributions of the thesis, as well as possible future work that can be conducted in the field.

6.1 Research Questions Discussion

The Research Questions defined in the beginning of this thesis are going to be discussed, in order to show the contribution of the thesis to the Human Computation and Ontology Evaluation fields of research:

1. **RQ1:** What are typical engineering errors in beginners' ontologies in practise, compared to the common errors introduced in the literature?

The literature describes a variety of engineering errors regarding the ontology, therefore a comparison was carried on beginners' ontologies, in order to acquire information about common novices' ontology engineering errors in practice. To achieve this, two automated tools were used.

With the help of the OOPS! tool [19], the main errors found in novices' ontologies regard:

- readability
- disjointness of classes
- not declaring inverse relationships
- the confusion between logical "and" and "or"

The second tool, based on the Automatic Defect Candidate Detection [20] heuristic, was used for errors that cannot be identified automatically, but for which defect candidates may be elicited. Based on the results, it seems that the main problem of beginners' ontologies constitutes the missing of disjointness axioms, which supports one of the findings of the first comparison tool.

2. **RQ2:** What is a good methodology to solve problems regarding Ontology engineering with Human Computation?

The VeriCoM [25] methodology offers a way of evaluating conceptual models with Human Computation, by formalizing the problem and developing a generic approach. While the VeriCoM approach was exemplified on a Software Engineering use case by its authors, this thesis adapted the approach to an Ontology engineering use case, regarding the following modelling errors: missing existential restriction, universal

restriction instead of existential restriction, missing universal restriction, missing disjointness, confusion between linguistic and logical "and" (intersection versus union). All the stages of VeriCoM were applied, considering the enumerated defects, on a specific, well-known ontology and an experiment was conducted, in order to find out the performance of subjects, when verifying the presence of these defects in Human Computation tasks.

The experiment shows that, when using the VeriCoM approach to design Human Computation tasks, the average performance of the workers is at 78% and the average speed of completing a task is at 55 seconds, proving that this Human Computation methodology is able to achieve high performance regarding the verification of specific defects in ontologies. This means that the VeriCoM approach and Human Computation are a viable alternative to expert evaluation of ontologies, which are expensive, time-consuming, and have limited scalability.

3. **RQ3:** Considering error classification, to what extent is the overall performance influenced, when having multiple error types in one Human Computation task and when the variety of errors increases from three to five?

The experiment is based on initial work of [27], where a similar experiment was conducted, but with only two of the five defects stated above, present in singular error tasks. This means that each task contained a model with only one possible error. The experiment of this thesis had an increased number of error types of five, but also tasks with multiple errors, respectively, such that the performance compared to the first experiment decreased from 92.58% to 78%, while the response time remains on average at slightly less than one minute.

Even though the performance reduced by 14.58%, considering the fact that multiple error types were present in a Human Computation task and that the variety of error increased, the overall performance is still high and did not change drastically (over 30%). This goes to show that Human Computation is still reliable, even when the ontologies to be verified contain multiple and different errors, and is a viable approach for Ontology verification in general.

As an outline of the main contribution of this thesis, it has been shown (1) what are typical engineering errors in beginners' ontologies in practice, (2) that VeriCoM is a good Human Computation approach to verify ontologies and (3) that considering error classification, having multiple error types in one Human Computation task and increasing the variety of errors from three to five slightly reduces the overall performance of crowd, which still remains at a high level of accuracy.

6.2 Limitations and Future work

This last section concludes the research of the thesis by presenting some limitations and possible future work.

Firstly, the conducted comparison on beginners' ontologies relies on available tools for identifying Ontology engineering errors. Since some of the errors need human knowledge to be identified, only some heuristic approaches are available. As soon as the technology advances in this field, another comparison or even a benchmark can be carried out, to extend or refute the current findings of what are the Ontology engineering errors most present in novices' ontologies in practice.

