University of Zurich^UZH

Protégé Plugin for Change and Impact Visualization

Master's Thesis November 10, 2020

Mirko Serbak
of Gretzenbach SO, Switzerland

Student-ID: 15-701-147
mirko.serbak@uzh.ch

Advisor: **Romana Pernischová**

Prof. Abraham Bernstein, PhD
Institut für Informatik
Universität Zürich
http://www.ifi.uzh.ch/ddis

Acknowledgements

I would like to thank Prof. Abraham Bernstein and Romana Pernischovà for giving me the opportunity to work on this thesis and supporting me throughout the whole process. You have made this part of my studies very interesting and enjoyable for me. I extend this to all other members of the DDIS since many of you have given input and some even participated in my experiment. Of course, I give my thanks to the other participants of my experiment as well.

In addition, I thank my family and friends for always supporting and believing in me. All of you have made it possible for me to be successful on my path by giving me so many opportunities in my life.

Last but not least, I used Protégé as a resource in this work, and Protégé is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

Zusammenfassung

Mit dem Aufkommen des Semantic Web hat die Anwendung von Ontologien in vielen verschiedenen Bereichen zugenommen. Damit einhergehend ist die Entwicklung von Ontologien zu einem aktiven und vielfältigen Forschungsfeld geworden. Ein noch unerforschter Aspekt ist, dass sich viele Ontologieentwickler der Folgen ihrer Modifikationen nicht bewusst sind (Pernischová et al., 2020). Um dieses Problem anzugehen, stellt diese Arbeit ChImp vor. ChImp ist ein Protégé-Plugin, das Informationen über die Auswirkungen von Veränderungen einer Ontologie anzeigt. Darüber hinaus umfasst die Arbeit auch eine Evaluation dieses Plugins mit einer technischen Analyse und einem Nutzerexperiment. Die technische Evaluierung führte zu der Schlussfolgerung, dass das Plugin im Allgemeinen stabil ist und voraussichtlich auf grosse Ontologien skaliert. Das Benutzerexperiment zeigte, dass den Entwicklern die Visualisierung des Plugins im Allgemeinen gefällt. Ob das Plugin einen wahrgenommenen Informationsnutzen vermittelt, konnte im Rahmen der Arbeit nicht festgestellt werden.

Abstract

With the emergence of the Semantic Web, the application of ontologies has increased in many different fields. Along with that, the development of ontologies has become an active and diverse research field. One yet unexplored aspect is that many ontology developers are unaware of the consequences of their modifications (Pernischová et al., 2020). To address this problem, this thesis presents ChImp, a Protégé plugin that displays change impact information about an ontology. Furthermore, the thesis also covers an evaluation with a technical analysis and a user experiment. The technical evaluation resulted in the conclusion that the plugin is generally stable and expected to scale to large ontologies. The user experiment showed that developers generally like the visualization of the plugin. The thesis was not able to determine if the plugin conveys a perceived information benefit.

Contents

1 Introduction

Knowledge graphs, ontologies, and the Semantic Web are the focus of an active and diverse research community. They have their application in many fields such as linguistics, chemistry, or biology (Navigli and Ponzetto, 2012; Hastings et al., 2011; Consortium, 2018). In addition, they are increasingly important for business systems as they can function as a conceptual backbone for IT systems. Their main role in all of these applications is to provide a formalization of a common understanding of a domain. This ability to capture a portion of the real world in a logical framework that is machine-processable lays the foundation for the interoperability of applications and machine learning (Stojanovic, 2004). One area of interest is how ontologies change and evolve. In fact, there are already studies that evaluate the evolution of large ontologies over long time spans and give a lot of insight into how ontologies change (Goncalves et al., 2011). There are also studies that investigate the impact changes have on an ontology. One example of this is the work by Gross et al. (2012) that looks at the impact of changes on functional enrichment analyses. Another example is Pernischová (2019) who proposes a methodology to predict the impact of changes. Studies like these highlight the importance of knowing how changes impact an ontology. Pernischová et al. (2020) observe that ontology engineers often do not know what the consequences of their actions are when they are editing an ontology. In an effort to address this problem, they conducted a user survey to elicit preferences of how changes are displayed. With the help of these preferences, they derived requirements for a Protégé plugin that displays information about the impact of the changes in an ontology. This thesis is a continuation of the work by Pernischová et al. (2020). It describes the process that implemented the proposed requirements and resulted in the ChImp (Change Impact) plugin. Since the implementation of software is always accompanied by requirements and expectations, this thesis also contains a technical analysis and a user evaluation of the plugin.

The technical analysis evaluates the plugin on the dimensions of functionality and stability by addressing the following technical research questions:

RQ1 *How long does it take for the plugin to visualize a change?*

RQ2 *Does the plugin block while calculating metrics?*

RQ3 *How does the plugin perform with ontologies of different sizes?*

The user evaluation consists of an experiment that aims to answer the following research questions:

RQ4 *Do developers like the plugin and its visualizations?*

RQ5 *Is the content useful and informative to developers?*

The first chapter contains a deep dive into the related work pertaining to ontologies, ontology change, and visualization techniques. Subsequently, Chapter 3 describes the implementation of the plugin - from requirements to design, architecture, and the application components. After that, the evaluation of the plugin is presented in two chapters. Chapter 4 is devoted to the technical evaluation and Chapter 5 to the user evaluation. Finally, chapters 6 and 7 outline the limitations, future work, and the contribution of this thesis.

2

Related Work

2.1 Terminology

Depending on the field of study, the term "ontology" can refer to different concepts. Guarino and Giaretta (1995) discuss this multiplicity and vagueness of meaning and argue for clear terminological choices when discussing such terms. Often cited is Gruber (1993) who uses the definition of a conceptualization by Genesereth and Nilsson (1987) to define an ontology as an explicit specification of the objects, concepts, entities, and relationships that are presumed to exist in an area of interest. While this definition is very general, it is sufficient for the purpose of this thesis.

Since the plugin proposed in this thesis focuses on ontology change and change impact, these terms also need to be defined. Flouris et al. (2008) survey the field of ontology change and come to the conclusion that the terminology is not used consistently. They define the term ontology change as "the generic process of changing an ontology in response to a certain need" (p. 118). While this definition might sound simple, the underlying problem is very complex because it encompasses subfields such as heterogeneity resolution, the fusion of ontologies, and versioning. They define the subfield ontology evolution as the modification of an ontology without data loss or the negation of its validity. In contrast, they define ontology versioning as the same task, but over many versions of an ontology. Expanding on that definition, ontology evolution can be seen as a process that determines what needs to be changed, how the change is going to be resolved, and how consistency is ensured (Stojanovic, 2004). In a process-centric view like this, the distinction between ontology evolution and versioning can become blurred. Hence, some authors argue for a broader view that encompasses both aspects (Zablith et al., 2015). An example that investigates ontology evolution and versioning is COnto-Diff, an approach to produce a mapping between versions of ontologies (Hartung et al., 2012).

In the context of ontologies, change impact can be categorized with various criteria to enable the analysis from different angles: structural and semantic impact, addition and deletion impact, ontology, annotation and content impact and Abox and Tbox impact (Abgaz et al., 2011). Another approach by Gonçalves et al. (2011), classifies changes into categories of effectual and categories of ineffectual changes. The idea behind this approach is to distinguish between changes that have a logical impact and changes that do not. As mentioned in the introduction, there are also authors that try to capture impact by analyzing their effect on statistical applications (Gross et al., 2012). In addition, There are authors that attempt to capture the impact with the help of graph theory (Pernischová et al., 2019; Pernischová, 2019).

The current implementation of the ChImp plugin is aimed at incremental ontology modifications in the ontology editor Protégé. It is not concerned with the actual modification itself. Instead, it is attempting to display ontology changes in real-time. That being said, it cannot always depict the whole change and therefore should be seen as a visual summary of incremental modifications in Protégé.

2.2 Ontology Metrics

An important and non-trivial problem is the assessment of the quality of an ontology. Inherent to the makeup of an ontology is the concept of consistency, which describes the assessment of its logical correctness. While reasoners usually determine the consistency, there are also other types of manually calculated logical indicators such as conflict measures. Both the conflicts and the consistency can be used to assess the quality of ontologies (Arpinar et al., 2006). The logic of an ontology reveals important information. However, it only covers one viewpoint. To remedy this, researchers use metrics in their evaluations. Metrics allow them to succinctly summarize a large range of viewpoints about an ontology. One example for this is Orme et al. (2007) who propose several metrics that reflect the complexity and cohesion of an ontology. Another example is Duque-Ramos et al. (2013) who attempt to assess the general quality of an ontology with a framework that includes metrics. Burton-Jones et al. (2004) draw inspiration from semiotic theory and present a metrics suite focused on the syntactic, semantic, pragmatic, and social aspects of an ontology. Others derive their metrics from e.g. software quality standards (Duque-Ramos et al., 2011; Orme et al., 2007) or graph theory (Zhang et al., 2010). The usefulness of such metrics may be limited to a specific area of interest or purpose. For example, Manouselis et al. (2010) evaluate metrics in the biomedical domain. As a result, many researchers propose metrics that are highly specialized to a specific use case. A special variant of this are metrics that depend on inputs (Tartir and Arpinar, 2007) or metrics that can be weighed (Duque-Ramos et al., 2016). In theory, metrics like these can be adjusted to certain use cases due to their dynamic nature.

Since metrics can be calculated on any ontology, they can also be used for multiple ontology versions. Duque-Ramos et al. (2016) show this by applying their quality framework on multiple ontology versions. Closely related to this is the concept of semantic drift, which is defined as the change in meaning that occurs over different versions of an ontology (Stavropoulos et al., 2016). This means, that the understanding of a concept can gradually change over time (Gulla et al., 2010). In this context, similarity measures between two versions of a label in an ontology can be used to construct metrics that capture semantic drift (Stavropoulos et al., 2019).

There are attempts to compare and assess the quality of metrics (Sicilia et al., 2012).

However, there are no universally objective measures against which metrics can be tested. For this reason, authors such as Duque-Ramos et al. (2013) consult expert opinions about their metrics suite. Duque-Ramos et al. (2013) found that even though the experts provided them with improvement suggestions the experts also considered that some metrics are only useful in a specific context.

To encompass a wide range of aspects, the current implementation of ChImp uses metrics that are proposed in many different studies. The main criterion for the selection is how many times the metrics are used in past research. However, the list is also a reflection of personal preferences. For an overview of all the metrics that are used in the plugin consult Section 3.4.1.

2.3 Visualization Techniques

The goal of the ChImp plugin is to support developers in understanding the impact of their changes. One approach to better convey this information is visualizing the ontology and the metrics. To find an appropriate visualization method, this thesis surveys research in the fields of ontology visualization and software visualization. Pernischová et al. (2020) also employ some of these aspects and techniques.

2.3.1 Ontology Visualization

Katifori et al. (2007) and Dudáš et al. (2018) are two important contributions that survey the state of the art in ontology visualization methods. Even though they were published a decade apart, they both observe that there is not one specific method that has established itself as the de-facto standard. Moreover, they both reason that this is because specific applications require custom-fit solutions and argue in favor of approaches that offer more than one visualisation method. In addition, Dudáš et al. (2018) argue that there are not many authors that adapt existing solutions. With that being said, their works are very insightful because they present a large array of methods and explore areas such as the temporal dimension used in versioning. Many methods, as well as the two surveys above use requirements that are based on taxonomies like those of Shneiderman (1996) or user studies like Kriglstein (2009). One prominent problem is the fact that a lot of these tools are directed at domain experts. Some approaches attempt to address this by creating an intuitive experience for their visualizations (Kuhar and Podgorelec, 2012).

The Protégé plugin library also already contains various plugins that use visualization techniques to present their content (e.g. Lohmann et al., 2014; Sintek, 2003; Hussain et al., 2014). In fact, there are already Protégé plugins that concern themselves with the display of changes in ontologies. The Change Analysis Tab of the Change Management plugin by Falconer et al. (2011) allows users to explore all changes that were stored with Collaborative Protégé. Change View provides a list of all modifications that were performed in a Protégé session (Drummond, 2011). Existing plugins such as these function similarly to the ChImp plugin proposed in this thesis. However, no plugin that I am aware of displays change impact information about modifications in a Protégé session in real-time.

2.3.2 Software Visualization

Another research area that employs techniques to visualize a type of evolution is software visualization. One technique in this field are polymetric views. Lanza and Ducasse (2003) introduce this concept, which is meant to enhance the users' mental image of the software. Concretely, software metrics are mapped to visualization attributes such as size, color, or position. As a consequence, the visualization is able to indirectly display multiple aspects of the software. Polymetric views can also be used to compare different software versions (Lanza and Ducasse, 2005). The idea here is to visualize the difference in the metrics. In a previous approach, Lanza and Ducasse (2002) focus solely on software evolution by using matrices that display different versions in their columns. Other methods that focus on software evolution employ the third dimension (Gall et al., 1999; Lanza et al., 2009) or introduce new graphical elements such as Evo-clocks, which display versions as sections in a pie (Alexandru et al., 2019). Another approach is the use of kiviat graphs to visualize software evolution (Pinzger et al., 2005). With their help, it is possible to display many different metrics over multiple versions that are represented by areas in the graph.

3 Implementation

As stated in the introduction, Pernischová et al. (2020) already presented a version of the ChImp plugin. The following sections will recapitulate certain elements from that paper and present additional documentation about the implementation of the ChImp plugin.

3.1 Design Phase

3.1.1 Initial Designs

The research presented in Chapter 2 shows how metrics, graphs, and other elements are used to display the state of ontologies. An initial design phase focused on how these techniques can be adapted to ontology change impact. The following elements were the result:

An element that shows the last changes that were made in the ontology:

This element is supposed to show the change by listing the specific axioms that Protégé added or removed. The initial design phase did not focus on this element since it is merely a list. However, the discussion of the element brought up the question of how color is used in the plugin. If for nothing else, colors can be useful in order to separate two different aspects. The discussion resulted in the decision that colors indicate additions and removals throughout the plugin but do not impose a value judgement. This means that the colors cannot have a positive or negative connotation - as would be the case with red and green - because it is not trivial to judge the value of ontology changes. Ultimately, Pernischová et al. (2020) use these deliberations to define their fourth requirement.

An element that shows ontology metrics:

This element is supposed to display metrics about the ontology and if possible about the last change. Figures A.1-A.10 in the appendix contain initial designs of this element. These designs are heavily influenced by Protégé's ontology metrics view. Other influences are stock market apps such as the yahoo finance mobile app (yah). Apps like that display daily stock price changes behind the prices with red or green percentages. The same principle was applied to ontology metrics. These change indicators were chosen for all future designs since they are able to elegantly convey change information about a metric. As an alternative, inspired by Lanza and Ducasse (2002), matrices that display multiple metrics (see Figure A.4) were considered. However, they were ruled out due to space constraints.

An element that shows the consistency of the ontology:

This element is also simple since its only purpose at that point was to display the consistency of the ontology. However, it highlighted the importance of reasoners in the plugin. Nevertheless, there are no early designs for it.

A graphical element that visualizes ontology change and impact

In the early stages of the design, a lot of effort went into the design of a graphical element that could present change and impact information about an ontology. One such approach was a node-link type diagram that put the changed node in the center of the diagram and arranged all affected classes around it (see Figure A.7). This diagram opened up many possibilities to display information about the change. The discussion included, for example, polymetric views (see Section 2.3.2). Polymetric views can increase the information density and expressiveness of a diagram by mapping metrics to nodes, edges, and colors. Ultimately, complex approaches like this were dropped in favor of other simpler ones that require less design and development effort. In addition, the goal of the plugin is to visualize ontology changes independent of the subject area of the ontology. As discussed in Section 2.3.1, this is not trivial as there is not a single approach that fits all purposes.

The initial design does not contain any elements that use the third dimension. Such elements would likely require a lot of design effort and be computationally expensive. It is vital that the plugin does not contain anything that would slow down Protégé as a whole. In addition, visually intensive elements would distract users and do not reduce complexity in an already complex development environment.

As can be seen in Figures A.1-A.10 in the appendix, many early approaches are line or area charts. A major advantage of such graphs is that they are easy to understand and able to convey changes over time. One major challenge is the display of multiple metrics in such a chart. Since primitive metrics are not on the same scale, a line chart can at most display two of them at once. Area charts such as in Figure A.5 suffer from the same problem. If drawn cumulatively, line and area charts are able to display slightly more metrics. However, those metrics need to be on a similar scale. Otherwise the chart will be too large. Inspired by Pinzger et al. (2005), Figure A.3 shows a Kiviat diagram. As stated by the authors, it enables the display of many metrics over many versions.

Equipped with these early designs, Pernischová et al. (2020) derived prototypes that they used in their survey. Even though the results did not paint a clear picture, they were able to derive the requirements listed in Section 3.2.

3.1.2 Further Development

After internal discussions about the existing designs, I decided that the main goal was to make ChImp look like it is part of the Protégé development suite. Hence, designs such as tiles that did not look similar were dropped. In fact, the design was adjusted to mirror that of Protégé's own ontology metrics view.

Initially, the ChImp plugin did not contain a graph of any kind. However, in a later version, a line chart was added. Figures A.11-A.15 show a new design iteration for this component. These designs are all similar and mainly revolve around the button placement and chart arrangement. The results from Pernischová et al. (2020) do not indicate that there is enthusiasm for a specific chart. Hence, the ChImp plugin includes the most simple variant. Note that these designs already contain the other panels. For a more detailed description of the current state of the design consult Section 3.4.2.

3.2 Requirements Analysis

One important contribution by Pernischová et al. (2020) is the elicitation of a list of requirements for visualizing and conveying ontology changes. To investigate the visual requirements, the authors conducted a user survey with 12 experienced semantic web practitioners. The answers from that survey resulted either directly or indirectly in the following 7 requirements.

R1 *ChImp should list the applied changes.*

R2 *ChImp should inform the user about the consistency of the loaded ontology.*

R3 *ChImp should show primitive and composite measures in a table, visualizing the new value and its difference to the old value based on the applied changes.*

R4 *ChImp should use colors to indicate changes.*

R5 *Ontology release notes should include the number and types of changes.*

R6 *Ontology release notes should include the result of a consistency check.*

R7 *Ontology release notes should report changes to the materialization*

In addition, they also formulated three requirements that are based on experience and general best practices.

R8 *ChImp should allow the user to chose between the presentation of metrics either in absolute values or as percentages.*

R9 *ChImp should let the user choose between using only the last change or all changes for the calculation of primitive and composite measures.*

R10 *ChImp should be responsive.*

3.3 Architecture

3.3.1 Protégé Plugin

Protégé is an OWL 2 ontology editor written in Java. The project was initially started in the 1980s and Protégé has become the most widely used software for building and maintaining ontologies (Musen, 2015). Closely connected to Protégé is the OWL API, which enables the handling of OWL 2 ontologies (Horridge and Bechhofer, 2009).

Protégé uses OSGI, which is a Java framework for the implementation of modular software systems. It consists of a set of specifications that define how modules should be implemented and how they should interact (osg). The main problem that OSGI tackles is complexity. The modular architecture breaks down software into modules. It also implicitly enforces principles such as loose coupling and semantic versioning. Protégé uses the Maven Bundle Plugin to create its OSGI architecture. In this architecture, individual plugins represent the modular components. The goal is that all Protégé plugins only expose interfaces with which they interact with their surroundings.

Protégé plugins are by definition inside an OSGI architecture. However, that does not say anything about the inner workings of an individual plugin since a plugin could theoretically try to defy the core principles. With that being said, the architecture of ChImp plugin was set up to fit seamlessly into its environment.

3.3.2 Plugin Architecture

Figure 3.1 shows the class diagram of the ChImp plugin. It shows that the ChImp plugin is a module of its own and that it is separated from the Protégé editor. The following paragraphs highlight several design aspects.

Metric Strategy

All metrics inside the ChImp plugin extend the abstract `Metric` class. The metrics themselves implement their individual calculation strategy by overriding the inherited `calculateMetric()` function. This design pattern enables the usage of inheritance polymorphism throughout the application. Concretely, this means that all metrics can be accessed through a common interface. In addition, it makes the plugin very extendable to new metrics. If a new metric can be implemented as a child of `Metric`, it can be used interchangeably with other metrics.

Protégé Interface

ChImp interacts with the Protégé editor by using the classes that Protégé exposes as its interface. Specifically, ChImp extends the class `AbstractOWLViewComponent` through which Protégé exposes its `OWLWorkspace`. With this, ChImp has access to the Protégé workspace, which includes ontologies, managers, and reasoners. The `ChimpPlugin` class then injects the relevant attributes of the `OWLWorkspace` into the display panels of the ChImp plugin. The injection of the reasoner and the active ontologies into the

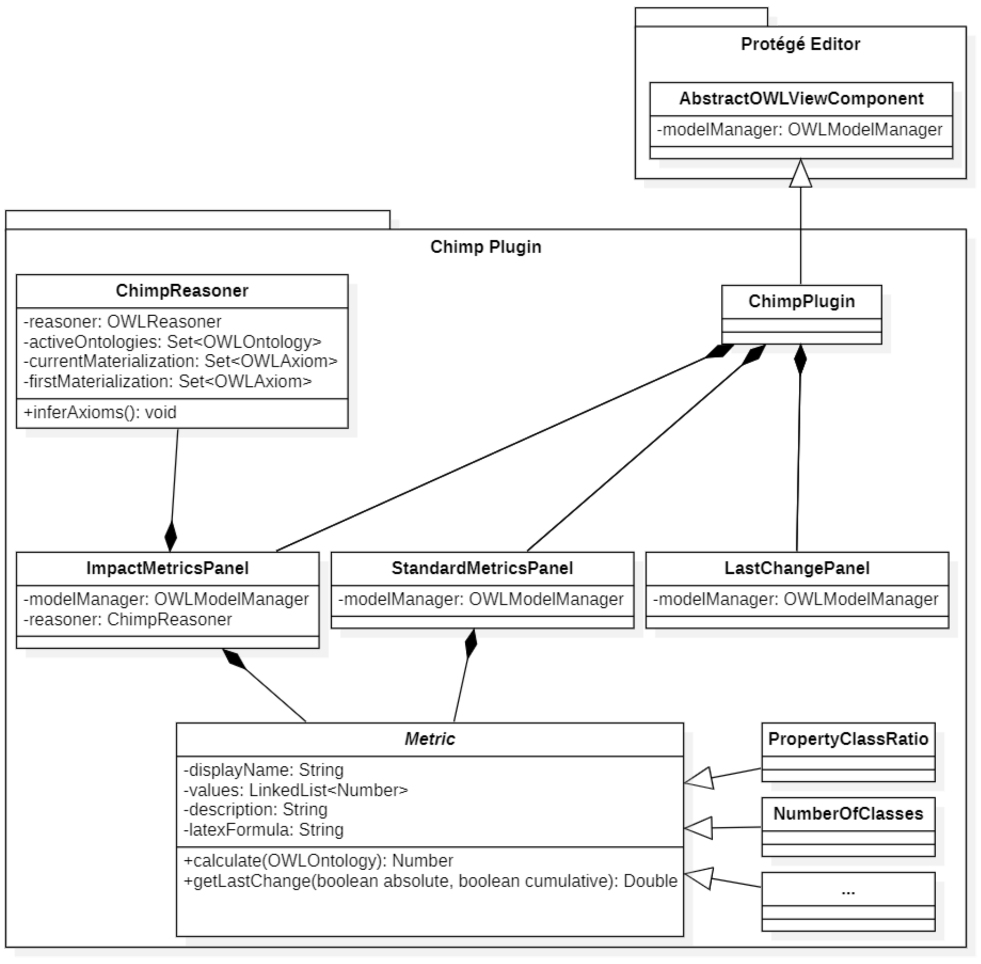

Protégé Editor
AbstractOWLViewComponent
-modelManager: OWLModelManager
Chimp Plugin
ChimpReasoner
-reasoner: OWLReasoner
-activeOntologies: Set<OWLOntology>
-currentMaterialization: Set<OWLAxiom>
-firstMaterialization: Set<OWLAxiom>
+inferAxioms(): void
ChimpPlugin
ImpactMetricsPanel
-modelManager: OWLModelManager
-reasoner: ChimpReasoner
StandardMetricsPanel
-modelManager: OWLModelManager
LastChangePanel
-modelManager: OWLModelManager
Metric
-displayName: String
-values: LinkedList<Number>
-description: String
-latexFormula: String
+calculate(OWLOntology): Number
+getLastChange(boolean absolute, boolean cumulative): Double
PropertyClassRatio
NumberOfClasses
...

Figure 3.1: ChImp Class Diagram

`ChimpReasoner` class follows the same principle: `ImpactMetricsPanel` has access to the `OWLReasonerManager`, through which the `OWLReasoner` is accessed.

