Software Requirements Specifications Document

Semantify.io:

A web-based tool to semantify/ontologize the knowledge of a standards.

Team:
Aleksandr Korovin
Shinho Kang
Alexey Karpov
Omar Gutiérrez

Supervisor
Irlán Grangel

1. Introduction

This document is a Requirement Specification for Semantify.io. This is the initial draft for the requirement specification and it will be used for the extensions.

This document is divided in four chapters. The first part introduces the document and gives an brief overview of the system. The overall description of the project and the future system is in the second chapter, where the product perspective, product function, user characteristics and constraints are presented. The detailed description of the requirements, such as functional and nonfunctional requirements, is in the third chapter. The last chapter is an appendix with figures, that are referenced in other chapters.

1.1 Purpose

This document defines the software requirements involved in the project. This document is to be used in the software development process by stakeholders, software developers, testers and project managers in order to identify, validate, design and implement the overall technical features and limitations of the system.

1.2 Scope

This document describes the an overall description and both functional and non-functional requirements for the system. It has an overall description with product perspective, product functions, user characteristics and constraints. It also contains functional and nonfunctional requirements such as performance, reliability, usability, interoperability and security.

1.3 Definitions and Abbreviations

- OPC UA: OPC Unified Architecture
- RDF : Resource Description Framework
- SPARQL: Semantic query language for databases
- OWL : Web Ontology Language
- RDFS: Schema provides a data-modelling vocabulary for RDF data

1.4 System Overview

The dynamic of today’s world imposes new challenges to the enterprises. The globalization, the ubiquitous presence of the internet and the development of hardware systems are some of the technological improvements that provoke changes everywhere. In the engineering and manufacturing domain, there is currently an atmosphere of departure to a new era of digitized production. In different regions, initiatives in these directions are known under different names, such as industrie du futur in France, industrial internet in the US or Industrie 4.0 in Germany. [1]

Industry 4.0 (I4.0) is a term coined in Germany to refer to the fourth industrial revolution. This is understood as the application of concepts such as Internet of Things (IoS), Cyber-physical Systems (CPS), the Internet of Services (IoS) and data-driven architectures in the real industry. [2] Under this concept we can find the term Machine to Machine (M2M) which refers to the communication between devices using a specific communication channel. One of the protocols created to implement M2M interoperability is OPC Unified Architecture (OPC UA)

The proposed system should visualize and analyze OPC UA standard specification documents and provides the functionality of parsing to create a vocabulary and allowing the user to edit vocabularies. This process should be supported with existing fundamental ontologies to simplify the integration into existing conceptual models.
The development is intended to be done using modern web technologies such as Node.js, Express, React.js, Bootstrap etc.

2. Overall Description

2.1 Product perspective

The conceptual process with respect to the handled data is explained as below:

- Parser load categories, features, and values of OPC UA schema into an internal model, which specifies ontology classes, properties, and individuals.
- In the transformation step, the internal model, consisting of entities for classes, properties, and individuals, is turned into an RDF model that describes the final ontology
- Ontology model is serialized as RDF/XML, and all other files that can present ontology.

2.2 Product functions

The use case diagram is presented in follows.

