Alamzeb Nasar

Widgets based ontology enhancement

Master's Thesis in Information Technology

27. huhtikuuta 2020

Jyväskylän yliopisto

Faculty of Information Technology

Author: Alamzeb Nasar

Contact information: alamzebnasar@gmail.com

Supervisor: Oleksiy Khriyenko

2nd Supervisor: Vagan Terziyan

Työn nimi: Widgets based ontology enhancement

Title in English: Widgets based ontology enhancement

Project: Master's Thesis in Information Technology

Page Count: - 35

Abstract:

We are already in the middle of era of automated machine communication. With a goal to integrate various systems and components (devises, sensors/actuators, applications and services, communication channels, etc.), we are approaching Internet of Things - a forthcoming technological revolution that will radically change our environment and enable innovative applications and services. Every minute huge amount of data is exchanged between machines for various purposes within various sectors. To achieve interoperability between heterogeneous things and move beyond application-specific devices, IoT will require interoperability at multiple levels and rely on the benefits of the semantic technologies at data communication level in particular. To make machine-to-machine communication flexible and allow interoperability between heterogeneous smart entities, *Semantic Web* is a really important concept to make this feasible.

Domain Ontology creation and visualization is a key concept in the world of *Semantic Web*, which is to characterize the domain into classes and describing each class through object and data properties. Currently there are a lot of tools available which allow creation and visualization of ontology like Protégé, NeOn Toolkit, Neologism, OWLGrEd and many more. All of them don't provide a better user interface to the user to create and visu-

alize domain ontology with better usability and context of the class and properties. The widgets ontology will allow the creation and visualization will allow the users a better more contextual way to capture and visualize the domain. The scope of this thesis is limited to data properties only.

The research method used is constructive, as there would be a new module would added to the world of *Semantic Web* and context based GUI would be proposed in this paper.

The end result of this module would a system which has more semantically annotated and easy to use system which gives in depth details about all the properties for example for data property like location or address, there would be a map on which you can mention the address, for temperature there would be a temperature gauge rather than a text field on which you can mention the temperature and much more.

While user creates domain ontology using domain ontology creator tool, he is given suggestions to add the widgets ontologies to choose from according to the domain he is creating ontology for. User selects the relevant widgets ontology and the widgets are imported into the visualization of the tool and user can see the widgets while creating and visualizing the data properties.

Keywords: Semantic web, Ontology, Context-based

Glossary

GUI:	Graphical User Interface
API:	Application Programming Interface

Contents

1 Introduction

In the age where automation is used in every machine and service, *Semantic Web* is one of the key components in the realization of this concept. There is an abundance of data on the web, which is neither categorized nor machine-readable. This makes a person wonder whether the available data is still being appropriately used or not.

Data is becoming more important than the services itself, because the analysis of *Big Data* is now at its peak. With the help of *Semantic Web*, all data can be made machine-readable, and all machine-to-machine communication can be made more feasible. The major hurdle in the way of machine-to-machine communication is the vendor specific API. Each API has its own interface and methods defined, which makes it hard for the machine to automatically detect and read the other newly available machine. For example, Vendor A has a refrigerator with a defined API, which allows to control the temperature of refrigerator. Vendor B has a home temperature control system also with a defined API, which allows to control the overall temperature of the home. If the temperature of the refrigerator is to be changed based on the current temperature of the home, then the home temperature control system must communicate with the refrigerator. The problem is that, if both have different defined APIs, then the communication would not happen. However, with the help of the *Semantic Web* concepts, a middle layer can be defined. This would automatically translate the API of refrigerator to that of the home temperature control system, thus allowing them to interact without any human assistance.

In order to create the middle layer interface, ontologies need to be created. This is the classification of the entities used in the API. For example, the ontology of a smart home would include class for appliances, such as refrigerator, home temperature control and television. Their data properties would define their quantitative and descriptive attributes. The relationship between the different instances and classes is defined by object properties.

There already exist ideas regarding the interoperability of things proposed in other research papers. One such idea is of *Smart Interface*. Khriyenko explains the concept of *Smart Interface* in his research paper "Context-Sensitive Visual Resource Browser". *Smart Inter-*

face visualizes different resources based on context with the help of *MetaProviders*. *Smart Interface* has a GUI-Shell, which allows the user to browse through resources and ontologies.

1.1 Research Question

How to improve the ontology annotation tools using better context-based widgets library for non-technical users?

With the advent of internet of things, interoperability and automated machine-to-machine communication is the way to the future technologies. The ideas of smart homes, smart cars, smartphones and smart spaces are helping the human society to acquire a better and easier way of life and to save energy.

Semantic Web plays an important role in establishing the communication seamless between machines. It allows us to annotate data in a way, which can be read by both machines and humans. If a web page (which is normally created to be visualizing data for humans) is semantically annotated, it can also be read by machines. This means that machines would be able to understand the data on the web the same way humans do. Similarly, it is important for interoperability of things that all APIs have a semantic layer, which can make it possible for appliances from different vendors (having different API descriptions) to communicate with each other.

Tom Gruber states in his online article that "Ontology is a formal specification of a shared conceptualization" [5]. Ontology defines the domain by classifying entities and describing them by properties. Annotation structure for the *Semantic Web* is described through ontology. Unfortunately, there has not been much focus given on improving semantic annotation tools (also known as ontology creation and visualization tools). Protégé, NeOn Toolkit, Fluent Editor and OWLGrEd are few tools that are available on the web. The problem with these tools is that they are very plain and do not describe or visualize the ontology properly.

There are few properties requiring dynamic visualization, for example data properties (such as temperature, location, address, distance, body parts, person and age). Currently, the available tools are not good enough to visualize them according to the context. They just treat them like any other textual or numerical property and no other dimensions have been explored.

Khriyenko has proposed the idea of contextually visualizing resources and entities in his research paper "Context-Sensitive Multidimensional Resource Visualization". He explains that the system should be smart enough to dynamically visualize resources based on context. For example, human relationships can be shown with the help of a family tree. In this thesis, the idea of contextual visualization has been applied to data properties in ontology definition.

The research method taken to approach this problem is constructive as a construction of ontology of widgets is being proposed. In addition, a tool will look for the appropriate widgets according to the domain. During the creation of the domain ontology, the user is given the suggestion in the semantic annotation tool of the available widgets according to the classes in the domain. This can visualize the ontology in a better way. When the user selects the option, the appropriate widgets are imported and used in the SAT (Semantic Annotation Tool) that the user is using.

With the help of the widgets library, annotation tools would be able to visualize the data properties of the entities. For example, data property 'location' can be shown with the help of a map and be pinned on the map. Temperature can be indicated on a thermometer and can be changed by dragging the temperature bar up and down. These widgets would give semantic meaning to each data property of the entity.

The result of adding this module would help the normal non-technical users to create their own ontologies more easily. It is difficult for these non-technical users to understand the concept of ontologies and how to describe them.

1.2 Methodology

The research methodology opted and taken to tackle this problem of usability in semantic annotation tools in this thesis is constructive. Lukka defined the constructive research approach in his journal article as “a research procedure for producing innovative constructions, intended to solve problems faced in the real world and, by that means, to contribute to the theory of the discipline in which it is applied” [6]. This methodology is used to create new artifacts, which can be any sort of solutions or innovations helping to solve real-life problems.

The solution for one of the tools mentioned above ‘Protégé’ will be discussed. A generic solution, which could be applied to other softwares in addition to Protégé, will be explained. Temperature, weather, location2D, location3D, time, age, geography and statistical charts are taken as examples. In addition, the solution visualizing these properties will be shown and proven. The changes needed to the OWL files (which include links to the widgets so that widgets ontology can be created) will be described. There is an extension to the Protégé, which will enable it to search for the relevant widgets ontology and allow it to import it into the User Interface. The scope of the solution is limited to data properties.

The solution would be implemented in the future, but only the theoretical description will be provided (for this thesis). The complexity of the solution is such that it exceeds the scope of this thesis.

2 Semantic Web

Sematic Web is a common framework, which permits the sharing and reuse of data across application, enterprise, and community boundaries [7].

According to W3C, *Semantic Web* is the web of data [7]. There is a large amount of data on the web, which is owned by applications. However, it is not shared across platforms. In case of internet of things, this concept is quite essential as data is shared between different sensors and applications (some of which are from different vendors). In addition, during the communication between different machines or applications, *Semantic Web* provides a layer, which allows automated communication based on common semantic API.

