TEKNISKA HÖGSKOLAN

HÖGSKOLAN I JÖNKÖPING

Content Ontology Design Pattern Presentation

Sheheryar Lodhi

Zaheer Ahmed

MASTER THESIS 2011

INFORMATICS

Presentation av designmönster för ontologier

Sheheryar Lodhi

Zaheer Ahmed

Detta examensarbete är utfört vid Tekniska Högskolan i Jönköping inom ämnesområdet informatik. Arbetet är ett led i masterutbildningen med inriktning informationsteknik och management. Författarna svarar själva för framförda åsikter, slutsatser och resultat.

Handledare: Eva Blomqvist
Examinator: Vladimir Tarasov

Omfattning: 30 hp (D-nivå)

Datum: 2011- 08- 05

Arkiveringsnummer:

Abstract

Ontology design patterns are used for creating quality modeling solutions for ontologies. The presentation of ontology design patterns is concerned with reusability of ontologies from a user perspective.

The purpose of this research is to identify improvement areas in the presentation of content ontology design patterns. The objective is to analyze different content ontology design patterns and provide suggestions for possible changes in current templates and pattern presentation.

The ontology design pattern templates were compared with existing templates of other patterns to identify improvement areas. After this, two surveys were conducted with novice users and expert ontology engineers to improve the readability and usability of content ontology design patterns from the user perspective and to discover differences in opinion while using the patterns. Based on the findings of comparison and survey results, we proposed suggestions to improve the current template and presentation of content ontology design patterns.

Sammanfattning

Designmönster för ontologier används för att skapa modelleringslösningar av hög kvalitet för ontologier. Presentationen av designmönstren möjliggör förståelse och enkel återanvändning av designmönstren från ett användarperspektiv. Syftet med detta examensarbete är att identifiera förbättringsområden i presentationen av mönster för ontologidesign.

Målet har varit att analysera olika designmönster för ontologier och ge förslag på eventuella förändringar i aktuella mallar och presentationen av mönstren. Mallar för designmönster för ontologier jämförs med befintliga mallar för designmönster inom andra områden, för att analysera skillnader mellan dem. Efter detta genomfördes två undersökningar, dels med nybörjare och dels med expertanvändare, för att upptäcka om det finns några skillnader mellan behov av information hos noviser och expertanvändare, angående deras användning av mönster för ontologidesign. Baserat på resultaten från jämförelsen och undersökningarna föreslår vi ett antal förbättringar i den aktuella mallen och hur designmönster för ontologier bör presenteras.

Acknowledgements

Key words

Ontology Design Pattern, Software Engineering Pattern, Content Pattern, Web Ontology Language.

Contents

List of Figures

List of Tables

List of Abbreviations

No	Abbreviations	Meaning
1	ODP	Ontology Design Pattern
2	OWL	Web Ontology Language
3	RDF	Resource Description Framework
4	OIL	Ontology Inference Layer
5	XD	eXtreme Design
6	ERM	Entity-Relationship Model
7	XML	Extensible Markup Language
8	XOL	Ontology Exchange Language
9	GUI	Graphical User Interface

1 Introduction

Ontology design patterns (ODPs) are semantic patterns that are used for creating quality modeling solutions for ontologies. ODPs play an important role in learning and teaching ontology engineering. They facilitate the automatic and semi-automatic ontologies construction and provide a base for creating ontologies in different domains. The presentation of ODP is concerned with reusability of ontology from user perspective. So far only a small catalogue of patterns exist which is available online at the ontology design pattern portal. In this portal, ODPs are described using a template with a set of headings that should be filled out when entering a new pattern. The template defines a standard way for constructing new patterns. There are possibilities to discuss modeling issues and review and suggest changes in patterns.

