

Aalto University  
School of Science  
Master's Programme in Computer, Communication and Information Sciences

Hai Phan

# **An adaptive grading system for semantic network communication compatibility**

Master's Thesis  
Espoo, July 9, 2019

Supervisors: Professor Petri Vuorimaa, Aalto University  
Advisor: D.Sc (Tech) Sanna Suoranta

<b>Author:</b>	Hai Phan	
<b>Title:</b>	An adaptive grading system for semantic network communication compatibility	
<b>Date:</b>	July 9, 2019	<b>Pages:</b> vii + 58
<b>Major:</b>	Computer Science	<b>Code:</b> SCI3042
<b>Supervisors:</b>	Professor Petri Vuorimaa	
<b>Advisor:</b>	D.Sc (Tech) Sanna Suoranta	
<p>The semantic network is becoming more and more important nowadays because it provides a general standard for data formats. In this way, computers can understand and know how to link data to other data. There are many different data formats, and therefore integrity of data must be tested carefully.</p> <p>Nowadays, many programs and IoT-devices collect data, but the data format may not follow the standards, which is why it cannot be used immediately and must be converted in the right format before use. This thesis focuses on testing and correctness of the incoming data. Also, the usability of the SemanticNow application is included.</p> <p>The study aims to develop automatic testing in the ServiceNow platform, which automatically tests ontology that the end-user has created. The name chosen to the application is SemanticNow. This Master's thesis will also include the development of a grading system that motivate the end-user to improve their ontologies. The ontology will have a grade and suggestions on improvements.</p> <p>In the result of the work, the SemanticNow application works as intended. The automated testing pointed out the syntax errors of the Turtle and the ontology creation was successful. The grading system helped the user to improve the ontology creation by showing the ontology mistakes immediately.</p>		
<b>Keywords:</b>	Semantic Web, automated testing, ServiceNow, grading system, RDF, Turtle	
<b>Language:</b>	English	

<b>Tekijä:</b>	Hai Phan		
<b>Työn nimi:</b>			
<b>Päiväys:</b>		<b>Sivumäärä:</b>	vii + 58
<b>Pääaine:</b>	Tietotekniikka	<b>Koodi:</b>	SCI3042
<b>Valvojat:</b>	Professori Petri Vuorimaa		
<b>Ohjaaja:</b>	TkT Sanna Suoranta		
<p>Semanttinen verkko on yhä tärkeämpi nykypäivänä, koska se antaa yleisen standardin data formaateille. Tällä tavoin tietokoneet ymmärtävät ja osaavat linkittää datat toisiinsa. Data formaatteja on paljon erilaisia ja sen takia datan eheys pitää testata tarkasti. Tämä diplomityö keskittyy ontologioiden testaukseen ja arvosanajärjestelmän kehitykseen, jonka tarkoitus on saada loppukäyttäjät parantamaan itse tehtyjä ontologioitaan.</p> <p>Nykypäivänä on todella paljon ohjelmia, jotka keräävät dataa. Datan formaatit eivät ole kuitenkaan välttämättä standardimuodoissa. Tässä diplomityössä kehitetään automaatiotestaus, joka testaa sisään tulevan datan oikeellisuuden.</p> <p>Tutkimuksen kohteena on rakentaa automaatiotestaus ServiceNow alustalla, jossa testataan loppukäyttäjän tuottamaa ontologiaa. Applikaation nimeksi valittiin SemanticNow. Tutkimukseen kuuluu myös sellaisen arvosanajärjestelmän rakentaminen, jossa loppukäyttäjän ontologia arvostellaan, sekä parannusehdotukset annetaan ja arvosana annetaan.</p> <p>Työn tuloksena havaittiin, että SemanticNow toimii kuten oltiin suunniteltu. Automaatiotestaukset näyttivät Turtle syntaksivirheet ja ontologian teko onnistui. Arvosanajärjestelmä auttoi käyttäjää parantamaan itse tehtyä ontologiaa näyttämällä ontologiassa olevat virheet.</p>			
<b>Asiasanat:</b>	Semanttinen verkko, automaatiotestaus, ServiceNow, arvosana-järjestelmä, RDF, Turtle		
<b>Kieli:</b>	Englanti		

# Acknowledgements

I would like to express my gratitude and appreciation to my supervisor professor Petri Vuorimaa for his guidance and encouragement throughout the master thesis. The thesis work has gave me a lot of knowledge and understanding of the importance of Semantic Web. I would like to express my special thanks to Sanna Suoranta for the thesis advice during the process and Jani Hursti for providing an interesting topic for the master thesis.

Finally, I want to thank my family and friends for the motivation and inspiration during the hard times of the thesis. Even after the time when I had a downhill skiing accident, I found strength to continue my thesis thanks to my close friends and family support.

Espoo, July 9, 2019

Hai Phan

# Abbreviations and Acronyms

ACL	Access control list
API	Application Programming Interface
BCG	Breast cancer grading
CMS	Course management system
HTML	Hypertext Markup Language
IDE	Integrated development environment
ITSM	IT service management
JSON-LD	JavaScript Object Notation for Linked Data
OWL	Web Ontology Language
PaaS	Platform-as-a-Service
RDF	Resource Description Framework; A W3C standard for defining linked data
RDFS	Resource Description Framework Schema
S-CMS	Semantic course management system
SaaS	Software-as-a-Service
SPARQL	SPARQL Protocol and RDF Query Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
XaaS	Anything-as-a-Service
XML	Extensible Markup Language

# Contents

Abbreviations and Acronyms	v
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.2 Research Questions and Development Goals . . . . .	2
1.3 Research Methods . . . . .	3
1.4 Thesis Outline . . . . .	3
<b>2 Standards Environment for Semantic Web</b>	<b>5</b>
2.1 Semantic Web . . . . .	5
2.2 Extensible Markup Language (XML) . . . . .	7
2.3 Resource Description Framework (RDF) . . . . .	8
2.3.1 RDF Data Model . . . . .	8
2.3.2 RDF Syntax . . . . .	10
2.3.3 Turtle Syntax . . . . .	11
2.4 Resource Description Framework Schema (RDFS) . . . . .	12
2.5 OWL Web Ontology Language . . . . .	13
2.6 SPARQL . . . . .	13
<b>3 ServiceNow</b>	<b>15</b>
3.1 The ServiceNow System . . . . .	16
3.2 ServiceNow Studio . . . . .	18
3.3 Client Script . . . . .	19
3.4 Business Rule . . . . .	20
3.5 Script Include . . . . .	20
<b>4 Methods</b>	<b>22</b>
4.1 Process . . . . .	22
4.2 Use Cases . . . . .	23
4.2.1 Inserting a Turtle Snippet to ServiceNow . . . . .	24
4.2.2 Editing ServiceNow Ontology . . . . .	24

4.3	User Testing . . . . .	25
4.4	User Types . . . . .	25
<b>5</b>	<b>Grading System</b>	<b>27</b>
5.1	Functionality . . . . .	27
5.2	Related Works . . . . .	28
5.2.1	Grading in Breast Cancer Histopathology Images . . .	28
5.2.2	Course Management System Using the Semantic Web .	29
5.2.3	Scaling of Technical Lignin Grades . . . . .	30
5.3	Design for Grading System . . . . .	32
<b>6</b>	<b>SemanticNow Design</b>	<b>34</b>
6.1	User Access to SemanticNow . . . . .	34
6.2	Application Menu . . . . .	35
6.3	Turtle Validation . . . . .	36
6.4	Turtle Import to ServiceNow . . . . .	37
6.5	Grading . . . . .	37
6.6	Access Control . . . . .	38
6.7	Query in SemanticNow . . . . .	39
<b>7</b>	<b>Implementation</b>	<b>41</b>
7.1	Turtle Validator . . . . .	41
7.2	Creation of Records . . . . .	42
7.3	Grading System and Scale . . . . .	46
<b>8</b>	<b>Evaluation</b>	<b>48</b>
8.1	Evaluating Research Questions and Development Goals . . . .	48
8.2	Evaluation of Process and Design . . . . .	49
8.3	Test Cases and User Feedback . . . . .	50
8.4	Evaluation of SemanticNow Implementation . . . . .	51
8.5	Discussion . . . . .	51
<b>9</b>	<b>Conclusions</b>	<b>53</b>
<b>A</b>	<b>Test Questions and Tasks</b>	<b>58</b>

# Chapter 1

## Introduction

Ontologies are an important part of Computer Science since they are explicit conceptual knowledge models. They are used in information systems to make domain knowledge available. Ontologies are part of Semantic Web and play an important role since they provide the semantic vocabulary, which in turn are used in websites and makes it easier for machines to interpret the information. [11]

The quality of the ontologies determines the machine's understanding of the information on the website. Therefore, the ontologies that are created by users should be tested and graded. This thesis will go through different kinds of automated tests that are made to run through every time user have created or updated ontologies. Another main topic of the thesis is to develop a grading system that will be shown after every test run. The grade would be counted from how well is the ontology made, and the idea of the grading system is to make the user want to improve the created ontology.

The thesis topic is provided by a company called Asema, which is collaborating with Aalto University. Asema aims to address major environmental issues in municipalities and cities by encouraging private sector enterprises and public sector organisations to support smart infrastructures and environmental initiatives [3]. Asema is like a database where people and organisations can put their collected data and sell it to others. For example, an ice cream kiosk owner wants to alter the ice creams' price tag based on how warm or cold the outside temperature is. The kiosk owner can buy the needed temperature values from someone selling them in Asema. This way, the ice cream kiosk owner gets to name the price depending on the weather temperature. If the weather would be hotter than usual, then the price would rise a little bit, and ice creams are discounted if the weather is colder than the usual.

The selected platform to develop the automated tests and grading system

is ServiceNow. ServiceNow is an excellent tool for making applications and is very flexible. The creation of the semantic web in ServiceNow can bring new paths for future development. For example, ServiceNow contains many incident records, and it would be great if similar incidents can be grouped automatically without any manual work. The created application will be called SemanticNow. Another reason to choose ServiceNow as a platform is the ServiceNow Agent, which is an artificially intelligent chatbot. The chatbot helps the user to navigate in the system. When the data is semantic, then this chatbot can even more easily get the needed information and understand the user better.

## 1.1 Motivation

Collecting data is like a trend nowadays. Much information is collected from the user when he or she is using the internet, social media or playing games. The services and games like Google search engine, Facebook or Counter-Strike, are good examples. For example, Google collects words that the user have used in the Google search and uses them to show the relevant advertisements for the user in the Google advertisements [9]. This case was an excellent example of collecting information, using the information and getting a profit of it. The data collection is non-profitable most of the time because there is no idea how to actually make it profitable.

The motivation of this research is to be able to make good use of people's collected data and make sure the data quality is the best possible. The main objective of this research is to get the user to build a high-quality ontology when he or she is inserting data to the system. The quality of the ontology is graded by automated tests that check the created ontology. The user will get feedback after the test has finished. If the received grade is not perfect, the mistakes would be pointed out and the user is recommended to improve the ontology.

## 1.2 Research Questions and Development Goals

There are two main research questions in this thesis:

1. How to define a grading system for ontologies?
2. How to get user inspired to improve his or her ontology creation?

The development goals of this research are:

1. Implementing automated testing solution for ontologies with ServiceNow platform.
2. Designing and implementing a grading system for ontology creation.

### 1.3 Research Methods

The research was conducted by implementing automated tests for ontologies in the JavaScript programming language. The tool that is used to build the whole application is ServiceNow. The point of this work is to create an application that could be used as a plugin in ServiceNow. The created application is called SemanticNow, and the future goal is to get it into ServiceNow Store, where other people can buy the product and plugin to their development instances.

A grading system was implemented, and the grade is calculated by using the self-created automated tests. Different test cases is graded in the grading system, and the overall score is returned. The implementation is also done with JavaScript programming language.

### 1.4 Thesis Outline

The rest of the thesis is organised as follows. Chapter 2 of the thesis goes through the basic knowledge of the semantic web and the most common syntaxes that is used in semantic web. The second Chapter will also cover Web Ontology Language (OWL) and SPARQL protocol and RDF query language (SPARQL). The third Chapter is going to tell about ServiceNow organisation, system and the basic functionalities that are used in this work. The used methods of this thesis will be discussed in the fourth Chapter. The Chapter will go through the process, use cases and the user testing. The fifth Chapter consist of functionalities of a grading system and goes through related works to get ideas for a grading scale that is used in this thesis. In the end of the fifth Chapter, I will talk about the design of the grading system. The sixth Chapter is about the design of SemanticNow, which includes how the user access and navigates in SemanticNow, Turtle syntax validation, grading and querying in SemanticNow. The seventh Chapter is about the implementation of the SemanticNow. It goes through the Turtle syntax validation, creation of records and grading scale. The evaluation of work is in eighth Chapter. I am evaluating the research questions, design of the work, test cases and user feedback and implementation. The eighth Chapter ends

with a discussion. The ninth Chapter consist of the conclusion of the thesis work.

## Chapter 2

# Standards Environment for Semantic Web

This chapter will go through some basic knowledge about the technologies, approaches and terms that are needed to present the information in this research and Web services. First, I will cover the following terms Semantic Web and Resource Description Framework (RDF) that are the basics needed to understand this thesis. Second, I will go through the data model and syntaxes of RDF in a more profound level. The data model visualises the meaning of the RDF and syntaxes envision on the programming level. Third, the knowledge of Web Ontology Language in the semantic web is needed to understand the ontologies. Last section will go through query language for RDF, which is called SPARQL.