2.1 Ontology

Tom Gruber mentions in his article that "Ontology is a formal specification of a shared conceptualization" [5]. Ontology defines the domain by classifying entities and describing them by properties. Annotation structure for the *Semantic Web* is described through ontology and links are created between resources. Ontology is the description of the domain with classes to annotate entities and properties to describe their attributes and relations between entities.

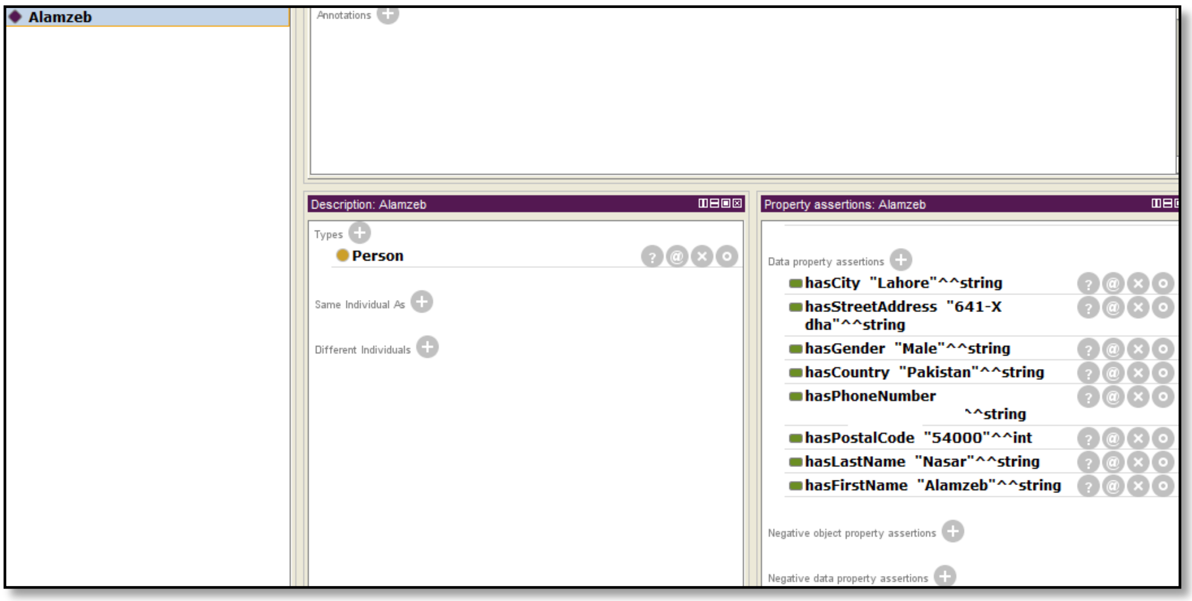
2.2 Data Properties

Data properties are properties used to define the data values associated with the OWL-class. Each data property has a particular data type, which is used to store the values of data property, such as temperature property. The temperature property has an integer or string data type for storing temperature.

If a property *q* is used where a data property is expected, then there should be a triple [8].

q rdf:type owl:DatatypeProperty

Data properties are the focus of this thesis as widgets are being proposed widgets for better semantic visualization of these properties in Ontology Annotation tools and specifically protégé.



Alamzeb

Annotations

Description: Alamzeb

Types
Person

Same Individual As

Different Individuals

Property assertions: Alamzeb

Data property assertions
hasCity "Lahore"^^string
hasStreetAddress "641-X dha"^^string
hasGender "Male"^^string
hasCountry "Pakistan"^^string
hasPhoneNumber ^^string
hasPostalCode "54000"^^int
hasLastName "Nasar"^^string
hasFirstName "Alamzeb"^^string

Negative object property assertions

Negative data property assertions

Figure 1: Entity Properties

Figure 1 shown above portrays the data properties related to an instance of a class person. Data properties are properties, which describe the individual entity and all its characteristics, such as name, city, gender, country and number. These mentioned data properties are manually entered in the tool.

"*Alamzeb*" has been used an example to illustrate the use of the data properties. The entity's full name is "*Alamzeb Nasar*", whose address is 641-X DHA Lahore 54000, Pakistan. It should be noted that these data properties can be better described using widgets.

2.3 Semantic Annotation Tools

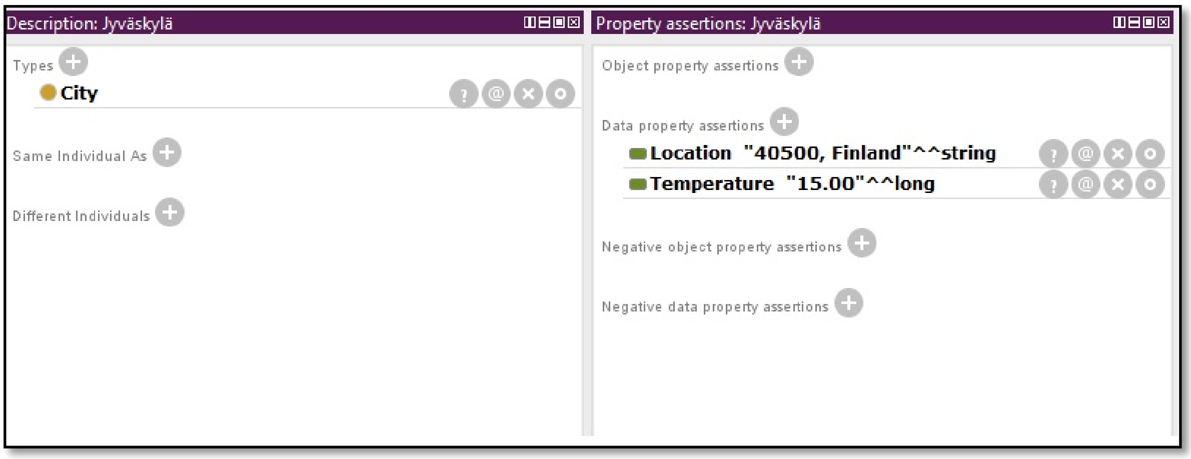
2.3.1 Protégé

Protégé[1] is a free, open source ontology editor and knowledge-based framework.

[1] https://protege.stanford.edu/

The Protégé platform supports two main ways of modeling ontologies via the Protégé-Frames and Protégé-OWL editors. Protégé ontologies can be exported into a variety of formats including RDF, RDFS, OWL, and XML Schema.

“Protégé is based on Java, it is extensible, and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development. Examples are a visual editor for OWL (called OWLViz), storage back-ends to Jena and Sesame, as well as an OWL-S plugin, which provides some specialized capabilities for editing OWL-S descriptions of Web services” [9].


Description: Jyväskylä
Types
City
Same Individual As
Different Individuals

Property assertions: Jyväskylä
Object property assertions
Data property assertions
Location "40500, Finland"^^string
Temperature "15.00"^^long
Negative object property assertions
Negative data property assertions

Figure 2: Protégé data properties GUI

Figure 2 above shows how the protégé visualizes the data properties. *“Jyväskylä”* is used as the example of city entity. The two data properties displayed above are ‘location’ and ‘temperature,’ which are manually entered in the tool. The city is in Finland having a zip code of 40500. The temperature is “15.00” degrees Celsius. It can be observed that the location of the city is difficult to read and visualize without the use of maps.

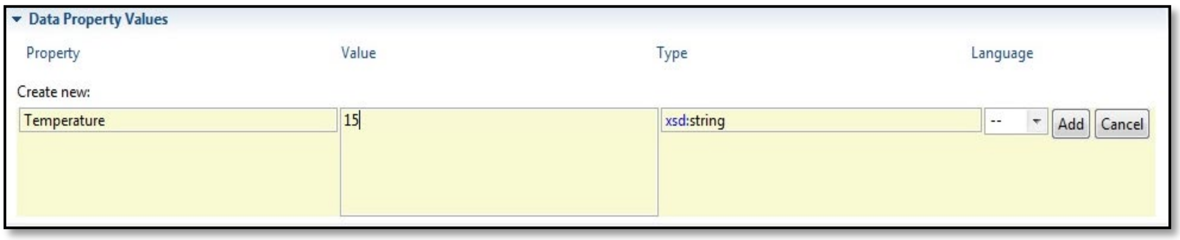
The data types available in current version of Protégé are listed below:

XMLLiteral, usignedShort, usignedLong, usignedInt, usingedByte, token, string, short, real, rational, positiveInteger, Normalizedstring, NonPositiveInteger, nonNegativeInteger, NMToken, negativeInteger, NCName, Name, Long, Literal, language, integer, int, hexbinary, float, double, decimal, datetimestamp, datetime, byte, boolean, base64Binary, anyURI.