As for using Human Computation to solve the problem of verifying ontologies, it has been shown that this is a viable approach, but the list of defects it has been researched on is limited. In the literature there are still other defects which cannot be identified automatically, which could be approached with the help of Human Computation and Crowdsourcing. Another future aspect that can be considered is extending the experiment on a larger crowd which doesn't imply only students, to find out if workers without specific knowledge or demographics would still achieve such high performances when evaluating errors in ontologies. Furthermore, the feedback of the students to the experiment lead to a list of possible improvements for next experiments:

- Having a tutorial, as well as instructions and examples that can be accessed during the tasks, is considered useful for solving problems
- Increasing the variety of the defects and questions, such that they are not repetitive and patterns cannot be easily recognised
- Choosing models or preparing data which is as small as possible, in order for the model to be readable
- Improving answering possibilities, such that they are unambiguous

List of Figures

1.1	Research questions, methodologies used and contributions	4
1.2	Overview of the experiment process [29]	5
3.1	OOPS! pitfall distribution in beginners' ontologies	16
3.2	Histogram of each of P08, P13, P11 and P19	17
3.3	Automatic Defect Candidate Detection error distribution in beginners' ontologies	18
3.4	Histogram of Missing disjointness axioms	18
4.1	Correct modelling of Hawaii pizza	22
4.2	Existential restriction example	23
4.3	Universal restriction example	23
4.4	Disjointness example	23
4.5	Union example	24
4.6	Intersection example	24
4.7	D1: missing existential restriction on Pizza Hawaii	25
4.8	D2: universal restriction instead of existential restriction on Pizza Hawaii	26
4.9	D3: missing universal restriction on Pizza Hawaii	27
4.10	D4: missing disjointness on Pizza Hawaii	27
4.11	D5: confusion between linguistic and logical "and" (intersection versus union) on Pizza Hawaii	28
4.12	HIT: Mushroom Pizza with defects D2 and D5	33
4.13	HIT: Instructions	33
4.14	HIT: Detailed Instructions	34
4.15	HIT: Examples	35
5.1	Overview of the experiment process [29]	37
5.2	Overview of the experiment planning phase [29]	40
5.3	Background knowledge of experiment subjects as per self-assessment	43
5.4	Background knowledge of experiment subjects as per qualification test	44
5.5	Experiment operation steps	46
5.6	Accuracy and speed of subjects' responses over time	52

List of Tables

5.1 Performance of subjects in verifying each of the 30 tasks	51
---	----

Bibliography

- [1] The rise of crowdsourcing. <https://www.wired.com/2006/06/crowds/>. Accessed: 2021-06-18.
- [2] Vowl: Visual notation for owl ontologies. <http://vowl.visualdataweb.org/v2/>. Accessed: 2022-04-10.
- [3] H. Alani, C. Brewster, and N. Shadbolt. Ranking ontologies with aktiverank. In *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*. Springer, 2006.
- [4] V. Basili and H. Rombach. The tame project: towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6):758–773, 1988.
- [5] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *ScientificAmerican.com*, 05 2001.
- [6] E. P. Bontas, M. Mochol, and R. Tolksdorf. Case studies on ontology reuse. In *Proceedings of the IKNOW05 International Conference on Knowledge Management*, volume 74, page 345. Citeseer, 2005.
- [7] C. Brewster, H. Alani, S. Dasmahapatra, and Y. Wilks. Data driven ontology evaluation. In *LREC*. European Language Resources Association, 2004.
- [8] A. Burton-Jones, V. Storey, V. Sugumaran, and P. Ahluwalia. A semiotic metrics suite for assessing quality of ontologies. *Data & Knowledge Engineering*, 55:84–102, 10 2005.
- [9] C. J. Cai, S. T. Iqbal, and J. Teevan. Chain reactions: The impact of order on microtask chains. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 3143–3154, New York, NY, USA, 2016. Association for Computing Machinery.
- [10] A. Duque-Ramos, J. Fernandez-Breis, R. Stevens, and N. Aussenac-Gilles. Oquare: A square-based approach for evaluating the quality of ontologies. *Journal of Research and Practice in Information Technology*, 43:159–176, 05 2011.