1.1 Background

In computer and information science ontology is defined as a "formal, explicit specification of a shared conceptualization" [1]. One of the main problem areas is reusability of ontologies. The existing ontologies are available at online ontology repositories which provide guidelines to ontology users. The existing ontologies provide limited assistance in using unfamiliar logical structures. Good practices must be discovered from literature. This problem is solved by implementing common solution as we learn in software engineering [2]. The patterns facilitate and to some extent automate the construction of ontologies. The development of patterns in the ontology field is very popular as that in software engineering. The patterns are defined for reuse and aim at facilitating the construction process very much like the way it is done in software engineering or architectural planning of buildings [3]. The purpose of design patterns is to solve the design problems. The patterns provide a useful way for handling the problems of reusability in a construction setting. In software engineering the common way to build software is to use design and architecture patterns. This also becomes true in ontology engineering [4]. Ontology design patterns provide modeling solutions of ontologies design problems. They provide a base for creating ontologies in different domains. Patterns are also used for evaluation of ontologies [2]. Ontology design patterns (ODPs) are of several types. They are divided into 6 families; Content patterns, Structural Patterns, Presentation patterns, Correspondence Patterns, Lexico-Syntactic Patterns and Reasoning Patterns [5].

This thesis deals with the presentation of content ontology design pattern. It describes the design issues of the presentation of ontology design patterns.

1.2 Purpose and Objectives

The purpose of this research is to identify improvement areas in the presentation of content ODPs. Improvement in presentation can ultimately improve the understandability of a pattern from user perspective. Our objective is to analyze different content ODPs and provide suggestions for possible changes in current templates and pattern presentation. It also includes determining the most important information about patterns which can help an ontology engineer in selecting an appropriate pattern.

Presentation of design patterns is related to issues such as reuse, guidance and communication. Our main goal is to evaluate the current patterns presentation. The evaluation is focused on the analysis of current patterns. We conduct a survey with experts and collect their suggestions to improve the readability and usability of ontology design patterns.

1.3 Research Questions

In this research, we answer the following research questions related to ontology design patterns:

1. What is the most important information, present in current templates, for understanding and reusing content ontology design patterns?

2. Is there a difference, with respect to question 1, between novice and expert ontology engineers?

3. What is missing in the current template for content ontology design patterns?

4. How can the current template and the presentation of content ontology design patterns be improved?

The template of an ontology design pattern consists of many parts, the first question is to identify the most important and vital information concerning the design patterns. This information would help an ontology engineer to select an appropriate design pattern for the required ontology. The second question is about the users who work with ontology design patterns. Generally, users are divided into two categories; novice and expert ontology engineers. Novice users are the end-users who use design patterns to implement in the ontologies. Expert ontology engineers are those who actually develop ontology design patterns. Each category of user has its own information requirement regarding design patterns.

There may be certain information that an ontology user need to understand a pattern but it is not available in the description, our next task will be to examine the missing information in the current ontology design pattern templates. Finally, based on the results of above questions, we will present suggestions for the possible improvement in the current templates and design pattern presentation.

1.4 Method

Here is the summary of the research methodology of our thesis. Refer to Chapter 3 for more details. The work starts with a literature review. First, we analyze what current templates exist in other fields and then compare them to ontology design pattern templates to analyze the difference. In the next phase, we conduct two different types of online surveys. The first online survey was conducted with novice users. They were asked to provide their opinion by responding to a structured sequence of questions. The second online survey was conducted with experts. This survey was posted on the ontologydesignpatterns.org website. At the end, results of the comparison of different patterns and the survey results are evaluated.

1.5 Limitations

The ontology design patterns (ODPs) have been grouped into six different families: Structural ODPs, Content ODPs, Reasoning ODPs, Presentation ODPs, Correspondence ODPs and Lexico-Syntactic ODPs. In this thesis, our focus of research will be the Content ODPs. The comparison of ODPs is limited to software patterns and data model patterns because these are the most used and well known patterns. There are many ontology languages available for development but we will only focus on OWL ontologies.

The respondents of the surveys are divided into two groups; novice and expert ontology engineers. The first survey is conducted with novice users which have some experience of working with ontology design patterns. The second survey is conducted online and the respondents are the members of the quality committee of the ontologydesignpatterns.org website.