It is to be noted that there are no specialized data types for attributes such as temperature, pressure and location.

2.3.2 NeOn

The NeOn Toolkit[2] is the ontology engineering environment originally developed as part of the NeOn Project and now supported, together with other technologies from NeOn, by the NeOn Foundation. open source multi-platform ontology engineering environment, which provides comprehensive support for the ontology engineering life-cycle. The toolkit is based on the Eclipse[3] platform, a leading development environment, and provides an extensive set of plug-ins covering a variety of ontology engineering activities [10].



Property	Value	Type	Language
Temperature	15	xsd:string	--

Figure 3: NeOn Data properties GUI

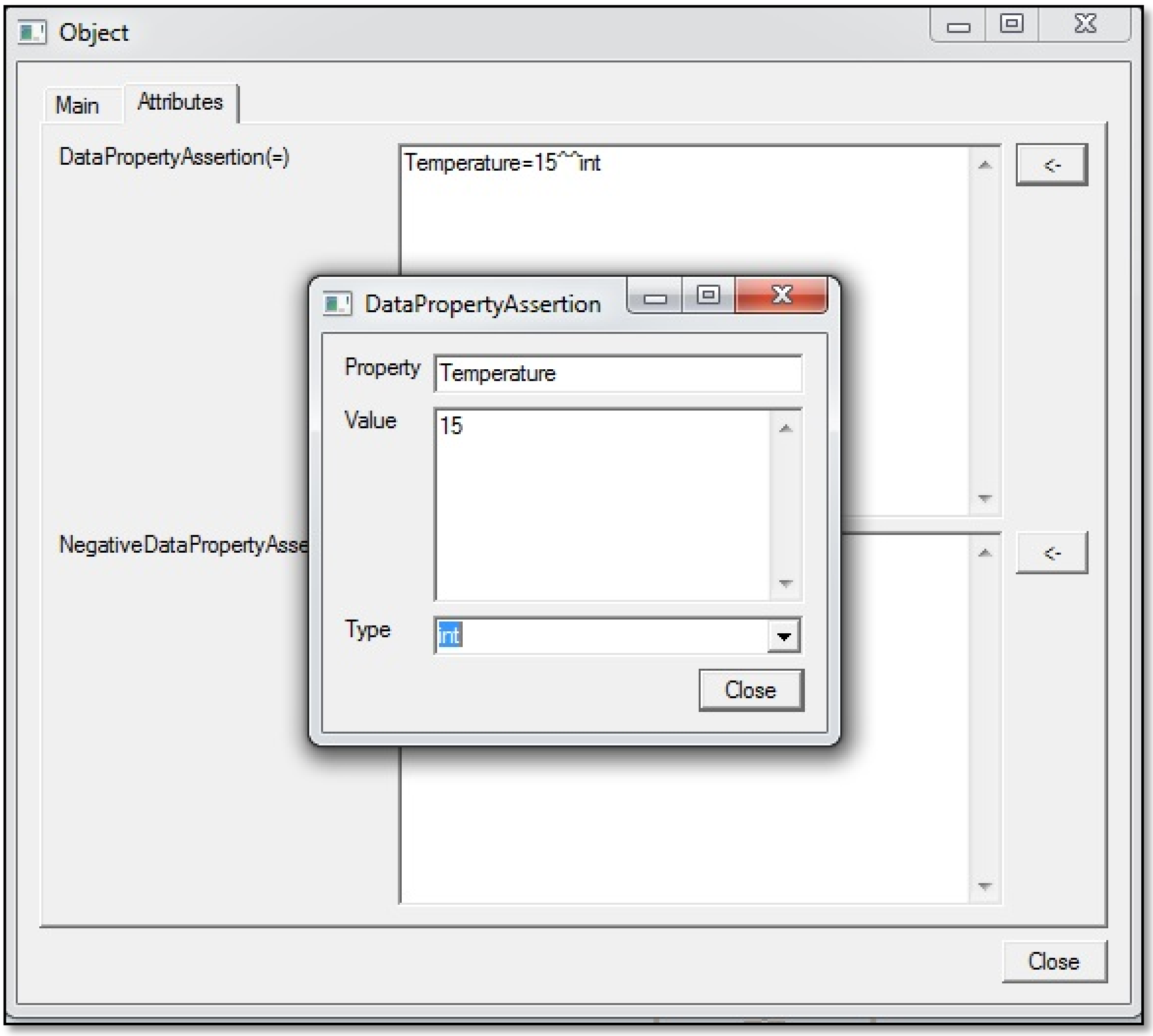
Figure 3 above shows how data properties (e.g. temperature) can be defined in Neon Toolkit annotation tool. This tool displays the data property in terms of name, value, type and language. Just as in Protégé, the input tool or field is described in plain text or integer values instead of contextual way.

2.3.3 OWLGrEd

OWLGrEd[4] is a free UML style graphical editor for OWL ontologies. It has additional features for graphical ontology exploration and development, including interoperability with Protégé [11].

[2] http://neon-toolkit.org/
[3] https://www.eclipse.org/
[4] http://owlgred.lumii.lv/

In case of the representation of data properties, it is not different to any other semantic annotation tools available.



Temperature=15^^int

DataPropertyAssertion

Property: Temperature

Value: 15

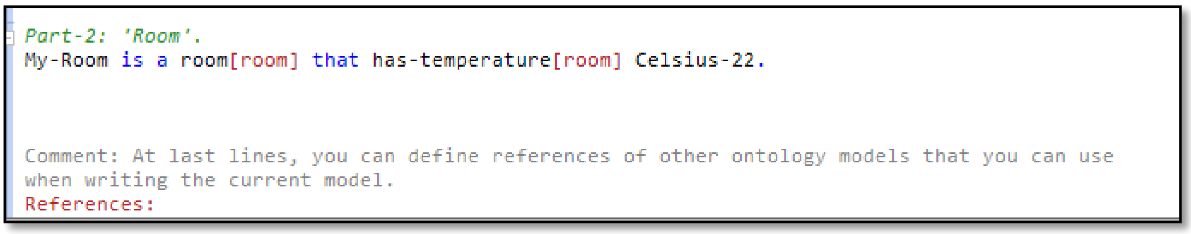
Type: int

Figure 4: OWLGred temperature data property

It can be observed in Figure 4 above that the data property (in this case temperature) is also represented here in terms of name, value and data type (same as Figure 3). In addition, there is no visual or more semantic representation of temperature available.

2.3.4 Fluent Editor

Fluent Editor[5] is a tool for editing, manipulating and querying complex ontologies written in OWL, RDF or SWRL. Fluent Editor is fully compatible with most of the *Semantic Web* W3C standard (i.e. OWL, RDF, SPARQL and SKOS). However, it has an intuitive user interface that uses the Ontorion Controlled Natural Language (OCNL). OCNL is a human friendly alternative to XML ontology language (such as OWL or RDF), but is completely compatible with OWL2, RDF and SWRL. Furthermore, the OCNL can also be used as a query language compatible with SPARQL [12].



```
Part-2: 'Room'.
My-Room is a room[room] that has-temperature[room] Celsius-22.


Comment: At last lines, you can define references of other ontology models that you can use
when writing the current model.
References:
```

Figure 5: Fluent Editor data property representation

The Fluent Editor data property representation also has the same way of displaying data properties with some basic data types, but there is no way to visualize them in a more semantic description way.

[5] https://www.cognitum.eu/Semantics/FluentEditor/

3 Visual Enhancement of Ontology

3.1 Widgets based facilitation for data properties

3.1.1 Widgets

“A widget is an element of a graphical user interface (GUI) that displays information or provides a specific way for a user to interact with the operating system or an application.

Widgets include icons, pull-down menus, buttons, selection boxes, progress indicators, on-off checkmarks, scroll bars, windows, window edges (that let you resize the window), toggle buttons, form, and many other devices for displaying information and for inviting, accepting, and responding to user actions” [17].

Widgets are plugins or tools, which are used to extend to an already created functionality. The great advantage of having widgets is that they can be used across platform, which helps in the reusability of code.

Widgets are extendable GUI components, which will be used to better contextually define the data properties of entities.