Listeners

The updates in ChImp are handled with the observer pattern by listening to the following two OWL listeners: `OWLModelManagerListener`, `OWLOntologyChangeListener`. The difference between these two listeners is that the `OWLModelManagerListener` fires events pertaining to the status of the loaded ontology and the reasoner plugin whereas the `OWLOntologyChangeListener` solely reports the concrete changes in the loaded ontology.

3.4 Plugin Components

The plugin is structured into several components. The section titles used in this section correspond to the package names used in the codebase.

3.4.1 Metrics

As mentioned in Section 3.3.2, all metrics in the ChImp plugin inherit the abstract class `Metric`. This class defines the fundamental attributes and methods that every metric has. In the current implementation, every metric has a name, a description, a LaTeX formula, and a list of values. In addition, the class regulates all retrieval functions for the metrics. The metrics themselves only override the `calculateMetric()` function to define their individual calculation strategy.

The metrics inside the ChImp plugin are split into three groups: primitive metrics, ratio metrics, and impact metrics. Table 3.1 shows the primitive and ratio metrics that are part of the current version of the ChImp plugin. The metrics listed, are the ones that I found to be the most widely used in research. To show where the metrics have been used thus far, the table also contains references to the specific works. Primitive metrics are metrics that present a count of elements in an ontology. These metrics do not in fact perform calculations but count and sort axiom sets in the underlying ontology. Table 3.2 shows how the metrics are calculated with the OWL API. Note that this table as well as the thesis thus far both use ontology in the singular. This is not necessarily the case since an ontology can have an imports closure. In the current implementation of the ChImp plugin, all metrics cover the whole imports closure for their calculations. However, the current implementation only displays changes that are made in a single session. Hence, the plugin does not display changes if they are performed in the imports closure.

Ratio metrics are closely connected to primitive metrics since they depend on them. In fact, they all calculate ratios of primitive metrics. However, the ratio metrics do not conform to a common scale. For example, the property class ratio can theoretically be a decimal between zero and infinity. The bottom part of Table 3.2 shows the formula for their calculation. All of these formulas exclusively reference primitive metrics. Hence, they are completely dependent.

Impact metrics are a new form of metric. They are derived from the materializations of ontologies. Concretely, most of them calculate ratios of materialization set measures. The impact metrics used in the current version of ChImp are reproduced in Table 3.3 since they are not publicly available (see Appendix A.1). An additional impact metric that is used in the plugin is the graph distance impact from Pernischová (2019). That paper defines the measure as:

$$impact = 1 - e^{-\frac{I(M_i)-I(M_{i+1})}{|\delta_i|}}$$

where $I(M)$ is the topological index:

$$I(M) = \sum_{u \in V(M)} \frac{1}{\sqrt{d(u)}}$$

$V(M)$ are all nodes in the materialization and $d(u)$ is the degree of node u. Consult Pernischová (2019) for more information about this. For the materialization, all impact metrics are dependent on a reasoner. A more detailed explanation of the reasoner in ChImp can be found in Section 3.4.2 and Section 3.4.3. In contrast to the other metrics, the impact metrics in the ChImp plugin can only be calculated cumulatively. The reason for this is that incremental calculations would require the metrics to save the materializations for each modification step. Hence, for large ontologies, the plugin would require a large amount of in-memory data storage. This could potentially be avoided by only saving the change sets and recalculating the individual materializations on the fly. The current implementation does not contain such a functionality due to time constraints.

3.4.2 Views

The main class of the ChImp plugin is called `ChimpPlugin`. Every Protégé plugin has an xml file that determines its type, its layout, and the classes that are ultimately loaded into the editor. In the main xml file inside ChImp, `ChimpPlugin` is defined as the main view component. Consequently, `ChimpPlugin` initializes whenever the plugin is opened inside Protégé. The class is relatively small since its only role is to initialize the other views and inject the Protégé workspace into them. In addition, the class checks on startup whether or not an ontology is loaded in Protégé. If this is not the case, the sentence "No ontology loaded" is displayed instead of any of the views. To react to changes inside the Protégé editor, it has an `OWLModelManagerListener`. If a new ontology is loaded, the plugin is reinitialized.

The individual views inside ChImp are all in a `JScrollPane` to make them scrollable. In addition, ChImp uses two `JSplitPane` classes to make the views adjustable.

Last Change Panel

The first view of the ChImp plugin is the last change panel. The purpose of this view is to display the last changes that were made in a Protégé session as defined in **R1** *ChImp should list the applied changes*. Figure 3.2 shows the last change panel in its current form. The actual last change is below the title. Under the subtitle "Previous Changes", the view shows all previous changes.

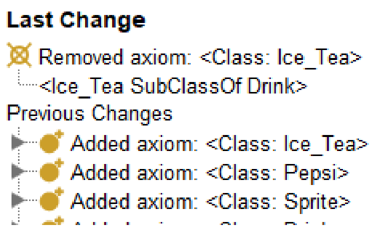

Last Change
Removed axiom: <Class: Ice_Tea>
<Ice_Tea SubClassOf Drink>
Previous Changes
Added axiom: <Class: Ice_Tea>
Added axiom: <Class: Pepsi>
Added axiom: <Class: Sprite>

Figure 3.2: Cutout of the last change panel

Any change in this view appears as a set of axioms. The reason for this is that

Protégé internally stores sets of axioms to represent changes. This makes changes very expressive. However, one problem with it is that it is not trivial to determine which axiom best summarizes a change. The fact that Protégé stores changes as sets of axioms is relevant because sets do not have an order. The current approach simply chooses the first one of these axioms to represent the rest.

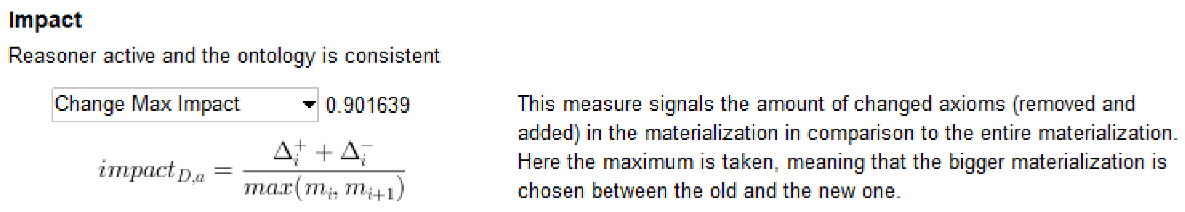
The view is realized with a `JTree`. All changes are nodes in this tree. Their label is the axiom that represents the change. All other axioms are added to the node as sub-nodes. By clicking on the nodes of this tree, users can choose which nodes are collapsed or open at any point. As a default, all of the nodes in the tree are collapsed.

To update the tree with new changes, the last change view uses an implementation of the `OWLOntologyChangeListener`. The view accesses this listener via the `OWLModelManager` that is injected in its constructor. The raw changes are then processed to display strings. Specifically, the first part of the string describes if it was an addition or a removal operation. The second part of the string contains the type of the axiom and its actual name.

In both functionality and appearance, this view is very similar to the Change View by Drummond (2011). However, a manual test showed that the current version of Change View, which is installed by default in Protégé, displays change sets with random class names when an axiom is deleted. The last change view presented in this section filters out these classes.

Impact Metrics Panel

The impact metrics panel displays the impact metrics presented in Section 3.4.1 as well as the reasoner status. With that, it fulfills **R2** *ChImp should inform the user about the consistency of the loaded ontology.* When the ChImp view is added to Protégé and no reasoner is loaded, the `ImpactMetricsPanel` class opens a popup to inform the user about this circumstance. When the plugin is open but no reasoner is running inside of Protégé, the view displays an information string that informs the user how a reasoner can be started. Figure 3.3 shows the view if a reasoner plugin is running inside of Protégé.



Impact

Reasoner active and the ontology is consistent

Change Max Impact 0.901639

$$impact_{D,a} = \frac{\Delta_i^+ + \Delta_i^-}{max(m_i, m_{i+1})}$$

This measure signals the amount of changed axioms (removed and added) in the materialization in comparison to the entire materialization. Here the maximum is taken, meaning that the bigger materialization is chosen between the old and the new one.

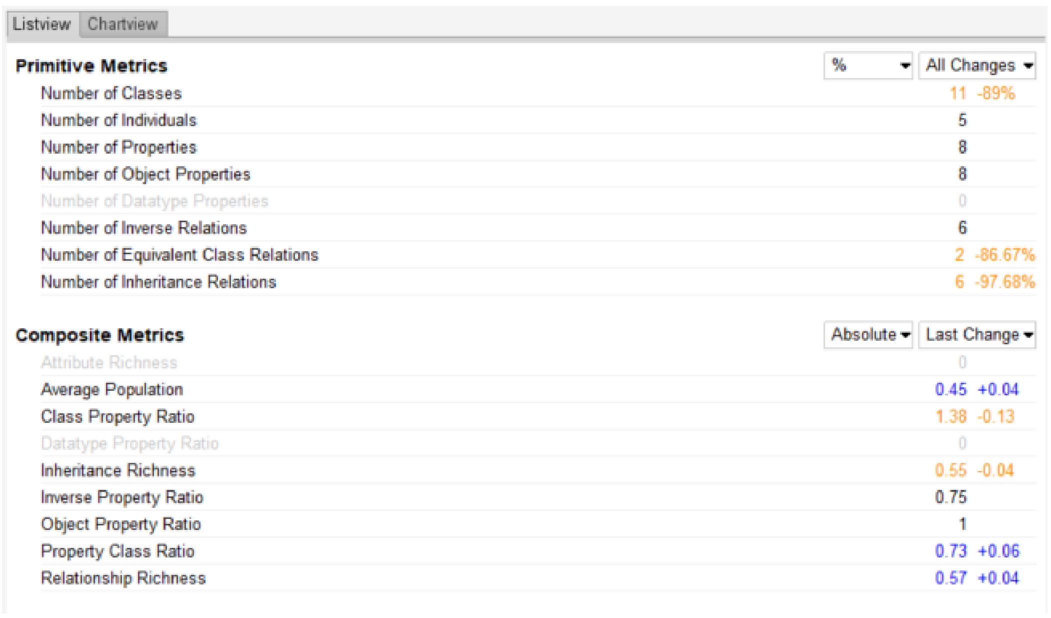
Figure 3.3: Cutout of the impact metrics panel

Right below the title, this view contains the reasoner status. Currently, the strings are reasoner status messages defined by the Protégé editor package. The rest of the panel displays a chosen impact metric. The metric name is printed in a dropdown panel, with which other impact metrics can be chosen. However, only one of them can be shown at any point in time. The metric itself is a number with five decimal places. Its description is a short text and a LaTeX formula printed with JLatexMath (see Section A.3.4).

To initialize the impact metrics, the `ImpactMetricsPanel` creates a `ChimpReasoner` instance (see Section 3.4.3). Compared to the other panels, the change updates in the impact view are more involved. Even though it listens to changes with an implementation of the `OWLModelManagerListener`, these changes are not necessarily realized. The reason for this is that the reasoner synchronization can be managed manually through the Protégé editor. This option to synchronize the reasoner gives users the possibility to regulate when the reasoner recalculates the materialization. For more information consult Section 3.4.3.

Standard Metrics Panel

This panel is dedicated to the display of the primitive and ratio metrics listed in Section 3.4.1 and with that fulfills **R3** *ChImp should show primitive and composite measures in a table, visualizing the new value and its difference to the old value based on the applied changes.* The panel is realized with a `JTabbedPane` to organize both the chart and the metric list as tabs. Figure 3.4 shows the list view that lists the primitive and ratio metrics.



Primitive Metrics	%	All Changes
Number of Classes	11	-89%
Number of Individuals	5	
Number of Properties	8	
Number of Object Properties	8	
Number of Datatype Properties	0	
Number of Inverse Relations	6	
Number of Equivalent Class Relations	2	-86.67%
Number of Inheritance Relations	6	-97.68%

Composite Metrics	Absolute	Last Change
Attribute Richness	0	
Average Population	0.45	+0.04
Class Property Ratio	1.38	-0.13
Datatype Property Ratio	0	
Inheritance Richness	0.55	-0.04
Inverse Property Ratio	0.75	
Object Property Ratio	1	
Property Class Ratio	0.73	+0.06
Relationship Richness	0.57	+0.04

Figure 3.4: Cutout of the list view in the standard metrics panel

The panel splits the metrics into two groups: primitive metrics and ratio metrics. Both groups are listed similar to Protégé's ontology metrics view. However, the difference between the last change is also shown next to each metric. This change display is customizable with the dropdowns at the top of each group. The first dropdown regulates if the change is displayed as a percentage or an absolute number. With the second dropdown, a user can choose to either display the last change or the cumulative change over all changes made in the session. With these functionalities, the view also fulfills **R4** *ChImp should use colors to indicate changes*, **R8** *ChImp should allow the user to chose between the presentation of metrics either in absolute values or as percentages* and **R9** *ChImp should let the user choose between using only the last change or all changes for the calculation of primitive and composite measures.*

The second tab of the view displays a graph of a selected metric. Figure 3.5 shows the panel with this tab in focus.

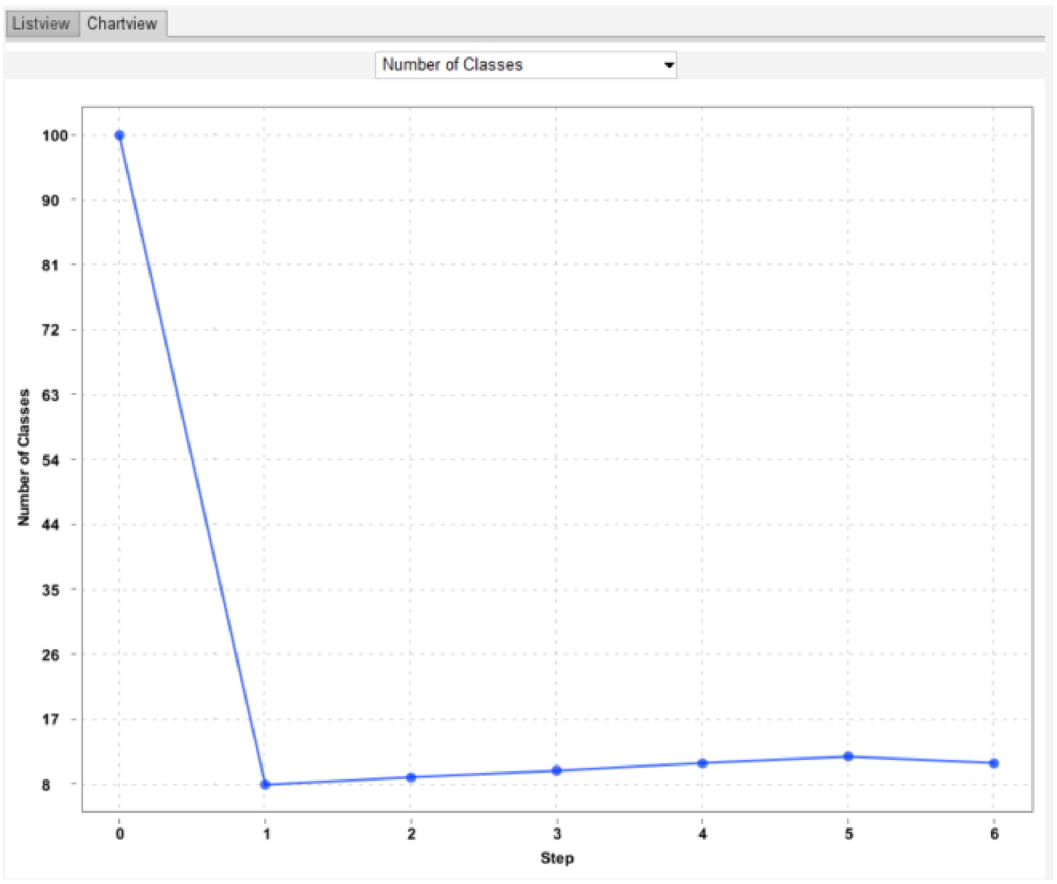


Figure 3.5: Cutout of the chart view in the standard metrics panel

At the top of the panel, there is a dropdown that regulates the metric that is displayed. The graph only ever displays one metric at a time. The current change step in the ontology is shown on the x-axis. The y-axis shows the value of the metric. The axis label range adjusts itself to the type of number that is displayed.

The standard metrics panel updates all changes by listening to changes with an implementation of the `OWLOntologyChangeListener`. Since both its tabs use the same metrics, this update only needs to be done once.

3.4.3 Reasoner

All of the impact metrics listed in Section 3.4.1 depend on a reasoner to calculate the materialization of the underlying ontology. The `ImpactMetricsPanel` described in Section 3.4.2 initializes the `ChimpReasoner` class to initialize the metrics. This class is initialized with a reasoner that implements the `OWLReasoner` interface. All reasoner plugins inside Protégé can be accessed through the `OWLReasonerManager` which is exposed by the `OWLModelManager`. Inside of ChImp, `ChimpReasoner` acts as an interface to interact with any reasoner that is loaded. In addition, since ChImp displays all impact metrics cumulatively, it saves the first materialization. The `OWLReasonerManager` also allows access to the reasoner status while the plugin is running.

The Protégé interface allows users to manually synchronize the current reasoner during a session. With this, a user can choose when to perform large reasoner calculations and potentially delay calculations to minimize the computational load. The synchronization itself forces the reasoner to recalculate the current materialization.

In an early phase, the ChImp plugin used a fixed version of the Hermit reasoner (see Hermit in Section A.3.4) However, I decided to work with the Protégé interface to allow the plugin to be more flexible in the future. For this reason, the ChImp plugin is independent of the reasoner and different users can satisfy their specific needs with their own reasoner choice. To interact with the ChImp plugin, a reasoner only needs to fulfill the requirement of implementing the `OWLReasoner` interface. Hence, this includes incremental reasoners as well.

3.4.4 Testing

The current implementation of the metrics is supported by several tests. The following sections present two different ways of testing. The first builds the tests manually and the second compares the metrics to previously calculated values.

Standard Tests

Ontologies can be constructed programmatically with the OWL API. This is done by creating axioms with the `OWLDataFactory` and adding them to an empty ontology with the `OWLOntologyManager`. Since the axioms are added manually, the result of the primitive metrics and the ratio metrics can be determined.

Unfortunately, not all of the metrics in the current version of ChImp have tests due to time constraints. There are 7 tests for the primitive metrics and 5 for the ratio metrics. The primitive tests, evaluate the metrics with different numbers and types of axioms. In addition, they have checks for empty ontologies. The ratio metrics have checks for which the numerator and/or the denominator of their calculations is zero. They also test extreme cases for the numerator and the denominator of their calculations. As an example, the annotation richness is tested with an ontology that contains a single class and 999 annotations. There are plans to test the impact metrics in this fashion as well. However, the construction of the ontologies is more complicated in their case since they only depend on the materializations.

Ontology Tests

In addition to the standard tests, the codebase includes ontology tests. These tests are performed on ontologies for which previous calculations exist. These calculations were performed with kbci_py, which is a library that can calculate various metrics. The code for kbci_py as well as the precomputed results are available online[1]. Unfortunately, kbci_py does not consider the imports closure for the calculation of what this thesis describes as primitive and ratio metrics. Since the precomputed ontologies contain imports, the results of the ChImp plugin differ by a lot. The same is not true for the impact metrics. In fact, kbci_py offered results for 9 of the 11 impact metrics that are currently part of ChImp. For the impact metrics, the library does include the imports closure. Since the impact metrics depend on the reasoner implementation, a slight deviation is to be expected. Most of the tested metrics deviate by less than 20%. Only the subsumption impact deviates by approximately 40%.

[1]https://gitlab.ifi.uzh.ch/DDIS-Public/chimp-mat

	Description	References
c	number of classes	(Sicilia et al., 2012; Manouselis et al., 2010; Tartir et al., 2010; Tomassen and Strasunskas, 2009)
i	number of individuals	(Sicilia et al., 2012; Manouselis et al., 2010; Tartir et al., 2010; Tempich and Volz, 2003; Tomassen and Strasunskas, 2009)
p	number of properties	(Sicilia et al., 2012; Manouselis et al., 2010; Orme et al., 2007; Tartir et al., 2010; Tempich and Volz, 2003; Tomassen and Strasunskas, 2009)
p_O	number of object properties	(Lantow and Sandkuhl, 2015; Shen et al., 2018; Tempich and Volz, 2003)
p_D	number of datatype properties	(Tempich and Volz, 2003)
h	number of subclasses	(Shen et al., 2018)
a	number of annotations	(Shen et al., 2018)
eq	number of equiv. classes	(Shen et al., 2018; Tempich and Volz, 2003)
inv	number of inverse relations	(Osumi-Sutherland et al., 2018)
i/c	average population	(Sicilia et al., 2012; Manouselis et al., 2010; Duque-Ramos et al., 2013; Gangemi et al., 2006; Tartir et al., 2010)
h/c	inheritance richness	(Sicilia et al., 2012; Manouselis et al., 2010; Djedidi and Aufaure, 2010; Lantow and Sandkuhl, 2015; Tartir et al., 2010)
a/c	annotation richness	(Duque-Ramos et al., 2013; Tartir et al., 2010)
p_D/c	attribute richness	(Tartir et al., 2010; Lantow and Sandkuhl, 2015; Djedidi and Aufaure, 2010)
$p/p+h$	relationship richness	(Sicilia et al., 2012; Manouselis et al., 2010; Duque-Ramos et al., 2014; Tartir et al., 2010)
p/c	property class ratio	(Duque-Ramos et al., 2013; Tempich and Volz, 2003; Gangemi et al., 2006; Tartir et al., 2010)
inv/p	inverse property ratio	(Djedidi and Aufaure, 2010; Gangemi et al., 2006; Tartir et al., 2010)
c/p	class property ratio	(Lantow and Sandkuhl, 2015; Gangemi et al., 2006; Alm et al., 2013)
p_D/p	datatype property ratio	(Tempich and Volz, 2003)
p_O/p	object property ratio	(Tempich and Volz, 2003)

Table 3.1: Description and references of the primitive and ratio metrics in ChImp. The top section contains the primitive metrics. The bottom section contains the ratio metrics.

	Description	Implementation
c	number of classes	o.getClassesInSignature().size()
i	number of individuals	o.getIndividualsInSignature().size()
p	number of properties	o.getObjectPropertiesInSignature().size() + o.getDataPropertiesInSignature().size()
h	number of subclasses	o.getAxioms(AxiomType.SUBCLASS_OF).size()
a	number of annotations	o.getAnnotations().size()
eq	number of equiv. classes	o.getAxioms(AxiomType. EQUIVALENT_CLASSES).size()
inv	number of inverse relations	o.getAxioms(AxiomType. INVERSE_FUNCTIONAL_OBJECT_PROPERTY).size() + o.getAxioms(AxiomType. INVERSE_OBJECT_PROPERTIES).size()

Table 3.2: Description and implementation of the primitive metrics in ChImp. `o` is the instance of the ontology within each metric implementation. The implementation is performed on all ontologies in the imports closure.

Formula		Description
$impact_{\Delta^+,m_i}$	$= \frac{\Delta_i^+}{m_i}$	added inference old ratio
$impact_{\Delta^-,m_i}$	$= \frac{\Delta_i^-}{m_i}$	removed inference old ratio
$impact_{\Delta^+,m_{i+1}}$	$= \frac{\Delta_i^+}{m_{i+1}}$	added inference new ratio
$impact_{\Delta^-,m_{i+1}}$	$= \frac{\Delta_i^-}{m_{i+1}}$	removed inference new ratio
$impact_{\Delta^+,\cap}$	$= \frac{\Delta_i^+}{m_{i,i+1}}$	added inference impact
$impact_{\Delta^-,\cap}$	$= \frac{\Delta_i^-}{m_{i,i+1}}$	removed inference impact
$impact_{\Delta,a}$	$= \frac{\Delta_i^+ + \Delta_i^-}{max(m_i, m_{i+1})}$	change max impact
$impact_{\Delta,\cap}$	$= \frac{\Delta_i^+ + \Delta_i^-}{m_{i,i+1}}$	change impact
$impact_{\Delta_\sqsubseteq,\cap}$	$= \frac{h_{\Delta_i}}{m_{i,i+1}}$	subsumption change impact
$impact_{\Delta_\sqsubseteq,\cap_\sqsubseteq}$	$= \frac{h_{\Delta_i}}{m_{i+1} - m_i}$	subsumption impact
m_i	$= \lvert M_i \rvert = \lvert G_i \backslash O_i \rvert$	Materialized part of the graph at time i
$m_{i,i+1}$	$= \lvert M_i \cap M_{i+1} \rvert$	number of axioms shared between the materialized versions
Δ_i^+	$= \lvert M_{i+1} \backslash M_i \rvert$	New axioms part of M_{i+1} but not in M_i
Δ_i^-	$= \lvert M_i \backslash M_{i+1} \rvert$	Old axioms part of M_i but not in M_{i+1}
h_{Δ_i}	$= \lvert SubClassOf(\cdot,\cdot)_{m,i} \rvert$	Number of $SubClassOf$ axioms in Δ_i^+ and Δ_i^-

Table 3.3: Description of the impact metrics in ChImp. They are reproduced from an internal document listed in Appendix A.1.