1.6 Thesis outline

Chapter 2 provides a detailed background of ontologies and ontology design patterns. Chapter 3 includes the research methodology for the thesis. In Chapter 4, results gathered from the two surveys are presented. The analysis of the survey results and the comparison of different patterns are included in Chapter 5. Finally, the conclusions of the work are presented in Chapter 6.

2 Theoretical Background

This chapter covers the major concepts and theories related to our research topic. Before presenting the related work, this chapter first includes definitions and descriptions of basic concepts. As the purpose ODPs is to facilitate the development of ontologies, the major concepts related to ontologies are presented in section 2.1. The process of ontology engineering and a list of ontology languages and tools for ontology development are explained in this section. Section 2.2 presents the background of knowledge reuse with focus on ontology reuse. In Section 2.3, software and data model patterns are described in detail. These patterns are compared with content ODPs in chapter 6. Section 2.4 presents ontology patterns in general and ontology design patterns in particular. Finally in section 2.5, the template of content ODP is explained as it is the focus of our research.

2.1 Ontology

Ontology, in philosophy, refers to a specific "system of categories accounting" which provide a certain view of the world. This specific system is not dependent on a particular language. For instance, Aristotle's ontology, which does not depend on the language that has been used to describe it, has always been the same. On the contrary, ontology, as used in AI, refers to an engineering artifact. These engineering artifacts are made up of a vocabulary (which is specific in nature and which describes a certain reality) and a set of explicit assumptions (where these assumptions relate to the intended meaning of the vocabulary words). The assumptions referred to above are, usually, of first-order logical theory form in which vocabulary words are represented in unary or binary predicate names and are called concepts and relations respectively. Ontology, in the simple cases, explains a conceptual hierarchy which is related by subsumption relationships. However, in cases that are more sophisticated, we add suitable axioms so that other relationships between concepts can be expressed and their intended interpretation be constrained [6], [7].

A formal definition of ontology is given below:

"An ontology is a formal, explicit specification of a shared conceptualization."[8]

We can understand ontologies as logical theories and, by using algebra, we can express ontologies as:
O = (S, A)
Where
 S = ontological signature for describing the vocabulary, and

A = set of ontological axioms for specifying the intended interpretation of the vocabulary.

Since the structured knowledge about a particular domain is clearly and specifically stated and can be communicated, it is explicit. This knowledge, unlike implicit knowledge, need not to be derived through logical deduction [9].

2.1.1 Relation to Other Models

Individual designations, rather than software classes, are modeled in an ontology. Ontologies are similar to database schemas in concepts. For instance, ontology classes are analogous to the entities of an entity-relationship model (ERM). Similarly, there is an analogy between the attributes and relationships of an ERM and the relations and properties in most ontology languages. Formalization of constraints as axioms in ontologies provides the machine-readable meaning as the basis for automated reasoning. In order to enable exchange of information between humans and heterogeneous decentralized applications, the meaning of notions is provided by ontologies [9], [10], [11].

2.1.2 Main Concepts of Ontologies

For ontologies formalization and implementation, there exist various knowledge representation formalisms (and corresponding languages). These varied knowledge representation formalisms furnish distinct components which facilitate these tasks. However, all of them have the following common minimal set of components [12].

Class: Broadly speaking, classes symbolize concepts. An example would be the travelling domain in which locations (cities, villages, etc.), lodgings (hotels, camping, etc.) and means of transport (planes, trains, cars, ferries, motorbikes and ships) can represent concepts. In ontologies, application of inheritance mechanisms can be done by usually organizing classes in taxonomies. Taxonomies of entertainment places can be represented by theaters, cinemas, concerts, etc and those of travel packages by economy travel, business travel, etc. In the paradigm of frame-based knowledge representation, we can also define metaclasses. Those classes whose instances are classes represent metaclasses. Because metaclasses can set up varied layers of classes, they usually provide gradations of meaning since where the classes are defined in the ontology they establish layers of classes [12].