3.1.2 Widgets Ontology

Widgets ontology would be the ontology defined for each widget. It will describe the characteristics of a widgets such as Name, Type, Path (to the GUI component), Data properties (which properties widget would be used for). This ontology would be used to create the relationship between the domain ontology and widgets.

This ontology can be read by the code for the widgets to select and show users the appropriate widget according to the selected data property in the domain ontology.

3.2 Widgets For Data properties

3.2.1 Widgets Library

There would be a library created using jQuery, CSS and html. This would be like a reusable library which in future could be used by other platforms as well. The code for displaying the data properties in protégé would need to be modified so that it reads the widget for displaying the specified property.

GUI would also be provided to the user to choose from the list of widgets when user specifies the data property for an object.

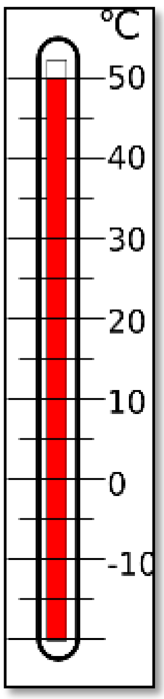


Figure 6: Thermometer for Temperature Property

Figure 6 above shows a widget of a thermometer displaying temperature between -20°C - +50°C. The user can select or adjust the desired temperature on the thermometer for the data property instance.

3.2.2 Embedding the widgets

An ontology is used for embedding the widgets, which will contain the link for each widget. This will also make it easier for user to create and embed their own widgets for their own custom data properties. Ontology data annotation tools like protégé would have to add this extension to their tools which would enable them to parse this widgets ontology and import all the widgets for data properties.

There would be link embedded with each data-property in owl description which would link it to the corresponding widget. When user creates a new data property a dropdown link would appear showing the available widgets to attach with this data property like shown in the figure below:

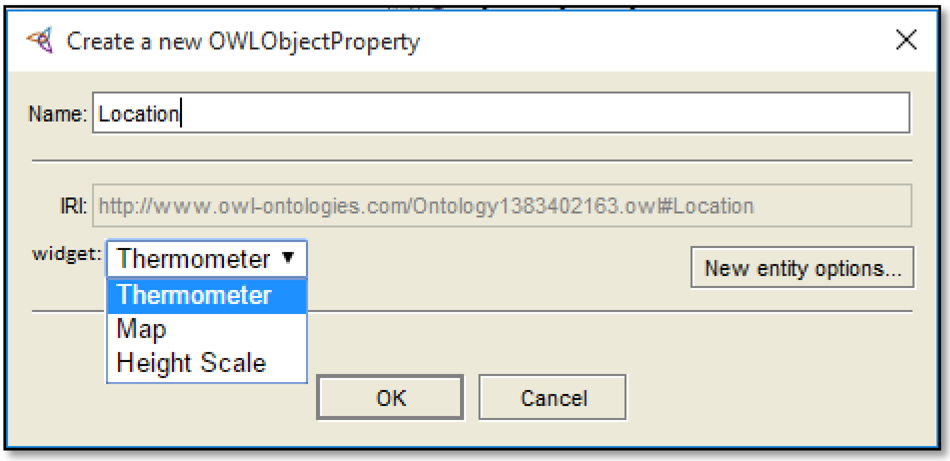

Create a new OWLObjectProperty
Name: Location
IRI: http://www.owl-ontologies.com/Ontology1383402163.owl#Location
widget: Thermometer
Thermometer
Map
Height Scale
New entity options...
OK
Cancel

Figure 7: Creating Data properties with Widgets

It can be seen in Figure 7 above that the user can select the desired widget from a drop-down menu of available widgets for the data type property location. This information is saved with the data type property description in OWL.

This link would be embedded in OWL in the following manner:

<owl:DatatypeProperty
rdf:about="http://www.owlontologies.com/Ontology1383402163.owl#Temperature"
rdf:Widget="http://www.owlontologies.com/Ontology1383402163.owl#Thermometer
"/>

Therefore, the code will read the corresponding widget assigned to the data type property and show that widget when user is entering the value.

Architecture Diagram

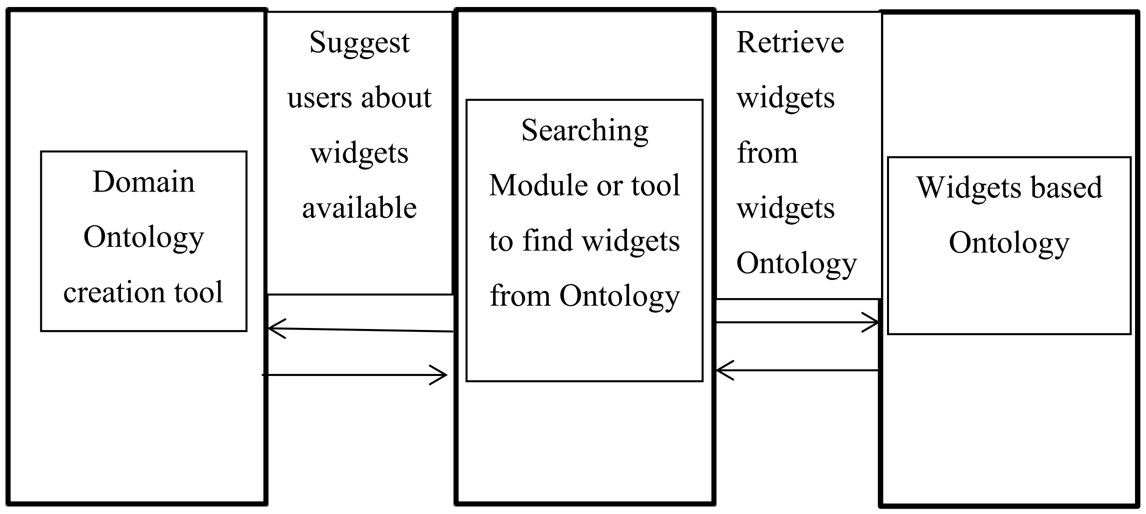

Domain Ontology creation tool
Suggest users about widgets available
Searching Module or tool to find widgets from Ontology
Retrieve widgets from widgets Ontology
Widgets based Ontology

Figure 8: Architecture design for widgets enhancement

Figure 8 above shows the overall architecture of the solution proposed. User, while creating the domain ontology using the annotation tools, will get suggestions when they create data properties for an entity. GUI of the tool will call a middle layer implemented in the plugin, which will in turn search into the widgets ontology library to look for the suitable widgets based on the matching rules defined.

Once the matching widgets are found, they would be retrieved back in the middle layer and then passed onto the GUI for the user to choose. Once the user chooses the correct widget from the list, widget would be retrieved and consequently rendered in the GUI for the user to use to define the data property.