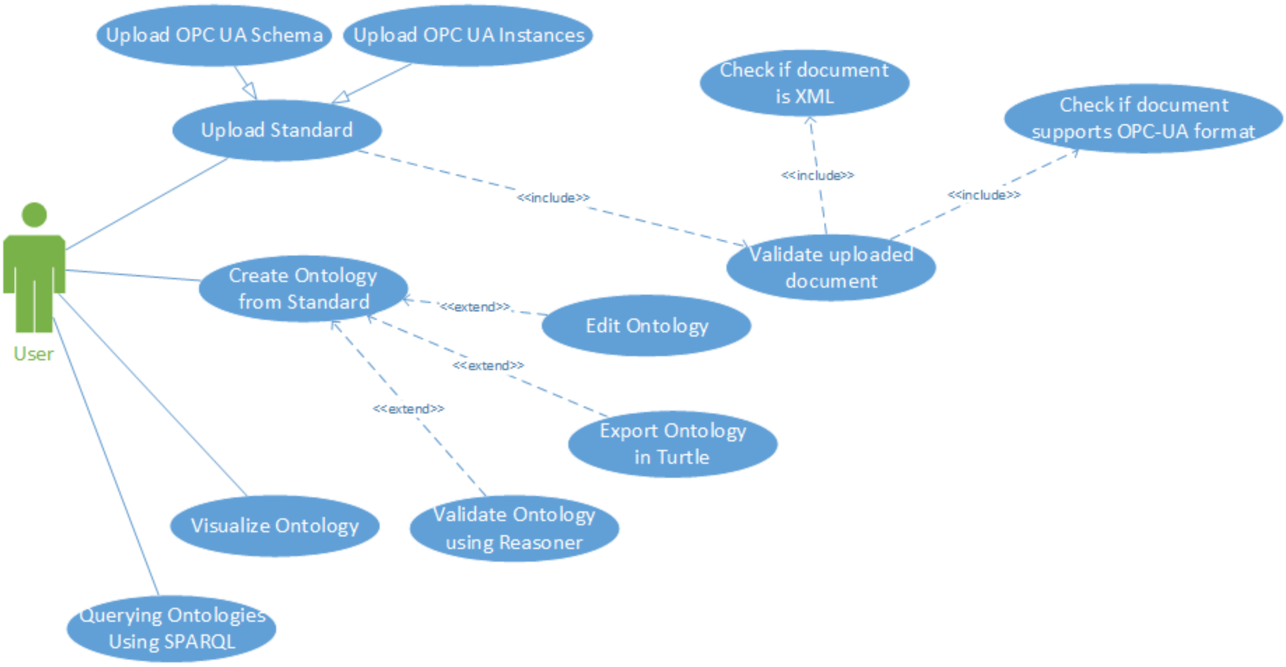

Upload OPC UA Schema
Upload OPC UA Instances
Check if document is XML
Check if document supports OPC-UA format
Upload Standard
<<include>>
<<include>>
<<include>>
Validate uploaded document
User
Create Ontology from Standard
<<extend>>
Edit Ontology
<<extend>>
<<extend>>
Export Ontology in Turtle
Visualize Ontology
Validate Ontology using Reasoner
Querying Ontologies Using SPARQL

Fig 2. Use case diagram

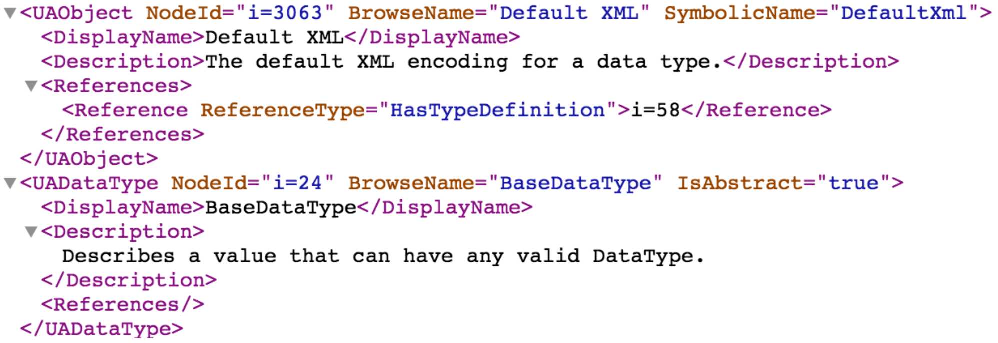
2.2.1 OPC Unified Architecture

OPC Unified Architecture (shortly, OPC UA) is a platform-independent standard through which various kinds of systems and devices can communicate by sending Messages between Clients and Servers over various types of networks. OPC UA defines a common infrastructure model to facilitate this information exchange.