Relations: Associations between concepts of the domain are represented by Relations. A formal definition of relations can state it (i.e. relation) to be "any subset of a product of n sets, that is: R ⊑ C1 × C2 × ... × Cn" [12].

Usually, ontologies consist of binary relations. In a binary relation domain represent the first argument of the relation and range represent the second. A simple example of the above is the arrivalPlace binary relation. Here, the concept travel is the domain of the relation and the range of this relation is the location concept. Knowledge from the domain can instantiate relations. Concept attributes (or slots) are often expressed using the binary relations [12].

Formal Axioms: Sentences that are always true can be modeled using formal axioms. Knowledge, (which other components cannot define formally), can be represented by formal axioms. Moreover, consistency of the ontology itself or of the knowledge stored in a knowledge base can be verified using formal axioms. Inference of new knowledge is facilitated by the use of formal axioms [1].

Instance: In an ontology, objects or individuals are represented by instances [12].

2.1.3   Ontology Levels

The above considerations open up opportunities of developing different kinds of ontologies on the basis of their level of generality, as shown in Fig. 1 below [13].
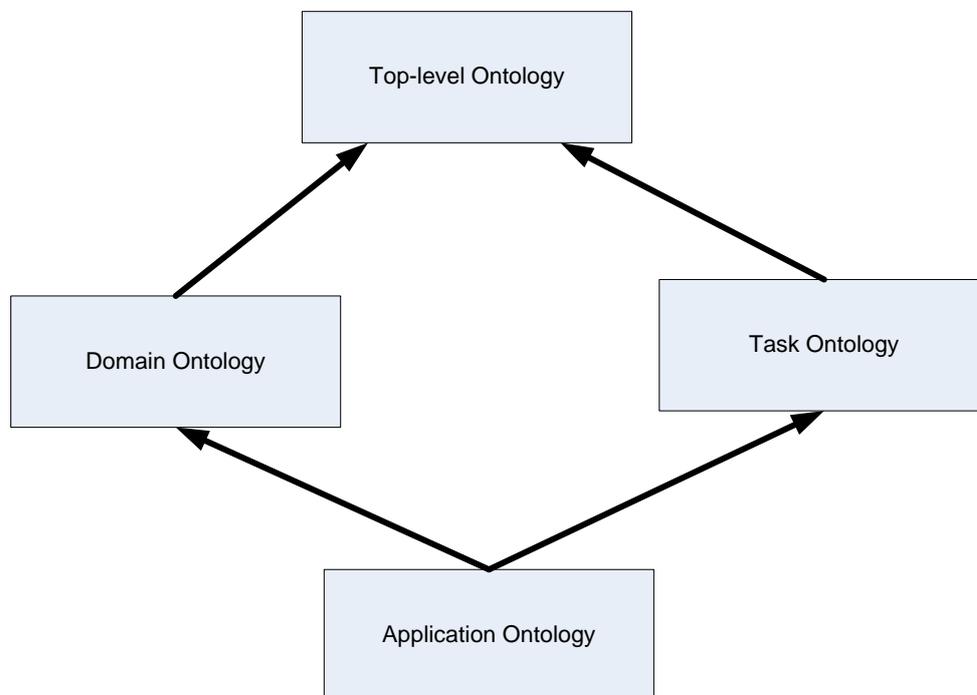
Figure 1- Kinds of Ontologies [13]

1. **Top-level ontologies:** The concepts which have no dependence on a particular problem or domain are described here. For instance, descriptions of very general concepts like space, time, matter, object, event, action, etc, are made in top level ontologies. Therefore there exist unified top-level ontologies for large communities of users [3].

2. **Domain ontologies and Task ontologies:** Specializing the terms introduced in the top-level ontology, these ontologies describe the vocabulary related to a generic domain (like medicine, or automobiles) and the generic tasks or activities (like diagnosing or selling) [3].