3.2.3 Showing Widgets in protégé

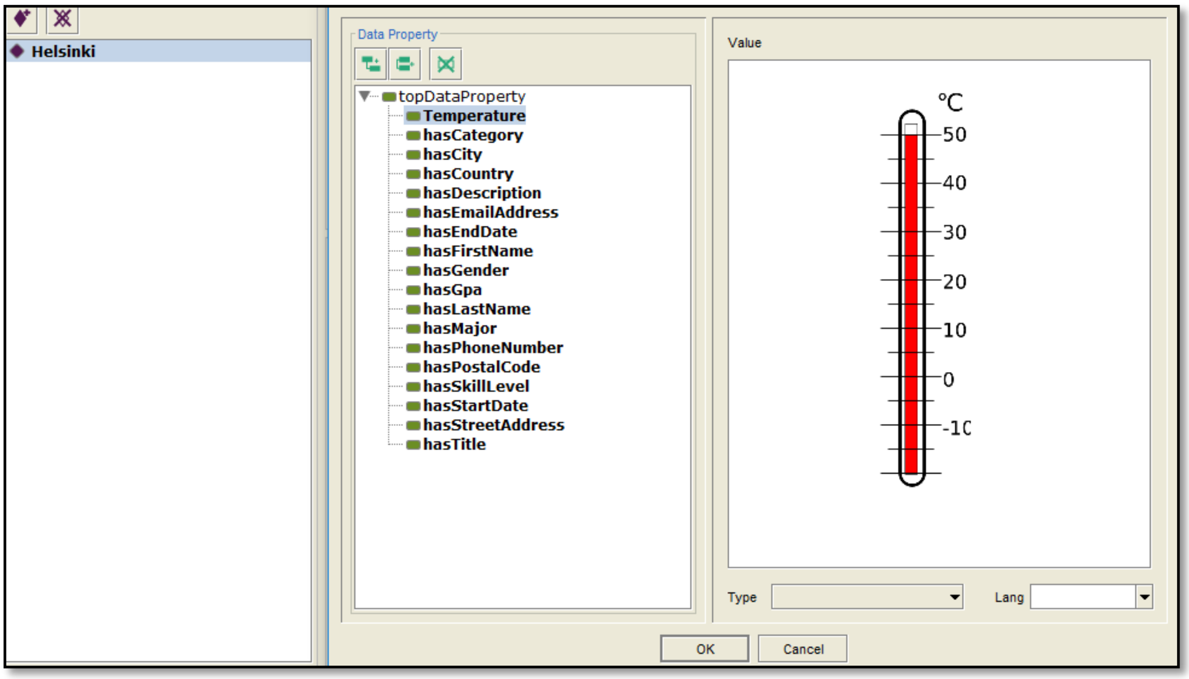


Figure 9: Protégé Temperature property visualization

Figure 9 above shows a city entity with a large list of data properties, which are manually created in the tool. *"Helsinki"* is used as the example of the city entity. The data-property "temperature" is focused in this section. Once the data property "temperature" is created, it can be visualized using the thermometer widget as displayed in the figure above. It should be noted that the 'thermometer' is selected as the widget from the drop-down menu (as explained earlier in Figure 7). In addition, the user can adjust the temperature to the value that needs to be shown.

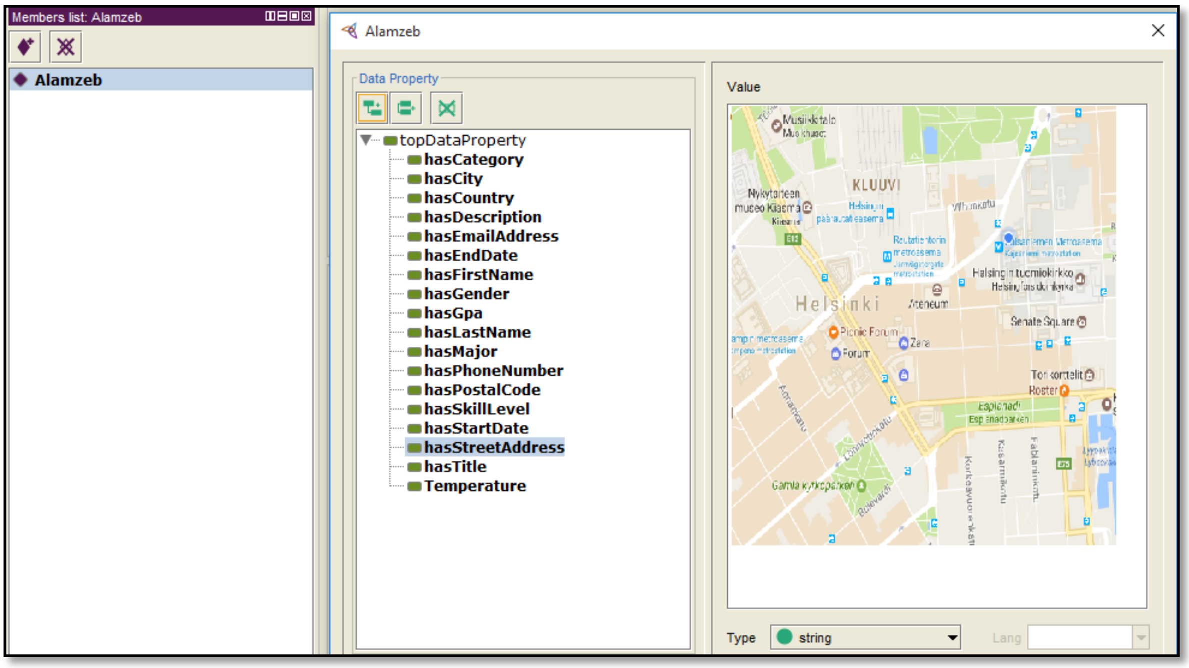


Figure 10: Protege Address property visualization

Figure 10 shows a person entity with a large list of data properties. The data property "address" is focused in this section. Currently, this data property is entered with string text. However, the address can better be described with the help of the map widget. Furthermore, points of interest can be placed on the map. Therefore, it would be easier to visualize the address and see exactly where the person is residing.

3.2.4 Plugin for protégé

It should be noted that this paper has an overview of the solution and not the practicalities of coding the extension and the steps for implementation.

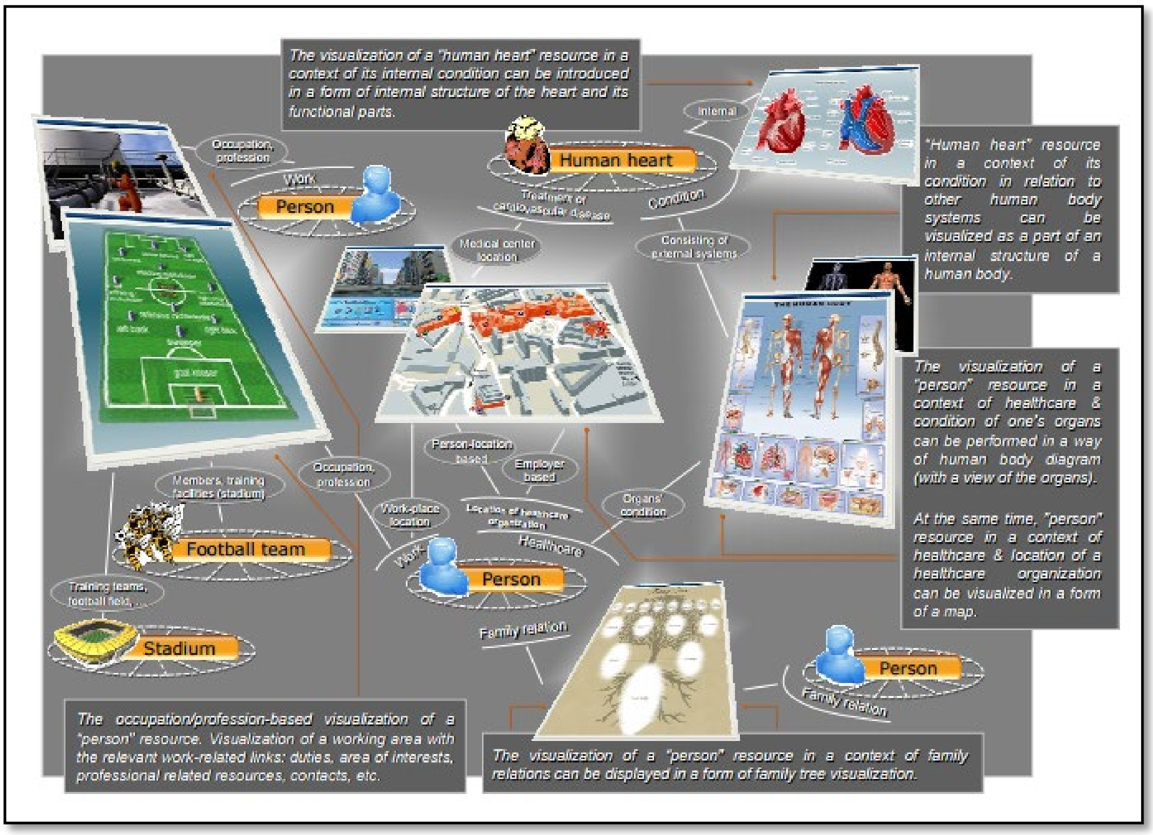
Below is a brief description of writing a plugin for protégé as described in the documentation of Protégé [14].

Protégé 4.1 or higher supports Java 5 and Java 6, therefore the correct version of Java needs to be installed on the computer system. Protégé plugin can be created using Eclipse's plugin for Protégé OWL editor in addition to Maven for the compilation and running of Protégé. On the other hand, IntelliJ IDEA could also be used for creating the plugin. There are also other possible IDEs used for creating the plugin. All the documentation and details of the API are available on the website.

4 Related Work

4.1 Context-sensitive visual resource browser

This world has a huge amount of data generated every minute. Therefore, there is a huge need for displaying the data according to each user's context. Data can be location coordinates, personal medical health information, temperature statistics and any other form being generated from numerous sources. The system should have a more efficient way of showing data. For example, generating a map is the efficient way of displaying location coordinates. Another example is creating images of body parts, which can show the medical health of a patient efficiently. The third example is using a thermometer, which can indicate the temperature in an efficient way.