- [11] M. Fernández, C. Overbeeke, M. Sabou, and E. Motta. What makes a good ontology? a case-study in fine-grained knowledge reuse. In *Proceedings of the 4th Asian Conference on The Semantic Web, ASWC '09*, page 61–75, Berlin, Heidelberg, 2009. Springer-Verlag.
- [12] T. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *Int. J. Hum. Comput. Stud.*, 43:907–928, 1995.
- [13] N. Guarino and C. A. Welty. *An Overview of OntoClean*, pages 151–171. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
- [14] A. Gómez-Pérez. Towards a framework to verify knowledge sharing technology. *Expert Systems with Applications*, 11(4):519–529, 1996. The Third World Congress on Expert Systems.
- [15] A. Maedche and S. Staab. Measuring similarity between ontologies. In A. Gómez-Pérez and V. R. Benjamins, editors, *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pages 251–263, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
- [16] K. Munir and M. Sheraz Anjum. The use of ontologies for effective knowledge modelling and information retrieval. *Applied Computing and Informatics*, 14(2):116–126, 2018.
- [17] E. Paslaru Bontas Simperl, C. Tempich, and M. Mochol. *Cost Estimation for Ontology Development: Applying the ONTOCOM Model*, pages 415–528. Springer Netherlands, Dordrecht, 2007.
- [18] R. Porzel and R. Malaka. A task-based approach for ontology evaluation. *ECAI Workshop on Ontology Learning and Population, Valencia, Spain*, 01 2004.
- [19] M. Poveda-Villalón, A. Gómez-Pérez, and M. C. Suárez-Figueroa. OOPS! In *Innovations, Developments, and Applications of Semantic Web and Information Systems*, pages 120–148. IGI Global, 2018.
- [20] A. Prock. Hybrid human-machine ontology verification: Identifying common errors in ontologies by integrating human computation with ontology reasoners. Master's thesis, Technische Universität Wien, 2021.
- [21] R. Qarout. *Novel Methods for Designing Tasks in Crowdsourcing*. University of Sheffield, 2019.
- [22] A. J. Quinn and B. B. Bederson. Human computation: A survey and taxonomy of a growing field. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, page 1403–1412, New York, NY, USA, 2011. Association for Computing Machinery.

- [23] A. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe. Owl pizzas: Practical experience of teaching owl-dl: Common errors & common patterns. volume 3257, pages 63–81, 10 2004.
- [24] M. Sabou, L. Aroyo, K. Bontcheva, A. Bozzon, and R. Qarout. Semantic web and human computation: The status of an emerging field. *Semantic Web*, 9:1–12, 03 2018.
- [25] M. Sabou, D. Winkler, P. Penzerstadler, and S. Biffl. Verifying conceptual domain models with human computation: A case study in software engineering. In *HCOMP*, 2018.
- [26] S. Tartir, I. B. Arpinar, M. Moore, A. P. Sheth, and B. Aleman-Meza. OntoQA: Metric-based ontology quality analysis. In *Proceedings of IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*, 2005.
- [27] S. S. Tsaneva. Human-centric ontology evaluation. Master’s thesis, Technische Universität Wien, 2021.
- [28] P. Warren, P. Mulholland, T. Collins, and E. Motta. Improving comprehension of knowledge representation languages: A case study with description logics. *International Journal of Human-Computer Studies*, 122:145–167, Feb. 2019.
- [29] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.

Appendices

Appendix A: Self Assessment Test

Self-Assessment Test

Your answers will be treated anonymously. Your name will only be used to connect your answers to your inspection records. No individual information will be made public in any form.

* Required

1. Email *

2. Name *

3. Student ID (Matrikelnummer) *

English-Language Skills

For the question below, please consider the following levels:

1 - no understanding: My understanding of English is very basic.

2 - little understanding: I can understand and use familiar everyday expressions and very basic phrases.

3 - some understanding: I can understand the main points of clear standard input on familiar matters regularly encountered in work, school, leisure, etc.

4 - expert understanding: I can understand the main ideas of complex text on both concrete and abstract topics, including technical discussions in my field of specialization, and can recognize implicit meaning.

4. Q1: How would you rate your understanding of English documents? *

Mark only one oval.

1 2 3 4

no understanding expert understanding

Experience with Formal Logics

For the question below, please consider the following levels:

- 1 - no knowledge: I don't have experience in the area.
- 2 - little knowledge: I am aware of the basic symbolic notation and understand the meaning of logical conjunctions and quantifiers.
- 3 - some knowledge: I have an understanding of logical axioms and can derive the conveyed implications from them.
- 4 - expert knowledge: I fully understand logical axioms and can derive explicit and implicit implications from them.

5. Q2: How would you rate your knowledge of formal logic? *

Mark only one oval.

	1	2	3	4	
no knowledge	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	expert knowledge

General Modeling Skills

For the question below, please consider the following levels:

- 1 - no knowledge = I have no knowledge in the area.
- 2 - little knowledge = I am aware of the basic model components and can recognise them in graphical and textual representations.
- 3 - some knowledge = I have performed modeling as part of my education/study assignments.
- 4 - expert knowledge = I have performed extensive modeling during my professional employment.