4 Technical Evaluation

Chapter 1 lists **RQ1**, **RQ2**, and **RQ3** about the implementation of the ChImp plugin. The following sections address these questions by looking at the implementation of the plugin.

4.1 Performance

Before answering the research questions it is important to first identify their target. **RQ1** *How long does it take for the plugin to visualize a change?* is a good segue into the topic of task separation in the Protégé environment. In the environment of the ChImp plugin, it is important to distinguish between plugin tasks and Protégé tasks. Analyzing the performance of Protégé tasks is not part of this thesis because it would be hard to isolate the scope of such an assessment. Hence, the research question above is specifically directed at the visualization performance of the plugin.

Before the plugin can show anything it has to perform its own internal tasks. Primitive metrics use the OWL API to access the ontology that is loaded in the Protégé workspace (see Section 3.4.1). Testing the performance of the OWL API is also not part of this thesis. Nevertheless, the specific calls give insight into the computational complexity behind the calculations. Therefore, for this evaluation, it is important to determine the computational complexity of the function calls listed in Table 3.2. Looking at the individual calls in detail, it becomes clear that their computational complexity is in fact linear. As an example, the call to the number of classes in the signature of an ontology is returning the length of a precomputed set. The Protégé editor, stores all classes of an ontology in such a set. The caller now only needs to calculate the length of that set to determine the number of classes. The same logic applies to most primitive metrics because the most complex operation they perform is a set count. Hence, their computational complexity is linear. Exceptions are the metrics `NumberOfEquivalentClassRelations`, `NumberOfInheritanceRelations` and `NumberOfInverseRelations`. For these metrics, the sets have to be filtered to get a set that only contains the appropriate relation. Even though this is different from the other primitive metrics, this filtering effort is linear as well and therefore its effect is negligible.

The ratio metrics are completely dependent on primitive metrics. In fact, all of their calculations are divisions of primitive metrics, which means that they also solely depend on set counts. Hence, their computational complexity is linear too. The actual division operation is negligible.

In contrast to the primitive metrics, the computational complexity of the impact metrics depends on the OWL API as well as the reasoner. The calculations the reasoner performs are very involved and go beyond the scope of this thesis. In addition, there is already research that analyzes reasoner performance (e.g. Dentler et al., 2011; Abburu, 2012; Oesch, 2018). Therefore, they are not considered in the assessment of the computational complexity of the impact metrics. Once the impact metrics have obtained the materializations from the reasoner, they also perform rather simple calculations, since they only depend on the OWL API. In fact, their computational complexity is linear as well. The reason for this is that most of the metrics calculate a ratio of the sizes of the current and last materialization, which are also determined by set counts. A special case is the calculation of the `GraphDistanceImpact` presented in Section 3.4.1. It requires the computation of the topological indices of both the initial materialization and the current materialization. This calculation is dependent on how the chosen reasoner stores the underlying ontology. If a reasoner saves the underlying ontology as a graph of nodes, the calculation of the degree of the individual nodes can be determined in linear time. Once the topological indices are determined, the rest of the calculation runs in linear time as well.

In conclusion, **RQ1** mostly depends on Protégé and the chosen reasoner and not the ChImp plugin. The calculations inside the plugin are trivial in comparison to the complex tasks that Protégé performs for each change operation. The explanation above also provides an answer to **RQ3** *How does the plugin perform with ontologies of different sizes?*. Since the computational complexity of the internal calculations of ChImp is linear, the overall performance mainly depends on Protégé. Based on this linear computational complexity, there is no reason to assume that the calculations do not scale to larger ontologies.

For the same reason, it is very hard to improve the performance of the plugin. Whenever it is possible, the metrics already save intermediate results to avoid unnecessary calculations. This is, for example, the case for the impact metrics. Since all of them compute their result cumulatively, they reuse the initial materialization in subsequent calculations. In the ChImp plugin, this is handled by the `ChimpReasoner` class. It calculates the first materialization so that the impact metrics can access it without an additional computational effort.

4.2 Stability

RQ2 *Does the plugin block while calculating metrics?* inquires about possible blockages during the calculation of the metrics. Considering the remarks in Section 4.1, I think it is unlikely that the internal calculations run into problems. I think it is far more likely that the reasoner or Protégé itself lock up during a change operation. With that said, it is possible that ChImp contains bugs or other code-related issues that result in an error. The testing mentioned in Section 3.4.4 does not extend to the visual components. Therefore, errors are more likely to occur there. In addition, even though several metrics have been tested, this is not a guarantee that bugs cannot occur.

While testing manually, I observed that on a normal computer Protégé can easily be pushed to its limits by performing a large delete operation. If a reasoner is active during this operation, the delay is even greater and may even result in a crash. In my experience, Protégé also crashes whenever an ontology is inconsistent and a reasoner is active. This scenario can be created by making any class the subclass of owl:Nothing. These crashes are not related to the ChImp plugin since they also occur without it.

5

User Evaluation

This chapter presents an experiment conducted with the ChImp plugin to provide insight into **RQ4** and **RQ5** from Chapter 1. These two research questions ask how developers rate the appeal of the plugin and if they find its content useful and informative. The chapter is structured as follows: First, Section 5.1 states the hypotheses of the experiment. Subsequently, Section 5.2 presents the design of the experiment. After that, Section 5.3 presents the results and Section 5.4 discusses them.

5.1 Hypotheses

The following paragraphs recapitulate the research questions that the experiment aims to answer and, if applicable, derive corresponding hypotheses.

RQ4 *Do developers like the plugin and its visualizations?*

This research question aims to assess the perceived appeal developers have when they have performed a task with the plugin. There are no hypotheses for this research question since it will be answered with descriptive statistics.

RQ5 *Is the content useful and informative to developers?*

This research question aims to ascertain if the ChImp plugin in fact fulfills an actual use case. It cannot be proven that there is not any use case for which the plugin is useful. However, given a specific use case, users can be asked if they perceive the plugin to be useful in that context. To obtain a sensible result, such a comparison needs to include participants that use the plugin and participants that do not. Therefore, a concrete experiment has two groups that both perform the same task. One group has the plugin during this task and the other one does not. For this comparison to work, it is vital that the participants do not know of the existence of the plugin beforehand, since this would otherwise affect their expectations.

> $H1_0$: *It is unclear if developers that perform a task with the ChImp plugin perceive to be better informed about the impact of their changes than their*

> *contemporaries that completed the same task without the plugin.*
> $H1$: *Developers that perform a task with the ChImp plugin perceive to be better informed about the impact of their changes than their contemporaries that completed the same task without the plugin.*

In addition, the ChImp experiment also investigates if the participants' perception of how well informed they are depends on whether or not they have previously performed a task with the plugin. The reason for this is that the results cannot be considered valid if the participants simply learn to appreciate the plugin. To do this, the group that used the plugin in the first task performs a second task without it, and vice versa for the other group. The baseline comparison looks at the tasks that the groups performed without the plugin. The goal of this is to determine if participants' perceptions are influenced by the previous task.

> $H2_0$: *Developers that have used the ChImp plugin and afterwards perform a task without it have a different perception of how well informed they are without it than developers that perform a task without having seen the plugin beforehand.*
> $H2$: *Developers that have used the ChImp plugin and afterwards perform a task without it have the same perception of how well informed they are without it than developers that perform a task without having seen the plugin beforehand.*

5.2 Experiment Design

The experiment was conducted remotely on the computers of the participants. A few days before the experiment, all participants were asked to install Protégé on their computers. This was done earlier to ensure that there would be enough time for troubleshooting. 15 minutes before the experiment, participants received an email with the survey and two pizza ontologies[1], one for each task. In addition, during the experiment participants were allowed to ask questions via email or chat.

5.2.1 Survey Structure

In order to give insight into the stated research questions and hypotheses, the experiment confronts the participants with tasks and inquires about their experience. Since the experiment was conducted remotely, the survey is set up to guide participants throughout its duration. For this to work, the survey explains every step of the process in great detail. The whole survey can be found in Appendix A.4.

The participants of the ChImp experiment are split into two groups. Both groups have two distinct surveys that differ in when they introduce the ChImp plugin. Figure 5.1 shows how the surveys are structured for the two groups. The start of both surveys

[1]https://protege.stanford.edu/ontologies/pizza/pizza.owl

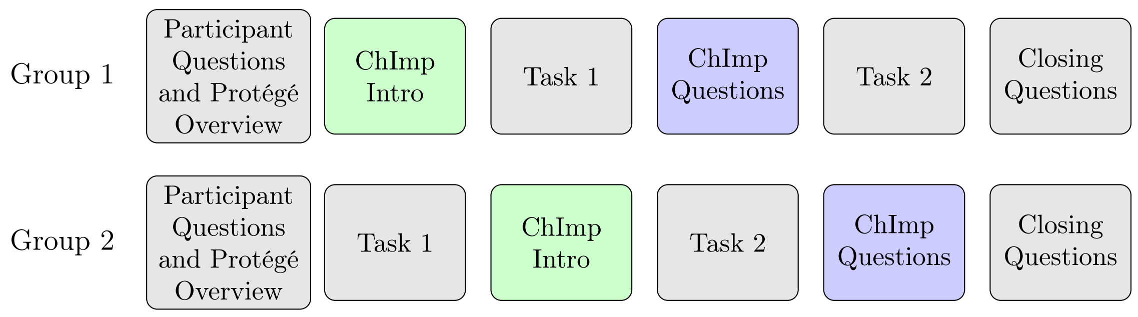


Figure 5.1: Steps in the experiment from left to right. The first group performed the first task with ChImp and the second task without ChImp. The second group performed the first task without ChImp and the second task with ChImp.

contains questions about the participants' experience with ontologies and Protégé and presents an overview of Protégé and its basic functionalities. After this first step, the surveys present the tasks. Depending on the group, they introduce the ChImp plugin at a different time and ask questions about it afterwards. It is important to note, that the first group is asked to close the ChImp plugin after answering the ChImp questions and not use it in the second task. In the end, both surveys contain the same closing questions. These closing questions consist of general questions about ChImp and the experiment and text fields for possible feedback.

To provide insight into **RQ4** *Do developers like the plugin and its visualizations?*, participants rate the design of the plugin and its views after they have used it in one of the tasks. For the first group, this is after the first task and for the second group, this is after the second task. Specifically, they are asked to rate the visualization of each view (see Section 3.4.2) and the plugin as a whole on a scale from 1 to 5 with the following instructions: "with 1 being you don't like it at all and 5 being you like it very much".

As described in Section 5.1, the approach to investigate **RQ5** *Is the content useful and informative to developers?* is more involved since it is two-pronged. The first aspect is the direct comparison between two groups that perform the same task but only one of them has the plugin. In the experiment, the first task is identical and therefore ideal for this comparison.

The second aspect revolves around whether or not participants learn and adapt during the experiment. The goal of this aspect is to determine if participants are able to correctly assess how well informed they are or if they change this assessment if they have a situational comparison. If the participants are actually unbiased by the fact that they have used the plugin before, the results from the second task can also be used to answer the first aspect. Each group performs a task with and without the ChImp plugin to answer this question. The first group starts the first task with the ChImp plugin and the second group starts the same task without it. In the second task, the same situation is in reverse. For each of the tasks, the participants respond to the same questions.

5.2.2 Experiment Tasks

The tasks in both surveys are the same. The only thing that is different is that the ChImp plugin is introduced at a different point. Before each task, participants are asked to open one of the ontologies that were sent before the experiment. Each task has two parts, that do not depend on each other. In each part, participants are asked to alter the ontology so that the ChImp plugin can display changes.

After each part, the survey reminds the participants that the experiment at hand revolves around the impact their changes have on the ontology as a whole. Furthermore, it asks participants to take a look at all the tabs and views that are currently open in Protégé. Following that, the participants need to fill out a scale with the following question: "Rate how well informed you are about the impact the previous changes had on the ontology as a whole (with 1 being not informed at all and 5 being very well informed)". In addition, they are asked to write down the view or element on the screen that was most crucial for answering the previous question. To see the tasks consult Appendix A.4.

Task 1

In the first part, the survey asks the participants to open the class hierarchy view in the entities tab of Protégé and add a class "Drink" to the ontology. In addition, this class has the subclasses "Coke", "Sprite" and "Ice Tea". The individual steps to add this structure with Protégé are explicitly stated in a list. The resulting hierarchy is shown as additional help.

In the second part, the survey again asks the participants to open the class hierarchy view. However, in this part, the instruction for the participants is to delete the "Pizza" class that is already in the ontology. The surveys show the participants which button needs to be clicked for this deletion operation and which deletion option needs to be selected.

Task 2

The first part of the second task mirrors the first part of task 1. However, instead of adding "Drink", participants have to add the class "Burger" and the subclasses "Cheeseburger", "Hamburger" and "Veggieburger".

The second part instructs participants to open the object property hierarchy view in the entities tab of Protégé. The survey then instructs the participants to delete the property "hasIngredient". The instructions are the same as in the second part of the first task but adopted for object properties.

5.2.3 Participant Demographics

In the first phase, I contacted university students that partook in the class *Semantic Web Engineering* in the fall of 2019. With them having taken this class, it is fair to assume that they are aware of the concept of an ontology. To expand the participant pool, I extended the invitation to all members of the DDIS. In addition, I invited students that took the seminar *Current Trends of Dynamic and Distributed Information Systems* in the spring of 2020. Altogether, 30 people were invited to the experiment. 15 of these replied and 13 eventually completed the experiment. The first group in the experiment had seven participants and the second group had six participants.

Seven participants reported that they have had at least one course at university that revolved around ontologies. However, only two participants replied that Protégé was used in a university course they attended. Three of the participants stated that they have used Protégé outside of university courses. One of them stated to have 4 months of experience with the tool. Another stated to have 6 months. In addition, two said that they have used Protégé in a professional setting. However, only the participant that stated to have 4 months of experience with Protégé in the previous question, also expressed to have used Protégé in a professional setting. The other participant that used Protégé in a professional setting reportedly does not have any experience with Protégé. Since this is a conflicting statement, it is not clear how much experience this participant really has with Protégé. In summary, at least 10 of the 13 participants do not have any experience with Protégé.

5.3 Results

The execution of the experiment spanned three weeks. On average, the participants took 42 minutes to complete the experiment. Five participants asked questions during the experiment. Most of the asked questions were about issues with Protégé or the survey itself, not the content of the tasks. Three of the participants had a technical issue during the experiment. However, all of these issues occurred at the beginning of the experiment and the participants were able to switch to a different computer or I was able to resolve the issue. All of the answers in the experiment are either on a scale from one to five or in text.

Figure 5.2 shows the results of the questions that were asked at the end of the experiment. The specific questions are in the figure description. In general, the participants liked the plugin and the experiment as a whole. However, the difficulty level was too low for most participants. In addition, on average the participants did not feel strongly about the validity of their assessment.

The textual feedback at the end of the study was mostly positive as well. Several participants stated that they like the idea of presenting ontology change and that they did not find other views in Protégé that did this. However, one person replied that the last change view is too small. In addition, two people remarked that the impact metrics were not explained well enough. One of them suggested adding further information in the description such as concrete examples or a better explanation about the intuition behind the formulas. One participant also replied that the chart view is currently limited to the progress of change and therefore not able to display a summary of change. That participant also suggested the use of a spider plot. Such a plot would show several metrics in one figure, and with that summarize the change.

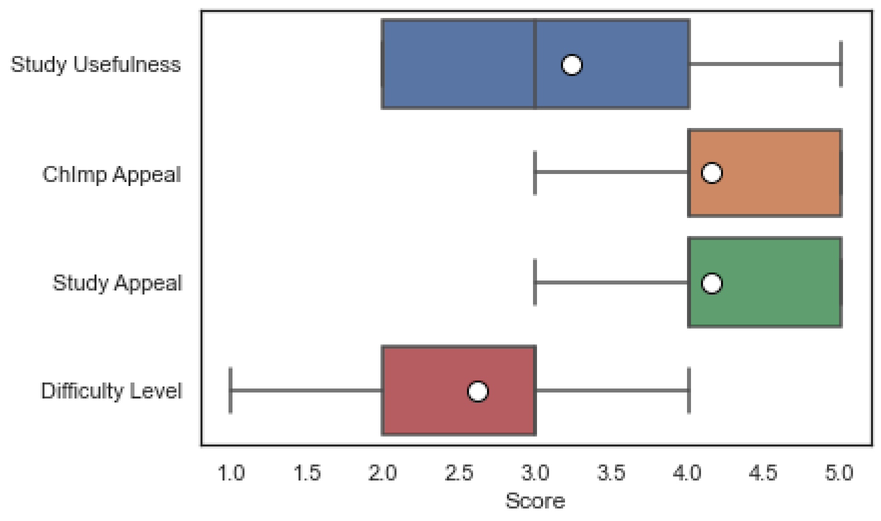


Figure 5.2: General feedback about the study and ChImp. The white circles represent mean values. 1 is the lowest score and 5 is the highest. Study Usefulness: How confident do you feel that your assessment of the usefulness of the plugin would hold up in a real-life use case? ChImp Appeal: How did you like the ChImp plugin? Study Appeal: How did you like the study overall? Difficulty Level: The difficulty level for me was?

Participants felt that the instructions were, for the most part, clear and easy to follow. One participant remarked that the changes in the experiment were really simple and inquired if ChImp was also able to display more complex changes. That participant also suggested testing ChImp in a code-review like situation, where one developer interprets the ChImp view of another developer.

5.3.1 Plugin Appeal

After participants used the plugin in a task, they answered more detailed questions about the plugin's visualization. Figure 5.3 shows the results of the visualization rating. The results suggest that people generally like the visualization of the list view and the chart view. The scores for the visualization of the last change view and the impact view are harder to interpret. Even though they show a slight positive trend, the whiskers show that the underlying data has a larger variance. The visualization of the plugin overall was also well received.

5.3.2 Plugin Content

As mentioned in Section 5.2.2, participants answered a question about how well informed they felt after each part of each task. Figure 5.4 shows the results of each task and both tasks combined. When looking at the chart for both tasks (Figure 5.4c), it seems clear

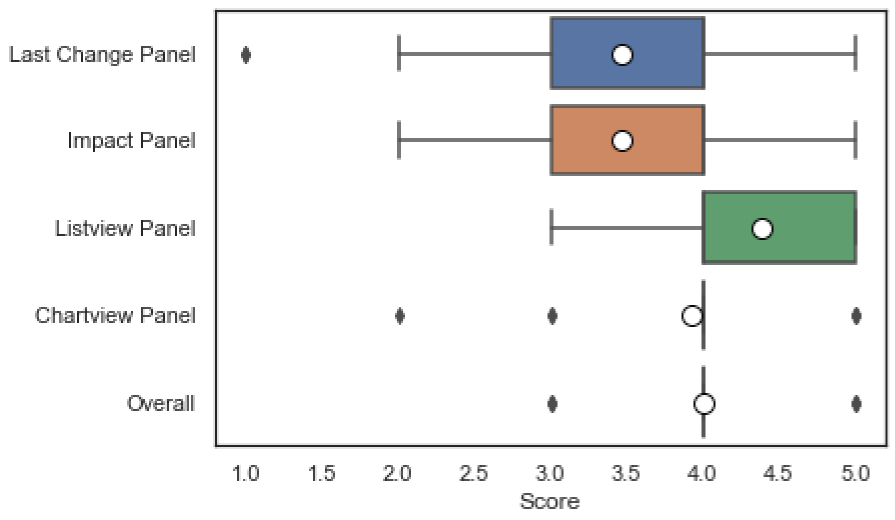


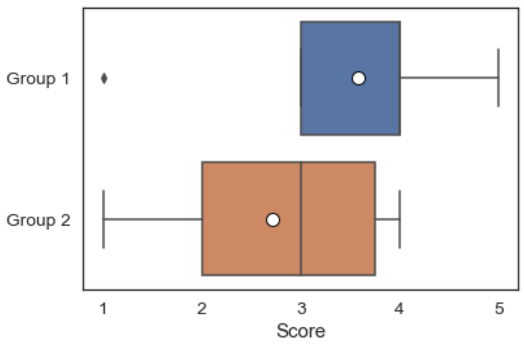
Figure 5.3: Visualization rating for the plugin and its views. The white circles represent mean values. 1 is the lowest score and 5 is the highest.

that participants with the ChImp plugin perceived to be better informed than those without it. A t-test confirms that the means of the two data sets are indeed significantly apart ($p = 0.00002$). The same appears to be true for the individual tasks, even though the difference in Figure 5.4a looks smaller. A t-test of the data sets of the first task results in a p-value of 0.058. Hence, the mean difference in the first task is not significant if a p-value of 0.05 is the threshold. A t-test of the data of the second task is more clear with a p-value of 0.000002.

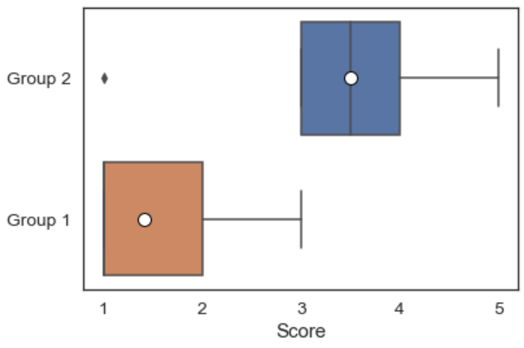
By looking at Figure 5.4a and Figure 5.4b it becomes apparent that the two groups did not behave consistently for both tasks. The scores for group 1 in task 2 look like they are lower than those of group 2 in task 1 even though both groups performed these tasks without the plugin. Figure 5.5b shows a direct comparison of their results. A t-test signals that the rating mean difference is significant ($p = 0.0016$). Hence, the participants that had seen the plugin beforehand, had a different perception of how well informed they were than the participants that had not yet seen the plugin. The same cannot be said of the tasks performed with the ChImp plugin. The scores in Figure 5.5a look very similar and their difference is not significant ($p = 0.84$).

After each part of each task, participants also needed to write down the element that was most crucial for the score they gave for how well informed they were. For the tasks that were completed with the plugin, most of the answers included either the plugin or a component of it. However, the responses mentioned most views of the ChImp plugin with the same frequency. There was not a single view that rose above the others. In addition, three participants stated other views and one replied with 'none'. Like in the general study feedback, some responses remarked that the impact metrics were not explained well enough.

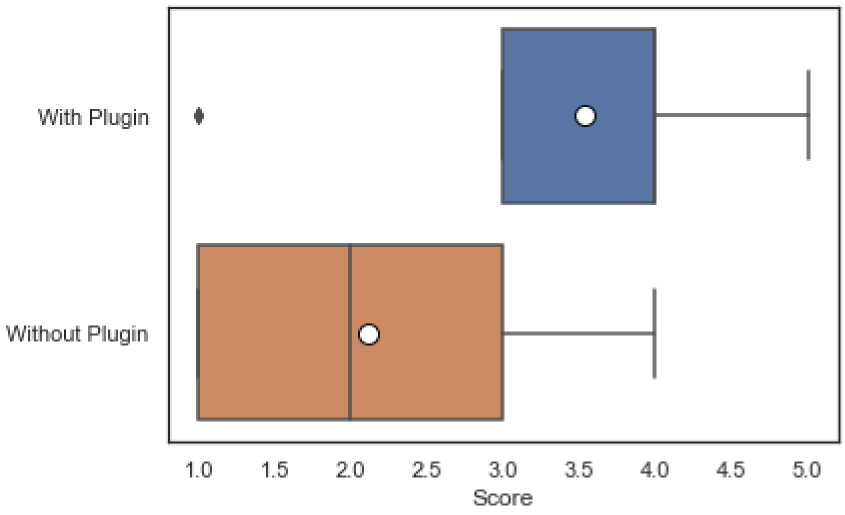
For the tasks that were completed without the plugin, most of the participants stated



(a) Task 1

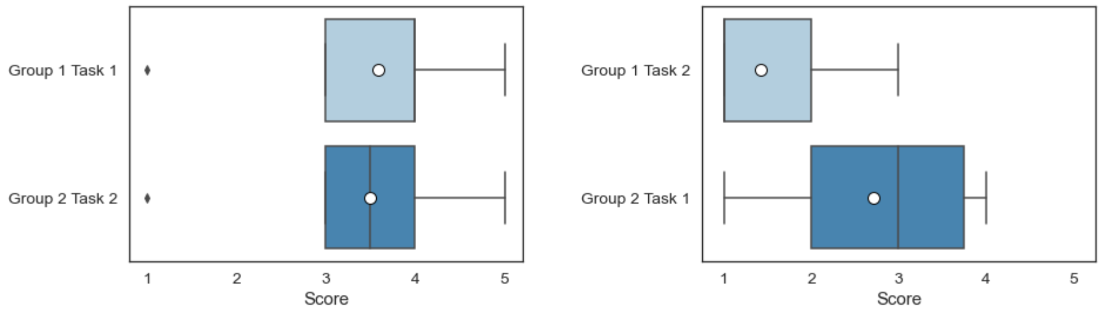


(b) Task 2



(c) Both Tasks

Figure 5.4: Scores of how well informed participants felt during the tasks. The top whiskers in all three plots are the tasks that were completed with the ChImp plugin. The bottom whiskers correspond to those that were done without the plugin. The white circles represent mean values. 1 is the lowest score and 5 is the highest.