OPC UA is applicable to manufacturing software in application areas such as Field Devices, Control Systems, Manufacturing Execution Systems and Enterprise Resource Planning Systems. These systems are intended to exchange information and to use command and control for industrial processes. [3]

The OPC UA specifications are layered to isolate the core design from the underlying computing technology and network transport. This allows OPC UA to be mapped to future technologies as necessary, without negating the basic design. Two data encodings are defined:

- XML/text
- UA Binary



```
<UAObject NodeId="i=3063" BrowseName="Default XML" SymbolicName="DefaultXml">
  <DisplayName>Default XML</DisplayName>
  <Description>The default XML encoding for a data type.</Description>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=58</Reference>
  </References>
</UAObject>
<UADataType NodeId="i=24" BrowseName="BaseDataType" IsAbstract="true">
  <DisplayName>BaseDataType</DisplayName>
  <Description>
    Describes a value that can have any valid DataType.
  </Description>
  <References/>
</UADataType>
```

Fig. Sample OPC UA Schema (XML)

2.3 User Characteristics

The user shall be an ontology engineer with extensive knowledge on Semantic Web, ontologies, meaning and usage. The user’s knowledge may be backed up by theoretical experience or by practical experience.

3. Specific Requirements

3.1. Functional Requirements

Requirements	Description	Priority
Validate XSD/XML files-	Be sure that XSD (Schema files) are a valid XML . XML Instances must complain with the Schema definition	Very high
Create ontologies from Standard.	Load and read OPC UA Schema and Instance files and translate it into OWL vocabularies/ontologies.	Very high
Visualization of ontologies	The translated file will be visualized graphically in a form of graph/tree	High
Edition of vocabularies/ontologies	Once it has been obtained the ontology the user will be able to modify and update it using a visual editor	High
Management and storage of vocabularies	The user will be able to store and handle the vocabularies, having functionalities such as save, delete, save as, etc.	High
Validate ontologies	The system will be able to validate the logical consistency of the ontologies using a Reasoner.	Medium
SPARQL queries support	The system will be able to execute SPARQL queries on the vocabularies/ontologies.	Very low

3.2. Non-functional Requirements

Requirements	**Description**	**Priority**
Handle different format files	Manage different format files: • RDF/XML (.xml) • Turtle (.ttl) • JSON-LD (.jsonld) • RDF-JSON (.rj) • OWL (.owl, .owx, .rdf) • N-Triples (.nt) • N-Quads (.nq) • Notation3 (.n3)	Medium
Render 100~150 nodes at time	The system will be able to load large OPC UA files and efficiently handle and visualize them keeping a constant time performance. The system will only render a reasonable number of nodes in the edition and visualization.	Medium
Responsive interface	The system will be able to be visualized in different devices such as smart-phones, tablets and computers	Low

3.3. Use-case specifications

Title	**Upload OPC UA Schema**		
ID	UC1		
Description	Standard files are translated into OWL files, that is, all the values in the XML files are transformed into its OWL equivalent		
Preconditions	The user is logged in the system and is located in the main interface		
Postcondition	An ontology file is created from the OPC UA Schema		
Main course		**Actor input**	**System**
	1	The user press the button “Upload Schema”	
	2		The system shows a file explorer to the user
	3	The user chooses a file	
	4		The system validates the ontology
	5		The ontology is saved
Alternate course	1	The user press the button “Upload Schema”	
	2		The system shows a file explorer to the user
	3	The user chooses a file	
	4		The system validates the ontology
	5		The ontology is saved

Title	**Upload OPC UA Instance**		
ID	UC2		
Description	Standard files are translated into OWL files, that is, all the values in the XML files are transformed into its OWL equivalent		
Preconditions	The user is logged in the system and is located in the main interface. It does exist the correspondent ontology created from an OPC UA Schema file.		
Postcondition	An ontology file is created from the OPC UA Instance		
Main course		**Actor input**	**System**
	1	The user press the button “Upload Instance”	
	2		The system shows a file explorer to the user
	3	The user chooses a file	
	4		The system validates the ontology
	5		The ontology is saved
Alternate course	1	The user press the button “Upload Instance”	
	2		The system shows a file explorer to the user
	3	The user chooses a file	
	4		The system validates the ontology
	5		The system show the error “The ontology is not valid”