6. Q3: How would you rate your knowledge in Model-Driven Engineering? *

Mark only one oval.

	1	2	3	4	
no knowledge	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	expert knowledge

7. Q4: Select the areas in which you already have at least some experience:

Check all that apply.

- Entity–Relationship (ER) diagrams
- Unified Modeling Language (UML)
- Stock and Flow diagram (SFD)
- Causal Loop Diagram (CLD)
- Business Process Model and Notation (BPMN)
- Event-Driven Process Chain (EPC)
- Other: _____

Ontology Modeling Skills

For the questions below, please consider the following levels:

1 - no knowledge = I have no knowledge in the area.

2 - little knowledge = I am aware of the basic components of ontologies and can recognise them in graphical and textual representations.

3 - some knowledge = I have an understanding of the implications of ontology axioms and restrictions.

4 - expert knowledge = I can perform reasoning with ontology models, as well as compare and relate them to each other.

8. Q5: How would you rate your knowledge in ontology modeling? *

Mark only one oval.

	1	2	3	4	
no knowledge	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	expert knowledge

9. Q6: How would you rate your knowledge of web-based knowledge representation languages (e.g., RDF(S), OWL)? *

Mark only one oval.

	1	2	3	4	
no knowledge	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	expert knowledge

10. Q7: Select the areas in which you already have some experience:

Check all that apply.

- Web Ontology Language (OWL)
- RDF Schema (RDFS)
- Simple Knowledge Organization System (SKOS)
- Other: _____

Crowdsourcing

Crowdsourcing is an idea made popular by Jeff Howe where a high number of people can solve tasks that cannot yet be fully automated, online. [The Rise of Crowdsourcing.

<https://www.wired.com/2006/06/crowds/>]

11. Q8: Do you have any experience with Crowdsourcing platforms (e.g., Innocentive, Openideo, Amazon Mechanical Turk, etc.)? *

Mark only one oval.

- Yes
- No

Thank you for your participation!

This content is neither created nor endorsed by Google.

Google Forms

Appendix B: Qualification Test

Qualification Test

Make sure to accept the HIT before you start answering the questions.



Previewing Answers Submitted by Workers



This message is only visible to you and will not be shown to Workers.

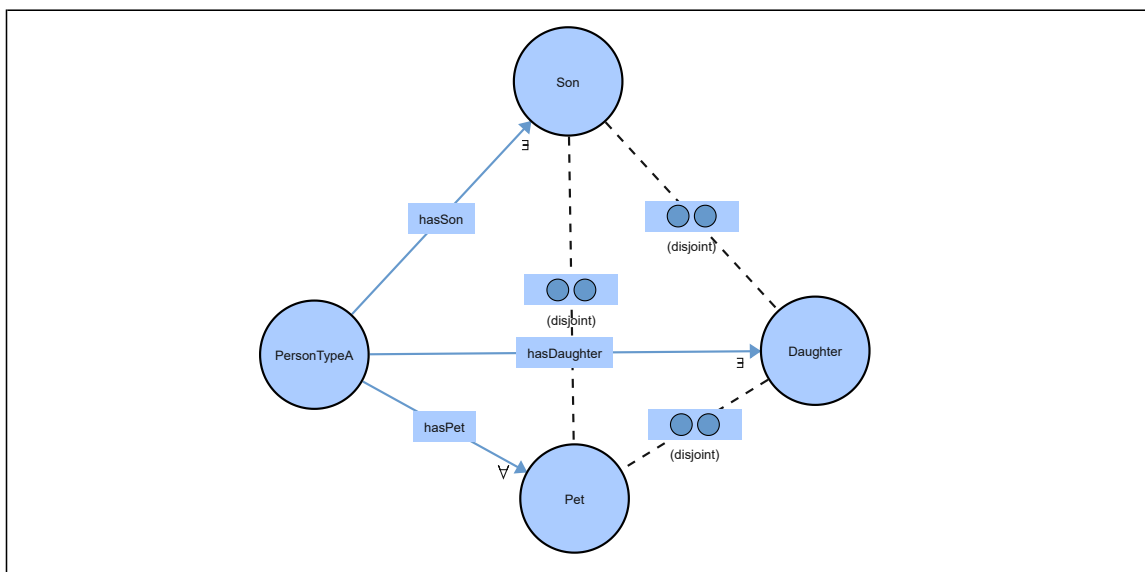
You can test completing the task below and click "Submit" in order to preview the data and format of the submitted results.