The visualization of a "human heart" resource in a context of its internal condition can be introduced in a form of internal structure of the heart and its functional parts.

"Human heart" resource in a context of its condition in relation to other human body systems can be visualized as a part of an internal structure of a human body.

The visualization of a "person" resource in a context of healthcare & condition of one's organs can be performed in a way of human body diagram (with a view of the organs).

At the same time, "person" resource in a context of healthcare & location of a healthcare organization can be visualized in a form of a map.

The occupation/profession-based visualization of a "person" resource. Visualization of a working area with the relevant work-related links: duties, area of interests, professional related resources, contacts, etc.

The visualization of a "person" resource in a context of family relations can be displayed in a form of family tree visualization.

Figure 11: Resource Visualization [4]

Figure 11 above shows the example taken from the research paper “Context-Sensitive Multi-Dimensional Resource Visualization” by Oleksiy Khriyenko [4]. It thoroughly explains the concept of resource visualization as shown in the diagram above. For example, the football field seen in the figure visualizes a football team in terms of the formation of the team along with the pictures of the players. This way, both the team’s players and the formation can easily be recognized. The figure also has a family tree, which visualizes family relationships. It makes it more contextual to understand the connections from ancestors to grandchildren, also the relations present at the same level, such as siblings and cousins.

Another example is that of the heart, which could be visually displayed with all the inner components (i.e. the chambers, arteries, valves, vessels and the wall) as shown in Figure 11. This would make it easier to study and analyze the issues and recognize the affected area. Similarly, the whole body is displayed in a contextual way with each body part displayed explicitly. In this way, even the non-medical users (such as patients) could also understand when medical users (such as doctors) are explaining the issues to them.

Last example given in the diagram is that of a person’s workplace and home location, both of which could be displayed using a map. The images of both could be shown on the map along with the distance and the fastest route between them.

There are many other cases apart from the ones mentioned above, which are already in use. However, they still require contextual based visualization. For example, the ontology visualization tools, which is the focus of this thesis, are just plain text and not so easy to use. Introducing contextual based widgets would have a major effect on usability and demand for these annotation tools.

4.2 Smart Interface

Smart Interface has been defined as “the smart adapter for human in Human-Resource communication and intelligence through resource browser” [13]. In the Internet of things, *Semantic Web* can be used to achieve interoperability of different components by different vendors. Every component can expose its service as a *Semantic Web* service and can communicate with other devices through control unit [13].

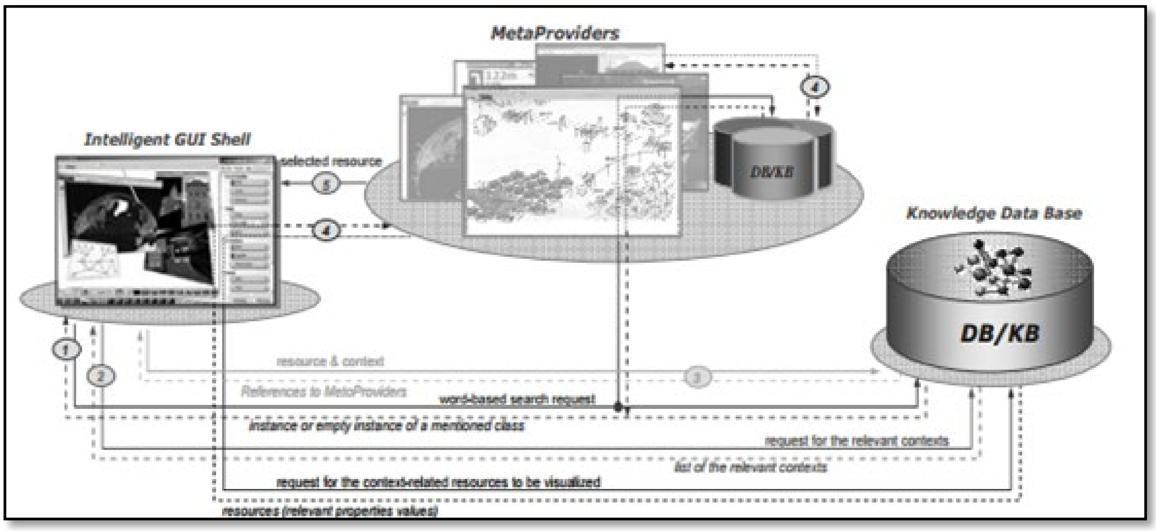

MetaProviders
Intelligent GUI Shell
selected resource
5
4
4
DB/KB
Knowledge Data Base
DB/KB
1
2
3
resource & context
References to MetaProviders
word-based search request
instance or empty instance of a mentioned class
request for the relevant contexts
list of the relevant contexts
request for the context-related resources to be visualized
resources (relevant properties values)

Figure 12: Smart Interface Interaction model

4.3 Personal Assistant:

Personal Assistant (PA) is the supportive agent assigned to every user. One main objective is to deal with human personality and adapt to his/her personal ontology and personal perception of environment. Another objective is to support common semantic standards and approach to be interoperable with other surrounding digital world entities (applications, services and systems) [14].

This concept is presently becoming common and popular. The two most known examples are the personal assistants developed by Microsoft (i.e. Cortana), Apple (i.e. Siri) and Amazon (i.e. Alexa). Personal Assistant keeps track of a person's personal ontology and has connections to ontologies of other services, applications and people (with whom the communication is wished to be made). For instance, when a person wants to order a pizza, his /her PA checks their ontology for their preference of pizza, then checks the nearby available services selling the pizza of interest and in the final step, orders the pizza on the person's behalf.

Figure 13 displays the process of adding intelligence to the Personal Assistant agents. It has been explained in the research paper “Adding Semantic Web Knowledge to Intelligent Personal Assistant Agents” by Garrido, Guetl and Martinez [15]. The authors mention a proposal regarding providing semantic web knowledge to the Intelligent Personal Assistant (IPA) agents. In this way, these agents will behave more like humans and provide solutions to peoples’ questions or problems in the most realistic manner. In addition, these agents will provide guidance to people either by showing them on-screen or communicating by voice.

According to the research paper, an IPA agent will process the solution through four main areas (i.e. Commonsense, Rational, Association and Behavioral). Semantic repositories are added to the IPA agents, with which they will consult and construct appropriate responses in an automated way.

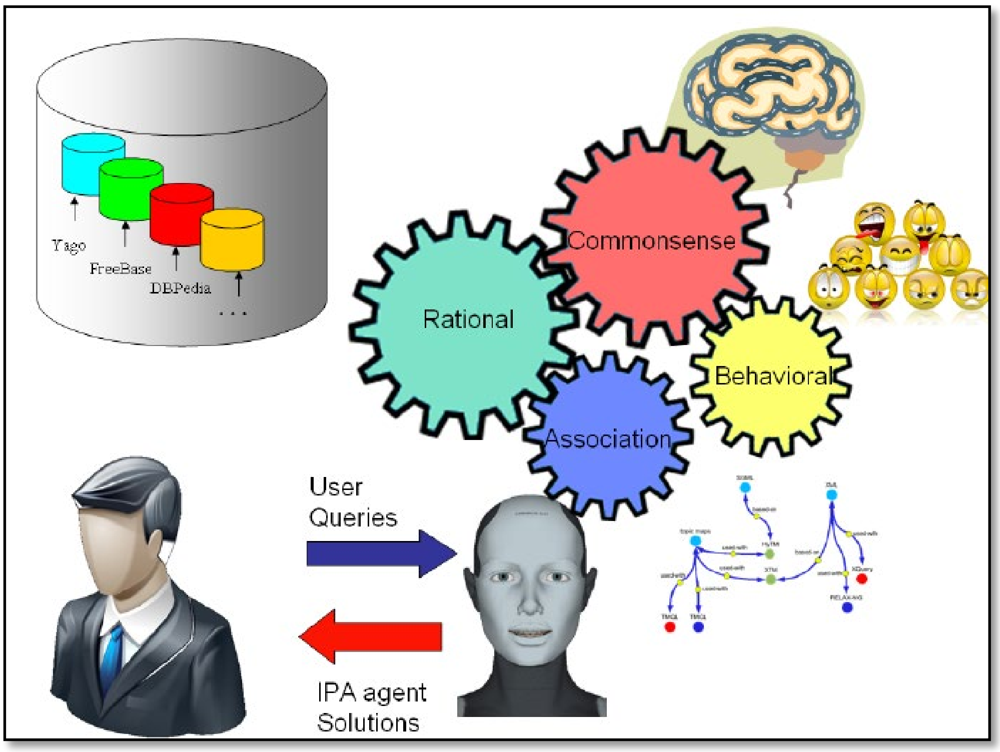


Figure 13: Intelligent Personal Assistant [15]

This is one of the major breakthroughs in history of machine learning. This is because, machines are mimicking humans at a much faster rate. Adding *Semantic Web* knowledge to machines would accelerate this process.