3. **Application ontologies:** These ontologies describe those concepts that have dependence on both, the particular domain and task. The application ontologies are often specialized form of the above method ontologies. These concepts, while performing a certain activity, play roles analogous to those played by domain entities, like an extra component [13], [14].

2.1.4　Ontology Language

We can classify the existing logical languages into different groups such as traditional, markup or web-based languages. The development of traditional ontology languages was done in the field of artificial intelligence in the early 1990s [9]. Figure 2 provides an overview.
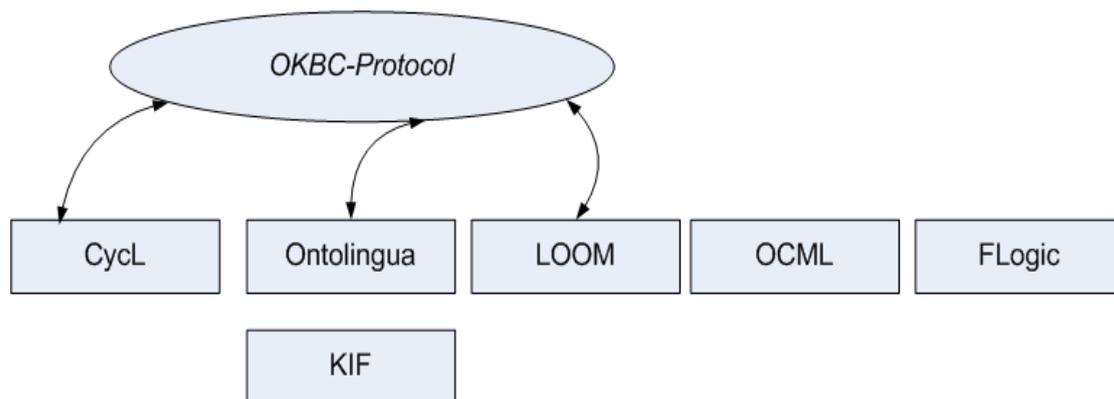
Figure 2: Traditional ontology languages [12]

Traditional Ontology Languages
The first language to be developed on the basis of frames and first order logic was CycL. A frame (which is a data network of nodes and relations) represents a typified situation. Changes can be represented when this situation is adapted and combined with other frame systems. After CycL

came into existence another language called Knowledge Interchange Format (KIF) which is based on first-order logic. Moreover, Ontolingua was developed so that the building of ontologies in KIF could be simplified. LOOM is a language which was developed on description logics. A language having the same style as Ontolingua is the Operational Conceptual Modeling Language (OCML). It was developed in 1990. When frames and first-order logic is combined, a language that comes into existence is FLogic. Knowledge base can be accessed in a standardized fashion using the Open Knowledge Base Connectivity (OKBC) protocol in a way similar to the Open Database Connectivity (ODBC) protocol used in the databases [15], [9].

Web-based Ontology Languages
XML is the basis of development of the syntax of web-based ontology languages. The degree of expressiveness that each of this web-based ontology language displays is the point of difference among them [9]. Figure 3 provides an overview.

OIL	DAML+OIL	OWL

RDF(S)

SHOE	XOL	RDF

XML
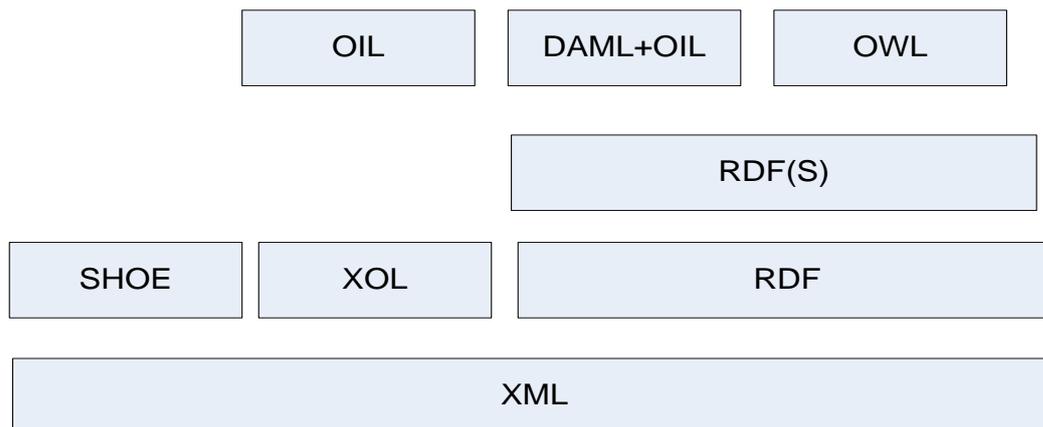
Figure 3: Web-based Ontology Languages [12]