Please enter your Student ID:

Section 1

This section tests your understanding of basic ontology components and the ability to recognise them in graphical representations.

Consider the model, represented in VOWL, and answer questions 1 & 2 below.



1. Identify the main model components from the model

How many named classes can you identify from the model?

How many relations (including disjoint relations) can you identify from the model?

2. Identifying the different quantifiers from the model

How many universal restrictions (owl:allValuesFrom) can you identify in the model?

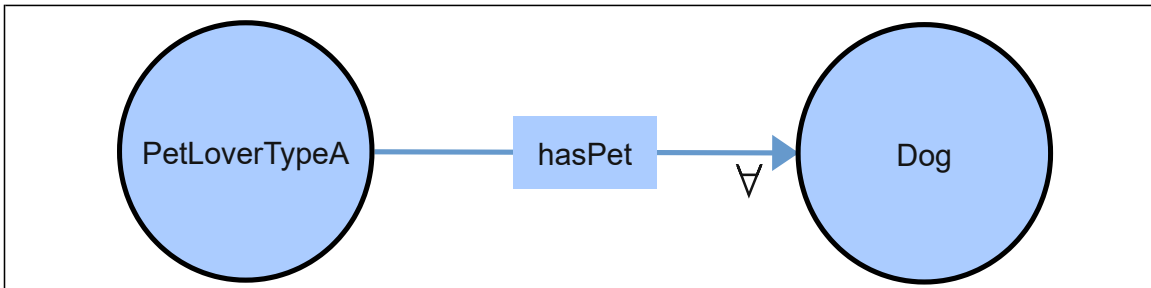
How many existential restrictions (owl:someValuesFrom) can you identify in the model?

[Continue to Section 2](#)

Section 2

This section tests your understanding of the implications of ontology axioms and restrictions.

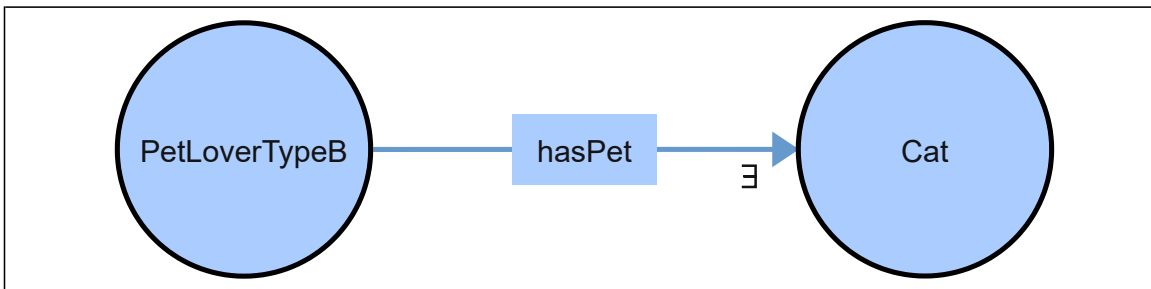
Consider the model, represented in VOWL, and answer question 3 below.



3. Select the statement that describes instances of PetLoverTypeA correctly.

- Instances of PetLoverTypeA must have a Dog pet and cannot have other types of pets.
- Instances of PetLoverTypeA might not have a Dog pet and cannot have other types of pets.
- Instances of PetLoverTypeA must have a Dog pet and can also have other types of pets.
- Instances of PetLoverTypeA might not have a Dog pet and can also have other types of pets.

Consider the model, represented in VOWL, and answer question 4 below.