(a) Tasks with the ChImp plugin (b) Tasks without the ChImp plugin

Figure 5.5: Comparison of similar tasks. (a) compares the tasks that were completed with the ChImp plugin. (b) compares the tasks that were completed without the ChImp plugin. The white circles represent mean values. 1 is the lowest score and 5 is the highest.

one of the hierarchy views inside Protégé. Some of them mentioned that they were able to see the difference inside of the hierarchy tree. Other answers included Protégé's ontology metrics view and again 'none'. This 'none' was given by a different participant than the one given as a response for the task with the plugin. One participant remarked that the experiment never gave a definition of what impact is and inquired if the goal of the study is to determine the participants' definition of impact.

At the end of the experiment, all participants were asked to rate how informative each panel and the plugin as a whole were for them. Figure 5.6 shows an overview of these ratings. The scores indicate that the participants thought that the list view was very informative. The rest of the views received scores that are harder to interpret. Nevertheless, they indicate that the last change view and the chart view were perceived to be moderately informative and that the impact view was perceived to be less informative.

5.4 Discussion

In general, the results of the study have to be taken with a grain of salt due to the small sample size of 13. In addition, even though participants liked the plugin and the study, they did not give a clear indication of whether or not they thought that their responses were useful for a real-life use case (see Figure 5.2).

With these caveats in mind, the results seem to indicate that participants generally liked the visualization of the plugin, albeit not equally for every view. Well-liked were especially the list view and the chart view. Given these indications, **RQ4** *Do developers like the plugin and its visualizations?* can be answered as follows: The participants in the study generally liked the ChImp plugin and its visualizations.

The results from Figure 5.4c suggest that there is also a very clear-cut answer for **RQ5** *Is the content useful and informative to developers?*. However, as demonstrated in Figure 5.5b, the second task should not be considered for the comparison. The t-test for

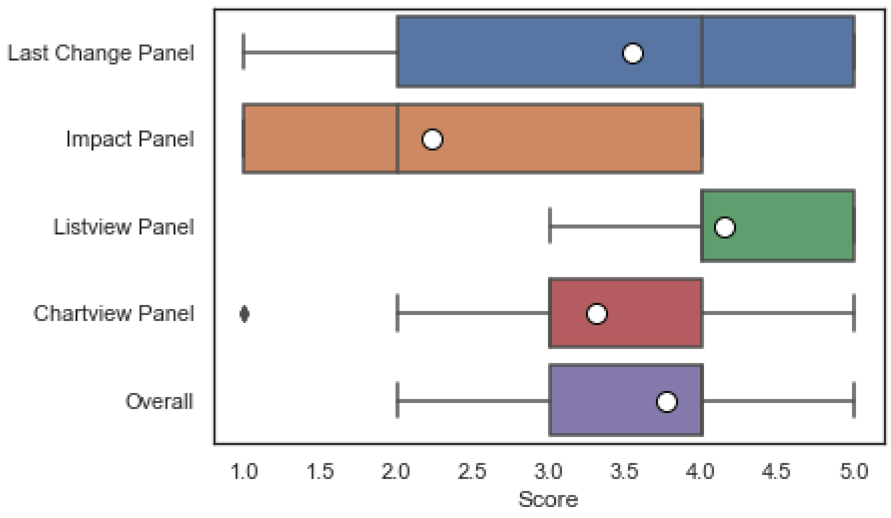


Figure 5.6: Rating of how informative the plugin and its views are. The white circles represent mean values. 1 is the lowest score and 5 is the highest.

this data shows that the fact that the participants had already seen the plugin affected their perceptions in the second task. As a consequence, $H2_0$ cannot be rejected. Despite this, the results from Figure 5.4a can still provide an answer since they are unaffected by this bias. In the first task, neither group had another task that they could compare it to. Nevertheless, these data are also do not allow for a meaningful conclusion since the means are not significantly distinct with a p-value of 0.058. Hence, $H1_0$ cannot be rejected either. It is noteworthy, that the threshold of a p-value of 0.05 is arbitrary. A higher significance level would indicate that the difference is in fact significant. However, considering the low number of participants and their lack of experience with Protégé, a conclusion based on the current data is not meaningful. The data do not give a significant indication of whether or not the participants perceive to be better informed with the plugin. The only indicator that is left, is the participants' own estimation of how informative the ChImp plugin and its views were after they completed a task with it. However, the results suggest that only the list view is clearly thought of as informative. In conclusion, the answer to research question **RQ5** is: The results do not provide a clear indication that the content of the plugin was useful and informative to developers.

6 Limitations and Future Work

The thesis at hand has several limitations that deserve mention. The following paragraphs cover the most important ones. In addition, the paragraphs outline possible future work.

6.1 Requirements

The implementation of the ChImp plugin heavily relies on the work by Pernischová et al. (2020). However, it is the only study that I could find that presents requirements for ontology change impact visualization. The fact that there is not more research could indicate that this particular niche is still in its infancy and more research is needed to establish a consensus. In addition, research in the field of ontologies shows that it is not likely that there is a visualization that fits all use cases (see Section 2.3.1). This aspect can be analyzed in more detail by eliciting more specific requirements and adapting ChImp to them.

6.2 Implementation

There are several concrete elements of the current implementation that need improvements. As mentioned in Section 3.4.2, the last change view does not summarize the change sets and simply chooses the first change in a set to represent it. However, summarizing a change set is not trivial. Hence, more research in this area is needed. On top of that, there is no limit to how long the strings inside the view can be. Therefore, the plugin requires an approach to summarize single changes to short strings. These two problems are underscored by the fact that several participants of the user experiment stated that the last change view is too small. Better change summaries could improve this perception. The feedback from the user experiment also shows that the other visual elements that are part of ChImp have space for improvements. The participants especially did not understand the impact metrics. Hence, more work is needed to figure out how their meaning can be better conveyed. All remarks like this should be considered in a future version. In addition to these aspects, requirements 5 to 7 from Pernischová et al.

(2020) are currently not covered in ChImp. A future release should include an export functionality that covers these three requirements.

Discussions during the implementation of Chimp also brought about several other ideas for features that are not part of this thesis due to time constraints. This is evident by the fact that many of the initial designs and concepts presented in Section 3.1 are not used in the current implementation. I believe, that a node-link diagram that uses polymetric views to display the last change and its dependencies, shows a lot of promise. One participant of the user evaluation also suggested the use of a kiviat diagram. Despite the fact that Pernischová et al. (2020) show that there is no indication that such a diagram is particularly useful, I believe that it also offers promising possibilities. Another idea that is not in the current implementation is to make ChImp more customizable. Concretely, this could mean that the user can choose the metrics that are displayed in ChImp's views or display only certain information about them. Alternatively, this could mean that whole views inside of ChImp are customizable or even interchangeable. In addition, the selection of metrics is currently very limited. A newer iteration should include more metrics that address aspects such as quality, semantics, or concept drift. Another proposed feature is the use of machine learning. With enough data, a regression model could learn to predict change impact or warn a user at a crucial editing step. Pernischová (2019) has already laid the foundation for such an approach.

6.3 Technical Evaluation

The goal of the technical evaluation was to answer questions about the performance and stability of the plugin. However, it became clear that the plugin itself does not contain calculations with high computational complexity. Hence, the research questions were largely answered by referring to Protégé's own limitations. Proper answers to the research questions would include Protégé in their considerations. Such a comprehensive approach could determine exactly when and in which scenario a blockage or time delay could occur.

In addition to such an approach, a future evaluation also needs to test the plugin's interaction with different reasoners. As of now, it is unclear how the plugin reacts to different reasoners or even different types of reasoners, such as incremental reasoners. This could be done by adding more tests to the implementation. Due to time constraints, the current testing suite is also very limited in its scope. More tests for the metrics as well as the components would lead to a more stable application in general.

6.4 User Evaluation

The experiment suffers from several shortcomings. First off, the sample selection was not appropriate for the task. Most of the participants did not have any prior experience with Protégé. On top of that, several of them only had one course about ontologies and were therefore beginners in the field. It is to be expected that more experienced developers would have been able to better evaluate how useful the ChImp plugin is in providing information about change impact. Next to the experience deficit, it also cannot be ruled out that the sample suffered from a self-selection bias. In a worst-case scenario, only people that think this thesis or theses in general are worthwhile participate in the study. Participants like that will attempt to maximize positive feedback. This effect is worse if there is an affiliation between the conductor and the participants of the experiment, which was the case for several of the participants.

In addition to the sample selection, the sample size was also lacking. With only 13 participants it is hard to elicit a result that is generalizable. As a consequence, the statistical tests that were made have very little validity. Welch's t-test was used for the determination of all the p-values. Even though this test is better at mitigating the difference in sample variance, it cannot do so because of the small sample size.

As for the experiment design, there are also several issues that need to be mentioned. I am not an expert user of Protégé and do not know what constitutes a change or impact that normally occurs while editing an ontology. Therefore, I have to assume that the tasks described in Section 5.2.2 might be too simplistic and do not actually simulate a realistic use case. In addition, one participant stated that the survey never defined impact as it is used in the questions. It also cannot be ruled out that the participants had a different conception of what impact is in this context. Another aspect that introduces more unknown variables is the fact that the study was conducted remotely. The participants' actions outside of Protégé were not monitored during the experiment. However, since the average completion time is rather close to my expectations, I assume that most participants did not do much else during the experiment.

With the above limitations in mind, the insights presented in Section 5.4 lose credibility. Even though participants liked the visualization of the plugin, it has to be assumed that most of them did not have enough experience with Protégé to make an assessment that is appropriate in this context. More experienced Protégé developers would have already seen many Protégé plugins and could therefore better assess the visualizations of the ChImp plugin. The same logic applies to the participants' assessment of how informative the ChImp plugin is. It is likely that most of them did not know what Protégé has to offer and how informed they should feel about a change at any point in time. Specifically, the experiment incorrectly assumed that they are able to find and compare other views inside Protégé to answer the questions about change impact. Another issue with the study itself was that it was not possible to ensure that the participants did not know about the ChImp plugin before the experiment. Even though it was never mentioned in any communication beforehand, several participants were aware of it due to their affiliation with the DDIS group. In summary, to answer the research questions a new study should be set up that has more expert users and more appropriate tasks to account for the mentioned limitations.

Another aspect that arose from the user evaluation is that participants might have had a different concept of what impact is in the context of ontology changes. More research into what developers' perceptions of change impact are could give more insight into the problem of displaying change impact.

7 Conclusions

In their work Pernischová et al. (2020) point out that ontology engineers often are not aware of the impact their changes have. Derived from requirements defined by them, this thesis presents the implementation of the ChImp plugin, a Protégé plugin dedicated to the display of change impact information. In addition, a technical and a user evaluation provide an assessment of the ChImp plugin and its functionalities.

This thesis covers the whole design process as well as the implementation of the plugin and provides insight into the intentions behind choices made during the development process. The technical evaluation resulted in the conclusion that the calculations performed inside the ChImp plugin are generally not computationally expensive. Hence, the plugin is not expected to delay Protégé during a change operation. Testing covers the calculation of several metrics. Calculation errors are therefore unlikely to occur. The user evaluation that Pernischová et al. (2020) already hinted at, did give a weak indication that developers like the visualizations of the plugin. However, the results did not provide an answer to the question of whether or not the content of the plugin is useful and informative to developers. The user evaluation did show that the experiment needs to be repeated with more participants that are experts with Protégé and tasks that are more appropriate to such experts.

The plugin currently only implements seven of the ten requirements stated by Pernischová et al. (2020). A planned future release covers the remaining three requirements by adding the functionality to export the impact information for release purposes. This future release also includes the functionality to dynamically choose the metrics that are displayed in the plugin. In addition, there is room for improvements and adjustments for the already implemented visualizations and the proposed designs and concepts provide a lot of possibilities for future releases.

The ChImp plugin aims to help ontology developers better understand what the consequences of their actions are, in order to improve the ontology evolution process as a whole. Future studies will be able to use the plugin to investigate ontology change impact. Hence, it might inspire more research in this area with the end goal of supporting ontology engineers' awareness and productivity.

References

What is osgi? – osgi^{TM} alliance. *https://www.osgi.org/developer/what-is-osgi/*. Accessed: 2020-10-19.

Yahoo finance app. *https://mobile.yahoo.com/finance*. Accessed: 2020-10-19.

Sunitha Abburu. A survey on ontology reasoners and comparison. *International Journal of Computer Applications*, 57:33–39, 2012.

Yalemisew Abgaz, Muhammad Javed, and Claus Pahl. A framework for change impact analysis of ontology-driven content-based systems. pages 402–411, 01 2011. doi: 10.1007/978-3-642-25126-9_52.

Carol V. Alexandru, Sebastian Proksch, Pooyan Behnamghader, and Harald C. Gall. Evo-clocks: Software evolution at a glance. In *2019 Working Conference on Software Visualization (VISSOFT)*, pages 12–22. IEEE, 2019. doi: 10.1109/VISSOFT.2019.00010.

Rebekka Alm, Sven Kiehl, Birger Lantow, and Kurt Sandkuhl. Applicability of quality metrics for ontologies on ontology design patterns. In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development - Volume 1: KEOD, (IC3K 2013)*, pages 48–57. INSTICC, SciTePress, 2013. doi: 10.5220/0004541400480057.

Ismailcem Arpinar, Karthikeyan Giriloganathan, and Boanerges Aleman-Meza. Ontology quality by detection of conflicts in metadata. *CEUR Workshop Proceedings*, 179, 01 2006.

Andrew Burton-Jones, Veda C. Storey, Vijayan Sugumaran, and Punit Ahluwalia. A semiotic metrics suite for assessing the quality of ontologies. 55(1):84–102, 2004. doi: 10.1016/j.datak.2004.11.010.

The Gene Ontology Consortium. The Gene Ontology Resource: 20 years and still Going strong. *Nucleic Acids Research*, 47(D1):D330–D338, 11 2018. doi: 10.1093/nar/gky1055.

Kathrin Dentler, Ronald Cornet, Annette Teije, and Nicolette de Keizer. Comparison of reasoners for large ontologies in the owl 2 el profile. *Semantic Web*, 2:71–87, 01 2011. doi: 10.3233/SW-2011-0034.

Rim Djedidi and Marie-Aude Aufaure. ONTO-EVO A L an ontology evolution approach guided by pattern modeling and quality evaluation. In *International symposium on foundations of information and knowledge systems*, pages 286–305, 2010.

Nick Drummond. ChangeView, March 2011. URL *https://code.google.com/archive/p/co-ode-owl-plugins/wikis/ChangeView.wiki*.

Marek Dudáš, Steffen Lohmann, Vojtěch Svátek, and Dmitry Pavlov. Ontology visualization methods and tools: a survey of the state of the art. 33:e10, 2018. doi: 10.1017/S0269888918000073.

Astrid Duque-Ramos, Jesualdo Tomás Fernández-Breis, Robert Stevens, and Nathalie Aussenac-Gilles. OQuaRE: A SQuaRE-based approach for evaluating the quality of ontologies. 43(2):18, 2011.

Astrid Duque-Ramos, Jesualdo Tomás Fernández-Breis, Miguela Iniesta, Michel Dumontier, Mikel Egaña Aranguren, Stefan Schulz, Nathalie Aussenac-Gilles, and Robert Stevens. Evaluation of the OQuaRE framework for ontology quality. 40(7):2696–2703, 2013. doi: 10.1016/j.eswa.2012.11.004.

Astrid Duque-Ramos, Martin Boeker, Ludger Jansen, Stefan Schulz, Miguela Iniesta, and Jesualdo Fernandez-Breis. Evaluating the good ontology design guideline (goodod) with the ontology quality requirements and evaluation method and metrics (oquare). *PloS one*, 9:e104463, 08 2014. doi: 10.1371/journal.pone.0104463.

Astrid Duque-Ramos, Manuel Quesada-Martínez, Miguela Iniesta-Moreno, Jesualdo Tomás Fernández-Breis, and Robert Stevens. Supporting the analysis of ontology evolution processes through the combination of static and dynamic scaling functions in OQuaRE. 7(1):63, 2016. doi: 10.1186/s13326-016-0091-z.

Sean M. Falconer, Tania Tudorache, and Natalya Fridman Noy. An analysis of collaborative patterns in large-scale ontology development projects. In *K-cap*, pages 25–32. ACM, 2011.

Giorgos Flouris, Dimitris Manakanatas, Haridimos Kondylakis, Dimitris Plexousakis, and Grigoris Antoniou. Ontology change: classification and survey. 23(2):117–152, 2008. doi: 10.1017/S0269888908001367.

Harald Gall, Mehdi Jazayeri, and Claudio Riva. Visualizing software release histories: the use of color and third dimension. In *Proceedings IEEE International Conference on Software Maintenance - 1999 (ICSM'99). 'Software Maintenance for Business Change' (Cat. No.99CB36360)*, pages 99–108. IEEE, 1999. doi: 10.1109/ICSM.1999.792584.

Aldo Gangemi, Carola Catenacci, Massimiliano Ciaramita, and Jos Lehmann. Modelling ontology evaluation and validation. In *European semantic web conference*, pages 140–154, 2006.

Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987. ISBN 0934613311.

Rafael S. Goncalves, Bijan Parsia, and Uli Sattler. Analysing the evolution of the NCI thesaurus. In *2011 24th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 1–6. IEEE, 2011. doi: 10.1109/CBMS.2011.5999163.

Rafael S. Gonçalves, Bijan Parsia, and Ulrike Sattler. Categorising logical differences between OWL ontologies. In *CIKM*, pages 1541–1546. ACM, 2011.

Anika Gross, Michael Hartung, Kay Prüfer, Janet Kelso, and Erhard Rahm. Impact of ontology evolution on functional analyses. *Bioinformatics*, 28(20):2671–2677, 2012.

Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199 – 220, 1993. ISSN 1042-8143. doi: 10.1006/knac.1993.1008.

Nicola Guarino and Pierdaniele Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. In *Towards very Large Knowledge bases: Knowledge Building and Knowledge sharing*, pages 25–32. IOS Press, 1995.

Jon Gulla, Geir Solskinnsbakk, Per Myrseth, Veronika Haderlein, and Olga Cerrato. Semantic drift in ontologies. volume 2, pages 13–20, 01 2010.

Michael Hartung, Anika Groß, and Erhard Rahm. COnto–diff: generation of complex evolution mappings for life science ontologies. 46(1):15–32, 2012. doi: 10.1016/j.jbi.2012.04.009.

Janna Hastings, Nico Adams, Marcus Ennis, Duncan Hull, and Christoph Steinbeck. Chemical ontologies: what are they, what are they for and what are the challenges. *Journal of Cheminformatics*, 3:1–1, 04 2011. doi: 10.1186/1758-2946-3-S1-O4.

Matthew Horridge and Sean Bechhofer. The owl api: A java api for working with owl 2 ontologies. In *Proceedings of the 6th International Conference on OWL: Experiences and Directions - Volume 529*, OWLED'09, page 49–58, Aachen, DEU, 2009. CEUR-WS.org.

Ajaz Hussain, Khalid Latif, Aimal Rextin, Amir Hayat, and Masoon Alam. Scalable visualization of semantic nets using power-law graphs. *Applied Mathematics & Information Sciences*, 8:355–, 01 2014. doi: 10.12785/amis/080145.

Akrivi Katifori, Constantin Halatsis, George Lepouras, Costas Vassilakis, and Eugenia Giannopoulou. Ontology visualization methods—a survey. 39(4):10, 2007. doi: 10.1145/1287620.1287621.

Simone Kriglstein. User requirements analysis on ontology visualization. In *2009 International Conference on Complex, Intelligent and Software Intensive Systems*, pages 694–699. IEEE, 2009. doi: 10.1109/CISIS.2009.37.

Sasa Kuhar and Vili Podgorelec. Ontology visualization for domain experts: A new solution. In *2012 16th International Conference on Information Visualisation*, pages 363–369. IEEE, 2012. doi: 10.1109/IV.2012.67.

Birger Lantow and Kurt Sandkuhl. An analysis of applicability using quality metrics for ontologies on ontology design patterns. *Intelligent Systems in Accounting, Finance and Management*, 22(1):81–99, 2015.

Michele Lanza and Stéphane Ducasse. Understanding software evolution using a combination of software visualization and software metrics. 8(1):135–149, 2002. doi: 10.3166/objet.8.1-2.135-149.

Michele Lanza and Stéphane Ducasse. Polymetric views - a lightweight visual approach to reverse engineering. 29(9):782–795, 2003. doi: 10.1109/TSE.2003.1232284.

Michele Lanza and Stéphane Ducasse. Codecrawler - an information visualization tool for program comprehension. In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, pages 672–673. IEEe, 2005. doi: 10.1109/ICSE.2005.1553647.

Michele Lanza, Harald Gall, and Philippe Dugerdil. EvoSpaces: Multi-dimensional navigation spaces for software evolution. In *2009 13th European Conference on Software Maintenance and Reengineering*, pages 293–296. IEEE, 2009. doi: 10.1109/CSMR.2009.14.

Steffen Lohmann, Stefan Negru, and David Bold. The protégévowl plugin: Ontology visualization for everyone. In Valentina Presutti, Eva Blomqvist, Raphael Troncy, Harald Sack, Ioannis Papadakis, and Anna Tordai, editors, *The Semantic Web: ESWC 2014 Satellite Events*, pages 395–400, Cham, 05 2014. Springer International Publishing. doi: 10.1007/978-3-319-11955-7_55.

Nikos Manouselis, Miguel Ángel Sicilia, and Daniel Rodríguez. Exploring ontology metrics in the biomedical domain. 1(1):2319–2328, 2010. doi: 10.1016/j.procs.2010.04.260.

Mark A. Musen. The protégé project: a look back and a look forward. *AI Matters*, 1(4): 4–12, 2015. doi: 10.1145/2757001.2757003.

Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193:217–250, December 2012. ISSN 0004-3702. doi: 10.1016/j.artint.2012.07.001.

Jérôme Oesch. Benchmarking incremental reasoner systems. Master's thesis, University of Zürich, 2018.

Anthony M. Orme, Haining Yao, and Letha H. Etzkorn. Indicating ontology data quality, stability, and completeness throughout ontology evolution. 19(1):49–75, 2007. doi: 10.1002/smr.341.

David Osumi-Sutherland, Enrico Ponta, Mélanie Courtot, Helen Parkinson, and Laura Badi. Using owl reasoning to support the generation of novel gene sets for enrichment analysis. *Journal of Biomedical Semantics*, 9, 12 2018. doi: 10.1186/s13326-018-0175-z.

Romana Pernischová. The butterfly effect in knowledge graphs: Predicting the impact of changes in the evolving web of data. In *Doctoral Consortium at ISWC 2019*, 2019.

Romana Pernischová, Mirko Serbak, Dell'Aglio Daniele, and Abraham Bernstein. Chimp: Visualizing ontology changes and their impact in protégé. In *Proceedings of the Fourth International Workshop on Visualization and Interaction for Ontologies and Linked Data co-located with the 18th International Semantic Web Conference, VOILA@ISWC 2020*. CEUR-WS.org, 2020.

Romana Pernischová, Daniele Dell'Aglio, Matthew Horridge, Matthias Baumgartner, and Abraham Bernstein. Toward predicting impact of changes in evolving knowledge graphs. In *ISWC satellites*, volume 2456 of *CEUR workshop proceedings*, pages 137–140, Aukland, NZ, October 2019. CEUR-WS.org.

Martin Pinzger, Harald Gall, Michael Fischer, and Michele Lanza. Visualizing multiple evolution metrics. In *Proceedings of the 2005 ACM symposium on Software visualization - SoftVis '05*, page 67. ACM Press, 2005. doi: 10.1145/1056018.1056027.

Ying Shen, Daoyuan Chen, Buzhou Tang, Min Yang, and Kai Lei. Eapb: Entropy-aware path-based metric for ontology quality. *Journal of Biomedical Semantics*, 9, 12 2018. doi: 10.1186/s13326-018-0188-7.

Ben Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages*, pages 336–343, 1996.

Miguel Ángel Sicilia, Daniel Rodríguez, Elena García-Barriocanal, and Salvador Sánchez-Alonso. Empirical findings on ontology metrics. 39(8):6706–6711, 2012. doi: 10.1016/j.eswa.2011.11.094.