Simple HTML Ontology Extensions (abbr. SHOE) is a frame-and-rule base language that provides a HTML Extension for the semantic annotation of web pages. This language provides opportunities of an HTML-extension. Rules refer to the system behavior specifications (according to pre set assertions). The pre set assertions are predicates indicating the truthfulness of something and can be tested for. The XOL, which is the XML based Ontology Language, is a superset of those language elements which are based on the OKBC protocol. The XOL has been developed to be a platform of exchange for the formal knowledge models in bioinformatics. The ideas of frame based representation languages have influence on XOL [9].

The Resource Description Framework (RDF) is language that has often found to be used in basic ontologies. RDF specifies information about web

resource. Attributes and values describe resources. Links to other resources help in describing resources [9].

The serialization of RDF in XML is most widespread. For representing lightweight ontologies, the Resource Description Framework can be used to serve as a fundamental general-purpose format. The semantics of the RDF elements can be defined by using the RDF Schema (RDF/S) [9].

The representation of classes and subclasses as well as properties for describing relations between information and instances respectively are included in the RFD-Schema. The built-in main meta-classes in RFD-schema are "`rdfs:Class`" and "`rdfs:Property`". The concepts of an ontology can, thus, be represented in this way.

It is, therefore suitable to use RDF/S for classification hierarchies. On the basis of the above, the Defense Advanced Research Projects Agency (the US-agency DARPA) developed the DARPA Agent Markup Language (DAML) as a language of communication for software agents [9].

The web language Ontology Inference Layer (OIL) was developed with the intentions that it should be an ontology language which should have a formal semantics and extensive deduction possibilities. The DAML+OIL provided the bases for developing the Web Ontology Language (OWL) [9].

The Web Ontology Language (OWL) is a semantic markup language for creating ontologies. It allows to describe terms and their relations in such a way that they become machine understandable. Technically, this language builds upon the RDF syntax and also employs the triple model, but it features more powerful possibilities for building expressions than RDF/S. As an extension of RDF and RDF/S, in OWL further language constructs are included, allowing for expressions formulated similarly to those with first-order logic. In order to provide the representation of relations between properties and classes (like equality, cardinality and basic constraints), the Vocabulary (function, Relation and constants) is added. There are three versions in which the Web Ontology Language exists and these three versions are built on top of each other [9].

 The lower and hence less expressive dialect becomes a subset of the dialect higher up. In accordance with the above, OWL Lite and OWL Full are considered respectively to be the least expressive dialect. The definition of constraints to different degrees is facilitated by these dialects [12], [9].

In a new version of the OWL ontology language is known as OWL 2. In OWL 2, scalability requirements are solved by profiles. The OWL 2 QL

profile, one of the subset of language is used to answer query via query rewriting. The OWL- DL language has computational complexity as compared to OWL 2 [16], [17].