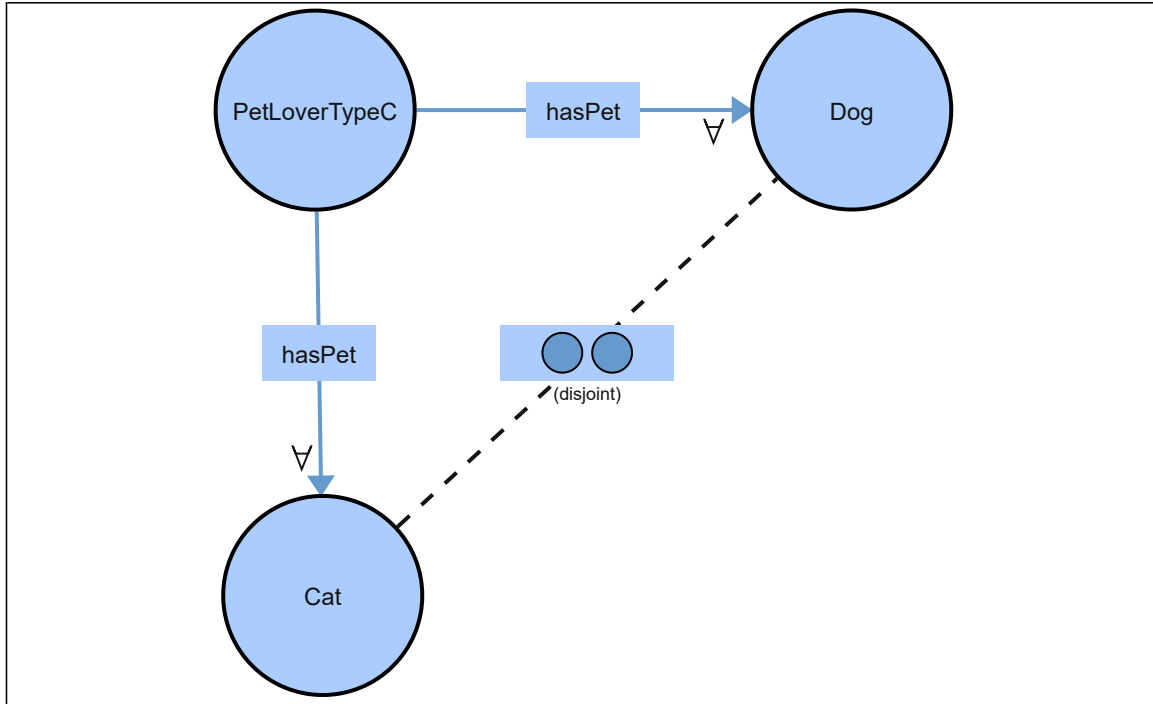


4. Select the statement that describes instances of PetLoverTypeB correctly.

- Instances of PetLoverTypeB must have a Cat pet and cannot have other types of pets.
- Instances of PetLoverTypeB might not have a Cat pet and cannot have other types of pets.

- Instances of PetLoverTypeB must have a Cat pet and can also have other types of pets.
- Instances of PetLoverTypeB might not have a Cat pet and can also have other types of pets.

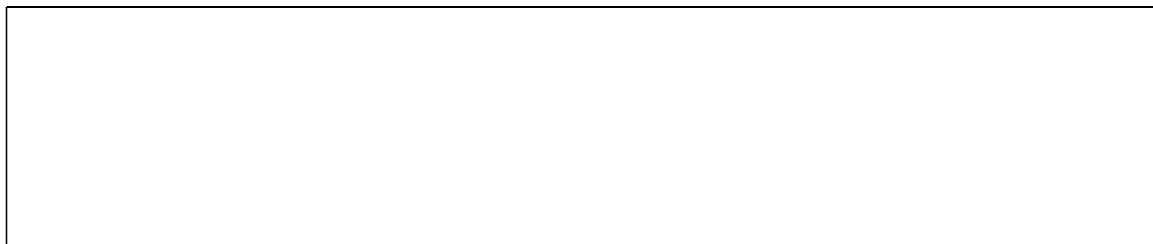
Consider the model, represented in VOWL, and answer question 5 below.