## 2.1 Semantic Web

The primary purpose of the semantic web is to make the information on the Internet more understandable for machines. The semantic web links the data to other data to make it even more usable and accessible. The Internet is full of data, and the data can be improved by following the five Linked Open Data principles by Tim Berners-Lee. Just by adding the unique identifiers as URIs to the things improves the data a lot. Data enhancement improves the collaboration between human and machine. [4]

The search engines could be better nowadays if the information on the Internet would be structured and follow the standards of the semantic web. For this reason, search engines cannot provide the exact document that the user is searching for, and some irrelevant documents are included. Searching with keywords often result in a list of documents and part of the result are

not related to the searched information. The needed document may not even result as the first on the search engine's list. The problem is the information that is produced on the Internet is only produced for visualisation and the computers cannot understand the information properly because of this. [2]

The international community called World Wide Web Consortium (W3C) works to develop Web standards and tries to solve the problem that is mentioned above [32]. One of their main objectives is to make the information on Internet readable for computers [32]. The way to display information nowadays is with HyperText Markup Language (HTML), and it is hardly understandable for computers. W3C aim to bring up a new way to display information without deleting the original markup language HTML. Many Web-based applications use the principles or technologies of the W3C Semantic web nowadays.

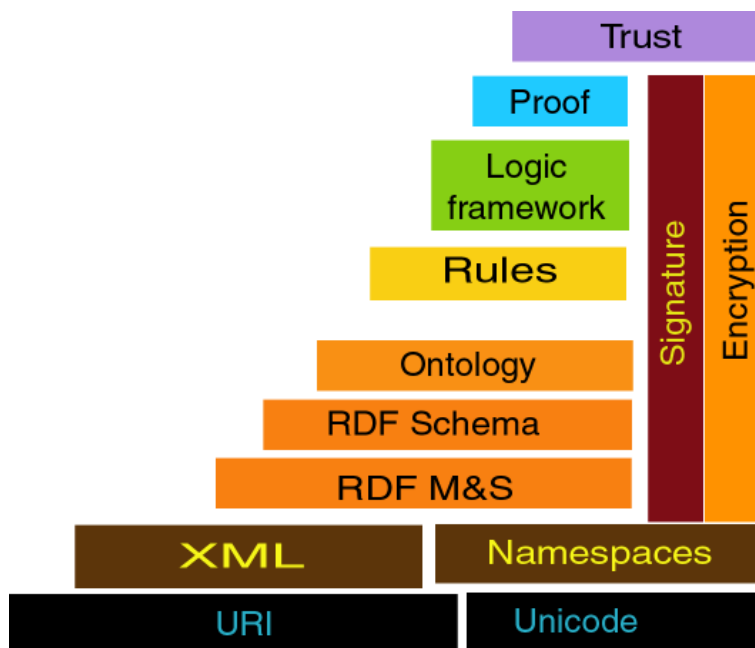


Figure 2.1: Semantic web layer cake [32]

Figure 2.1 depicts the semantic web layer cake that displays a stack where the upper layer is built on top of the lower one. Each layer uses capabilities and exploits the ones below it. This layer cake is the way to display how the standardised technologies for Semantic Web are organised to be able to make Semantic Web functional. Each layer is an extension of the previous ones and not a replacement. The lower layers consist of the standards of W3C: Resource Description Framework (RDF), Resource Description Framework

Schema (RDFS) and Web Ontology Language (OWL), which are all originally from Extensible Markup Language (XML). Next, I will go through the standards that are used in automated testing to get a better understanding of the standard and syntax.

## 2.2 Extensible Markup Language (XML)

Extensible Markup Language (XML) is a markup language which objectives are simplicity, generality and usability everywhere on the Internet. XML has a set of rules to encode documents that makes it both human-readable and machine-readable. Even though the design of the XML is focused on documents, it is still commonly used to represent arbitrary data structures. [35]

The syntax of XML consists of markups and contents where contents are inside markups. The big problem with XML is that it can only describe the content of the document. XML does not have any specific mechanism to describe relationships between resources semantically. Resources are uniquely identifiable objects that can be identified with Uniform Resource Identifiers (URI). In the XML example below, humans can understand that the mail is sent to Anthony, who is a person. This kind of reasoning does not apply for machines because XML cannot interpret the data semantically.

```
<mail id="01">
  <to>Anthony</to>
  <from>Stephan</from>
  <title>Job Interview</title>
  <body>Welcome to job interview on Tuesday 2pm.</body>
</mail>
```

I want to present another example of how the machines cannot understand the data with a graph in Figure 2.2. The figure points out the facts:

1. Micky likes banana.
2. Banana is a fruit.

A human can understand both of the facts. On top of it, the human can interpret that Micky also likes other fruits in general because banana is a fruit. The relationship between the data is clear, but unfortunately, it is not for the machines. The machines cannot understand that Micky likes fruit if it is not defined. This is where the semantic web steps in and creates a

relationship between Micky and the fruit that is depicted in the bottom part of the Figure 2.2. The machine can now understand the linking between these two because the data is semantic. [29]

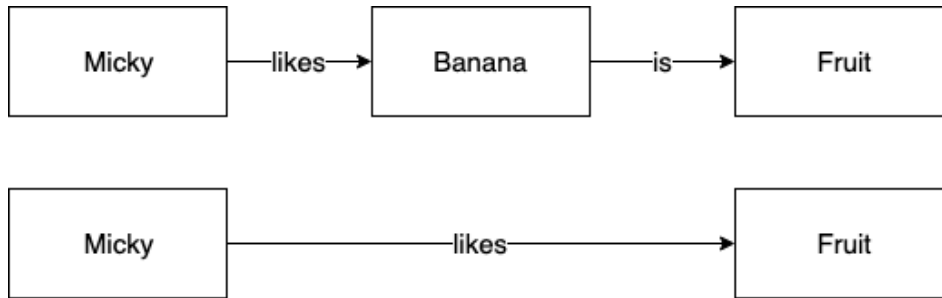


Figure 2.2: Simple data relationship with semantic web

## 2.3 Resource Description Framework (RDF)

Machines can read the XML, but they are not able to understand the data. The web contains much information, which is not machine-understandable because it is unstructured, unsemantic and originally built for human consumption. The metadata is used to describe the data on the web and it can fix the problem of unsemantic data. Metadata can be understood as a "data of data" or more specifically "data describing web resources. [16]

The infrastructure of the Resource Description Framework (RDF) makes it possible to encode, exchange and reuse structured metadata. RDF is an extension of XML and it also makes the machines to understand the data, which means that the metadata is expressed semantically. The common syntax of RDF is XML that is used in exchanging and processing the metadata. The data can be supported with the common conventions of structure, syntax and semantics with the RDF infrastructure. RDF does not make any assumption about a particular application domain and does not define the semantics of any particular application domain. Therefore, RDF is domain neutral even though it is suitable for describing information about any domain. [20]

### 2.3.1 RDF Data Model

The RDF data model is used to represent named properties and property values [16]. The property values can work as attributes of resources and

resemble attribute-value pairs. The basic structure of RDF data model contains subject, predicate and object, which are linked to each other. The benefit of the RDF model is that it also represent relationships between resources, and therefore resembles an entity-relationship diagram [7]. Property types express relationships.

The RDF data model has picked the approved and well-made principles from many different data representation communities. In object-oriented design, however, the resources and properties work differently. The resources correspond to objects and properties correspond to instance variables in the object-oriented design.

The syntax of RDF data model is independent and used for representing resources and their corresponding descriptions [20]. The equivalence of two RDF expressions can be represented with the RDF data model. The rule for two RDF expressions to be equivalent is that their data model representations are identical. The variation in expression can be seen, but the main point of the definition is still the same.

The following two statements are a demonstration of the use of the RDF data model:

1. "Jonathan is the owner of the house."
2. "The owner of the house is Jonathan."

Both statements mean the same for humans because they logically conclude it. These kinds of statements are not the same for machines because machines interpret the statements as strings and evaluate that the statements are different. To make these statements understandable for computers, the RDF model separates the resources, property types and corresponding values of the statement. Then, providing a method of expressing semantics and making encoding to enable the statements to be machine-readable. An example of how the statement is divided in subject, predicate and object is displayed in the table below.

Subject (Resource)	House
Predicate (Property)	ownerOf
Object (Literal)	"Jonathan"

Table 2.1: An example of statement division to subject, predicate and object

In the RDF model in Table 2.1, resources are associating with properties. Therefore, the resource must represent *House* before proceeding forward. The property type of owner and the corresponding value Jonathan must

also be declared. The RDF model meets the requirements for functioning semantically, and it is graphically expressed in Figure 2.3.

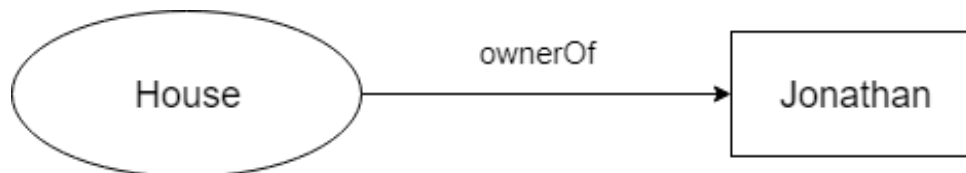


Figure 2.3: RDF graph of house owner Jonathan

### 2.3.2 RDF Syntax

The first syntax of the RDF data model is based on XML syntax, which is also known as RDF/XML [5]. There are many other syntaxes like N-Triple, Turtle, and JSON-LD. Each of the different syntaxes has its benefits. For example, N-Triple enumerates all triples separated by dots, and Turtle is focused on human readability. The RDF/XML syntax is needed to create and exchange metadata [16]. JSON-LD uses a popular JSON text format. RDF creates a proper structure on XML to represent it in semantics [20]. The example below demonstrate one of the possibilities to write information given in Figure 2.3.

```

<rdf:Description rdf:about="http://www.houseowners.org/jonathan">
  <s:hasName>Jonathan</s:hasName>
  <s:owner>House</s:owner>
</rdf:Description>
  
```

The XML namespace mechanism separates different meanings in resource description. XML namespaces can uniquely identify the vocabulary that defines the semantics. For example, the property type "author" can be understood differently when it is compared between Dublin Core vocabulary and Library of Congress vocabulary. The definition of "author" in Dublin Core vocabulary is: "person or organisation responsible for the creation of the intellectual content of the resource" and in Library of Congress vocabulary it is defined as following: "A person, family, or organization responsible for creating a work that is primarily textual in content, regardless of media type or genre". In the Figure 2.4, the vocabulary of Dublin Core is used in predicate by adding the abbreviation of Dublin Core and colon before "Author". The usage of namespaces makes the difference of how the predicate "Author" is supposed to be understood. [12]

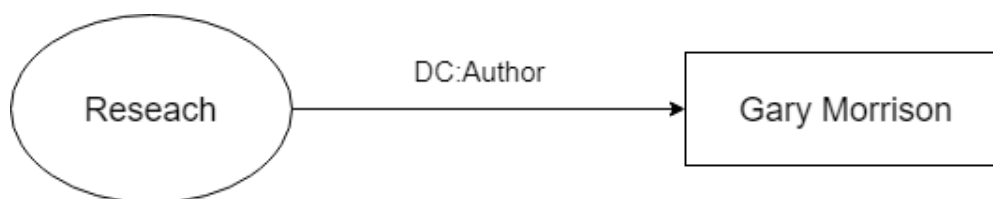


Figure 2.4: The usage of Dublin Core vocabulary in RDF Graph

### 2.3.3 Turtle Syntax

One of the RDF syntax and serialisation that I am going to go through is Turtle syntax. Turtle is one of many serialisations of RDF and is more intuitive for humans to read and write. Turtle is an extension of the N-Triples notation and also a subset of Notation 3. The difference between these serialisations are the syntaxes, which can be included or not. Below is an example of Turtle about the "Tuntematon Sotilas" novel, which is written by Väinö Linna. The illustration of the RDF graph that was created from this is in Figure 2.5. [31]

```

@prefix : <http://esimerkki.fi/> .
@prefix dc: <http://purl.org/dc/terms/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

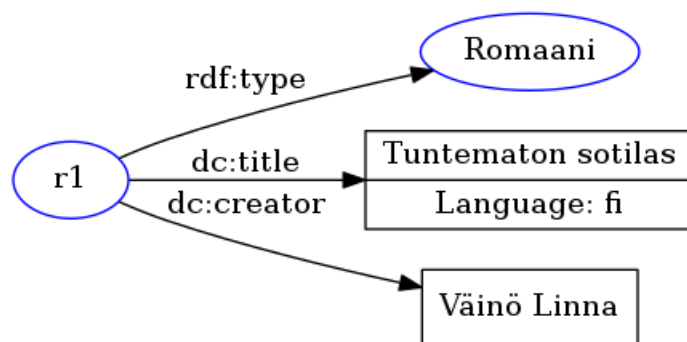
:r1 rdf:type :Romaani ;
     dc:title "Tuntematon sotilas"@fi ;
     dc:creator "Väinö Linna" .
  