Michaell Sintek. Ontoviz tab: Visualizing protege ontologies. 2003.

Thanos Stavropoulos, Stelios Andreadis, Efstratios Kontopoulos, Marina Riga, Panagiotis Mitzias, and Ioannis Kompatsiaris. Semadrift: A protégé plugin for measuring semantic drift in ontologies. In *1st International Workshop on Detection, Representation and Management of Concept Drift in Linked Open Data (Drift-a-LOD) in Conjunction with the 20th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, volume CEUR Vol-1799, pages 34–41, Bologna, Italy, 11 2016. CEUR Workshop Proceedings.

Thanos G. Stavropoulos, Stelios Andreadis, Efstratios Kontopoulos, and Ioannis Kompatsiaris. Semadrift: A hybrid method and visual tools to measure semantic drift in ontologies. *Journal of Web Semantics*, 54:87 – 106, 2019. doi: 10.1016/j.websem.2018.05.001. Managing the Evolution and Preservation of the Data Web.

Ljiljana Stojanovic. *Methods and Tools for Ontology Evolution.* PhD thesis, Universität Fridericiana zu Karlsruhe, 2004.

Samir Tartir and I. Budak Arpinar. Ontology evaluation and ranking using OntoQA. In *International Conference on Semantic Computing (ICSC 2007)*, pages 185–192. IEEE, 2007. doi: 10.1109/ICSC.2007.19.

Samir Tartir, I Budak Arpinar, and Amit P Sheth. Ontological evaluation and validation. In *Theory and applications of ontology: Computer applications*, pages 115–130. Springer, 2010.

Christoph Tempich and Raphael Volz. Towards a benchmark for Semantic Web reasoners-an analysis of the DAML ontology library. In *EON*, volume 87, 2003.

Stein L. Tomassen and Darijus Strasunskas. An ontology-driven approach to web search: Analysis of its sensitivity to ontology quality and search tasks. In *Proceedings of the 11th International Conference on Information Integration and Web-Based Applications amp; Services*, iiWAS '09, page 130–138, New York, NY, USA, 2009. Association for Computing Machinery. doi: 10.1145/1806338.1806368.

Fouad Zablith, Grigoris Antoniou, Mathieu d'Aquin, Giorgos Flouris, Haridimos Kondylakis, Enrico Motta, Dimitris Plexousakis, and Marta Sabou. Ontology evolution: A process-centric survey. *The Knowledge Engineering Review*, 30:45–75, 01 2015. doi: 10.1017/S0269888913000349.

Hongyu Zhang, Yuan-Fang Li, and Hee Beng Kuan Tan. Measuring design complexity of semantic web ontologies. 83(5):803–814, 2010. doi: 10.1016/j.jss.2009.11.735.

A

Appendix

A.1 Impact Metrics

University of Zurich^UZH

Department of Informatics

Definition of Impact Measures for Materialization over Evolving Ontologies

Romana Pernisch, Daniele Dell'Aglio

5. February 2020

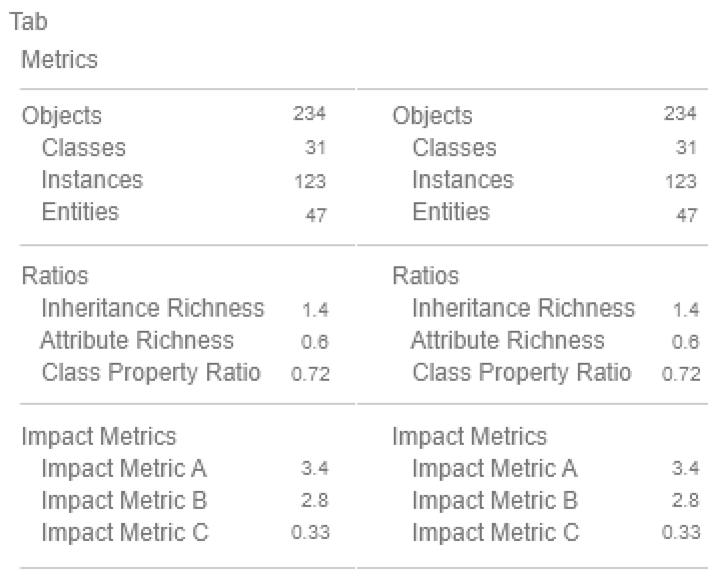
This document defines and describes measures for the quantification of impact over the materialization of evolving ontologies $\mathcal{O} = \{O_1, O_2, ...O_i, O_{i+1}, ...\}$. The materialization is a function executed with an ontology as input $M_i = materialization(O_i)$. The impact is calculated between two materializations M_i and M_{i+1}. Some variables and basic concepts are introduced below:

Formula		Description
m_i	$= \lvert M_i \rvert = \lvert G_i \backslash O_i \rvert$	Materialized part of the graph at time i
$m_{i,i+1}$	$= \lvert M_i \cap M_{i+1} \rvert$	number of axioms shared between the materialized versions
Δ_i^+	$= \lvert M_{i+1} \backslash M_i \rvert$	New axioms part of M_{i+1} but not in M_i
Δ_i^-	$= \lvert M_i \backslash M_{i+1} \rvert$	Old axioms part of M_i but not in M_{i+1}
h_{Δ_i}	$= \lvert SubClassOf(\cdot,\cdot)_{m,i} \rvert$	Number of $SubClassOf$ axioms in Δ_i^+ and Δ_i^-

Below, the different impact measures are defined:

Formula		Name
$impact_{\Delta^+,m_i}$	$= \frac{\Delta_i^+}{m_i}$	added inference old ratio
$impact_{\Delta^-,m_i}$	$= \frac{\Delta_i^-}{m_i}$	removed inference old ratio
$impact_{\Delta^+,m_{i+1}}$	$= \frac{\Delta_i^+}{m_{i+1}}$	added inference new ratio
$impact_{\Delta^-,m_{i+1}}$	$= \frac{\Delta_i^-}{m_{i+1}}$	removed inference new ratio
$impact_{\Delta^+,\cap}$	$= \frac{\Delta_i^+}{m_{i,i+1}}$	added inference impact
$impact_{\Delta^-,\cap}$	$= \frac{\Delta_i^-}{m_{i,i+1}}$	removed inference impact
$impact_{\Delta,a}$	$= \frac{\Delta_i^+ + \Delta_i^-}{max(m_i, m_{i+1})}$	change max impact
$impact_{\Delta,\cap}$	$= \frac{\Delta_i^+ + \Delta_i^-}{m_{i,i+1}}$	change impact
$impact_{\Delta_\sqsubseteq,\cap}$	$= \frac{h_{\Delta_i}}{m_{i,i+1}}$	subsumption change impact
$impact_{\Delta_\sqsubseteq,\cap_\sqsubseteq}$	$= \frac{h_{\Delta_i}}{m_{i+1} - m_i}$	subsumption impact

A.2 Designs



Tab

Metrics

Objects	234	Objects	234
Classes	31	Classes	31
Instances	123	Instances	123
Entities	47	Entities	47
Ratios		Ratios	
Inheritance Richness	1.4	Inheritance Richness	1.4
Attribute Richness	0.6	Attribute Richness	0.6
Class Property Ratio	0.72	Class Property Ratio	0.72
Impact Metrics		Impact Metrics	
Impact Metric A	3.4	Impact Metric A	3.4
Impact Metric B	2.8	Impact Metric B	2.8
Impact Metric C	0.33	Impact Metric C	0.33

Figure A.1: Prototype 1

A.3 About the Plugin

The plugin is available on the following repositories: public[1], private[2].

A.3.1 Build Instructions

Prerequisites

To build and run the plugin, the following items must be installed:

- Apache's Maven
- A Protégé distribution (5.0.0 or higher)

Build

In the chimp-plugin directory: `mvn clean package`

[1]https://gitlab.ifi.uzh.ch/DDIS-Public/chimp-protege-plugin

[2]https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-mirko-serbak

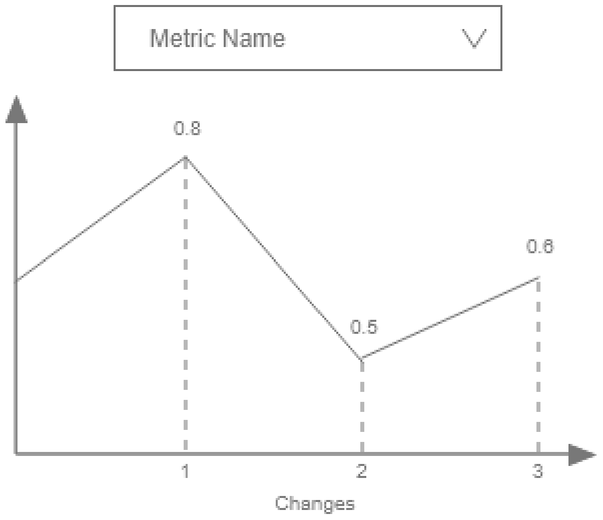


Figure A.2: Prototype 21

On build completion, the "target" directory will contain a chimp-plugin${version}.jar file. Copy the JAR file from the target directory to the "plugins" sub directory of your Protege distribution

A.3.2 Installation Instructions

- Download the plugin from the link provided above.
- Launch your Protege distribution.
- Open from menu: Window > Views > Ontology views > ChImp (Change Impact).
- Select About from the Help menu to verify successful installation

A.3.3 Usage Instructions

Once you have opened the plugin view in a Protégé tab, you can start using it. You need to start a reasoner to see content in the impact view.

A.3.4 Dependencies

The plugin uses several frameworks in the individual components. The following is a list of all dependencies for the compile scope. Thus, these are all dependencies that support the core functionalities of the plugin.

In order for the plugin to compile, the following libraries need to be added:

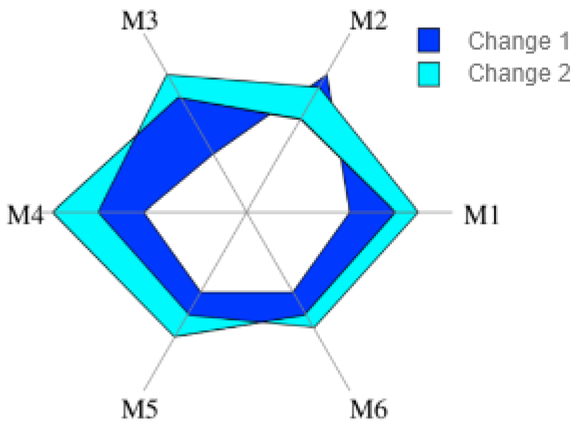


Figure A.3: Prototype 3

- protege-editor-owl 5.5.0[3]
  Every Protégé plugin needs to access the Protégé editor package in order to access Protégé.

- miglayout 3.7.4[4]
  Miglayout is a layout manager for Java applications. ChImp uses it to manage the Java Swing panels.

- jlatexmath 1.0.7[5]
  The metrics inside of the ChImp plugin have an optional String parameter with which a LaTeX formula can be added to its description. JLatexMath is used to print these formulas as images so that the plugin can display them.

- xchart 3.6.5[6]
  The graph inside of the chart view is implemented with XChart.

In addition, several libraries are used to test the plugin. They are necessary for compilation but are not part of the final code package.

- junit-jupiter 5.6.2[7]
  The tests listed in Section 3.4.4 are all written with JUnit.

[3]https://mvnrepository.com/artifact/edu.stanford.protege/protege-editor-owl
[4]https://mvnrepository.com/artifact/com.miglayout/miglayout
[5]https://mvnrepository.com/artifact/org.scilab.forge/jlatexmath
[6]https://mvnrepository.com/artifact/org.knowm.xchart/xchart
[7]https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api

- assertj-core 3.17.0[8]
  AssertJ supports the testing with JUnit.
- org.semanticweb.hermit 1.4.5.456[9]
  The Hermit reasoner is used to test the impact metrics.
- commons-csv 1.8[10]

A.4 ChImp Survey

[8]https://mvnrepository.com/artifact/org.assertj/assertj-core
[9]https://mvnrepository.com/artifact/net.sourceforge.owlapi/org.semanticweb.hermit
[10]https://mvnrepository.com/artifact/org.apache.commons/commons-csv

ChImp Survey

Ontology Change Impact Survey

Thank you for taking part in this survey. The survey guides you through a small experiment and is meant to be kept open during the whole process. Please follow the instructions carefully and answer all questions to the best of your ability.

There are 35 questions in this survey.

Participant Information

The following questions are about how proficient you are with ontologies.

In university, how many courses did you take that revolved around ontologies? *

☐ Only numbers may be entered in this field.
Please write your answer here:

In university, how many courses did you participate in that used Protégé as a development tool? *

☐ Only numbers may be entered in this field.
Please write your answer here:

Have you ever used Protégé outside of these courses? *

Please choose **only one** of the following:

○ Yes
○ No

How many years of experience do you have with Protégé?

Only answer this question if the following conditions are met: Answer was 'Yes' at question '3 [participantinfo3]' (Have you ever used Protégé outside of these courses?)

☐ Only numbers may be entered in this field.
Please write your answer here:

Have you ever used Protégé in a professional setting?

Only answer this question if the following conditions are met: Answer was 'Yes' at question '3 [participantinfo3]' (Have you ever used Protégé outside of these courses?)

Please choose **only one** of the following:

○ Yes
○ No

Introduction

This experiment uses Protégé as an ontology editing tool. Please execute the steps on this page. In the end you should have a Protégé window with an opened ontology.

Preparation

Before the experiment you should have received a link to download Protégé as well as a sample ontology. If you have not done this and do not have access to the download links that were provided, please contact the survey instructor. Do not download Protégé from the official website.

Startup

If you have already completed the downloads proceed as follows:

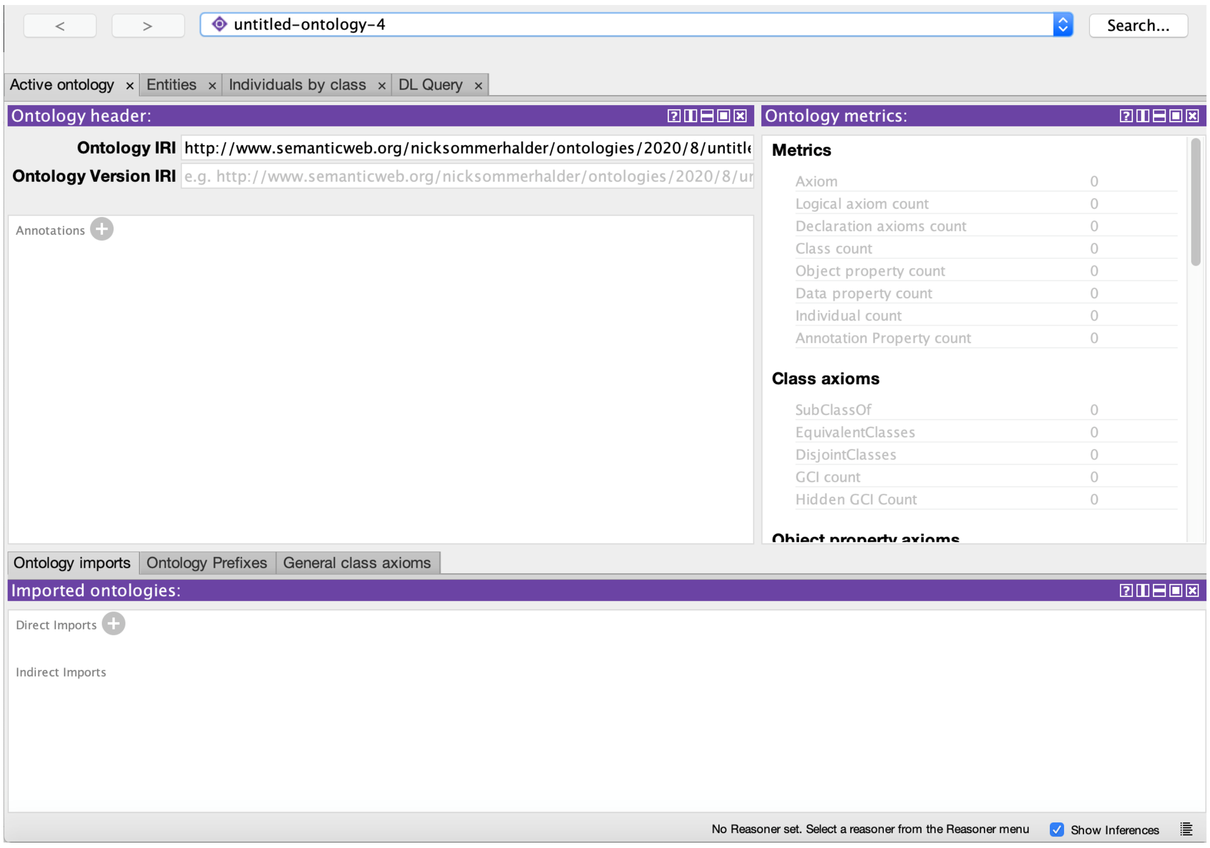
1. Unzip the Protégé distribution and place the extracted folder where you can easily find it (e.g. on your Desktop. You can delete the folder after the experiment).
2. Place the dowloaded ontology next to the extracted folder.
3. Open Protégé
   - Windows / Mac users can click on the application file inside the extracted folder
   - Linux users need to start Protégé by executing ./run.sh from the command line.

You will be presented with an update screen popup when you open Protégé. Close this popup.

Protégé Overview

The following gives a very brief overview of the most basic functionalities Protégé has to offer. If you have already worked with Protégé you may skip to the end of the page.

You should now have an open Protégé window. By default, you will be presented with the following screen:



Ontology IRI http://www.semanticweb.org/nicksommerhalder/ontologies/2020/8/untitle
Ontology Version IRI e.g. http://www.semanticweb.org/nicksommerhalder/ontologies/2020/8/ur

Metrics

Axiom	0
Logical axiom count	0
Declaration axioms count	0
Class count	0
Object property count	0
Data property count	0
Individual count	0
Annotation Property count	0

Class axioms

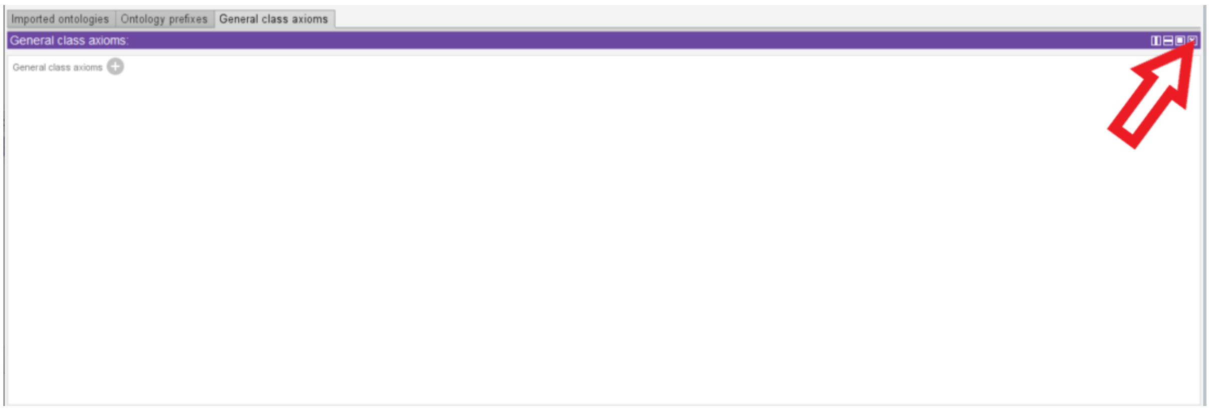
SubClassOf	0
EquivalentClasses	0
DisjointClasses	0
GCI count	0
Hidden GCI Count	0

Object property axioms

No Reasoner set. Select a reasoner from the Reasoner menu

On top, you can see the tabs Active ontology, Entities, Individuals by class and DL Query. These tabs contain various views that show information about the underlying ontology. In the tab Active ontology, you will see the views Ontology header, Ontology metrics and Imported ontologies.

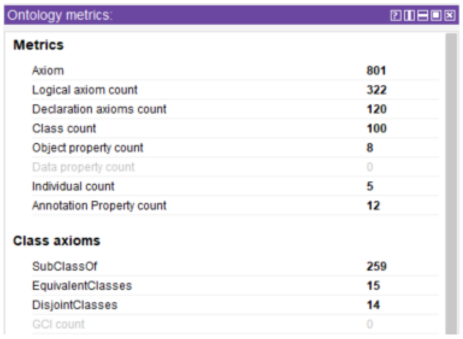
Please close the views at the bottom of the page by clicking on the close sign three times:



You should now only be able to see the Ontology header and the Ontology metrics. Please leave the other views as they are.

To see some real data in Protégé we can open an ontology file. Do this by clicking on *File* > *Open...* and then select the ontology file that you downloaded before the experiment.

You should now see that the views in the Active ontology tab are showing information about the ontology. e.g. Ontology Metrics displays several metrics such as the axiom and the class count of the ontology:


Ontology metrics:

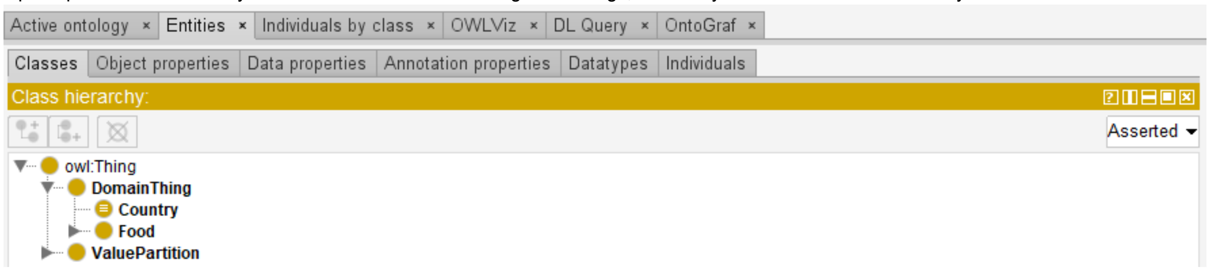
Metrics

Axiom	801
Logical axiom count	322
Declaration axioms count	120
Class count	100
Object property count	8
Data property count	0
Individual count	5
Annotation Property count	12

Class axioms

SubClassOf	259
EquivalentClasses	15
DisjointClasses	14
GCI count	0

Open up the Class hierarchy view on the tab Entities. Among other things, it allows you to edit the class hierarchy:



By selecting a class in the hierarchy, you can use the buttons above the hierarchy to edit it. These are the buttons from left to right:

Add a subclass | Add a class on the same level | Delete a class

Protégé usually also offers the all these options by rightclicking on an element.

Experiment

It is vital for the experiment that you do not add any third party plugins. Please use the views that Protégé already provides.

ChImp Introduction

In this experiment the ChImp plugin is used. The ChImp view presents changes and various metrics to visualize change impact. The following is a brief overview.

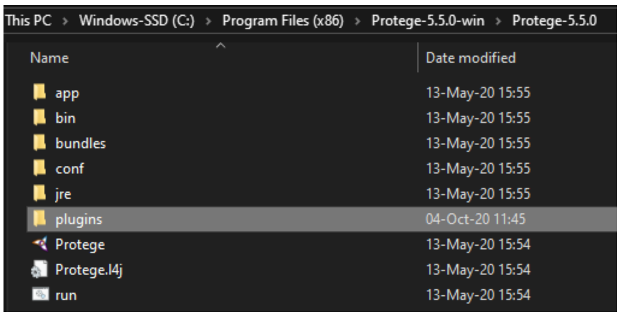
Adding the Plugin to Protégé

First, close Protégé.