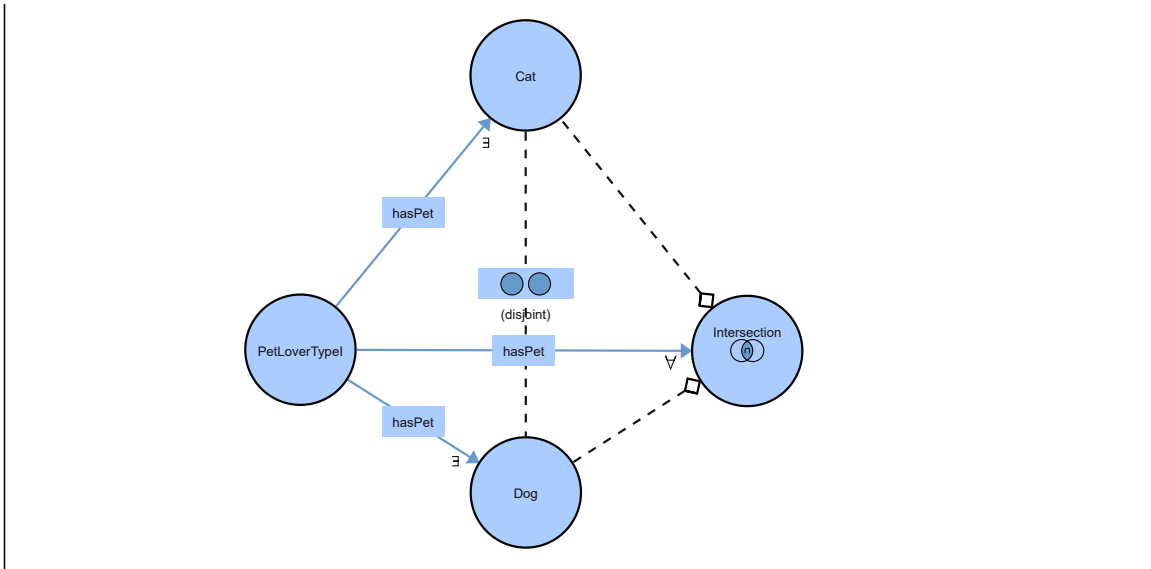


5. Select the statement that correctly represents instances of PetLoverTypeC.

- Instances of PetLoverTypeC must have 2 pets - a Dog and a Cat.
- Instances of PetLoverTypeC might have 2 pets - a Dog and a Cat but also might not have any pets.
- Instances of PetLoverTypeC cannot have any pets.
- Instances of PetLoverTypeC could have 0 to n pets from type Cat or 0 to n pets from type Dog but not both.

Consider the model, represented in VOWL, and answer question 6 below.





6. Select the statement that correctly represents instances of PetLoverType.

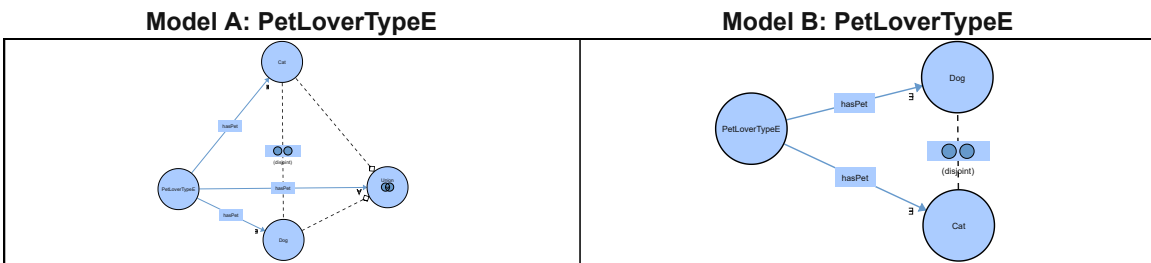
- Instances of PetLoverType must have 2 pets - a Dog and a Cat.
- Instances of PetLoverType must have 2 pets which are both Dog and Cat at the same time.
- Instances of PetLoverType cannot have any pets.
- Instances of PetLoverType could have 0 to n pets from type Cat and 0 to n pets from type Dog but not both.

Continue to Section 3

Section 3

This section tests your ability to reason with ontology models, as well as compare and relate them to each other.

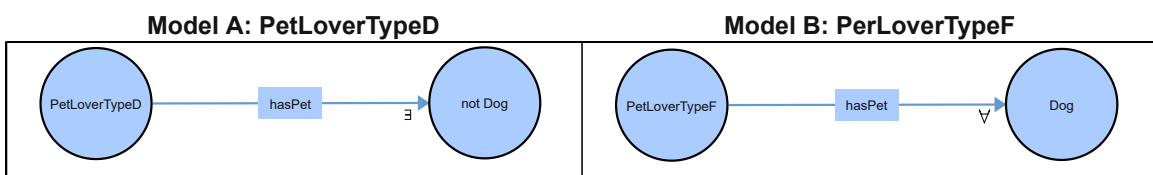
Consider models A and B both describing PetLoverTypeE, each represented in **VOWL**, and answer question 7 below.



7. Select the correct statement about models A and B describing PetLoverTypeE.

- Model A allows for instances of PetLoverTypeE to have a pet that is neither a Dog nor a Cat.
- Model B allows for instances of PetLoverTypeE to have a pet that is neither a Dog nor a Cat.
- None of the models allow for instances of PetLoverTypeE to have a pet that is neither a Dog nor a Cat.
- Both models allow for instances of PetLoverTypeE to have a pet that is neither a Dog nor a Cat.