```

The considered syntaxes that could be used for this thesis are XML/RDF, JSON-LD, and Turtle. XML/RDF is a syntax of RDF data model and it is more focused to be machine-understandable. The example of XML/RDF syntax is displayed below. JSON-LD uses JSON text format to allow people to read and write documents easily [28]. Turtle syntax expresses data in RDF data model similarly to XML/RDF and is the newest syntax of these three.

```

<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc="http://purl.org/dc/terms/">

  <rdf:Description rdf:about="http://esimerkki.fi/r1">
    <rdf:type rdf:resource="http://esimerkki.fi/Romaani"/>
  
```



Namespaces:  
 http://esimerkki.fi/  
 dc: http://purl.org/dc/terms/  
 rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#

Figure 2.5: An example of RDF Graph that was made from Turtle syntax

```

<dc:title xml:lang="fi">Tuntematon sotilas</dc:title>
<dc:creator>Väinö Linna</dc:creator>
</rdf:Description>

</rdf:RDF>

```

The main reason for choosing Turtle syntax for this thesis work over XML/RDF and JSON-LD is because it is easy to understand for humans. The people who are going to use SemanticNow are most likely not familiar with any of these syntaxes and Turtle syntax is the easiest to understand from the three syntaxes. The second reason is the popularity of Turtle. Most of the people who know the semantic web will very likely know how to read and write Turtle syntax. The other two syntaxes XML/RDF and JSON-LD are still possible syntaxes to implement in this work, but first I want to focus on Turtle. Other syntaxes could be part of future work.

## 2.4 Resource Description Framework Schema (RDFS)

Resource Description Framework Schema (RDFS) is a basic tool for users to define vocabularies, structures and constraints for expressing metadata about Web resources. The terms that are used in RDF statements are defined, and

specific meanings are assigned to them. RDFS can be thought of as a vocabulary library that has defined sets of property types for certain communities. The definitions are different depending on which domain has made it. The idea is not to reinvent the wheel but to fill in the missing parts or define things differently in its way. The schema is meant to be reusable after creating it. The XML namespace mechanism defines the identification for RDF Schemas. [6]

URI of the schema can be used to access both human and machine-processable description of RDF Schema. If the schema is machine-processable, many things can be learned from it. For example, the semantics of the property-types names can be learned and then the understanding of the schema is improved. RDF Schemas structure is based on the RDF data model. Even if an application does not understand a particular schema, it will still be able to fetch the description and separate resources, property types and corresponding values. [20]

## 2.5 OWL Web Ontology Language

OWL Web Ontology Language is created for applications that need to process information inside documents. The information is usually just presented to humans, and the information does not need any processing. OWL is one of the high-level languages that have better machine interpretability of Web content because it is supported by XML, RDF, and RDF Schema. Also, new vocabulary with a formal semantics is used in the process, and it improves understanding of the content. OWL has three very expressive sublanguages: OWL Lite, OWL DL, and OWL Full. [18]

Another feature of OWL is that it can represent the meaning of the terms in vocabularies and their relationships with each other. This representation is called ontology. In this thesis, the test subject is usually an ontology. Terms of ontology are reviewed, and relationships are checked. OWL is compelling language because it has facilities for expressing meaning and semantics much better than XML, RDF, and RDF Schema. [18]

## 2.6 SPARQL

SPARQL protocol and RDF query language (SPARQL) is a query language for RDF, and it is highly recommended by W3C [23]. SPARQL is also used for data management. SPARQL is based on the Turtle notation's triple patterns, and the patterns are matched against the RDF data graph. The

## CHAPTER 2. STANDARDS ENVIRONMENT FOR SEMANTIC WEB14

example of the query is shown below:

```
PREFIX ns: <http://www.domain.com/namespace/>
SELECT ?person ?name
WHERE {
    ?person ns:typens:person.
    ?person ns:name?name
}
```

SPARQL has various query forms, and the most used ones are SELECT, FROM and WHERE. SELECT returns a list of wanted variables. FROM adjusts how the query is restricted, and it is optional. WHERE is a condition expression query form and the most restricted condition will come first. [25]

## Chapter 3

# ServiceNow

ServiceNow was founded at the beginning of 2003 by Fred Luddy, and the popularity rose very quickly [17]. ServiceNow was the first system that offered Anything-as-a-Service (XaaS) type of cloud service. The cloud service provides an easy way to manage the data in the cloud, the scalability to extend storage and backup all the data. The service offered IT Service Management (ITSM) in a web browser, which means that everyone anywhere could use ServiceNow in any web browser [14]. Organisations use ServiceNow for creating systems that suite best for enterprise information technology, improving their work efficiency and reducing operating costs. The ServiceNow company offers an all-in-one cloud platform for service management, operations management and business management [21]. ServiceNow architecture is illustrated in Figure 3.2.

The main reason for the popularity was that ServiceNow worked straight from the browser. ServiceNow is a fast-growing company and to be able to compete with its competitors, the company releases a new version update of the software at least once a year. The versions are in alphabetic order and named after famous cities or mountains. Each release contains new features and fixes to known errors. ServiceNow version Madrid was released at the beginning of April in 2019. ServiceNow also has a online store where developers can upload their applications. The applications can be downloaded or bought from the ServiceNow store as a plugin. The possibility to get SemanticNow application to the ServiceNow Store is illustrated in Figure 3.1.

As mentioned above, ServiceNow provides XaaS services, which contains all other existing cloud service models [13], like Software-as-a-Service (SaaS) and Platform-as-a-Service (PaaS). The customer of ServiceNow receives a service platform, where they can manage their IT services of their company. Additionally, the customer can develop their implementations with

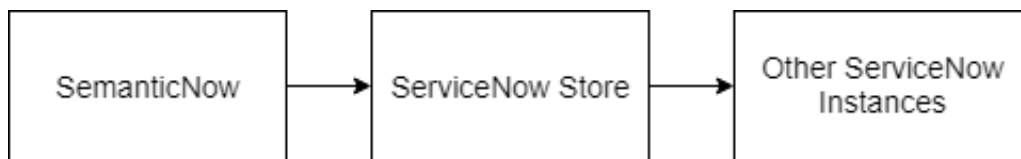


Figure 3.1: The scope of SemanticNow in ServiceNow

the ServiceNow platform, which is the most valuable and favourable feature in ServiceNow. For example, SemanticNow is created with a PaaS feature.

This thesis work can be completed with React, which is a JavaScript framework used to solve the challenges of complex user interfaces with changing datasets. React is widely used in web development and uses generally accepted workflows [10]. I chose ServiceNow over React for the following reasons. ServiceNow provides a user interface for forms and navigation. ServiceNow also provides a server to store the data and the querying in ServiceNow is simple. The ServiceNow development environment is free to use but the environment goes inactive if it is not used for 12 hours. I can focus on the grading system and the functionalities of the work when I chose ServiceNow as a tool.

This chapter is focused on the ServiceNow system and its various functionalities. The familiarisation with the tools is necessary to understand the implementation of SemanticNow. The first section reviews the ServiceNow system and its features. The second section describes the Integrated Development Environment (IDE) of the ServiceNow that functions on a web browser. After the second section, typical functions that are used in the implementation of SemanticNow and security are reviewed.

### 3.1 The ServiceNow System

The fact that the ServiceNow system works in browsers makes it very flexible. Additionally, ServiceNow can update the server of the client and add new features through the Internet on a request of the client. If the client wants to update the ServiceNow system to newer version, there is no need to switch to a new software. Therefore, the data of the client is kept in the database, and the ServiceNow version is just updated. The update usually brings new features and is more optimised to be more efficient than previous versions. ServiceNow offers both software support and hardware support for its customers, which means that the service is extensive and there is a support group for each service category. The client can make some custom changes

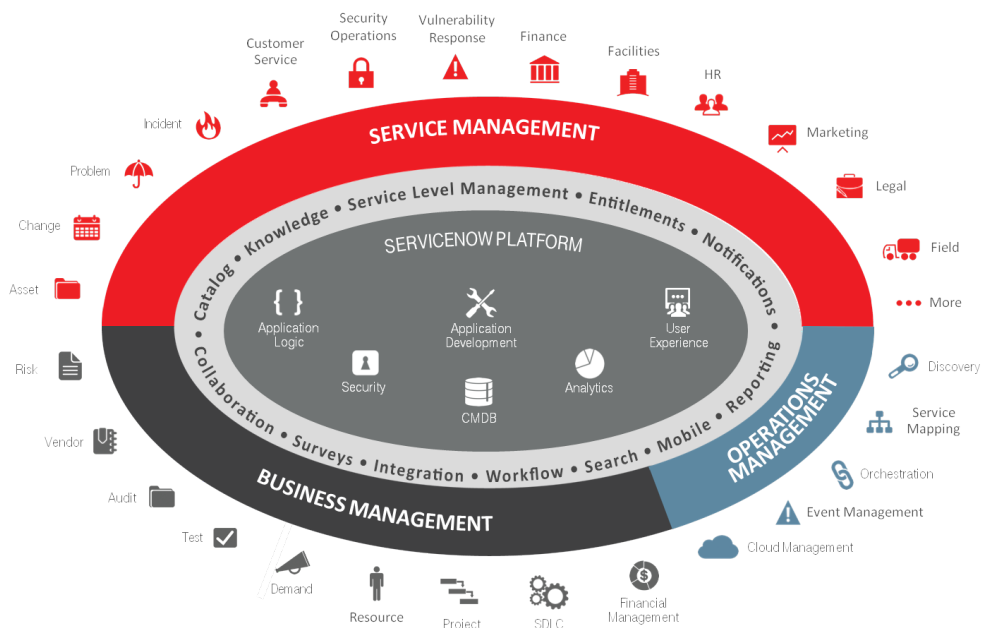


Figure 3.2: ServiceNow architecture of different services [27]

in their instances like changing the user interface or create rules of how a new incident ticket is created. The client has to contact ServiceNow service desk for more significant changes like upgrading ServiceNow Express version to ServiceNow Enterprise version. The express version is cheaper but it does not have features like scripting, plugins and user interface customisation. [14]

The ServiceNow system has various processes such as error, problem, request fulfilment and implementation management process. The system aims to automate these business processes to make the client's work more systematic, efficient and accessible for employees. For example, a client can create an incident ticket and depending on the details, the incident ticket is assigned to specific support group. Tickets can have priority level, urgency level and assigned to the right person or support team. There are three ways to book tickets:

1. A self-service portal, where the user creates an error or event directly into the system by filling in the portal form.
2. Email: ServiceNow's customer sends an email to the system's address, which automatically creates a ticket. The ticket will be directed to the right support group for resolution after this.

3. Created by ServiceNow first level customer support: The user talks with the support person on the phone or by email and the support personnel creates a ticket because the problem was not solved during the discussion.

The creation of a request ticket or incident ticket happens with a form that has pre-defined questions and fields to which the user has to answer. The first view of incident ticket creation is shown in 3.3.

The screenshot shows the ServiceNow incident creation form for incident INC0009001. The form is organized into two columns of fields. The left column includes: Number (INC0009001), Caller (David Miller), Category (Inquiry / Help), Subcategory (-- None --), Business service, Configuration item, Short description (Unable to post content on a Wiki page), and Description (I am not able to edit a wiki page). The right column includes: Contact type (-- None --), State (New), Impact (2 - Medium), Urgency (2 - Medium), Priority (3 - Moderate), Assignment group, and Assigned to. The form also features a top navigation bar with buttons for Follow, Update, Resolve, and Delete, and a bottom section for Related Search Results.

Figure 3.3: Basic ServiceNow incident creation form

## 3.2 ServiceNow Studio

ServiceNow Studio is a great tool that provides an IDE-like interface for application development. Usually, developers must download some IDE interface to begin coding, but ServiceNow has this already in the cloud platform, which does not require any downloads. Developers can have custom application centralised in one place. The interaction with the programming files while developing happens easily. On the left side, the files are separated into their categories. As it is shown in Figure 3.4, for example "tables", "business rules", "client scripts" and "script includes" are in their own layers. These ServiceNow functionalities are explained in the next sections. When things are organised in their categories, it is easier to search the programming file

even if the developer does not remember the name of the file. The developer can jump between the previously opened files and quickly edit them because the previously opened files are tabbed in the top menu of the ServiceNow Studio. [33]

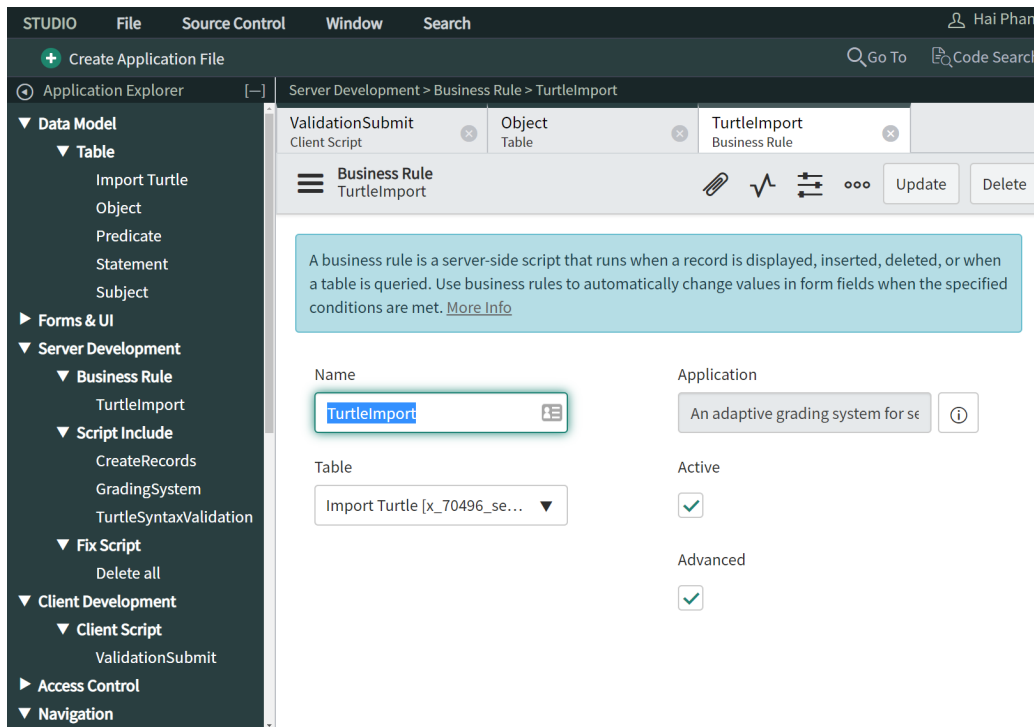


Figure 3.4: ServiceNow Studio IDE-like interface

### 3.3 Client Script

There are various possibilities that the developer can do with client scripts. Client scripts are run on the client with JavaScript programming language when client-based events are triggered, examples for when a form loads, after form submits, or when any field value is changed.