2.1.5   Ontology Engineering

The ontological engineering field has been subject to considerable study and research during the last decade. Ontological engineering contains different activities that facilitate the ontology development process. The notion of networked ontological engineering has come into play with the emergence of the Semantic Web, where one of the most relevant assumptions is that ontologies are distributed across different Web servers and ontology repositories and may have overlapping representations of the same or different domains [12]. Ontology engineering can be defined in a formal way as:

"*Ontological engineering refers to the set of activities that concern the ontology development process, the ontology life cycle, the principles, methods and methodologies for building ontologies, and the tool suites and languages that support them.*" [12]

The ontology requirement identification is one of the important activities during ontology development. The Ontology Resource Specification Document (ORSD) contains the description of ontology requirements. The ORSD facilitates several activities which include [20]:

- Finding and reusing the available knowledge resources so that they can be re-engineered into ontologies.
- Finding and reusing the already available ontological resources.
- Verifying the ontology during the ontology development process.

The important question is that how to construct an ontology? There have been a number of suggestions that have been made regarding the methodology and they reflect experiences of people who have built ontologies. Addressing the issues of maintenance and development of ontologies, several methodologies exist. We present below some important ones [18].

TOVE
The main objective of TOVE is to develop a standardized, reusable enterprise data model with characteristics that are defined below [19].

- For enterprise, TOVE gives shared terminologies which are mutually understood and used by every agent.

- TOVE characterizes the meaning of every term in a precise and clearly defined manner as much as possible.
- For automatic deduction of the answer for many common sense questions about the enterprise, TOVE use semantics in a set of axioms.
- For describing a term or concept, TOVE define symbols which are used in graphical context.

Common KADS

For developing a KBS (knowledge base system), common KADS is a one of the popular methodology. Common KADS covers many features of KBS project development. These features are project management, organizational analysis, knowledge acquisition, conceptual modeling, user interaction, system integration and design. Common KADS focuses on result oriented development rather than process oriented development. This method expresses KBS development in two perspectives. These are result perspective and project management perspective. In result perspective, a continuous improvement is done during a project life-cycle on a set of models which consists of different features of KBS. In project management perspective, a generic model covering certain risks is configured into a process according to the needs of a specific project [54].

ONIONS (Ontologie Integration of Naive Source)

The ONIONS (ONtologic Integration of Naive Sources) methodology focuses on the integration of heterogeneous sources of information in the process of knowledge acquisition. To address the issue of heterogeneity, it creates a formal domain ontology to integrate existing repositories of knowledge [18].

DILIGENT

The DILIGENT is a process of collaboratively building a shared ontology. It involves participants of different skill level having interest in collaboratively building or using an ontology. This process involves different kinds of participants including: (1) domain experts, who knows the domain of discourse very well; (2) ontology engineers, having required technical skills for building ontologies; (3) knowledge engineers, having knowledge about building ontology based information systems; (4) finally the users, who will use the ontology in their systems [21].

NeOn

The NeOn methodology is used for developing an ontology network. It is a scenario based methodology that focuses on the several activities such as requirements specification, ontological resource reuse, ontology design pattern reuse, non ontological resource reuse, reengineering, ontology

evaluation and ontology localization. The important aspects of Neon Methodology are described as [22]:

- NeOn methodology purpose a set of nine scenarios for constructing ontologies and ontology networks which focus on the reuse of ontological and non ontological resources, reengineering and merging, collaboration and dynamism.

- It describes a vocabulary of processes and activities which are used for identification and defining the processes and activities. These processes and activities are adopted when ontology networks are collaboratively built by teams.

eXtreme Design(XD) with Ontology Design Patterns

The eXtreme Design (XD) is used to solve ontology development problems by defining an approach, a set of methods and related tools based on the definition and application of ODPs. XD uses the idea of an ontology project [49]. There are two main sets contained in this development project namely:

1. The *problem space*, also known as the *local problems*, consists of real modeling problems that have to be solved during the project [49].

2. The *solution space*, constituted by reusable modeling solutions [49].

Several environments for ontology development have been created using the methodologies and languages. There are two groups into which ontologies tools can be classified. The first category of tools has a knowledge model which can be straightaway mapped to an ontology language. These tools are built as ontology editors for a particular language. The second category of tools is integrated tool suites. They offer a flexible architecture and their knowledge models are not dependent on ontology languages [12]. There are several tools for ontology development, we present some of them.