Please download the plugin with the following link. It is a jar file:
Plugin Download (https://drive.switch.ch/index.php/s/Pxq4j55vIUqIfep)

Once you have the file, go to your Protégé directory and add it to the plugin subdirectory:

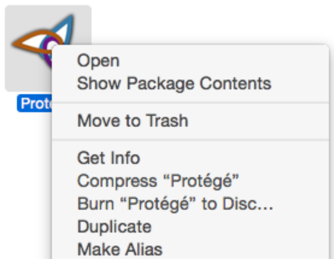
On Windows and Linux:



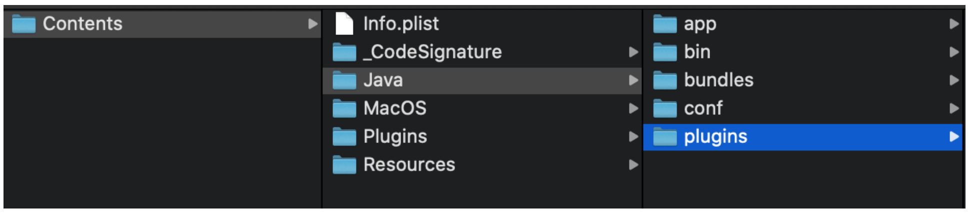
This PC › Windows-SSD (C:) › Program Files (x86) › Protege-5.5.0-win › Protege-5.5.0

Name	Date modified
app	13-May-20 15:55
bin	13-May-20 15:55
bundles	13-May-20 15:55
conf	13-May-20 15:55
jre	13-May-20 15:55
plugins	04-Oct-20 11:45
Protege	13-May-20 15:54
Protege.l4j	13-May-20 15:54
run	13-May-20 15:54

On Mac:
Rightclick on the Protégé app and select "Show Package Contents"



Then put the file in the following directory **(It's not the first Plugin directory, it's the one under Java)**:

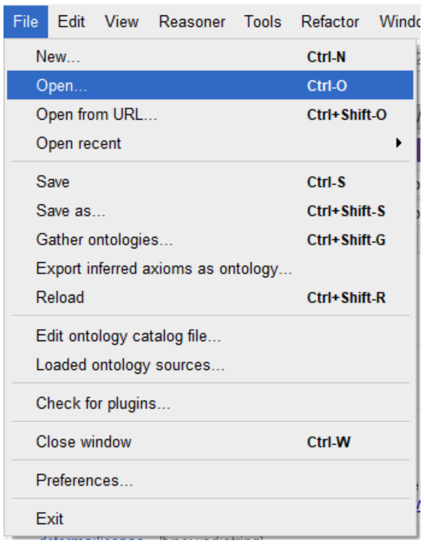


Once you have done this, you can open Protégé again.

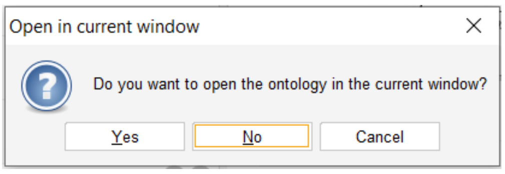
Open the Pizza Ontology

Please open a new pizza ontology to proceed.

Open the pizza.owl file by selecting File and Open... and then choosing the file on your computer.



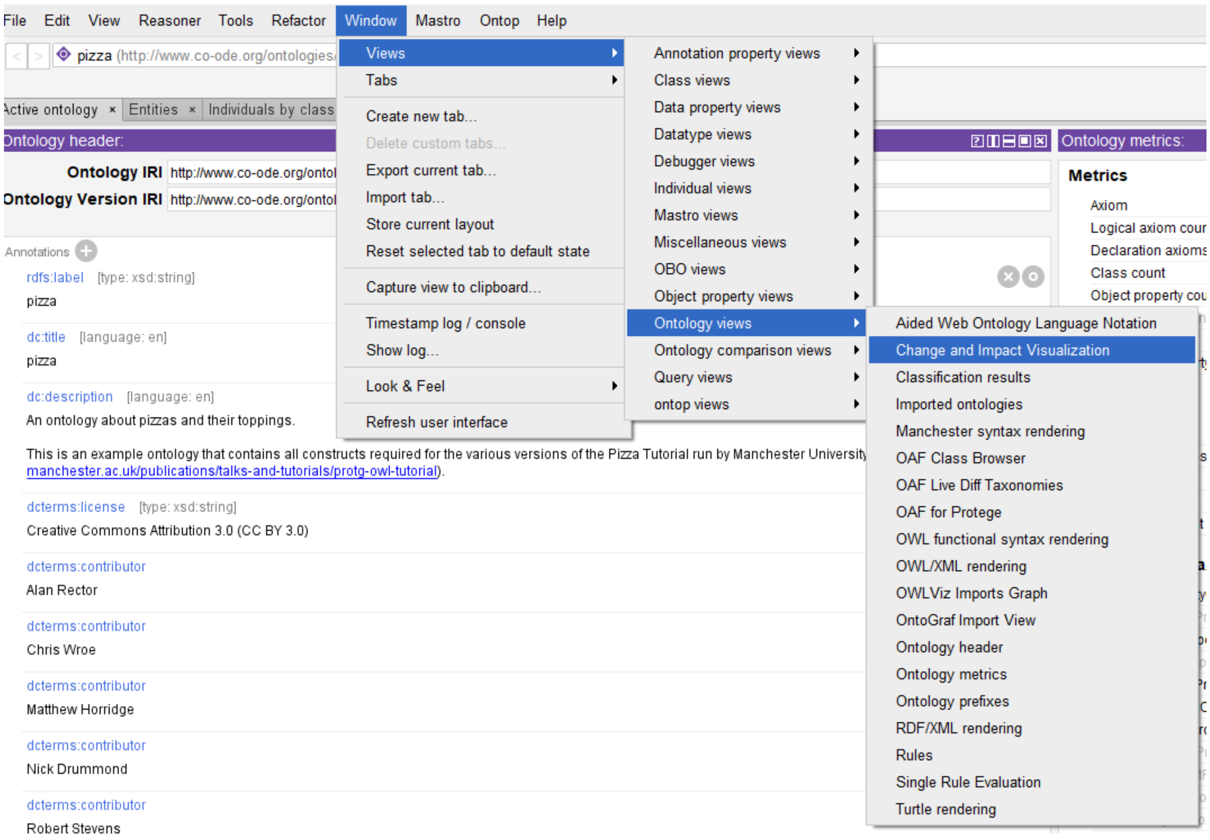
If the following dialog pops up, make sure to click **No**.


Open in current window

Do you want to open the ontology in the current window?

Adding the ChImp View

To add the ChImp view to a tab of Protégé, we select it from the top menu:



pizza (http://www.co-ode.org/ontologies/

Ontology IRI http://www.co-ode.org/ontol

Ontology Version IRI http://www.co-ode.org/ontol

Annotations

rdfs:label [type: xsd:string]
pizza

dc:title [language: en]
pizza

dc:description [language: en]
An ontology about pizzas and their toppings.

This is an example ontology that contains all constructs required for the various versions of the Pizza Tutorial run by Manchester University
manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial).

dcterms:license [type: xsd:string]
Creative Commons Attribution 3.0 (CC BY 3.0)

dcterms:contributor
Alan Rector

dcterms:contributor
Chris Wroe

dcterms:contributor
Matthew Horridge

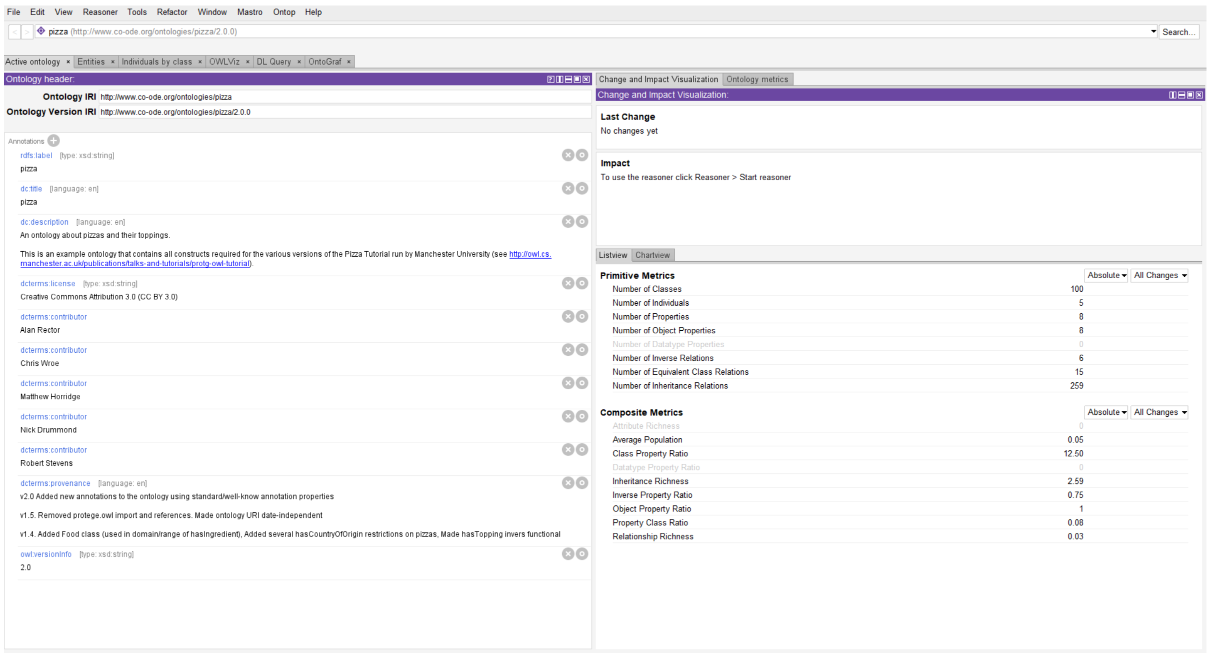
dcterms:contributor
Nick Drummond

dcterms:contributor
Robert Stevens

You are then given a pointer with which you can drag the view to a place of your choosing. Place it on top of the Ontology metrics view (this is the view on the right side of the Active ontology tab) so that the Ontology metrics view becomes a tab. You can also close the ontology metrics view.

Click OK if a warning message appears.

Your screen should now look similar to the following image:



pizza (http://www.co-ode.org/ontologies/pizza/2.0.0)

Ontology IRI http://www.co-ode.org/ontologies/pizza

Ontology Version IRI http://www.co-ode.org/ontologies/pizza/2.0.0

Annotations

rdfs:label [type: xsd:string]
pizza

dc:title [language: en]
pizza

dc:description [language: en]
An ontology about pizzas and their toppings.

This is an example ontology that contains all constructs required for the various versions of the Pizza Tutorial run by Manchester University (see http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial).

dcterms:license [type: xsd:string]
Creative Commons Attribution 3.0 (CC BY 3.0)

dcterms:contributor
Alan Rector

dcterms:contributor
Chris Wroe

dcterms:contributor
Matthew Horridge

dcterms:contributor
Nick Drummond

dcterms:contributor
Robert Stevens

dcterms:provenance [language: en]
v2.0 Added new annotations to the ontology using standard/well-know annotation properties

v1.5. Removed protege.owl import and references. Made ontology URI date-independent

v1.4. Added Food class (used in domain/range of hasIngredient), Added several hasCountryOfOrigin restrictions on pizzas, Made hasTopping invers functional

owl:versionInfo [type: xsd:string]
2.0

Last Change
No changes yet

Impact
To use the reasoner click Reasoner > Start reasoner

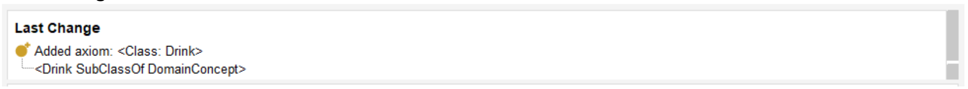
Primitive Metrics	
Number of Classes	100
Number of Individuals	5
Number of Properties	8
Number of Object Properties	8
Number of Datatype Properties	0
Number of Inverse Relations	6
Number of Equivalent Class Relations	15
Number of Inheritance Relations	259

Composite Metrics	
Attribute Richness	0
Average Population	0.05
Class Property Ratio	12.50
Datatype Property Ratio	0
Inheritance Richness	2.59
Inverse Property Ratio	0.75
Object Property Ratio	1
Property Class Ratio	0.08
Relationship Richness	0.03

The ChImp Plugin

The ChImp Plugin is split into three panels:

- **Last Change**

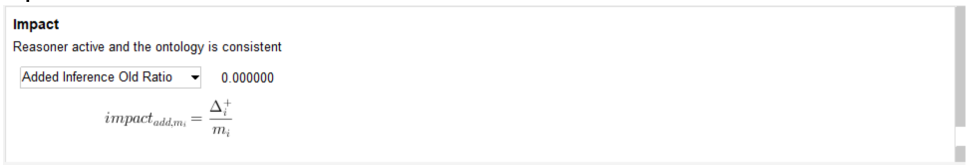


Last Change

Added axiom: <Class: Drink>
<Drink SubClassOf DomainConcept>

This view presents you with the last changes that were made. Protégé stores ontologies as sets of axioms. Hence, the changes here are also displayed in that way.

- **Impact**



Impact

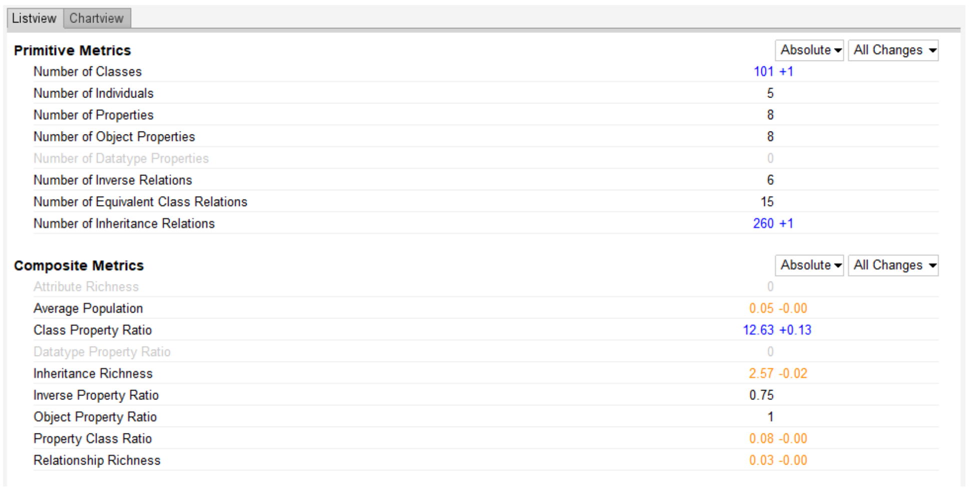
Reasoner active and the ontology is consistent

Added Inference Old Ratio 0.000000

$$impact_{add,m_i} = \frac{\Delta_i^+}{m_i}$$

The impact panel displays impact metrics. These are complex metrics calculated with the materializations of the ontologies.

- **Standard Metrics**



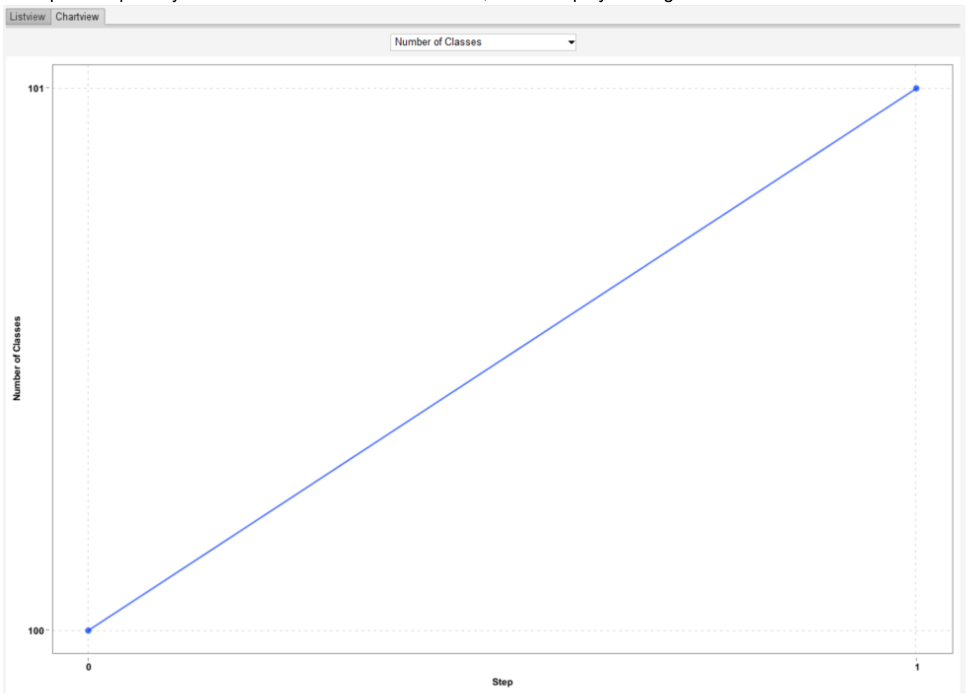
Listview | Chartview

Primitive Metrics	Absolute / All Changes
Number of Classes	101 +1
Number of Individuals	5
Number of Properties	8
Number of Object Properties	8
Number of Datatype Properties	0
Number of Inverse Relations	6
Number of Equivalent Class Relations	15
Number of Inheritance Relations	260 +1

Composite Metrics	Absolute / All Changes
Attribute Richness	0
Average Population	0.05 -0.00
Class Property Ratio	12.63 +0.13
Datatype Property Ratio	0
Inheritance Richness	2.57 -0.02
Inverse Property Ratio	0.75
Object Property Ratio	1
Property Class Ratio	0.08 -0.00
Relationship Richness	0.03 -0.00

The standard metrics view is similar to the Ontology metrics view by Protégé. It lists simple count and ratio metrics. However, it also shows change. With two dropdowns it can be adjusted to either display absolute or relative changes and all changes since the beginning or just the last change.

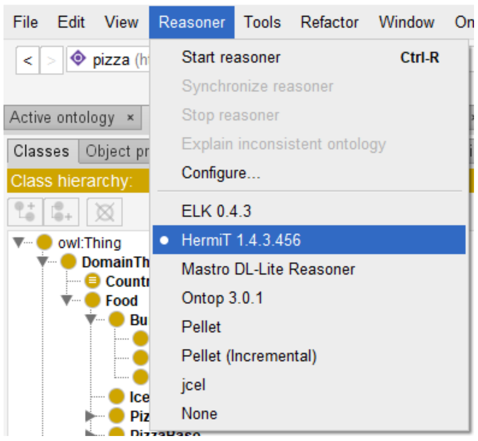
- On top of the panel you can also switch to the chart view, which displays a single metric over time:



Reasoner Initialization

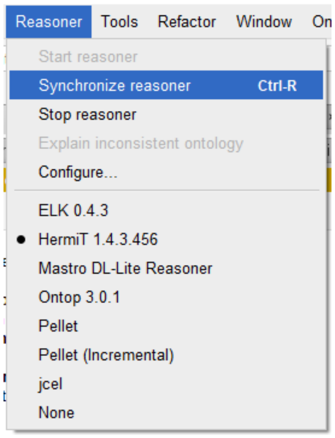
ChImp's Impact panel requires a reasoner. This is an algorithm that computes materializations. What is meant by that is that it tries to compute all logical inferences that can be made from the current ontology (Example: if A is a subclass of B and C is equivalent to B, then A is a subclass of C). The impact metrics that are displayed in the Impact panel are calculated with these inferences.

Make sure that you have an open view of ChImp. To start the reasoner, **select Reasoner and then HermiT** from the menu (HermiT comes preinstalled with Protégé. If you do not have it contact the survey instructor):



Once you have selected it, select **Reasoner and then Start Reasoner** from the same menu.

Please note that during the task, you have to synchronize the Reasoner to see changes in the Panel. To do this, select Reasoner and then Synchronize reasoner:

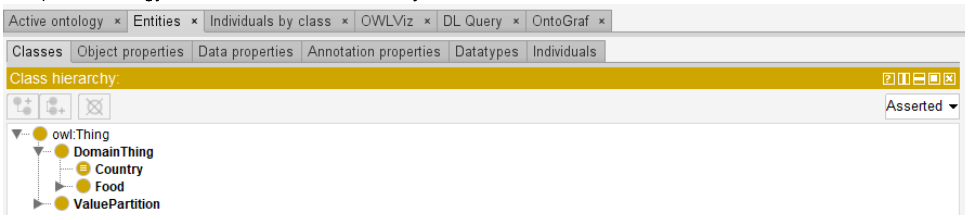


Task Addition

Your task is to alter the pizza ontology. Please ensure that you have it open in Protégé. Perform all of the steps listed below.

First Part

Open up the Class hierarchy view on the tab Entities.
The pizza ontology contains the classes "Country" and "Food" to define its domain:

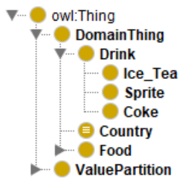


Add an additonal class "Drink" to the ontology. "Drink" is on the same level as "Country" and "Food" and has the subclasses "Coke", "Sprite" and "Ice Tea".

Hint:

1. Rightclick on Domain Thing and select Add subclass...
2. Write "Drink" into the name field and press Enter.
3. Rightclick on Drink and select Add subclass...
4. Write "Coke" into the name field and press Enter.
5. repeat for the other two drinks

After this process, you should end up with the following hierarchy:



The experiment at hand revolves around the impact your changes have on the ontology as a whole. Now for the evaluation, please take a look at all the views that are currently open in Protege (do not open new ones). **Please also take a look at other tabs such as the Active ontology.** *

Please choose the appropriate response for each item:

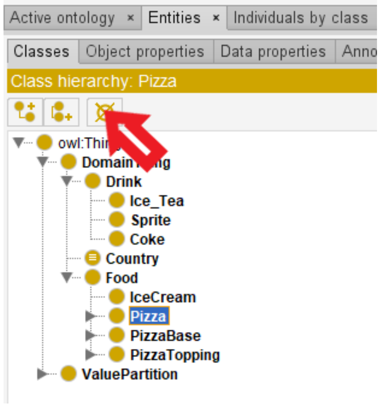
	1	2	3	4	5
Rate how well informed you are about the impact the previous changes had on the ontology as a whole (with 1 being not informed at all and 5 being very well informed)	◯	◯	◯	◯	◯

Write down the view / element on the screen that was most crucial for you for the previous question. *

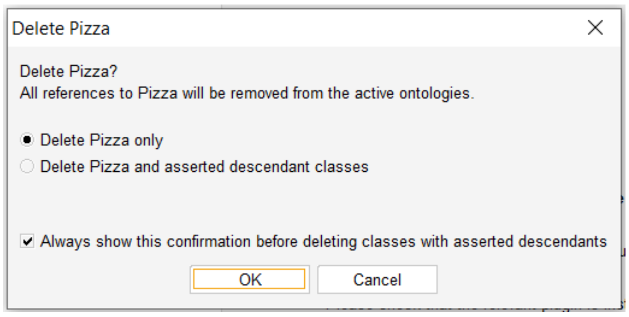
Please write your answer here:

Second Part

Now again open up the Class hierarchy view on the tab Entities.
For the second part, select the class Pizza in the domain Food and click the rightmost button in the view to delete the class.



You will be prompted with a delete prompt. Make sure that you have Delete Pizza only enabled:


Delete Pizza

Delete Pizza?
All references to Pizza will be removed from the active ontologies.

[✓ Delete Pizza only]
Delete Pizza and asserted descendant classes

[✓ Always show this confirmation before deleting classes with asserted descendants]

Please note, a lot of subclasses of Pizza now do not have a superclass anymore and are therefore moved to the upper hierarchy level. Your hierarchy might look a bit crowded.

Now again to evaluate the impact the last change had on the ontology as a whole, please take a look at all the views that are currently open in Protege (do not open new ones). **Please also take a look at other tabs such as the Active ontology.**
*

Please choose the appropriate response for each item:

	1	2	3	4	5
Rate how well informed you are about the impact the previous changes had on the ontology as a whole (with 1 being not informed at all and 5 being very well informed)	○	○	○	○	○

Write down the view / element on the screen that was most crucial for you for the previous question.
*

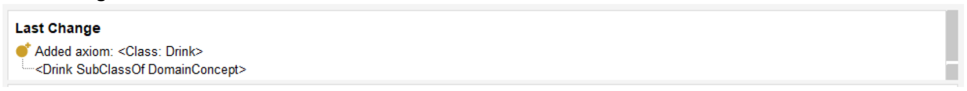
Please write your answer here:

ChImp Plugin Questions

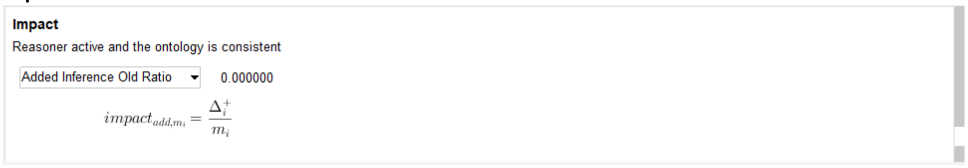
In the previous task you had the Chimp plugin view opened in Protégé. The following you are asked to answer several questions about it.