4.4 Visually facilitated Ontology Alignment (Or Interoperability of things):

Interoperability of things (IoT) is the next major development in the current area. The user should be given GUI in order to control units from different vendors for the interaction between the user and Personal Assistant. In addition, the ontology will be aligned for the words, which cannot be mapped automatically [13].

Visually facilitated Ontology Alignment is one of the major concepts in the “internet of things”, because this will help people to connect services and API through its mapping model. As a result, users will be allowed to match the properties in the ontologies, which are not automatically matched. Even though cognitive and semantic algorithms exist to match the properties under different names with certain defined rules, there can still be high chances of error. This is because some properties are not defined in mapping rules. Therefore, human interference is needed in such cases.

There would be a need of widgets after the ontologies have been matched in order to define and visualize the newly mapped properties.

Figure 14 displays the example of Smart Home proposed in the research paper “User-assisted Semantic Interoperability in Internet of Things” by Khriyenko, Terziyan and Kaikova [16]. In this example, both Vendors A and B have defined their own ontologies for the electronic appliances, such as television, refrigerator and central cooling system. When there is a first-time communication between the appliances of both vendors, the alignment service will read the two ontologies and create a mapping visual model for the user to review and modify (if needed). Once the communication is established, both appliances can interact with each other seamlessly. Hence, removing the barrier of the appliances of different brands will ease the compatibility issues.

At the moment, most appliances don’t have APIs based on ontologies. This is because different vendors are not co-operative while defining their API. Therefore, it is not practical to achieve Interoperability of things.

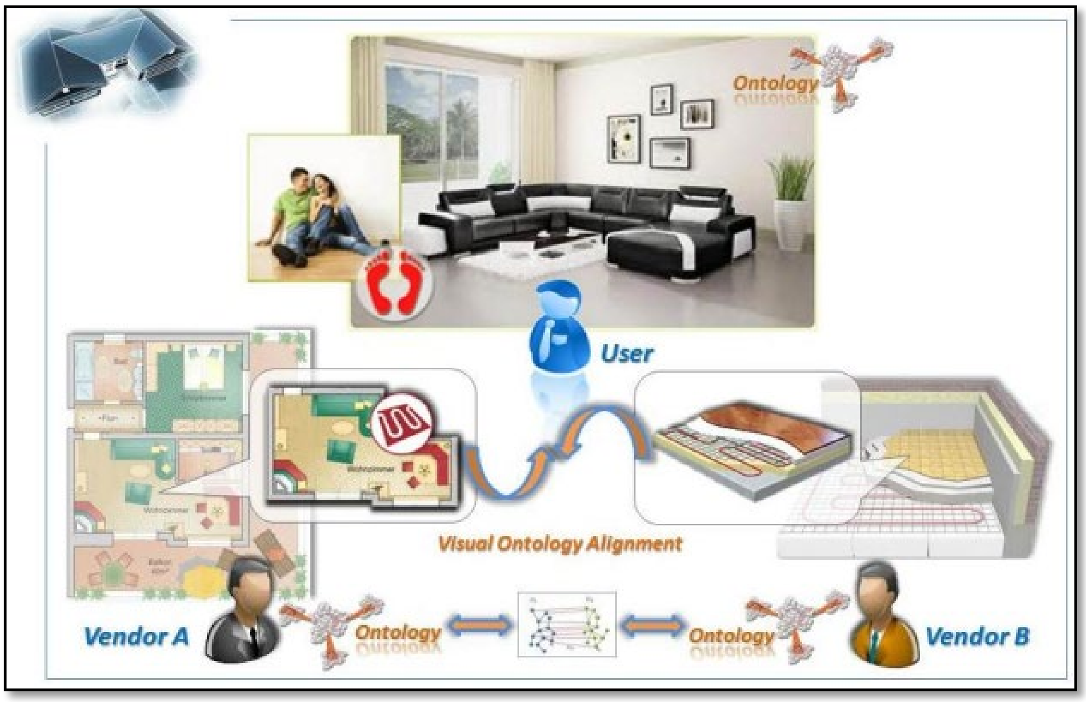


Figure 14: Visually-facilitated Ontology Alignment [16]

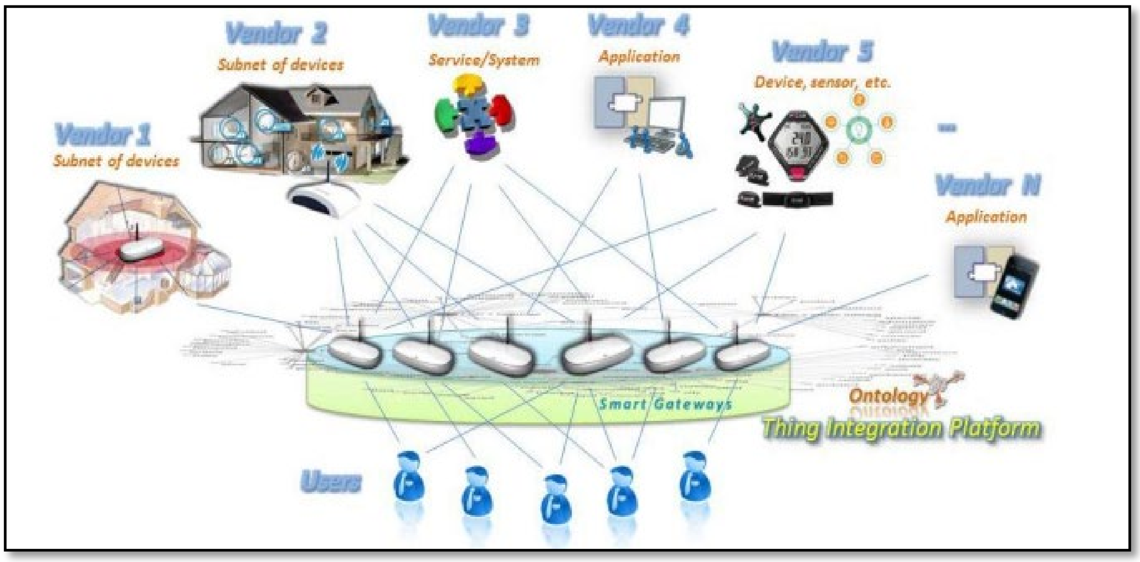


Figure 15: Semantic integration platform for IoT [16]

Figure 15 shows the concept of Smart Gateway. It has been explained in the same research paper “User-assisted Semantic Interoperability in Internet of Things” by Khriyenko, Terziyan and Kaikova [16]. “Smart Gateway” is a central hub for all the appliances and devices, which are used on a daily basis. The user can control different devices, sensors and networks by communicating through the Smart Gateway GUI. There is a possible scenario, where a Smart Gateway is established by the ontology alignment between different vendors. Once the alignment is achieved, the user can then easily control. Another situation is that a user can access multiple Smart Gateways at the same time, because each vendor has established their own Smart Gateway.

Visually facilitated Ontology Alignment can be a revolutionary invention for machine-to-machine communication.

5 Conclusion

5.1 Personal Motivation

The personal motivation for writing on this topic was the pursue of Master of Science degree in web intelligence and service engineering along with *Semantic web* and *Everything to everything interfaces* as basic topics of interest. All these topics involve defining ontologies for seamless communications between applications. Difficulties did occur while using these ontology visualizing softwares. Data-types are limited and hard to visualize the data when creating these tools. It should be noted that while it is difficult for a technical user to create ontology, it would be close to impossible for a non-technical to do the same task. Therefore, these enhancements (with the help of widgets) would greatly help in increasing the usability for ontology annotation tools.

Secondly, being a GUI developer and greatly interested in the usability side of the software, this provided an opportunity towards a better usability aspect of this new field of *Semantic Web* and ontology visualizing tools. In addition, the need of proposing the idea of widgets library for data properties, which could then be further expanded to object properties and better visualize the features of entities and relations between the entities.