The possible actions with client scripts are configure forms, fields on the forms, and field values at the same time when a user is using the form. Client scripts can do things like making fields hidden or visible, make fields read-only or writable, set the value in one field based on the value in other fields, and display messages based on a value in a field. [34]

Client scripts run on many occasion, which are the followings: on load, on change, on cell edit and on submit. The type of client script can be selected as the type on load that means the system renders the form before the user can enter any data. *On load* type is used when the developer wants to hide some fields or manipulate some field values. *On change* type occurs when the user changes some field values, and when the change occurs, the developer decides what will happen next. The type *on cell edit* works similarly to the on change type, but it runs only when the list editor changes a cell value. The last type is *on submit* that runs when the user has pressed the submit button and tries to submit the form. On submit usually validates the fields on the form and cancel the form submission if there are some errors. [15]

### 3.4 Business Rule

A business rule is a script that runs in server-side when a record is displayed, inserted, updated or deleted. The primary usage of a business rule is to automatically trigger some scripts or change values in form depending on the conditions. A business rule can create events for email notifications and script actions.

The business rules have four options on when it runs in the form. The first one is called *before*, and it is triggered after form submission but before anything is done to the record in the database. The second one is called *after*, which occurs after submission, and all the actions are done to the record in the database. The next one is more complicated, and it is called *async*. It occurs when a scheduler runs the scheduled job that is created from the business rule. The scheduled job is created from business rule after the submission and after any action on the records in the database. The last option is called *display*, which runs before the form is shown to the user and just after the data is read from the database. [34]

This thesis will use a business rule with after option, which means that the user has submitted the form, and all the actions are taken on the record in the database. The purpose of this business rule is to trigger a script include, which function will be described better in the next section.

### 3.5 Script Include

The script includes a server-side script that defines a function or class. Script includes are executed only when they are called explicitly by other scripts. In this research, script include is called from the business rule. Script includes

use JavaScript programming language, and server scripts use functions and classes of the script include. Each script include defines either a function or an object class, and there can be multiple functions in one script include. Securing the code is the reason for creating client scripts. Only the users with developer or admin roles can view and edit the script includes, but many other roles like "basic user" can view how the business rules functions in ServiceNow. [34]

# Chapter 4

## Methods

### 4.1 Process

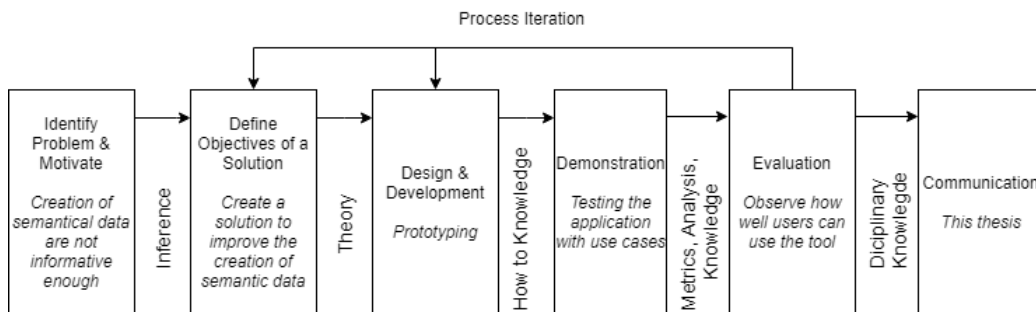


Figure 4.1: The process of the thesis work

The research work follows Peffers & al methodology, which is called design science research methodology. The methodology is divided into six activities [22]. The six activities that are used in methodology are problem identification and motivation, defining the objectives for a solution, design and development, demonstration, evaluation and communication. This thesis work will mainly focus on the design and development part because it will probably require reworks and iterations. The process is illustrated in Figure 4.1.

The first activity of the methodology is to identify the problem, which is the creation of semantical data that are not informative enough. Another problem is to get the user to improve the created ontology inside ServiceNow cloud platform. On top of these two, the creation of automated tests that will check the Turtle syntax before inserting it into the ServiceNow system

must also be done.

The second activity is to define the objectives of the solution. The objective of this thesis is to create a solution to improve the creation of semantic data. The aim of this work is to show the mistakes to the users that are creating new semantic data if there are any mistakes in creation.

The third activity of the process is design and development, which includes thinking of the whole process of how the Turtle syntax comes into the system and turn into ontology. The design will be discussed more precise in Chapter 6. The development of the prototype starts when the picture of the design process is clear, and the development of the prototype will be improved whenever the process is iterated.

The fourth activity is about the demonstration, which includes defining the use cases and testing the use cases. The use cases are described in Section 4.2.

The fifth activity is the evaluation of the work. In evaluation, the aim is to ask people to test the application and ask their opinions on its usability. For example, I would take notes while the user is testing and ask for feedback. After several tests have been gone through. The design and development can be improved. The iteration will move the process back to defining objectives of a solution and design and development. The process iteration can be done as many times as needed. When the iteration is completed as many times and there are no fixes needed then the work can move onto communication.

The sixth activity is communication, which is the documentation of this thesis. All the reasoning and decisions are documented in this work.

## 4.2 Use Cases

Defining use cases help to understand the core objectives of the work better. The process of each step becomes more transparent than the previous one, and some of the unnoticed things can be discovered. The whole picture of the work would be split into smaller parts and observed more closely. The use cases help when the design and development work is in progress. It visualises the task better for the developer, and it is sometimes easier to understand from use cases than from the definition of the task.

In the use cases, I have assumed that the person is a user of ServiceNow application, and he or she has the rights to access the application SemanticNow. The steps from opening a web browser to getting access to SemanticNow application is illustrated in Figure 4.2. There are usually many use cases, but I will define the two most important ones that are related to automated tests and their feedback. These cases are described next.

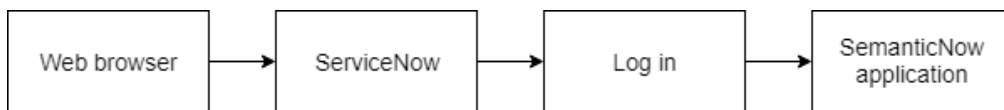


Figure 4.2: The process from web browser to SemanticNow application

### 4.2.1 Inserting a Turtle Snippet to SemanticNow

The main point of the first use case is that the user wants to create new ontology in SemanticNow and the user has a Turtle syntax code snippet. The first step that the user has to do is to go to the correct ServiceNow instance site where the SemanticNow application is located. Then, the user logs in and finds the module on the left side of the screen. The user chooses Import Turtle, which brings the user an empty field for the Turtle syntax code snippet. The user inserts his or her snippet and clicks on submit, and the code will be validated, and the ontology is created. The user can see the visualisation of his or her ontology and table of statements with their attributes: Statement, Subject, Predicate and Object. This use case is illustrated in Figure 4.3.

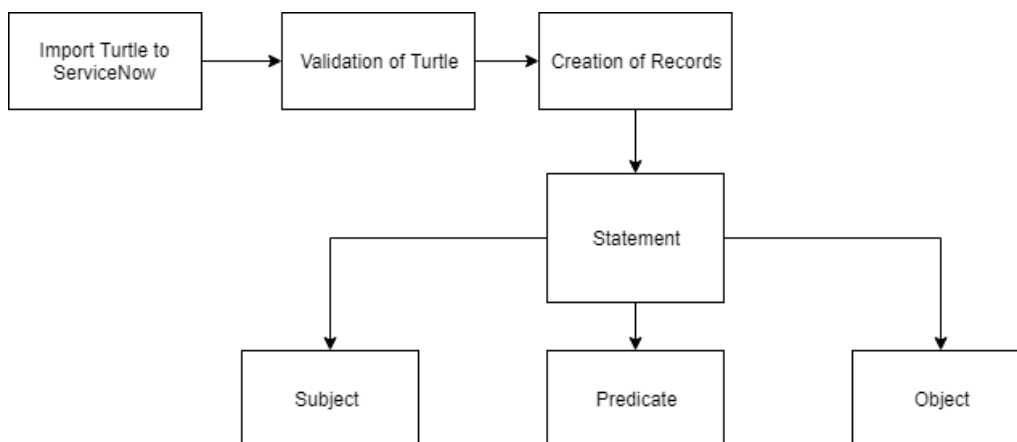


Figure 4.3: Creation of ontology in SemanticNow application

### 4.2.2 Editing SemanticNow Ontology

The desired feature of the second use case is the editing of the created ontology. After inserting the ontology into the system, the user wants to improve the ontology by editing it. I assume that the user has logged into the system

and is ready for editing the creation. The user can navigate to the Statement table where the ontology is located and select it. Then, the user can choose a subject, predicate or object from the selected statement. These tables have fields which the user can edit and save.

### 4.3 User Testing

The user tests were performed on ten people, five of whom were more experienced users and the rest was less experienced users. User types are defined in the next section. The users get a list of tasks that they have to complete. I have setup the users and demonstrated the navigation before they start completing the tasks. The list of tasks and feedback are in Appendix A. The tests were performed with the users' laptops. The users were instructed to go to SemanticNow website and log in with the provided log in information. The users began to complete the tasks and I was collecting notes and helping them.

The tests included navigation and semantic ontology creation in SemanticNow. I also asked the users if they understood the meaning of grading system and how was the usability of SemanticNow.

I had two prototypes of SemanticNow and the users performed the tasks in both of them. The first version had a grading scale from A to F and displayed grade information with JavaScript alert function. The second version had a grading scale from 1 to 10 and the "addInfoMessage" ServiceNow feature to display grade information.

### 4.4 User Types

It is good to define what kinds of users the system has because some users can be more technical, and some with less technical experiences. The users with technical background will more likely know how to use ServiceNow platform than the users without any. The reason is very likely to be the intuition to use the system. There are various technical users, but in this case, I will use the more experienced and less experienced ones. The definition of more experienced technical users is that they do programming in their work. The less experienced ones are the ones who do business in IT industry but does not have any programming skills.

Each of the user groups defined has to understand the semantic web and ontologies at some level. The experienced technical users know on the programming level and understand the Turtle syntax also. The less experienced

technical users have less knowledge about Turtle syntax, but they know the essential subject, predicate and object structure. They can write the basic and straightforward Turtle syntax.

## Chapter 5

# Grading System

The previous chapter has gone through the use cases of how the user creates and edits an ontology. This chapter will go through how the grading system works. There exist many different grading scale that could be used in SemanticNow. For example, academic grading in the United States commonly takes from five to seven letter grades. The grades traditionally are from A to F. A being the best and F being the worst one [26]. The first section will go through the definition and functionality of the grading system. In the second section, I am going through all the grading criteria and the point system of the work. In the third section, I am going to discuss related works that have a grading system with their semantic data. Some ideas from the other works can be used in the implementation of this thesis work.

### 5.1 Functionality

The functionality of the grading system is pretty straight forward. The grading system, like every other, reviews something and gives a grade of the review. In this thesis, the primary purpose of the grading system is to examine the imported Turtle syntax. The grading happens right after the Turtle syntax has passed the validation tests. The grading system goes through all the data and tries to find missing or incorrect data. The errors will be counted, and the final grade is calculated and returned to the user. The functionality of the grading system of the SemanticNow is illustrated in Figure 5.1.

The purpose of returning a grade is to show the user if there is room for improvements. The grading shows the user the fields that are empty or wrong and encourage the user to fill the field with the correct data. The user can reach the highest grade by populating all the empty fields and fixing

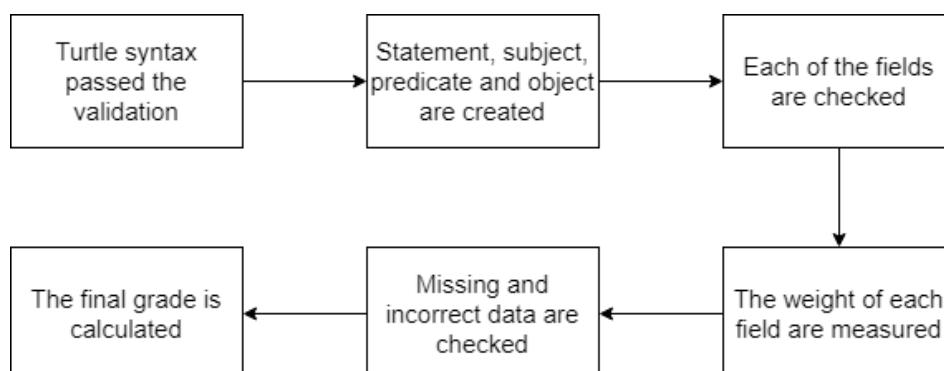


Figure 5.1: The basic functionality of grading in SemanticNow

the incorrect data. The user is making the ontology richer that way. If the user creates a very precise ontology, it is easier for machines to interpret and understand the data.