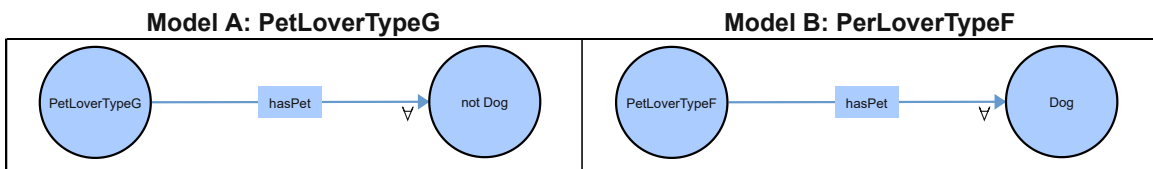
Consider models A and B describing PetLoverTypeD and PerLoverTypeF, each represented in VOWL, and answer question 8 below.



8. Is it true that PetLoverTypeD is disjoint to PetLoverTypeF? That is, there can be no instance that is at the same time of type PetLoverTypeD and PetLoverTypeF.

- Yes
- No

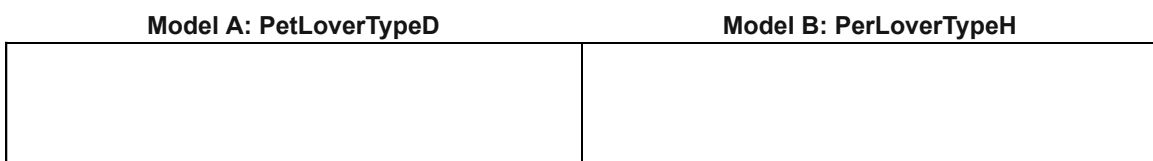
Consider models A and B describing PetLoverTypeG and PerLoverTypeF, each represented in VOWL, and answer question 9 below.

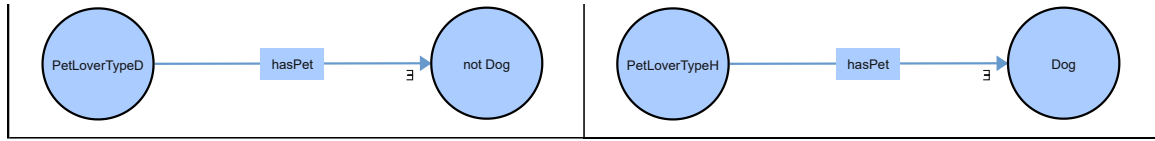


9. Is it true that PetLoverTypeG is disjoint to PetLoverTypeF? That is, there can be no instance that is at the same time of type PetLoverTypeG and PetLoverTypeF.

- Yes
- No

Consider models A and B describing PetLoverTypeD and PerLoverTypeH, each represented in VOWL, and answer question 10 below.





10. Is it true that PetLoverTypeD is disjoint to PetLoverTypeH? That is, there can be no instance that is at the same time of type PetLoverTypeD and PetLoverTypeH.

- Yes
- No

Appendix C: Post-study Questionnaire

Post-Study Questionnaire

Your answers will be treated anonymously. Your name will only be used to connect your answers to your inspection records. No individual information will be made public in any form.

* Required

1. Name *

2. Student ID (Matrikelnummer) *

Feedback Questions

3. Q1: Was the tutorial example useful for understanding the various modelling issues *
that can arise in ontologies?

Mark only one oval.

- Yes
 No
 Partially

4. Q2: Were you aware of (some of) the modelling issues prior to the Quiz?? *

Mark only one oval.

- Yes
 No
 Partially

5. Q3: What could have made the difference between modelling concepts clearer (the existential & universal restriction, disjointness, union & intersection) ?

6. Q4: Was it easy to recognise invalid representations of menu items? *

Mark only one oval.

- Yes
 No

7. Q5: Did you use the provided instructions while performing the judgements? *

Mark only one oval.

- never
 sometimes
 most of the time
 all the time

8. Q6: Were the provided instructions and examples helpful in understanding modelling issues? *

Mark only one oval.

- Yes
 No

9. O7: Is there any feedback you would like to share (general comments, positive aspects, improvements, etc.) ?

Thank you for your participation!

You will receive a feedback email with your performance scores in the next hours/days.

This content is neither created nor endorsed by Google.

Google Forms