As a reminder, the plugin contains the following panels:

- **Last Change**


Last Change
Added axiom: <Class: Drink>
<Drink SubClassOf DomainConcept>

- **Impact**



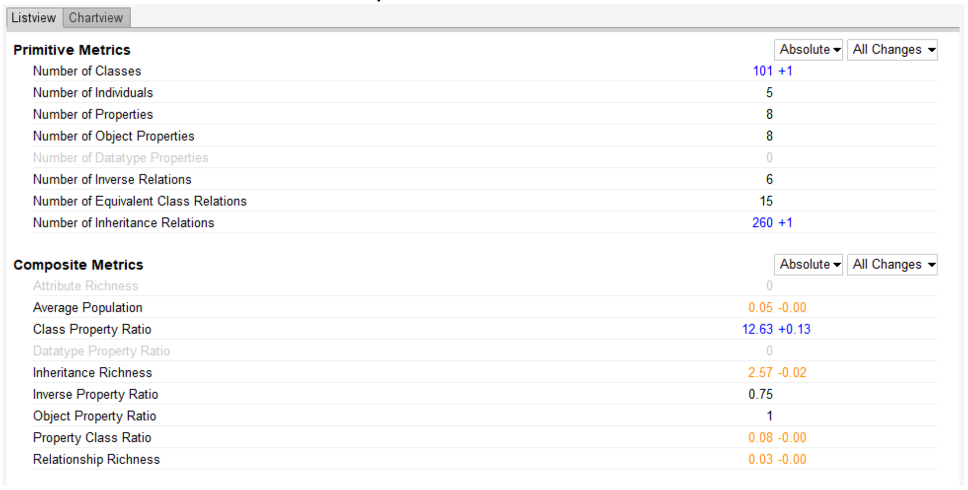
Impact

Reasoner active and the ontology is consistent

Added Inference Old Ratio 0.000000

$$impact_{add,m_i} = \frac{\Delta_i^+}{m_i}$$

- **Standard Metrics which contains the subpanels Listview and Chartview**



Primitive Metrics	Absolute / All Changes
Number of Classes	101 +1
Number of Individuals	5
Number of Properties	8
Number of Object Properties	8
Number of Datatype Properties	0
Number of Inverse Relations	6
Number of Equivalent Class Relations	15
Number of Inheritance Relations	260 +1

Composite Metrics	Absolute / All Changes
Attribute Richness	0
Average Population	0.05 -0.00
Class Property Ratio	12.63 +0.13
Datatype Property Ratio	0
Inheritance Richness	2.57 -0.02
Inverse Property Ratio	0.75
Object Property Ratio	1
Property Class Ratio	0.08 -0.00
Relationship Richness	0.03 -0.00

Content *

Please choose the appropriate response for each item:

	1	2	3	4	5
Rate how informative the Last Change panel was for you (with 1 being not informative at all and 5 being very informative)	○	○	○	○	○
Rate how informative the Impact panel was for you (with 1 being not informative at all and 5 being very informative)	○	○	○	○	○
Rate how informative the Listview panel was for you (with 1 being not informative at all and 5 being very informative)	○	○	○	○	○
Rate how informative the Chartview panel was for you (with 1 being not informative at all and 5 being very informative)	○	○	○	○	○
Rate how informative the whole plugin was for you (with 1 being not informative at all and 5 being very informative)	○	○	○	○	○

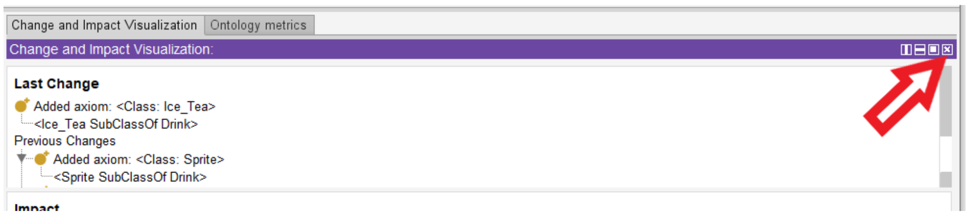
Visualization *

Please choose the appropriate response for each item:

	1	2	3	4	5
Rate the visualization of the Last Change panel (with 1 being you don't like it at all and 5 being you like it very much)	○	○	○	○	○
Rate the visualization of the Impact panel (with 1 being you don't like it at all and 5 being you like it very much)	○	○	○	○	○
Rate the visualization of the Listview panel (with 1 being you don't like it at all and 5 being you like it very much)	○	○	○	○	○
Rate the visualization of the Chartview panel (with 1 being you don't like it at all and 5 being you like it very much)	○	○	○	○	○
Rate the visualization of the plugin overall (with 1 being you don't like it at all and 5 being you like it very much)	○	○	○	○	○

Close the ChImp plugin

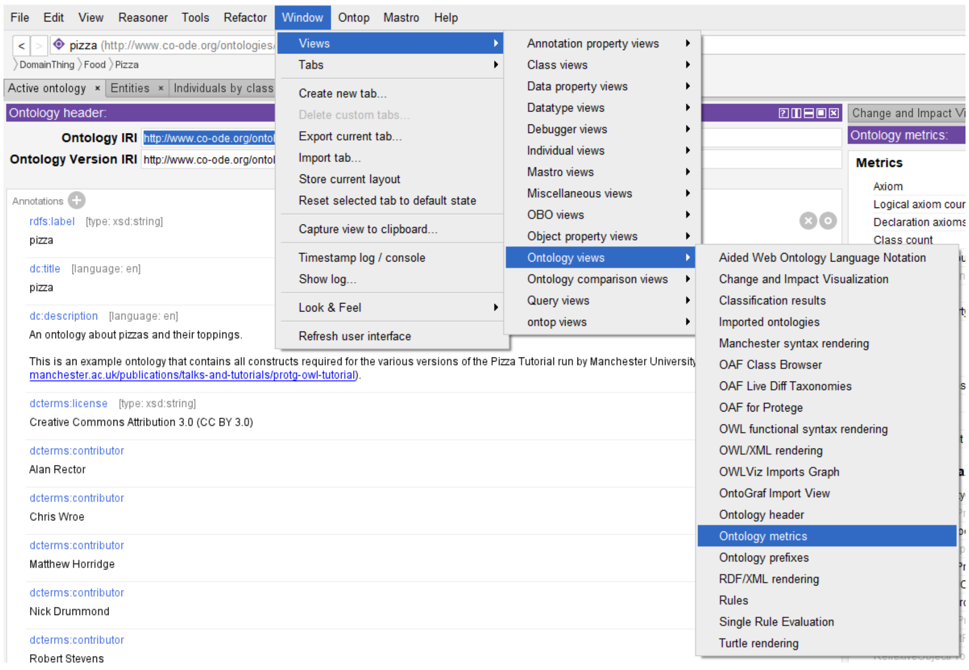
Close the ChImp plugin and do not use it for the rest of the experiment. To do this, click on the close button on the top right of the view:



Change and Impact Visualization
Ontology metrics
Change and Impact Visualization:

Last Change
Added axiom: <Class: Ice_Tea>
<Ice_Tea SubClassOf Drink>
Previous Changes
Added axiom: <Class: Sprite>
<Sprite SubClassOf Drink>
Impact

Make sure that you have the Ontology metrics view open in its stead. If you closed it, open it again by selecting:



Ontology IRI http://www.co-ode.org/ontol
Ontology Version IRI http://www.co-ode.org/ontol

Annotations
rdfs:label [type: xsd:string]
pizza
dc:title [language: en]
pizza
dc:description [language: en]
An ontology about pizzas and their toppings.

This is an example ontology that contains all constructs required for the various versions of the Pizza Tutorial run by Manchester University
manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial).
dcterms:license [type: xsd:string]
Creative Commons Attribution 3.0 (CC BY 3.0)
dcterms:contributor
Alan Rector
dcterms:contributor
Chris Wroe
dcterms:contributor
Matthew Horridge
dcterms:contributor
Nick Drummond
dcterms:contributor
Robert Stevens

If the plugin shows up again when you open a new window, close the plugin view and continue with the task. Do not use it for the next task.

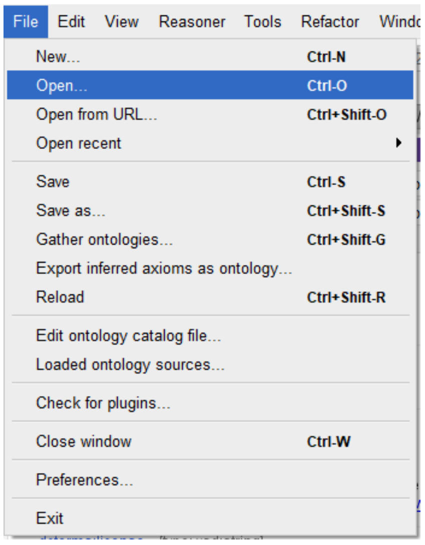
Task Deletion

Your task is to alter the pizza ontology. Please ensure that you have it open in Protégé. Perform all of the steps listed below.

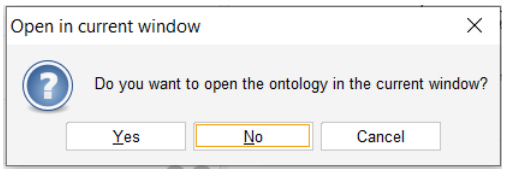
Open the Pizza ontology.

Please open a new pizza ontology.

Open the pizza.owl file by selecting File and Open... and then choosing the file on your computer.



If the following dialog pops up, make sure to click **No**.

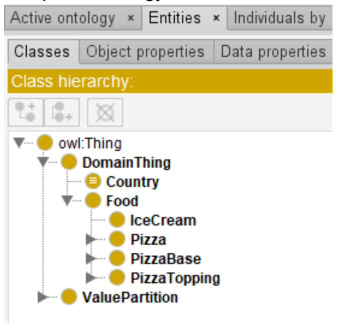


Open in current window

Do you want to open the ontology in the current window?

First Part

Open up the Class hierarchy view on the tab Entities.
The pizza ontology contains the class "Food" with several subclasses:



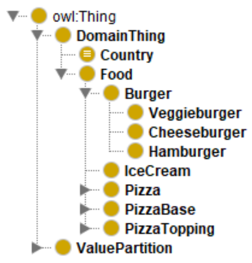
- owl:Thing
  - DomainThing
    - Country
    - Food
      - IceCream
      - Pizza
      - PizzaBase
      - PizzaTopping
  - ValuePartition

Add an additonal class "Burger" to the ontology. "Burger" is a type of "Food" and therefore on the same level as e.g. "PizzaTopping". It has the subclasses "Cheeseburger", "Hamburger" and "Veggieburger".

Hint:

1. Rightclick on "Food" and select Add subclass...
2. Write "Burger" into the name field and press Enter.
3. Rightclick on "Burger" and select Add subclass...
4. Write "Hamburger" into the name field and press Enter.
5. repeat for the other two drinks

After this process, you should end up with the following hierarchy:



- owl:Thing
  - DomainThing
    - Country
    - Food
      - Burger
        - Veggieburger
        - Cheeseburger
        - Hamburger
      - IceCream
      - Pizza
      - PizzaBase
      - PizzaTopping
  - ValuePartition

The experiment at hand revolves around the impact your changes have on the ontology as a whole. Now for the evaluation, please take a look at all the views that are currently open in Protege (do not open new ones). **Please also take a look at other tabs such as the Active ontology.**
*

Please choose the appropriate response for each item:

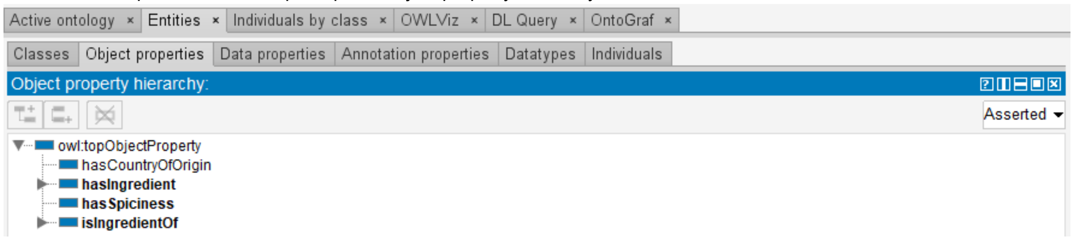
	1	2	3	4	5
Rate how well informed you are about the impact the previous changes had on the ontology as a whole (with 1 being not informed at all and 5 being very well informed)	○	○	○	○	○

Write down the view / element on the screen that was most crucial for you for the previous question.
*

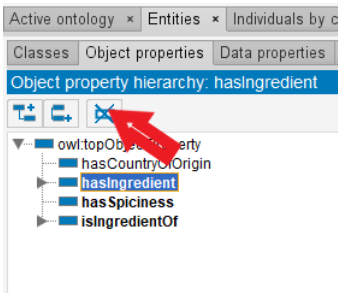
Please write your answer here:

Second Part

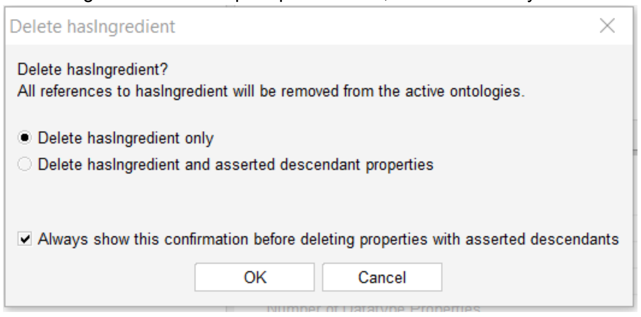
For the second part of the task, open up the Object property hierarchy in the Entities tab:



Your task now is to delete the "hasIngredient" object property. You do this by clicking on the rightmost button of the view:



You will again see a delete prompt. This time, make sure that you have Delete hasIngredient only selected:



Delete hasIngredient

Delete hasIngredient?
All references to hasIngredient will be removed from the active ontologies.

● Delete hasIngredient only
○ Delete hasIngredient and asserted descendant properties

✓ Always show this confirmation before deleting properties with asserted descendants

OK Cancel

Once you click on OK, you should not be able to see the property anymore.

Now again to evaluate the impact the last change had on the ontology as a whole, please take a look at all the views that are currently open in Protege (do not open new ones). **Please also take a look at other tabs such as the Active ontology.**
*

Please choose the appropriate response for each item:

	1	2	3	4	5
Rate how well informed you are about the impact the previous changes had on the ontology as a whole (with 1 being not informed at all and 5 being very well informed)	○	○	○	○	○

Write down the view / element on the screen that was most crucial for you for the previous question.

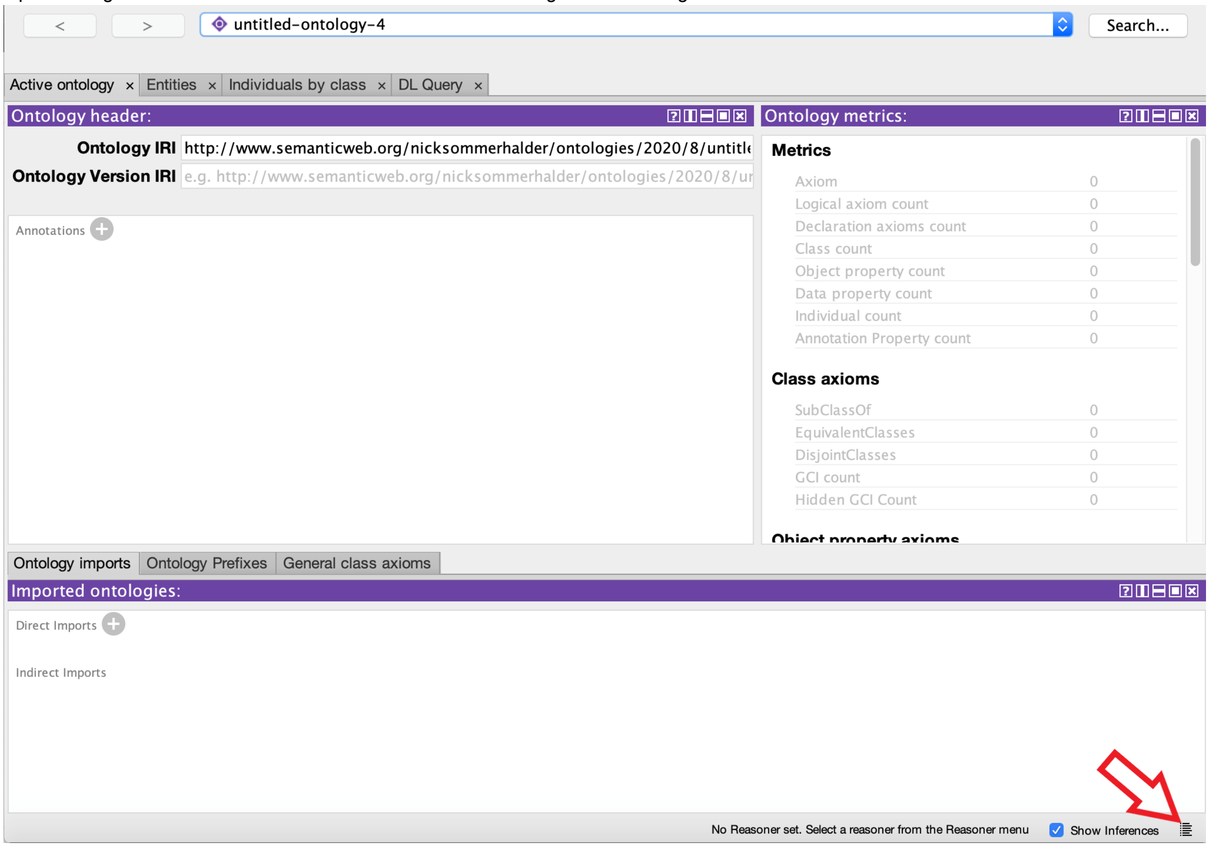
*

Please write your answer here:

Logfile Upload

To complete the experiment, please upload the Protégé log file. It contains the actions you performed inside Protégé during the experiment.

1. Open the log file view as follows with the button on the bottom right of the Protégé window:



2. Then click on Show log file, which should then open the location of the log file in your file explorer. **Copy the file to your desktop.**

3. Upload the file with the button below.

Upload a file

☐ Please upload at most one file

Kindly attach the aforementioned documents along with the survey

select the log file

Closing Questions

General questions about the ChImp plugin *

Please choose the appropriate response for each item:

	1	2	3	4	5
How confident do you feel that your assessment of the usefulness of the plugin would hold up in a real life use case? (with 1 being not confident at all and 5 being very confident)	○	○	○	○	○
How did you like the ChImp plugin? (with 1 being not at all and 5 being very much)	○	○	○	○	○

If you have any comments about what you liked or disliked about the ChImp plugin, please write them down here.

Please write your answer here:

General questions about the experiment *

Please choose the appropriate response for each item:

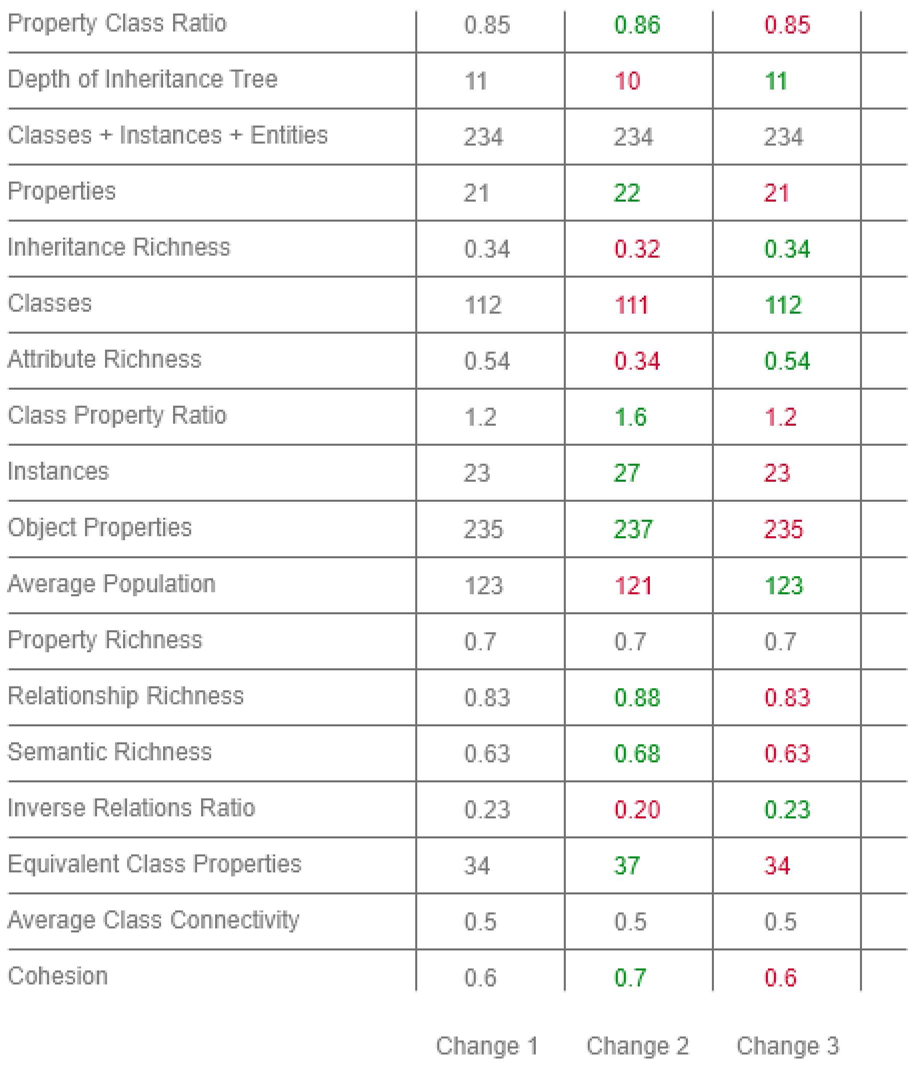
	1	2	3	4	5
How did you like the study overall? (1 being not at all and 5 being very much)	○	○	○	○	○
The difficulty level for me was (1 being too easy and 5 being too difficult)	○	○	○	○	○

If you have any other comments about the experiment in general, please write them down here.

Please write your answer here:

Thank you very much for participating!

Submit your survey.
Thank you for completing this survey.