## 5.2 Related Works

The related works shows variation in grading and ideas that could be used in this thesis work. Also, the benefits of the related works are discussed. The first related work is grading in breast cancer histopathology images that measures the breast cancer possibility from histopathology images. The grading system combines the scores of three different measures and returns the final grade. The second related work is about course management grading system. The factors that affect the final grade are assignments, projects and exams. The benefit of the work is the scaling between them, which happens automatically. The meaning of automatic scale is, for example, the professor have decided that assignments gives 40% of the final grade, the system automatically scales each of the assignments to have the same score weight. The third related work is scaling of technical lignin grades, which grades the suitability of lignins to potential applications. The lignins are measured in three different categories, which are combined and give a final score.

### 5.2.1 Grading in Breast Cancer Histopathology Images

Tutac et al. [30] investigate the guided paradigm for semantic indexing of histopathology images and apply it to Breast Cancer Grading (BCG). The

most used grading system in breast cancers is BCG. The criteria for giving the final grade are to combine nuclear pleomorphism, tubule formation and mitotic counts, the scores are given by each of them individually. The lowest score being 1 and the highest score being 3. Then, they are added together to give the final grade. [30]

The importance of the breast cancer grading system is the automation and saving time in manual work. The automatic grading system semantically indexes the images line. The system uses medical domain knowledge.

The idea of dividing different things during the grading could be used in SemanticNow. The fields that are related to particular prefixes like RDF and RDFS could be grouped. The point is to group prefixes that serve the same or almost the same things and give a separate grading. The user can then notice that something might be wrong with some prefix groups.

### 5.2.2 Course Management System Using the Semantic Web

Course management systems (CMS) are increasing in popularity, and the well-known CMSs are Blackboard.com and WebCT.com. Both of the sites focus on distance education opportunities [19]. The common tools that are included in CMS are a variety of functionalities, such as registration tools for students, course enrolment management, class project management, test administration, assessment tools, and online discussion boards [8].

Jorge Cardoso [8] developed an innovative Semantic Course Management System (S-CMS), which is a part of Strawberry project1. The project1 explores the use of semantic web technologies and tries to use the benefits of the semantic Web in S-CMS. The work's main purposes are to make the CMS more efficient and usable for students and faculty members. The example case is to automate various procedures that are involved when the students enrol for different class projects. The problem has always been in the comprehensive course management, and the class projects are complex and time-consuming. Many factors affect this complexity. Such factors are a large number of students, various rights for students that allow registering for particular projects, the background of the students and grades of the students. [8]

The data in the system is linked semantically, and it is on seven different layers. The lower layers have information like which faculty members teach which courses, student course enrollments, student degree enrollment, and personal information about students and teachers. Also, lower layers are responsible for connecting the data sources with a variety of protocols. The

higher layers will handle the integration and combination of data that means the data from other integrated systems can be used and queried. The other systems are the ones from other universities or schools, and the format of the data can be very different. [8]

The Strawberry project has a plug-in that has a grading feature. The feature enables the use of grading ontology that allows the grading of the students that have enrolled in a course. The teacher can define the his or her own grading policy using the plug-in for calculate the final grade. Teachers can create new evaluation items such as exams, exercises and attendance that affect the final grade. The weight of evaluation items can be different. For example, the final grade is given from the exam and the assignments. The weight of the exam can be 60%, and the assignments can have the weight of 40%. The teacher can add more assignments, and the plug-in would automatically adjust itself so that each assignments weight the same and all assignments are still affecting 40% of the final grade. [8]

The benefit of the ontologies is shown in the flexibility and reuse when modelling a grading domain. The ontologies of different schools and universities can be migrated with each other. Also, the new ontologies can be put on top of the existing ones, and the ontology would grow bigger. Some teachers have their way to give the final grade, which is why the automatic tool that can calculate the grade is essential. After the creation of the own grading model, it can be reused in other courses or even other teachers. The grading ontology structure of the CMS work is illustrated in Figure 5.2. [8]

Similarly than in the grading, weighting the evaluation items is an excellent idea to try in this thesis. The problem would be the measurement of each field because it is not that clear, which fields are more important than others. If there are many fields, then each of them must be weighted. Another option is to put a base weight on each field and then adjust them afterwards. All the weight of the fields would be counted together and then divided by the number of the fields.

### 5.2.3 Scaling of Technical Lignin Grades

The main objective of Nazu Akture's thesis [1] was to create a unified definition for a grading system that will work as a tool for finding suitable potential applications [1]. The applications are used for industrial-scale production. The other objectives that must be investigated before the grading system of the work was:

1. Analyse and study the differences between technical lignins.

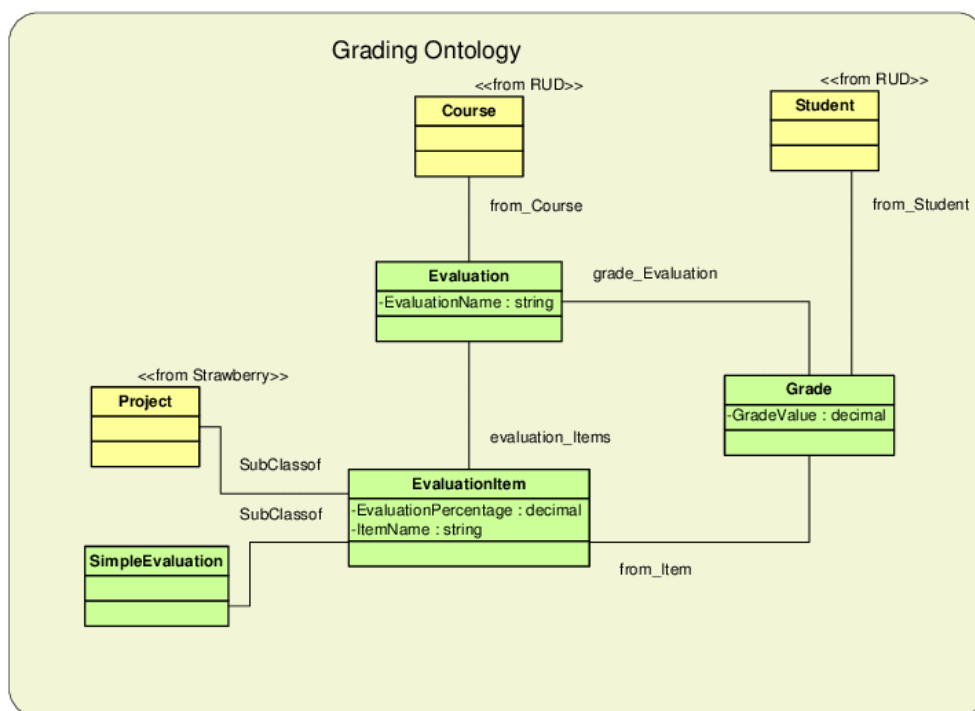


Figure 5.2: The structure of the grading ontology [8]

2. The definition of the desired properties for technical lignins by taking account of different end-use applications.
3. The creation of lignin matrix for a product catalogue for technical lignins.

The work consists of characterisation of three technical lignins and five samples. The characterisation of the samples was done in the laboratory by strictly following the methodologies of available standards and protocols. Each of the samples has properties that were elemental composition, molecular weight distribution, purity, reactivity, and structural characteristics. The ranking between lignins was performed. They were ranked regarding their suitability for eight potential applications, which were in three categories. The categories were aromatic chemicals, composites and energy pathway. [1]

The grading of this work were from A to D, and A is the best and D is the worst. The ranking is calculated by the suitability of lignins toward different applications. The grade A and B for lignins mean that they are highly suitable the selected applications. The grade C lignins need to be

improved, and grade D lignins are not suitable for the selected application. [1]

The grading scale of Nazli Akture's work is small, but for his research, it was precise enough. The point is not to have significant scaling in scores and keep it clear if the suitability is enough or not. The scaling of Nazli Akture's thesis work could work with few grades declaring if it is a pass or fails. It would not be very precise, but at least it would be transparent for the user.

### 5.3 Design for Grading System

The grading criteria is often decided before any tests. The grading criteria affects the final score because each violated criteria decreases the final score. The criteria that I am using in this work are the following. The weight of fields, missing field value, missing prefix and correct type criteria.

The weighting criteria of the ontology is the following. The essential and important fields are type, label, domain and range. Each of these important fields must have prefix to be able to pass the tests and successfully be imported to SemanticNow. These important fields and prefixes weight 60% of the final grade. The rest 40% of the final grade are given from the other fields equally. The weighting of the final grade is illustrated in Figure 5.3.

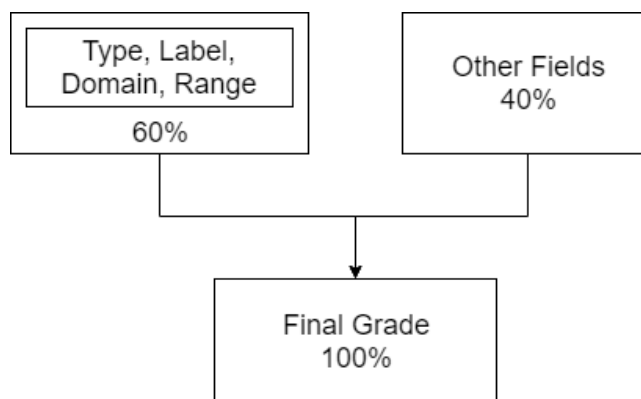


Figure 5.3: The weighting of the final grade in SemanticNow

The missing field value criteria and missing prefix criteria is considered important, especially when the missing field belongs to one of the important fields. The value is more important than prefix, which is why missing value would decrease the final score more than missing prefix. The weight of value is 60% and the weight of prefix is 40% of the field.

Specific fields require a correct type for the value. The correct type criteria penalises for using wrong value type. For example, the user cannot put a string value to the field that requires integer value. When this is graded, then the user is noted for having an incorrect value type and encouraged to change the value to the correct one. The incorrect value type decreases 20% of the field score. The correct type criteria is not considered important, which is why it does not weight much in the field score.

There are many scoring scales available. This work will try various grading scales and pick the best one for use. The scoring scale can be from A grade to F grade, 0 to 5, 0 to 10 or even more precise from 0 to 100. I used two different grading scales in my prototypes. The grading scale from A to F is used in the first prototype and grading scale from 0 to 10 is used in the second prototype.

## Chapter 6

# SemanticNow Design

The design of SemanticNow is simple and straight-forward. The main point of SemanticNow is to be able to bring users' ontologies into the ServiceNow platform that runs on the website. The SemanticNow should be usable for both more experienced users and less experienced users. The user interface is simple and does not have unnecessary buttons or actions.

This chapter will walk-through each process step in SemanticNow in all the details that are focused on the design. The first section goes through how a person can get access to the ServiceNow platform and have rights to see SemanticNow application. The second section is an explanation of the navigation in SemanticNow and explains the SemanticNow application menu and the modules inside of it. The third section, the Turtle validation, will review the steps that occur when the user inputs a correct and incorrect Turtle syntax. The fourth section will go through the import of the ontology, which occurs right after passing the Turtle validation. The fifth section is about the design of two different grading systems that I have implemented. I will review the user feedback about the two different versions and select the one that is better based on user feedback. The fifth section will also include the two version how the grading and errors will show up for the users.

### 6.1 User Access to SemanticNow

The very first step to being able to use SemanticNow is the access to the ServiceNow platform. The person has to demand a user to SemanticNow to have an access to it. The user access creation is illustrated in Figure 6.1. The steps to get user access to SemanticNow are the following:

1. Admin creates a SemanticNow user for the person that demanded it.

2. Admin gives the user rights to the user by either giving "semantic\_now\_role" or adding the user to SemanticNow group, which has the "semantic\_now\_role".
3. Admin gives the credentials to the person.
4. The person logs in to SemanticNow from SemanticNow website.
5. The user can create and edit own ontologies in SemanticNow.

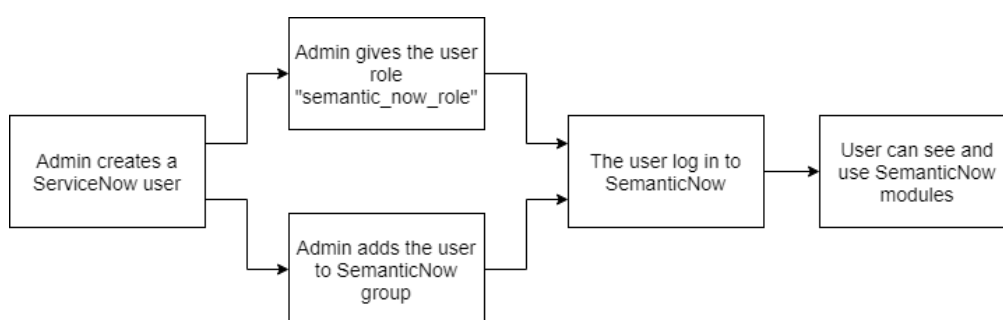


Figure 6.1: The user creation and giving access rights to SemanticNow application

The user credentials are linked to the personal email and the credentials are sent to the email. The timeout of SemanticNow is 12 hours that means the user is automatically logged out after inactivity of 12 hours. The reason for timeouts is to follow the activity of online users and for security reasons.

## 6.2 Application Menu

The SemanticNow application menu is on the left side of the browser, and it is illustrated with the modules in Figure 6.2. The SemanticNow application menu contains two different types of modules. The "Import Turtle" module directs the user straight to the import Turtle form. The other four modules "Statements", "Subjects", "Predicates" and "Objects", direct user to a selected table of the module that contains records of the chosen module. This design has all the things that it needs and no other unnecessary modules. The usability of SemanticNow is better when having fewer modules because it is easier for users to navigate.