Title	**Create ontology from standard**		
ID	UC3		
Description	Standard files are translated into OWL files, that is, all the values in the XML files are transformed into his OWL equivalent.		
Preconditions	An OPC UA Schema or an OPC UA instance has been already uploaded.		
Postcondition	An OWL ontology is created.		
Main course		**Actor input**	**System**
		The user upload the OPC UA file (see UC1 and UC2)	
	1		The module “fromOPCtoOWL” receives either the OPC UA Schema file or the OPC UA instance.
	2		The file is parsed to OWL ontology.
	3		The OWL file is validated (see UC5)
	4		The file is sent to the user
	5	The user receives the file	
Alternate course		The user upload the OPC UA file (see UC1 and UC2)	
	1		The module “fromOPCtoOWL” receives either the OPC UA Schema file or the OPC UA instance.
	2		The file is parsed to OWL ontology.
	3		The OWL file is validated (see UC5)
	4		The system shows
	5	The user receives the file	

Title	**Edit Ontology**		
ID	UC4		
Description	The user is able to add and delete triples.		
Preconditions	The user is logged into the system. An ontology file is already stored.		
Postcondition	A personalized ontology is obtained		
Main course		Actor input	System
	1		The system shows an interface with the graph of the ontology
	2	The user modify the ontology with help of an editor	
	3	The user press the button “Save” to store the ontology	
	4		The system validates the ontology (see UC5)
	5		The ontology is saved
Alternate course	1		The system shows an interface with the graph of the ontology
	2	The user modify the ontology with help of an editor	
	3	The user press the button “Save” to store the ontology	
	4		The system validates the ontology (see UC5)
	5		The system shows the next error message to the user: “The ontology is not valid”.

Title	**Validate ontology using a reasoner**		
ID	UC5		
Description	Using a reasoner the system verifies that the ontology is valid.		
Preconditions	The user is logged into the system. An ontology file is already stored.		
Postcondition	The ontology file has been validated		
Main course		Actor input	System
	1		An specific module in the system receives the ontology file
	2		An specific module in the system checks whether it is a valid ontology
	3		A message is sent to another module stating that the file is a valid ontology
Alternate course	1		An specific module in the system receives the ontology file
	2		An specific module in the system checks whether it is a valid ontology
	3		A message is sent to another module stating that the file is not a valid ontology

Title	**Export ontology**		
ID	UC6		
Description	Export an ontology.		
Preconditions	The user is logged and located in the interface of edition. An ontology file is already stored.		
Postcondition	A personalized ontology is obtained		
Main course		Actor input	System
	1		The system shows an interface with the graph of the ontology
	2	The user chooses a format	
	3	The user press the button “Export”	
	4		The system executes the validation of the ontology (see UC5)
	5		The system exports the ontology to the format that the user chose.
Alternate course	1		The system shows an interface with the graph of the ontology
	2	The user chooses a format	
	3	The user press the button “Export”	
	4		The system executes the validation of the ontology (see UC5)
	5		The system shows the next error message to the user: “The ontology is not valid”.

Title	**Query ontology using SPARQL**		
ID	UC7		
Description	Execute SPARQL queries on the vocabularies stored in the system.		
Preconditions	The user is logged and located in the interface of queries. An ontology file is already stored.		
Postcondition	The system returns the result in form of triples		
Main course		Actor input	System
	1		The system shows an interface with a text box giving the user the possibility to introduce SPARQL queries
	2	The user writes the SPARQL query and press the button “Execute”	
	3		The system executes the validation of the ontology (see UC5)
	4		The system executes the query
	5		The system returns the result in form of triples
Alternate course	1		The system shows an interface with a text box giving the user the possibility to introduce SPARQL queries
	2	The user writes the SPARQL query and press the button “Execute”	
	3		The system executes the validation of the ontology (see UC5)
	4		The system shows the next error message to the user: “The ontology is not valid”.

4. Appendix

4.1 References

[1] Grangel-González, Irlán, et al. "Towards a Semantic Administrative Shell for Industry 4.0 Components." arXiv preprint arXiv:1601.01556 (2016).
[2] Drath, Rainer, and Alexander Horch. "Industrie 4.0: Hit or hype?[industry forum]." Industrial Electronics Magazine, IEEE 8.2 (2014): 56-58.
[3] OPC UA Specifications: Part 1: Overview and Concepts(online) http://www.opcfoundation.org/UA/Part1/