It is hoped that with the help of this paper, the visualization of ontologies be revolutionized, and the usage of these tools be brought on a daily basis. For example, a non-technical (or non-IT) user will have the ability to create an ontology involving his/her own self and their relation to other entities (whether of same or different class).

Figure 16 below shows the main entity "*Thing*" with different sub entities in the Protégé tool. One of the sub entities is "*Person*" and the name given to this sub entity is *"Alamzeb"*. The ontology of *"Alamzeb"* is described in terms of related properties, which are Hobbies, Organization (where the person studies or work), Skills and Record. It can be observed in Protégé that the given information is not defined in a contextual way as it is just plain text and therefore, making it hard for the user to graphically relate the objects with their properties. Therefore, with the help of visual widgets, library users can easily visualize the relations.

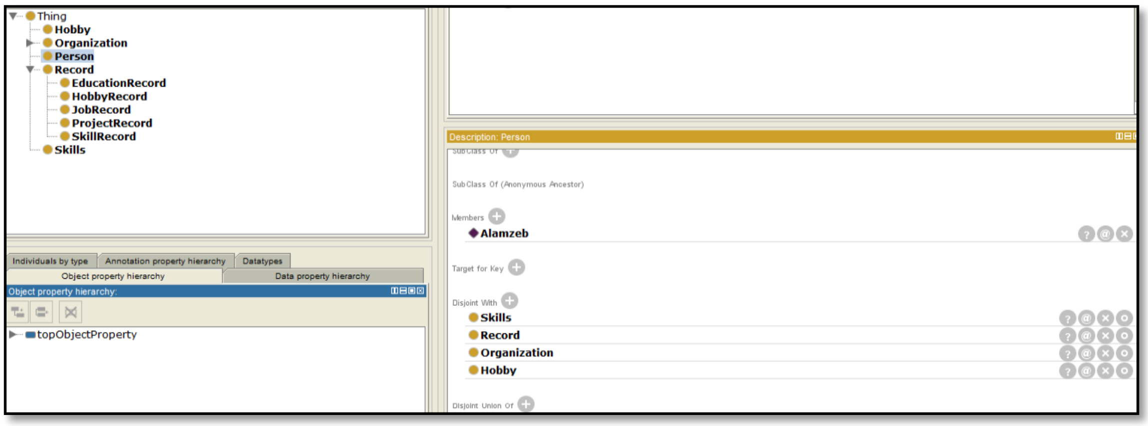


Figure 16: Personal Records Ontology

Figure 17 shows the height widget, which is a long scale for measuring the "height" property of a person. It can be measured in the unit selected by the user.

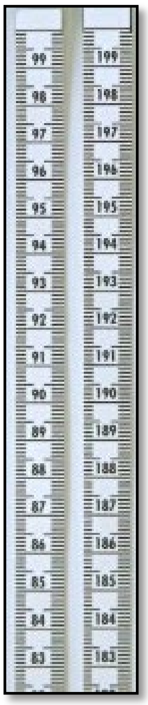


Figure 17: Height Widget

5.2 Target Group:

The first target group for this research paper are the students, who have an interest in *Semantic Web* and ontology concepts and wish to study them in the future. This thesis paper would provide them the basis for expanding the world of ontology creation, thus making it simple to incorporate linked web into the latest applications. These students can study this more in detail and possibly introduce widgets for the visualization of object type properties.

The second target group are the developers, who create applications, such as Protégé, Neon and Toolkit. They can think better ideas regarding making applications more contextual, usable and cognitive for day-to-day users. Therefore, users will most likely link their created data on basis of ontologies and machine-readable content, which could be available across the web. For example, when a user creates a profile on LinkedIn, his/her data information will be saved in a central repository system based on *Semantic Web*, from which it could then be used across the entire web. This means that the user would not have to manually enter the same data repeatedly, thus escaping the long process of filling the application forms while searching for jobs. The data imported from the LinkedIn API may be machine-readable, but it can't be shared across entire web.

5.3 Challenges and Dangers

There are many potential challenges and dangers with respect to this topic as this is a new concept. In addition, there may exist competing ideas that could potentially undermine this ontology-based approach.

The first potential danger is that there wouldn't be a need of creating the widgets library if the widgets are already present and made available in the annotation tools. In addition, they are made part of the API. However, it should be noted that (while these widgets are to be made available), they will most probably be in-built in some specific tools. Therefore, such widgets wouldn't be used universally (meaning in every tool).

The whole idea of *Semantic Web* and ontology design might be replaced by a more efficient and easy-to-use approach. In such case, the annotation tools wouldn't be used anymore. Therefore, this would be a danger to the widgets library.

The main challenge of creating such a library is to make it more robust and usable across all tools. Writing such a code could be a hectic task in terms of learning the works of the tools and creating an ontology-based library, which can be easily incorporated in all the tools.

Another challenge regarding the design of the library is to choose the suitable programming language to write it. JavaScript would be a better choice of language than jQuery, because it is faster and also easier to incorporate in most of the tools. However, there is a possibility that some tools have compatibility issues with a certain programming language, hence making it difficult for the language to be incorporated in the widgets library.

References

[1] RDF Working Group. (2014, February 25). Research Description Framework (RDF). Retrieved June 1, 2015, from https://www.w3.org/RDF

[2] World Wide Web Consortium (W3C). (n.d.). Semantic Web. Retrieved June 1, 2015, from https://www.w3.org/standards/semanticweb/ontology.html

[3] McGuffin, M. (2004, July 1). Graphical widget. Retrieved June 3, 2015, from https://en.wikipedia.org/wiki/Graphical_widget

[4] Khriyenko, O. (2007). Context-Sensitive Multidimensional Resource Visualization. Retrieved June 3, 2015 from http://www.cs.jyu.fi/ai/papers/VIIP-2007.pdf

[5] Gruber, T. R. (n.d.). What is an Ontology? Retrieved June 4, 2015, from http://www-ksl.stanford.edu/kst/what-is-an-ontology.htm

[6] Lukka, K. (2003). The Constructive Research Approach. L. Ojala & O-P. Hilmola (Ed.). *Case Study Research In Logistics* (pp. 83-101). Turku: Kirjapaino Grafia Oy.

[7] Herman, I. (2009). W3C Semantic Web Frequently Asked Questions. Retrieved June 4, 2015, from https://www.w3.org/RDF/FAQ

[8] owl:DatatypeProperty . (n.d.). Retrieved June 5, 2015, from http://www.infowebml.ws/rdf-owl/DatatypeProperty.htm

[9] World Wide Web Consortium (W3C). (2009, December 3). Protege. Retrieved June 5, 2015, from https://www.w3.org/2001/sw/wiki/Protege

[10] NeOn. (2014). Toolkit Wiki. Retrieved June 2015, from http://neon-toolkit.org

[11] World Wide Web Consortium (W3C). (2012, 15 April). OWLGrEd. Retrieved June 5, 2015, from https://www.w3.org/2001/sw/wiki/OWLGrEd

[12] World Wide Web Consortium (W3C). (2015, 15 January). Fluent Editor. Retrieved June 5, 2015, from https://www.w3.org/2001/sw/wiki/Fluent_Editor

[13] Khriyenko, O. (2008). Context-Sensitive Visual Resource Browser. Retrieved June 5, 2015 from http://www.cs.jyu.fi/ai/papers/CGV-2008.pdf

[14] Protégé Wiki. (n.d.). Protege4DevDocs. Retrieved June 5, 2015, from http://protegewiki.stanford.edu/wiki/Protege4DevDocs

[15] Garrido, P., Guetl, C., & Martinez, F. J. (n. d.). Adding Semantic Web Knowledge to Intelligent Personal Assistant Agents. Retrieved June 5, 2015, from http://ceur-ws.org/Vol-687/seres10_submission_2.pdf

[16] Kaikova, O., Khriyenko, O., & Terziyan, V. (2013). End -user Facilitated Interoperability in Internet of Things Visually-enriched User-assisted Ontology Alignment. *International Journal on Advances in Technology Internet*, *6*(1&2), 90–100. Retrieved from http://www.iariajournals.org/internet_technology/inttech_v6_n12_2013_paged.pdf

[17] Rouse, M. (2015, December 18). Widget. Retrieved February 1, 2016, from https://whatis.techtarget.com/definition/widget