The user can search for the wanted application menu or module by using the filter navigator on the top left side. The application menus and modules

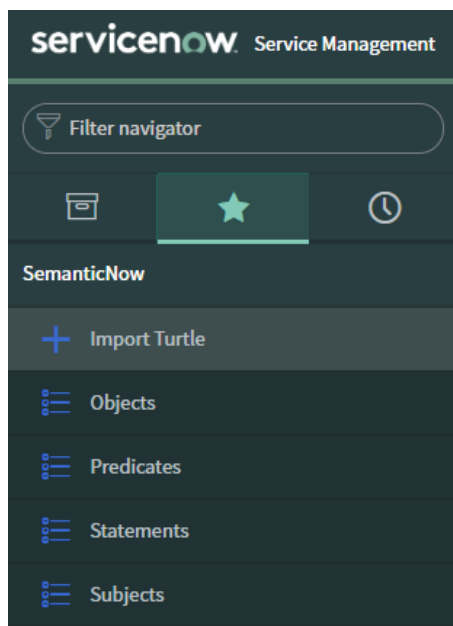


Figure 6.2: SemanticNow application menu

are hidden and restricted to the users that have the rights to see them. The three different categories are displayed below the filter navigator. The leftmost tab is showing all the application menus and modules, the middle tab is showing the favourites, and the rightmost tab is showing the history.

### 6.3 Turtle Validation

The validation of the Turtle occurs when the user has put the Turtle in the field called "Turtle Syntax" in the "Import Turtle" module and have pressed submit button. The view is illustrated in Figure 6.3 to get a better understanding of the form. A client script called "ValidationSubmit" will trigger on the press of submit button, and it will validate the Turtle of the user input. The client script is described in more detail in Section 7.1. The Turtle import will occur if the Turtle of the user passes the validation tests, and then the user is redirected to the created statement record in Statements table. The form prevents the user from proceeding if there are any validation errors. The errors will be shown as an alert, and it is illustrated in Figure 6.4.

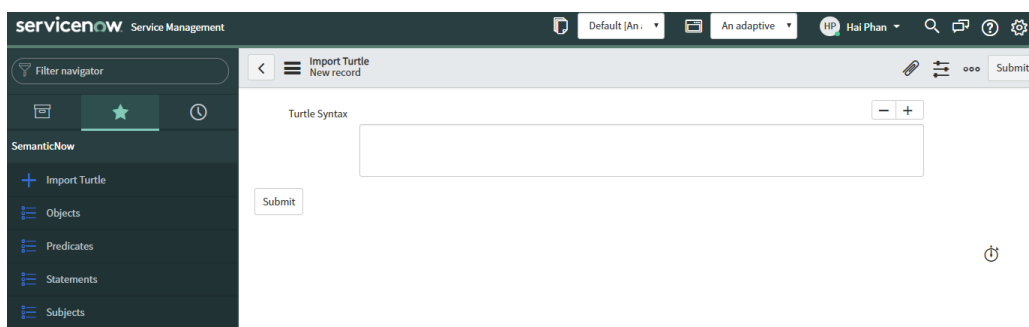


Figure 6.3: Turtle syntax form in SemanticNow application

## 6.4 Turtle Import to ServiceNow

The Turtle import occurs after the input of the user has passed the Turtle validation. A business rule called "TurtleImport" is triggered when the form submission is successful. The business rule will trigger a script include called "CreateRecords", and it will start to parse the Turtle input of the user. The script will create a statement, subject, predicate and object records, which are all created in their tables. These records are linked to each other with their automatically created unique value. The point of them being linked to each other is the accessibility to navigate between records smoothly and to see which records are linked to each other. The parsing will also automatically populate the fields of each record like prefix and namespaces.

The design of each record being in their own tables is for the user to separate statement, subject, predicate and object. For example, the user can see all the predicates that have been created by going to the predicates module from the SemanticNow application menu. This way, the navigation is simple and easy-to-use even for new users. The linking of the statement record is only one way because the statement is an upper layer of the other three tables. The illustration of statement being on the upper layer is illustrated in Figure 6.5.

## 6.5 Grading

The grading system will have two different designs, which will be decided after the first user testings. The first version includes the letters from A to F grades and A grade being the best and F grade being the worst. The second version of the grading system is grading from 0 to 10, which is clear and

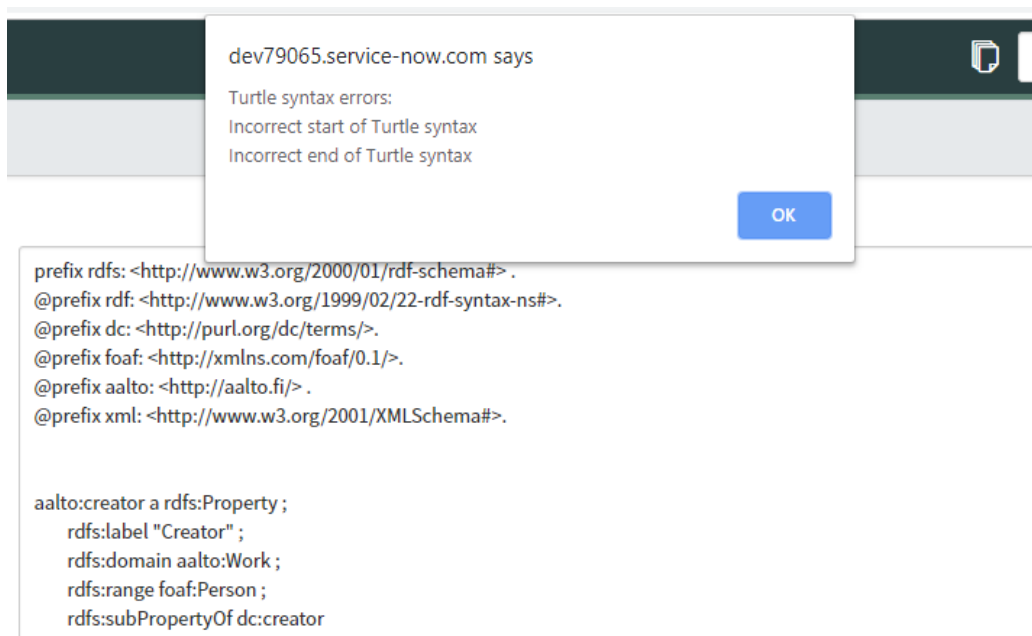


Figure 6.4: An alert of an incorrect Turtle validation

familiar for everyone. The best grade is 10 and worst grade is 0.

There will be two versions of how the grading will show up to the user when he or she has successfully inserted the ontology to SemanticNow. The first version uses alert that will pop up right after the insertion of Turtle syntax and all the grading information will be shown in the alert. To be able to continue using SemanticNow, the user must press the "Ok" button. The second version is the ServiceNow platform's function called "addInfoMessage", which shows the grade information on top of the form. The user does not need to press any buttons to confirm that he or she has acknowledged the message. The message will disappear when the user navigates to other page.

## 6.6 Access Control

ServiceNow has a robust security system that is built into all different levels of the system. This section is going to focus on the securities that are related to this research. Typical Access Control List (ACL) is used in the SemanticNow application. The ACL prevents users without permission to access and to create any records in SemanticNow application. Only users that have admin role or the SemanticNow role can access the application. The users that have

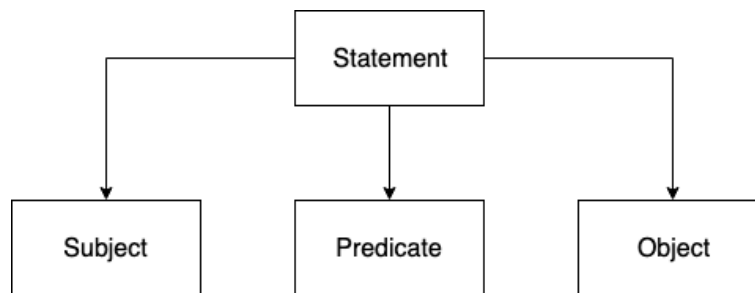


Figure 6.5: Statement relationship to subject, predicate and object

the SemanticNow role can access all the modules, but deeper security layers like script includes or security settings. [24]

## 6.7 Query in SemanticNow

The querying in ServiceNow happens from the point that the user runs the query, then the query is executed in the database, and it returns a table of records for the user. The condition query happens as follows: ServiceNow will show the fields that are in the table, and the user must choose one of the fields. The user must choose the necessary condition after selecting the field, and ServiceNow automatically gives the selectable conditions as a drop-down menu. For example, if the field only has Boolean value, then there are fewer conditions. The user can put the wanted value in the last field and press run to perform the query. The next example is a demonstration of how the query works in ServiceNow incident table, and the query conditions are:

1. The record is in an active state.
2. The category of the record is Software or Hardware.

The example of the query is illustrated in Figure 6.6, and the returned results are shown in Figure 6.7.

The reason for not selecting SPARQL as our query language is because there is no integration between SPARQL and ServiceNow. ServiceNow is providing efficient query, which is why I did not try to integrate SPARQL into ServiceNow.

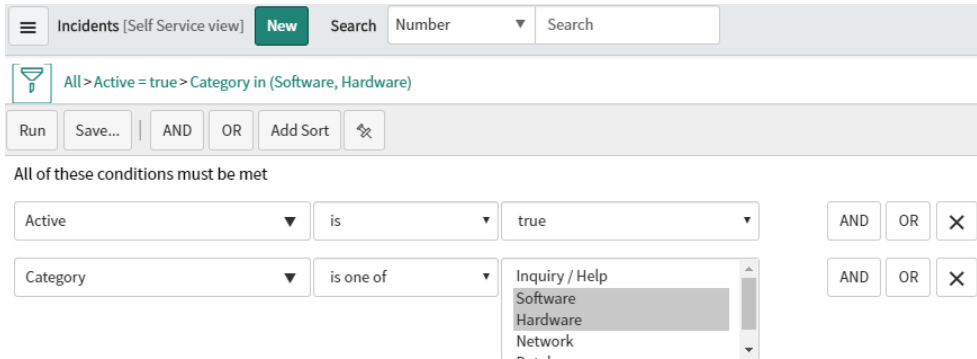


Figure 6.6: An example of ServiceNow query in incident table.

	Number	Active	Category	Opened	Short description
<input type="checkbox"/>	<a href="#">INC0009005</a>	true	Software	2018-08-31 21:35:21	Email server is down.
<input type="checkbox"/>	<a href="#">INC0007001</a>	true	Hardware	2018-10-16 22:47:10	Employee payroll application server is down.
<input type="checkbox"/>	<a href="#">INC0000054</a>	true	Software	2015-11-02 12:49:08	SAP Materials Management is slow or there is an outage
<input type="checkbox"/>	<a href="#">INC0000052</a>	true	Software	2019-03-08 12:48:40	SAP Financial Accounting application appears to be down
<input type="checkbox"/>	<a href="#">INC0000051</a>	true	Software	2019-03-08 12:48:32	Manager can't access SAP Controlling application
<input type="checkbox"/>	<a href="#">INC0000050</a>	true	Hardware	2019-03-08 13:58:24	Can't access Exchange server - is it down?
<input type="checkbox"/>	<a href="#">INC0000046</a>	true	Software	2019-03-08 14:04:15	Can't access SFA software
<input type="checkbox"/>	<a href="#">INC0000041</a>	true	Hardware	2018-12-29 16:44:53	My desk phone does not work
<input type="checkbox"/>	<a href="#">INC0000027</a>	true	Software	2018-12-25 15:55:55	Please remove the latest hotfix from my PC
<input type="checkbox"/>	<a href="#">INC0000025</a>	true	Hardware	2018-12-06 15:53:46	Need to add more memory to laptop
<input type="checkbox"/>	<a href="#">INC0000019</a>	true	Software	2018-12-16 15:44:39	Can't launch 64-bit Windows 7 virtual machine
<input type="checkbox"/>	<a href="#">INC0000016</a>	true	Hardware	2018-12-08 15:40:23	Rain is leaking on main DNS Server
<input type="checkbox"/>	<a href="#">INC0000015</a>	true	Software	2018-12-13 15:38:46	I can't launch my VPN client since the last software update

Figure 6.7: An example of query result in ServiceNow incident table.

## Chapter 7

# Implementation

This chapter will go through the implementation of SemanticNow step-by-step from the Turtle import validation to the creation of statement, subject, predicate and object. The essential functions will be reviewed and explained. The problems during the implementation would also be included in each section if there were any. The last section goes through the implementation of the two different versions of the grading system. The first version is based on the function of JavaScript, and the second version is based on the own feature of the ServiceNow. The feedback of the user testing is taken into account to decide, which version will be chosen.

### 7.1 Turtle Validator

There are few automated Turtle syntax validators in the Web. Either they did not provide any APIs that I could directly link to SemanticNow or they provided an API that would have required setting up a server. I decided to create my own Turtle syntax validator since there was no direct APIs available. I created my own Turtle validator by studying other validation programs by testing them with different Turtle syntax examples.

This work uses client scripts right at the beginning of the process to validate the Turtle syntax. The client script called "ValidationSubmit" is triggered when the user has pasted or written the data into the field of the import Turtle syntax form and pressed submit button. The client script parses the input each row and checks that it follows the standards of the Turtle syntax. For example, the client script checks the prefixes that they start with a "@" sign and the whole Turtle syntax ends with a "." sign. The client script prevents the user from submitting the Turtle syntax if it does not pass the validation. On top of prevention, the client script puts all the

errors in an array and return them to the user as an alert or in additional information box. The submission is accepted only when the major errors are fixed.

## 7.2 Creation of Records

The creation of records will appear in the SemanticNow tables. The records creation of statement, subject, predicate and object occur in the script include "CreateRecords", and the linking between them happens with a reference field. The reference field fetch the unique id of the other record and link them together. In this thesis work, the linking is done both ways for the convenience of moving from subject to predicate, predicate to object and another way around. The user can go to subject, predicate and object from statement record, but not another way around. The linking in subject records are different when compared to predicate and object records because in this work, one subject can have many predicates, and there can be only one link between predicate and object. The subject record uses a related list, which allows one subject record to have many predicate records.