	Change 1	Change 2	Change 3
Property Class Ratio	0.85	0.86	0.85
Depth of Inheritance Tree	11	10	11
Classes + Instances + Entities	234	234	234
Properties	21	22	21
Inheritance Richness	0.34	0.32	0.34
Classes	112	111	112
Attribute Richness	0.54	0.34	0.54
Class Property Ratio	1.2	1.6	1.2
Instances	23	27	23
Object Properties	235	237	235
Average Population	123	121	123
Property Richness	0.7	0.7	0.7
Relationship Richness	0.83	0.88	0.83
Semantic Richness	0.63	0.68	0.63
Inverse Relations Ratio	0.23	0.20	0.23
Equivalent Class Properties	34	37	34
Average Class Connectivity	0.5	0.5	0.5
Cohesion	0.6	0.7	0.6

Figure A.4: Prototype 4

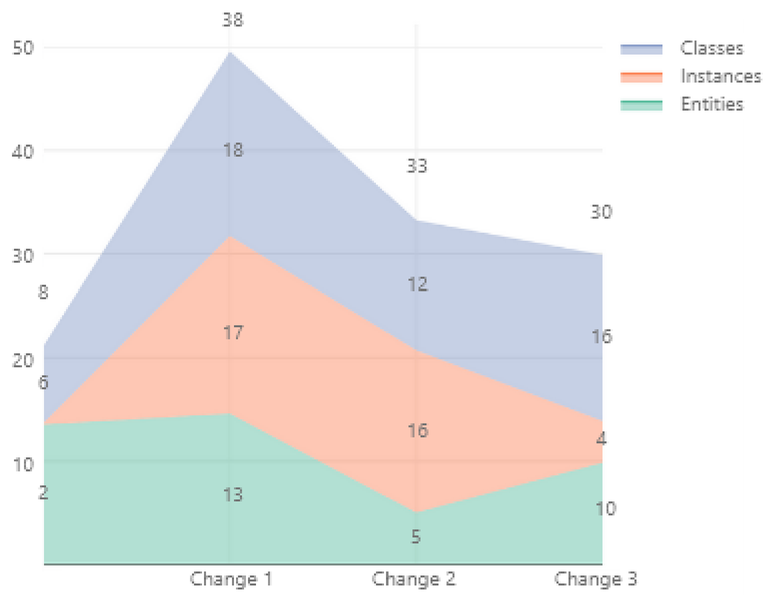


Figure A.5: Prototype 5

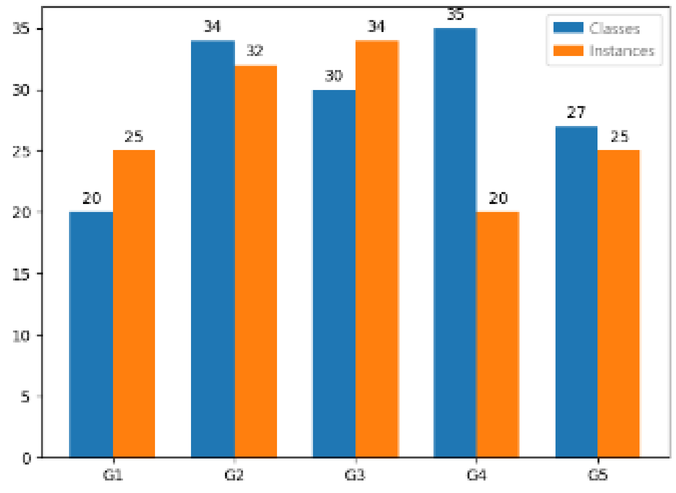
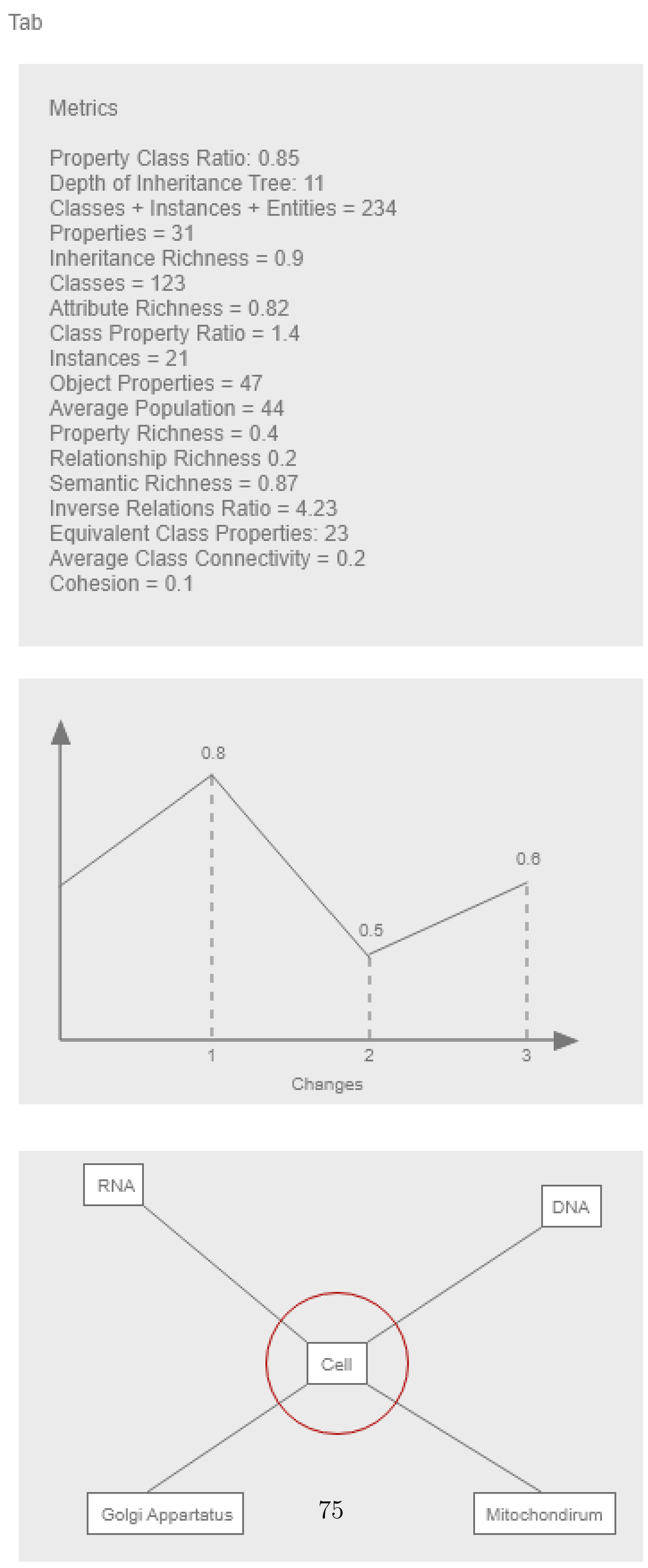


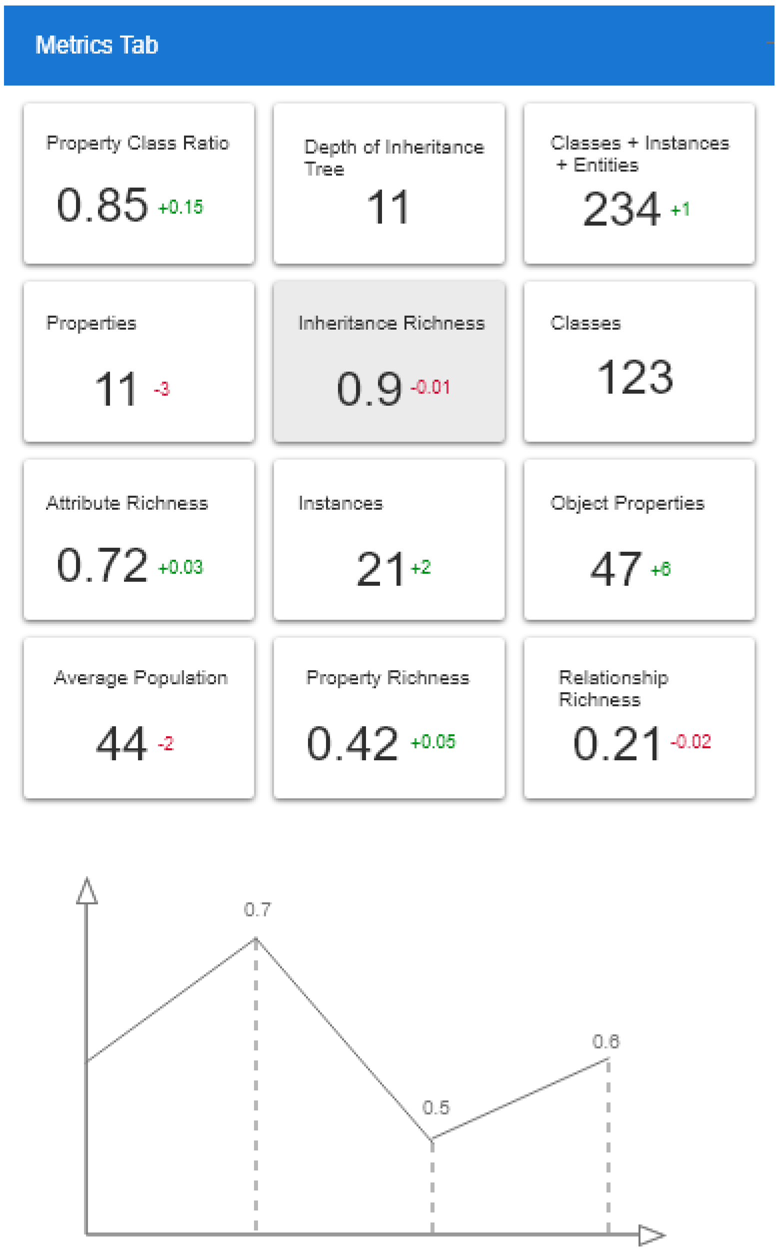
Figure A.6: Prototype 6


Tab

Metrics

Property Class Ratio: 0.85
Depth of Inheritance Tree: 11
Classes + Instances + Entities = 234
Properties = 31
Inheritance Richness = 0.9
Classes = 123
Attribute Richness = 0.82
Class Property Ratio = 1.4
Instances = 21
Object Properties = 47
Average Population = 44
Property Richness = 0.4
Relationship Richness 0.2
Semantic Richness = 0.87
Inverse Relations Ratio = 4.23
Equivalent Class Properties: 23
Average Class Connectivity = 0.2
Cohesion = 0.1

Figure A.7: Prototype 7



Metrics Tab

Property Class Ratio 0.85 +0.15

Depth of Inheritance Tree 11

Classes + Instances + Entities 234 +1

Properties 11 -3

Inheritance Richness 0.9 -0.01

Classes 123

Attribute Richness 0.72 +0.03

Instances 21 +2

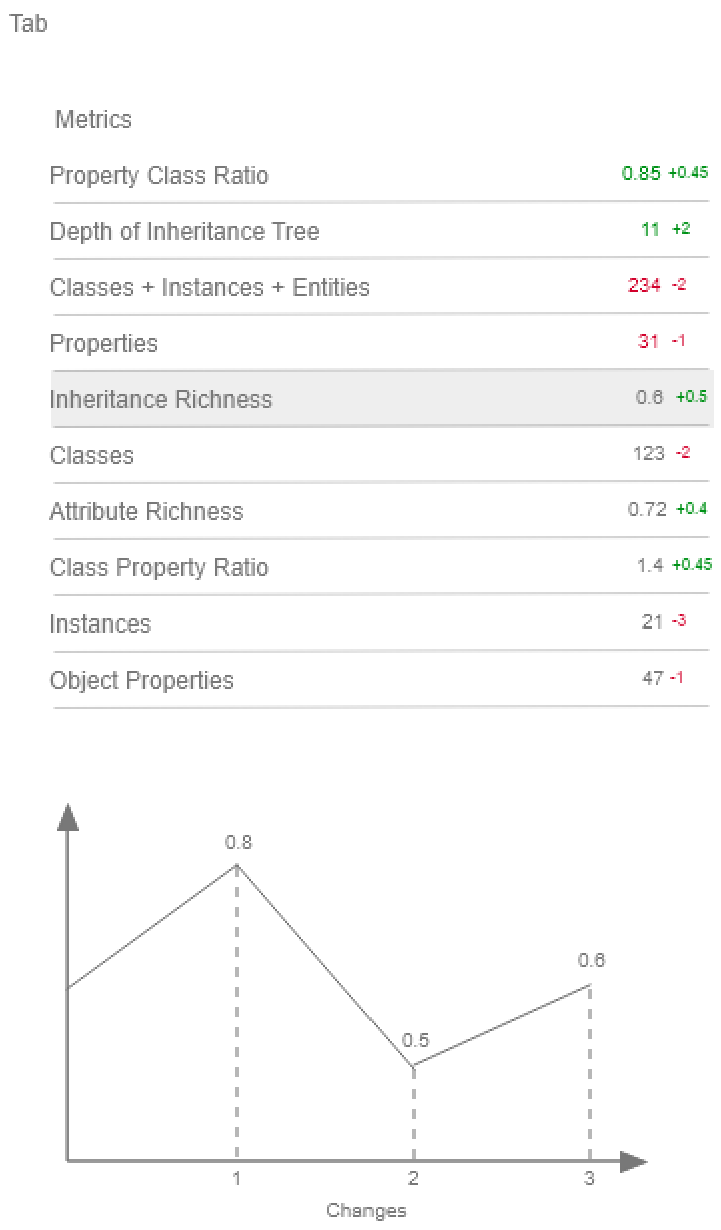
Object Properties 47 +6

Average Population 44 -2

Property Richness 0.42 +0.05

Relationship Richness 0.21 -0.02

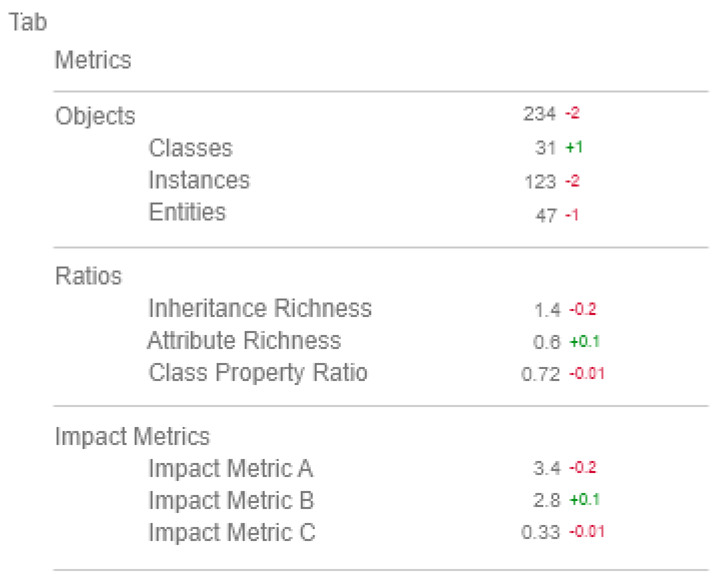
Figure A.8: Prototype 8



Tab

Metrics	
Property Class Ratio	0.85 +0.45
Depth of Inheritance Tree	11 +2
Classes + Instances + Entities	234 -2
Properties	31 -1
Inheritance Richness	0.6 +0.5
Classes	123 -2
Attribute Richness	0.72 +0.4
Class Property Ratio	1.4 +0.45
Instances	21 -3
Object Properties	47 -1

Figure A.9: Prototype 9

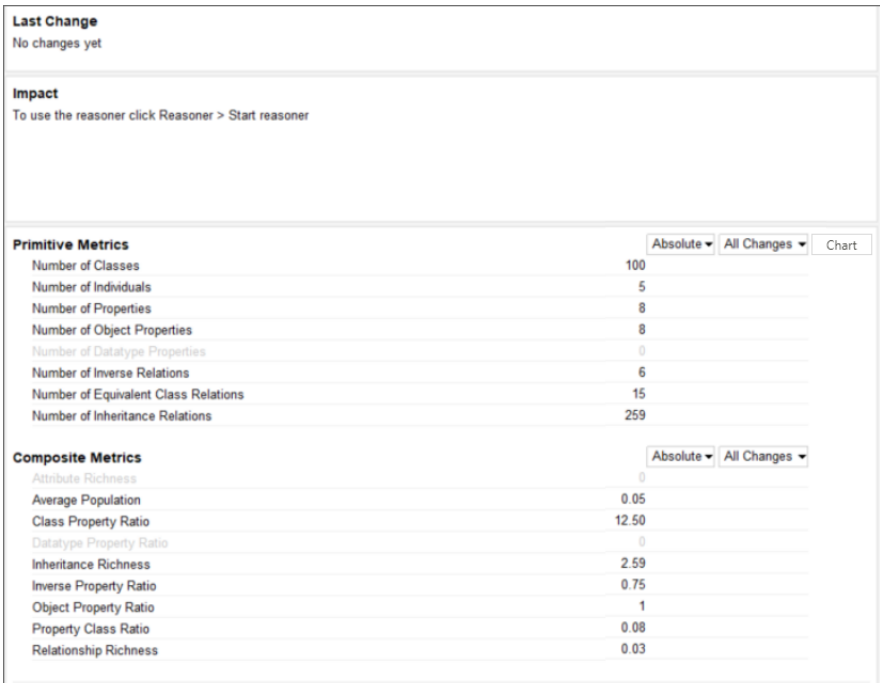


Tab

Metrics

Objects	234 -2
Classes	31 +1
Instances	123 -2
Entities	47 -1
Ratios	
Inheritance Richness	1.4 -0.2
Attribute Richness	0.6 +0.1
Class Property Ratio	0.72 -0.01
Impact Metrics	
Impact Metric A	3.4 -0.2
Impact Metric B	2.8 +0.1
Impact Metric C	0.33 -0.01

Figure A.10: Prototype 10



Last Change
No changes yet

Impact
To use the reasoner click Reasoner > Start reasoner

Primitive Metrics Absolute ▾ All Changes ▾ Chart

Number of Classes	100
Number of Individuals	5
Number of Properties	8
Number of Object Properties	8
Number of Datatype Properties	0
Number of Inverse Relations	6
Number of Equivalent Class Relations	15
Number of Inheritance Relations	259

Composite Metrics Absolute ▾ All Changes ▾

Attribute Richness	0
Average Population	0.05
Class Property Ratio	12.50
Datatype Property Ratio	0
Inheritance Richness	2.59
Inverse Property Ratio	0.75
Object Property Ratio	1
Property Class Ratio	0.08
Relationship Richness	0.03

Figure A.11: Chart Prototype 1

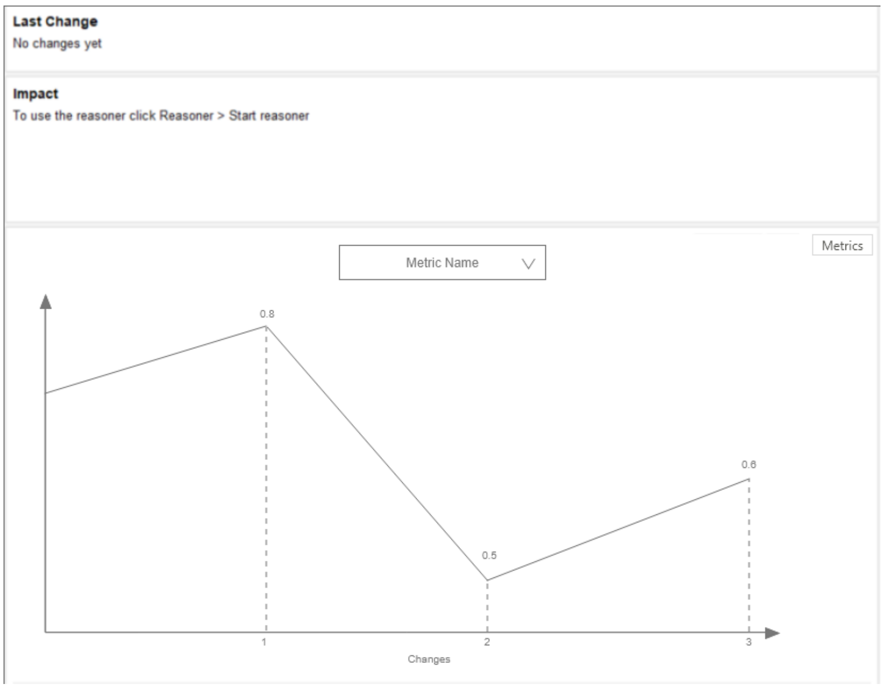
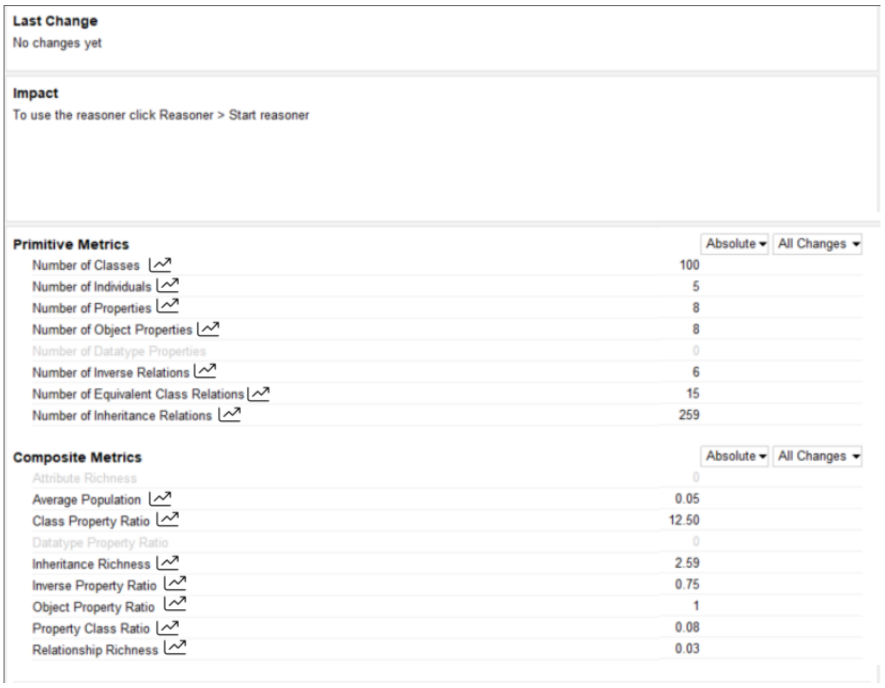


Figure A.12: Chart Prototype 2



Last Change
No changes yet

Impact
To use the reasoner click Reasoner > Start reasoner

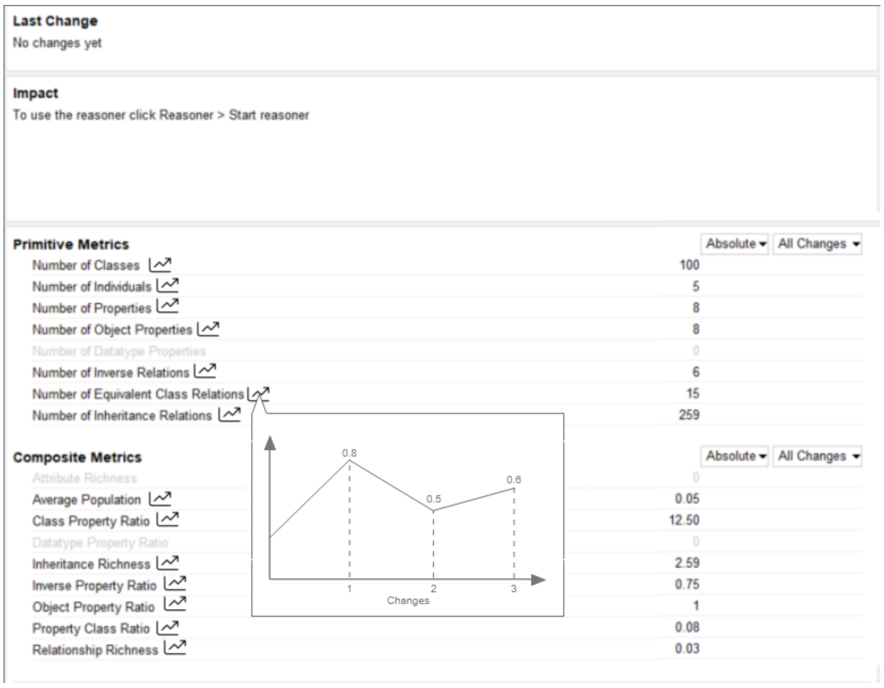
Primitive Metrics Absolute All Changes

Number of Classes	100
Number of Individuals	5
Number of Properties	8
Number of Object Properties	8
Number of Datatype Properties	0
Number of Inverse Relations	6
Number of Equivalent Class Relations	15
Number of Inheritance Relations	259

Composite Metrics Absolute All Changes

Attribute Richness	0
Average Population	0.05
Class Property Ratio	12.50
Datatype Property Ratio	0
Inheritance Richness	2.59
Inverse Property Ratio	0.75
Object Property Ratio	1
Property Class Ratio	0.08
Relationship Richness	0.03

Figure A.13: Chart Prototype 3



Last Change
No changes yet

Impact
To use the reasoner click Reasoner > Start reasoner

Primitive Metrics

Absolute ▾ All Changes ▾

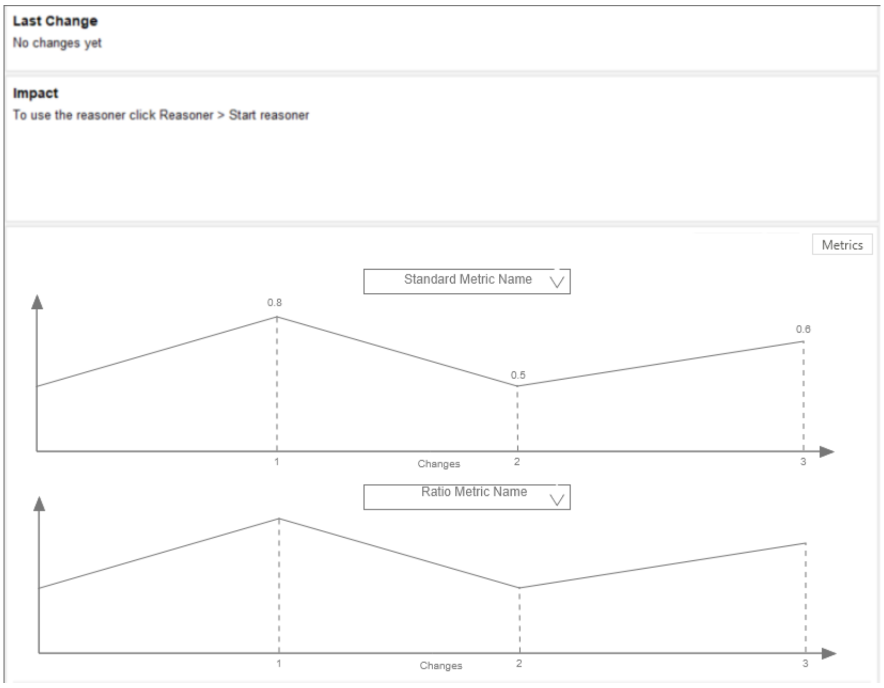
Number of Classes	100
Number of Individuals	5
Number of Properties	8
Number of Object Properties	8
Number of Datatype Properties	0
Number of Inverse Relations	6
Number of Equivalent Class Relations	15
Number of Inheritance Relations	259

Composite Metrics

Absolute ▾ All Changes ▾

Attribute Richness	0
Average Population	0.05
Class Property Ratio	12.50
Datatype Property Ratio	0
Inheritance Richness	2.59
Inverse Property Ratio	0.75
Object Property Ratio	1
Property Class Ratio	0.08
Relationship Richness	0.03

Figure A.14: Chart Prototype 4



Last Change
No changes yet

Impact
To use the reasoner click Reasoner > Start reasoner

Figure A.15: Chart Prototype 5

List of Figures

List of Tables