I will go through the creation of the records with the following Turtle syntax:

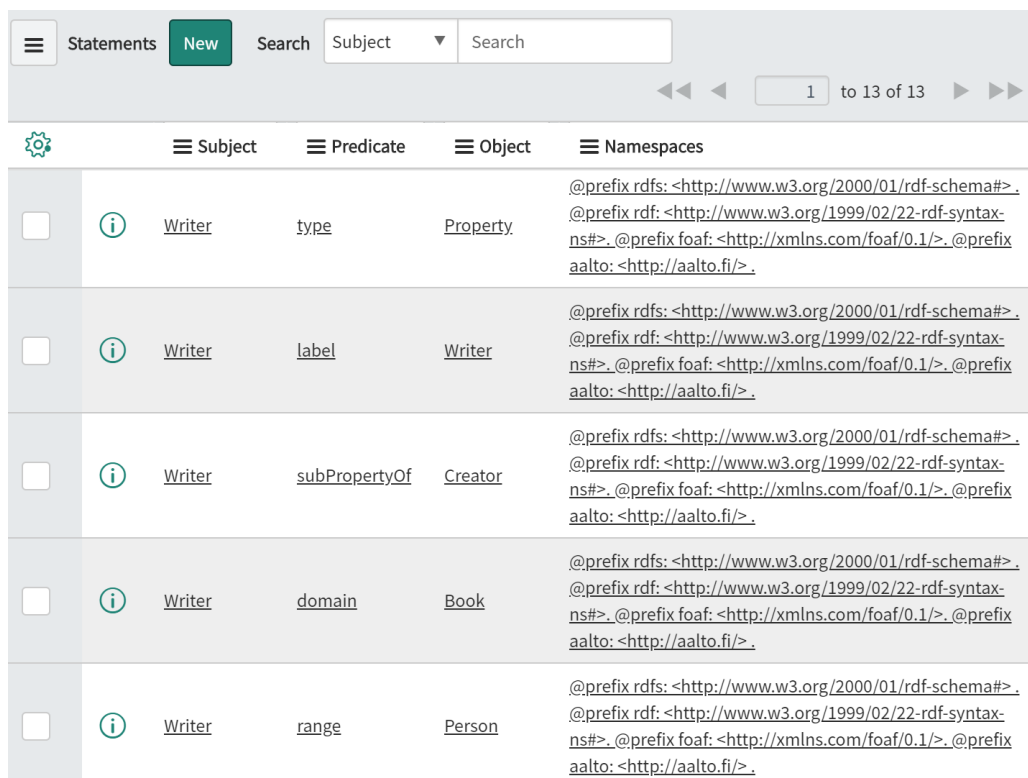
```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix aalto: <http://aalto.fi/> .
```

```
aalto:writer
  rdf:type rdfs:Property ;
  rdfs:label "Writer" ;
  rdfs:subPropertyOf aalto:Creator ;
  rdfs:domain aalto:Book ;
  rdfs:range foaf:Person .
```

The script include called "CreateRecords" parses the Turtle and takes all the necessary information, then populate them into the correct fields in the created records. The parsing happens line by line and I have created a set of rules that automatically populate the fields with correct values. For example, one of the rules checks the lines that starts with "@"-sign in the snippet and populates it in the field "Namespaces". The subject "writer" is

parsed from "aalto:writer" and when viewing the subject record, the prefix field is populated with "aalto" and name field is populated with "writer". The predicate is parsed from the lines that have indentation, for example, "rdfs:label" and "rdfs:subPropertyOf". Similarly, the object is parsed in the same line as predicate after the space between predicate and object. The information about namespaces, subject, predicates and objects are all necessary for building ontology. The creation of statement, subject, predicate and object co-occur, which make the linking in reference field possible.

The creation of the statement records occurs immediately after the submission. The illustration of the statements table is illustrated in Figure 7.1. The user can directly click on any subject, predicate or object to go to the record and edit it.



	Subject	Predicate	Object	Namespaces
<input type="checkbox"/>	<a href="#">Writer</a>	type	<a href="#">Property</a>	@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>. @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>. @prefix foaf: <http://xmlns.com/foaf/0.1/>. @prefix aalto: <http://aalto.fi/>.
<input type="checkbox"/>	<a href="#">Writer</a>	label	<a href="#">Writer</a>	@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>. @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>. @prefix foaf: <http://xmlns.com/foaf/0.1/>. @prefix aalto: <http://aalto.fi/>.
<input type="checkbox"/>	<a href="#">Writer</a>	subPropertyOf	<a href="#">Creator</a>	@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>. @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>. @prefix foaf: <http://xmlns.com/foaf/0.1/>. @prefix aalto: <http://aalto.fi/>.
<input type="checkbox"/>	<a href="#">Writer</a>	domain	<a href="#">Book</a>	@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>. @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>. @prefix foaf: <http://xmlns.com/foaf/0.1/>. @prefix aalto: <http://aalto.fi/>.
<input type="checkbox"/>	<a href="#">Writer</a>	range	<a href="#">Person</a>	@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>. @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>. @prefix foaf: <http://xmlns.com/foaf/0.1/>. @prefix aalto: <http://aalto.fi/>.

Figure 7.1: A newly created statement record in statements table

The statement record has a simple looking form that consists of namespaces, subject, predicate and object. The type of namespaces field is a string and it contains the namespaces that are parsed from the code snippet of the user. The other three fields are reference fields, which means that the user

must navigate to the record to see more information like the value of prefix in subject record. The user can easily change any of the fields by pressing the magnifying glass and choosing one of the options. The user can also go to the specific record by pressing the icon next to the magnifying glass, which is located on the right side of the form. The illustration of the form is in Figure 7.2.

Statement  
Created 2019-07-02 23:24:26

Update Delete

Subject

writer

Predicate

type

Object

Property

\* Namespaces

- +

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .  
 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
 @prefix foaf: <http://xmlns.com/foaf/0.1/> .  
 @prefix aalto: <http://aalto.fi/> .

Update Delete

Figure 7.2: A form of the statement record

The next record that is created is the subject, which contains two fields. Name and prefix field are both string type. The illustration of the view of the subject record is in Figure 7.3. The related lists are in the bottom of the record with the name of "Predicates". The related lists consist of all the predicate records that are linked with the selected subject record. Subject record does not have a direct link to the object record but the linking happens through predicate record.

The created predicate and object records are quite similar to each other. A predicate and object record has fields name and prefix, which type of string. Predicate record has reference fields to subject and object records,

Subject writer

Name  
writer

\* Prefix  
aalto

Update Delete

Predicates New Search Created Search

1 to 5 of 5

Subject = writer

	Created	Name	Prefix
<input type="checkbox"/>	<a href="#">2019-07-02 23:08:44</a>	range	rdfs
<input type="checkbox"/>	<a href="#">2019-07-02 23:08:31</a>	domain	rdfs
<input type="checkbox"/>	<a href="#">2019-07-02 23:08:18</a>	subPropertyOf	rdfs
<input type="checkbox"/>	<a href="#">2019-07-02 23:08:00</a>	label	rdfs
<input type="checkbox"/>	<a href="#">2019-07-02 23:07:40</a>	type	rdf

Figure 7.3: A form of the subject record with related list

but object record only has a reference field to a predicate record. The linking is not available from subject to object directly but through predicate, which is why the object record does not have a field of subject. The form of predicate record is illustrated in Figure 7.4.

After submission the user is redirected to a newly created statement record. The redirect improves the understanding of the creation, and the user can immediately see results of his or her ontology creation.

The image shows a web form titled "Predicate type". At the top, there is a header bar containing a back arrow, a hamburger menu icon, the text "Predicate type", and several utility icons (edit, list, update/delete) along with "Update" and "Delete" buttons. The main form area contains four input fields: "Name" (containing "type"), "Prefix" (containing "rdf"), "Object" (containing "Property"), and "Subject" (containing "writer"). The "Object" and "Subject" fields include search and information icons. At the bottom of the form are "Update" and "Delete" buttons.

Figure 7.4: A form of the predicate record

### 7.3 Grading System and Scale

I have created two versions of how the grading would be shown to the user and used two different scales in grading. I will go through the information display and grading scale of the first version and move to the second version.

The first version used the Javascript function called "alert". The grade and additional information is displayed to the user after successfully submitting the Turtle syntax in SemanticNow. The user must press "Ok" button to continue using SemanticNow that ensures the user has seen the grade and information. The information about the mistakes in ontology is pointed out to the user in the alert and the user is instructed to go to specific record to fix it.

The scaling from A to F was used in the first version that resembles the grading scale of American colleges. The purpose of this kind of grading scale is to test how well do users understand the point of the given grade and are they willing to improve the ontology to raise the grade.

The second version would use the "addInfoMessage" function from ServiceNow. The purpose of this function is to show the user the grade and additional information without the need of confirmation that the user has seen it. The information is displayed on top of the form clearly with a dark blue background. The mistakes are pointed out to the user and a hyperlink to the error is provided in the information box on top of the form.

The second version used scaling from 0 to 10 in which the higher number means the higher grade. In the Finnish education system, the higher number has always meant the higher grade. This makes it clear to Finnish people how the grading scale works.

## Chapter 8

# Evaluation

In this chapter, I will look back to the research questions and evaluate how well I have handled them. I will evaluate the methods that I have used in the design. I will also evaluate the implementation of SemanticNow in the ServiceNow platform and include some problems that I have faced. I will evaluate the usability tests and choose the best suited features for users in SemanticNow.

### 8.1 Evaluating Research Questions and Development Goals

This thesis work had two main research questions of how to define a grading system in ontologies and how to get the user to improve his or her ontology creation. I found various ways to grade the ontologies, which were used in other works. The grading from A to F grade and numbers from 0 to 10 was the options that I gave for the user testing. The majority of the users prefer the number grading, which I assume to be more familiar than the A to F grading.

I did many kinds of research about the grading systems that already exists during the thesis process. I came up with ideas of weighting the different fields because some fields are more important than others. The important fields weighted 60% of the final score and the rest 40% was from other fields. The weighting between important and less important fields was a success, because the test users focused on the important fields and did not left them empty. I also went through several grading systems and decided the best-suited one for the users. The primary purpose was to clearly show the user the grade and get him or her to improve the ontology creation.

There were two ideas of how the user could immediately improve the

ontology creation after submitting it in SemanticNow. The first one was an alert that will pop up after submission of the Turtle. There was a problem of not having the hyperlink in the alert, but instead, there was a help text that instructs the user to go to a specific record. The second one was an info message that showed up on top of the form with a hyperlink that will guide the user to the incorrect record. In the end, I chose the info message since it was more preferred according to the feedback.

The two development goals of this thesis were to implement an automated testing solution for ontologies in the ServiceNow platform and designing and implementing a grading system for ontology creation. The syntax to create ontologies in this thesis was Turtle. There was a problem to find an automated Turtle syntax validation API. There was no direct integration to the ServiceNow platform, which is why I created my own Turtle syntax validator by imitating the online Turtle validation sites.

## 8.2 Evaluation of Process and Design

When looking back at process method that I have used, the iteration of the process has been beneficial and improved the work outcome. The iteration was done only once because of the time limit. It was still enough to see the different between the first and the second version of the SemanticNow usage. The results of the application would not been that different with more iterations. The process consisted of definition of user types, prototyping, user tests, feedback, and checking the requirement list. There was a clear majority of the users that chose a particular type of how the ontology errors and grading would show up. Therefore, there were no difficulties in choosing between different versions of information messages.

The design of the work was little unclear for me until I got into the implementation part because I did not know how to design the semantic web in the ServiceNow platform. After trying out several things and finding out how the records could be linked together, I began to get a clear picture of the design. The design and the process helped me to develop SemanticNow step-by-step without moving in between different codes many times. Overall, the design helped me to implement the SemanticNow efficiently, and the process iteration made the application usable for the users.

### 8.3 Test Cases and User Feedback

I have made the test cases quite early of the work and many questions have come up on the way of the development. The questions of how the user should be able to create ontology in SemanticNow and how the grading should be displayed to the user to be as clear as possible. The use cases and feedback notes are in Appendix A. The test cases could be more accurate and asking even more questions from the test users, but the additional questions did not come in mind before the testing day.

All the users knew how to use the SemanticNow application and gave positive feedback. The navigation in SemanticNow was easy for all users because the navigation was quick and responsive. The usability of SemanticNow was clear and the users knew how to create or edit ontology without any help. I would have wanted to do another user testing with the additional questions but the time was limited, and I thought that the iteration from the first user testings was enough. Users did not encounter any major problems while using the SemanticNow, which is why I did not have another user testing session.

Much feedback was gotten from the users after the user tests and used for improvements. One of the suggestion was to the number of the row that caused an error. I have improved the alert by adding row numbers in the error messages. The row numbers will make it easier for the user to find the mistake and fix it.

The user testing result and the opinion from the users were that the "addInfoMessage" feature was better, because hyperlinks directed the user straight to the mistaken record. Hyperlinks were not supported in alerts. Some of the test users said that the alert would make sure that the user notices the grading and the information. The display of grade and information in second version was clear for the test users, which is why the confirmation button "Ok" was not needed. The chosen version of the grading system would be the second version with the "addInfoMessage" feature. The illustration of alert is in Figure 8.1 and the example of "addInfoMessage" is illustrated in Figure 8.2.

The two versions of the way to show the grade and additional information were shown to the test users, and the more suitable and familiar one is chosen. The first version had letters from A to F, and the second version had numbers from 0 to 10. The majority of the test users chose the numbers because it was more familiar to them. There was a discussion of the grading number range of 0 to 5 and 4 to 10 also, but the numbering range 0 to 10 was still the best.

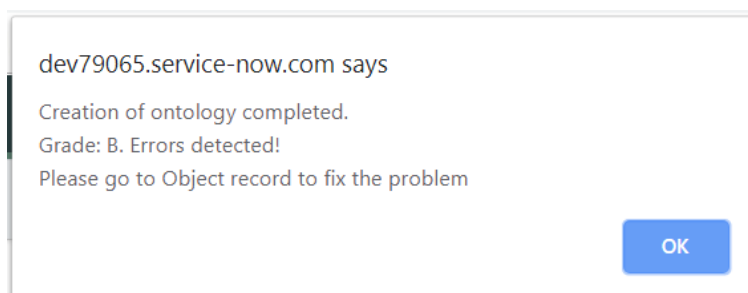


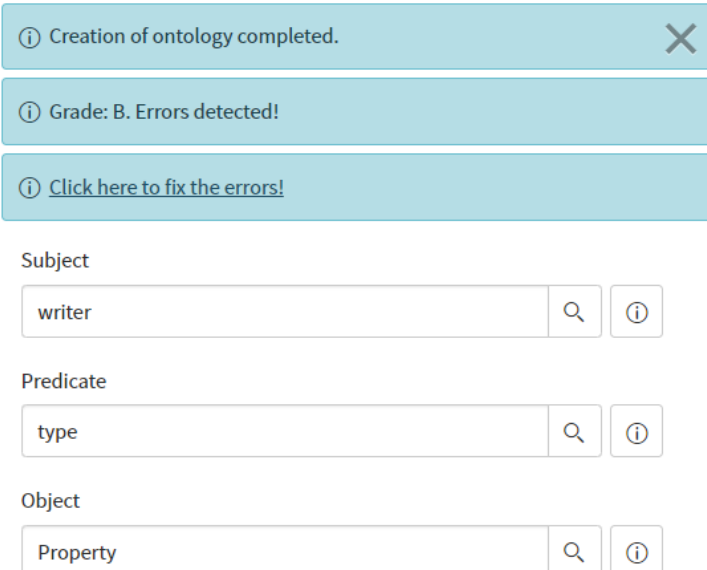
Figure 8.1: An example of JavaScript alert of the grading message

## 8.4 Evaluation of SemanticNow Implementation

With a few years of use experience of the ServiceNow, the implementation of SemanticNow was still a little bit challenging but exciting at the same time. The implementation of the work was more precise when I had made the design work carefully and created many step-by-step graphs to visualise the process. The implementation of Turtle syntax validation was kind of hard to do because I had to research from other online Turtle syntax validators and figure out how they work. I listed all the errors that I found and then I implemented them in SemanticNow. I would have been better if I could see all the possible error types and then implemented in my work. The usability of the SemanticNow was successful since the test users were able to do all the actions required in the tests. Overall, the implementation of SemanticNow was successful, and I managed to iterate the work once.

## 8.5 Discussion

During the time that I was designing the SemanticNow application, I have noticed that it could have a huge benefit for other ServiceNow customers. The way that the data is linked together and using it to find a relationship between different records pointed out to be very useful. SemanticNow can link similar incident records together automatically and collect some monthly statistics to visualise the number of certain incident group categories. The ServiceNow has a built-in feature of bringing data from charts, which the SemanticNow application can also utilise. The common problem with the chart import is that the data is separated from other data and not semantic.



The image shows a user interface for a grading system. At the top, there are three stacked light blue message boxes. The first box contains the text "Creation of ontology completed." with a close button (X) on the right. The second box contains "Grade: B. Errors detected!". The third box contains "Click here to fix the errors!". Below these messages are three input fields, each with a search icon and an information icon (i). The first field is labeled "Subject" and contains the text "writer". The second field is labeled "Predicate" and contains the text "type". The third field is labeled "Object" and contains the text "Property".

Figure 8.2: An example of "addInfoMessage" grading message

The thought of bringing to ServiceNow from the charts and creating an ontology model for it would decrease the error count and incorrect data.

I have planned future improvements for this thesis work, which would increase the number of users. This thesis work only supports Turtle syntax, and the future work is to extend the supported syntaxes to JSON-LD and RDF/XML. The SemanticNow would be able to test the syntax, parse it, and create ontologies to the application. On top of having more supported syntaxes, I have thought to bring the graphical way to present the data. The graphical presentation would present the relationship between the records and separate the subject, predicate and object. The graphical presentation help the users to see how the data is linked and see the data as a whole.

## Chapter 9

# Conclusions

The data on the Internet grows every day, but the structure of the data is far from the desired state. The use of the semantic web would improve the data quality and help machines to understand the data much better. Moreover, the testing and grading have to be included in the semantic web to get the best results out of the ontology creations. The users must learn how to create ontologies that have at least the essential information to make the machines understand the data.

The thesis goals was to design and implement an application called SemanticNow in the ServiceNow platform. The first objective of the application is to get the user input in Turtle format and check that there are no syntax errors. Even though the implementation of the automated tests was self-made instead of using the APIs from the other works, I believe that the testing is robust and working as planned. The second objective was the creation of the ontology from the Turtle syntax and making the data semantic. The parsing of the Turtle syntax and creating subject, predicate and object records was challenging because the relationships between them. Currently, the ServiceNow tend to create records without any relationships. The problem of the design was that the subject could have many predicates and ServiceNow system tend to have one to one record relationships. In the end, the feature called related list on the records made it possible to have many to one record relationships.

The grading system of the work was a motivation and inspiration for the user that created an ontology. The mistakes of the ontology were provided right after a successful ontology creation with SemanticNow application. The user was embraced to improve the ontology creation by giving a grade and links to the mistakes. The ability to go straight to the mistakes improved the chances that the user wants to fix them. The usability of the application was also taken into account, the user testing included technical and non-technical

users, and both of them performed very well. The feedback from the users helped the iteration of the work.

There is a good potential for SemanticNow future since the ServiceNow system is widely used in the world, and it is missing the semantic feature. The SemanticNow would bring a way to bring the semantic to data. The data would be easier to understand by the ServiceNow chatbot, and it can help the users in the customer service. The possible improvements for SemanticNow are an extension of syntax formats and graphical visualisation.

# Bibliography

- [1] AKTURE, N., ET AL. Creating a functioning lignin market: Unified definition of technical lignin grades and end-uses. M.Sc, thesis, Aalto University, 2019-03-12, pp. 99+9.
- [2] ANTONIOU, G., AND VAN HARMELEN, F. *A semantic web primer*. MIT press, 2004.
- [3] ASEMA ELECTRONICS. Asema, 2019. <https://www.asema.com/>. Accessed 21.5.2019.
- [4] BERNERS-LEE, T., HENDLER, J., LASSILA, O., ET AL. The semantic web. *Scientific american* 284, 5 (2001), 28–37.
- [5] BROEKSTRA, J., KAMPMAN, A., AND VAN HARMELEN, F. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *International semantic web conference* (2002), Springer, pp. 54–68.
- [6] BROEKSTRA, J., KLEIN, M., DECKER, S., FENSEL, D., VAN HARMELEN, F., AND HORROCKS, I. Enabling knowledge representation on the web by extending rdf schema. *Computer networks* 39, 5 (2002), 609–634.
- [7] CANDAN, K. S., LIU, H., AND SUVARNA, R. Resource description framework: metadata and its applications. *ACM SIGKDD Explorations Newsletter* 3, 1 (2001), 6–19.
- [8] CARDOSO, J. Developing course management systems using the semantic web. In *The Semantic Web* (2007), Springer, pp. 169–188.
- [9] CUI, Y., SHIVAKUMAR, N., CAROBUS, A., JINDAL, D., AND LAWRENCE, S. Content-targeted advertising using collected user behavior data, Jan. 27 2005. US Patent App. 10/649,585.
- [10] GACKENHEIMER, C. *Introduction to React*. Apress, 2015.

- [11] GRIMM, S., ABECKER, A., VÖLKER, J., AND STUDER, R. In *Ontologies and the semantic web* (2011), Springer, pp. 507–579.
- [12] HAYES, C., AND CUNNINGHAM, P. Shaping a cbr view with XML. In *International Conference on Case-Based Reasoning* (1999), Springer, pp. 468–481.
- [13] HEININGER, R. IT service management in a cloud environment: a literature review. *Studies of the Chair for Information Systems Technische Universität München*, 23 (2012).
- [14] JUVONEN, M., ET AL. Servicenow-tietämyskannan käyttöönotto ja kehitys. B.Sc. thesis, Kajaanin ammattikorkeakoulu, 2013, pp. 28+5.
- [15] KOTHA, V. Customer-centric service management using ServiceNow.
- [16] LASSILA, O., SWICK, R. R., ET AL. Resource description framework (RDF) model and syntax specification. Citeseer, 1998.
- [17] MÄKINEN, G., ET AL. ServiceNow-palvelunhallintajärjestelmän käyttöönotto: case: Lemminkäinen oyj. B.Sc. thesis, Lahden ammattikorkeakoulu, 2011, pp. 71+5.
- [18] MCGUINNESS, D. L., VAN HARMELEN, F., ET AL. OWL web ontology language overview. *W3C recommendation 10*, 10 (2004), 2004.
- [19] MEINEL, C., SACK, H., AND SCHILLINGS, V. Course management in the twinkle of an eye-LCMS: a professional course management system. In *Proceedings of the 30th annual ACM SIGUCCS conference on User services* (2002), ACM, pp. 281–283.
- [20] MILLER, E. An introduction to the resource description framework. *Bulletin of the American Society for Information Science and Technology* 25, 1 (1998), 15–19.
- [21] NECHYPORENKO, T., ET AL. ServiceNow as a platform—practical research. B.Sc. thesis, Haaga-Helia ammattikorkeakoulu, 2015, p. 31.
- [22] PEFFERS, K., TUUNANEN, T., ROTHENBERGER, M. A., AND CHATTERJEE, S. A design science research methodology for information systems research. *Journal of management information systems* 24, 3 (2007), 45–77.

- [23] PÉREZ, J., ARENAS, M., AND GUTIERREZ, C. Semantics and complexity of sparql. In *International semantic web conference* (2006), Springer, pp. 30–43.
- [24] REYBOK, R., HAUGSNES, A. S., KURT, J. Z. I., RHINES, J., GEDDES, H., OSYPOV, V., LEWIS, S., BRADY, S., AND MANNING, M. Techniques for sharing network security event information, July 24 2018. US Patent App. 10/032,020.
- [25] SBODIO, M. L. Sparqlent: A sparql based intelligent agent performing service matchmaking. In *Semantic Web Services* (2012), Springer, pp. 83–105.
- [26] SCHINSKE, J., AND TANNER, K. Teaching more by grading less (or differently). *CBE—Life Sciences Education* 13, 2 (2014), 159–166.
- [27] SERVICENOW. ServiceNow, 2019. <https://www.servicenow.com/>. Accessed 13.6.2019.
- [28] SPORNY, M., LONGLEY, D., KELLOGG, G., LANTHALER, M., AND LINDSTRÖM, N. Json-ld 1.0. *W3C Recommendation 16* (2014), 41.
- [29] SRIDEVI, K., AND UMARANI, D. R. A survey of semantic based solutions to web mining. *International Journal of Emerging Trends and Technology in Computer Science (IJETTS)* 1 (2012).
- [30] TUTAC, A. E., RACOCEANU, D., PUTTI, T., XIONG, W., LEOW, W.-K., AND CRETU, V. Knowledge-guided semantic indexing of breast cancer histopathology images. In *2008 International Conference on BioMedical Engineering and Informatics* (2008), vol. 2, IEEE, pp. 107–112.
- [31] VANEKOVÁ, V., BELLA, J., GURSKÝ, P., AND HORVÁTH, T. Fuzzy rdf in the semantic web: Deduction and induction. In *Proceedings of Workshop on Data Analysis (WDA 2005)* (2005), pp. 16–29.
- [32] W3C. W3C, 2019. <https://www.w3.org/>. Accessed 28.5.2019.
- [33] WOOD, M. *Mastering Srvicenow*. Packt Publishing Ltd, 2016.
- [34] WOODRUFF, T. *Learning ServiceNow*. Packt Publishing Ltd, 2017.
- [35] YERGEAU, F., BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C. M., AND MALER, E. Extensible markup language (XML) 1.0. *W3C Recommendation 4* (2004), 220.

## Appendix A

# Test Questions and Tasks

The user tests were performed after the first implementation was done and before the first iteration. There was two versions of the prototype. The first version included the grading scale with letters A to F and the alert displaying the grade and information. The second version included the grading scale from 1 to 10 and the "addInfoMessage" feature to display the grade and information. There were ten testers from whom five were more experienced and five were less experienced users.

ID	Task	Result	Notes and feedback
1	Import Turtle with incorrect data	10/10	Should show the line numbers where the error occurred
2	Understands and can fix the incorrect Turtle data	10/10	The error messages are clear and easy to understand
3	Import Turtle with correct data	10/10	Would be nice if it redirects to the created statement record
4	Understands the point of grading system	10/10	Better if grade would be numbered
5	Info message or alert feature	10/10	Most of the people chose info message
6	Would improve the ontology immediately	10/10	Yes. With the info message feature because of hyperlink
7	Edit Statement	10/10	Success
8	Edit Subject	10/10	Success
9	Edit Predicate	10/10	Success
10	Edit Object	10/10	Success