Ontolingua
The Ontolingua language follows the semantics and syntax of KIF. It follows a modular approach to ontology development. Ontolingua provides access to the previously stored library of ontologies but it can also be extended as new ontologies are added to the repository [18].

SWOOP

Swoop is an ontology browser and editor developed to be used with OWL and to help users in communicating with the data and documents in the present web applications [23].

KAON2

KAON2 is a latest development of KAON project after KAON1. The major difference between KAON1and KAON2 is the support of language. KAON1 support the exclusive extension of RDFS and KAON2 support the OWL-DL and F-Logic. The following features are included in KAON2 [24]:

- It provides an API for systematic management of OWL-DL, SWRL and F-logic ontologies.

- By using RMI its stand-alone server provides distributed access to the ontologies.

- A interface engine for processing the connected queries

- It provides a DIG interface (a standardized XML interface) which facilitates access from other tools.

- It contains a module which helps in importing ontology instance from relational databases.

OntoEdit

OntoExote is an ontology editor which facilitates ontology development and maintenance. OntoEdit has distinctive features as compared to other ontology tools because it is based on the recent methodology for development and it also provides a comprehensive use of inferencing. During ontology development, the OntoEdit methodology follows three steps. These steps are requirement specification, refinement and evaluation [21].

In first step all the requirements of the conceptual ontology are collected. This phase includes the domain and goal of ontology, design instructions, available knowledge sources, possible users, use case and ontology supported applications. The outcome of this step is a semi-formal description of the ontology. In refinement phase the semi-formal description is enhance and completely formalized into a suitable representation language. The selection of language is based on requirements for the ontology. From this phase we get mature ontology. In third step evaluation examine the target ontology with requirement specification. This phase described the usefulness of developed ontology [21].

TopBraid Composer

TopBraid Composer is a W3C compliant development tool for semantic web ontologies. It helps in developing and editing RDF/OWL files and executing SPARQL queries on them. It also includes several inference engines like OWLIM and Pellet [26].

NeOn Toolkit

NeOn Toolkit provides a complete support to a wide range of networked ontologies. It has an open architecture which consists of various modules to provide extensive ontology modeling options. The underlying platform for NeOn Toolkit is Eclipse, which is a very comprehensive development environment and suits to the modeling paradigm for ontologies [27].

Protégé

The Protégé platform is a light weight and free open source developing environment. Protégé provides a set of tools for user to develop the domain models and Knowledge base applications by using ontologies. Protégé has ability to enhance by adding the plug-in architecture and a java based application programming interface for developing knowledge base tools and application. Ontology is modeled in Protégé platform by the following two ways [28]:

- Protégé-Frames
- Protégé-OWL

Protégé-Frames

The Protégé-Frames editor is based on a fully featured user interface and knowledge server which facilitates users in developing and saving frame-base domain ontologies, updating data input forms and inserting data instance. Protégé-Frames support a knowledge model which is implemented with OKBCP (Open Knowledge Base Connectivity Protocol) described in detail in section 2.1.4 [28].

Protégé-OWL

The Protégé-OWL tool is an enhancement of Protégé in which OWL (web ontology language) is supported. The recent version, Protégé 4 also supports OWL 2. It helps the user in loading and saving OWL ontologies, editing and visualizing classes and properties and examining the ontology by using an OWL reasoner [28].

2.2   Knowledge Reuse

Knowledge reuse is common way to improve the quality of work artifacts. Pattern is a common way to improve reusability. There are other ways to support reusability, i.e. in object oriented programming the program components must be designed for reusability. To achieve reusability, there are set of design techniques that make the object oriented software more reusable. An obstacle for reuse methodologies is the lack of motivation among developers. Before starting the process, the developer needs to establish a reuse library which requires extra efforts. Reuse process is divided in to two steps *Design for reuse* and *Design by reuse*. To facilitate design by reuse, first the design for reuse process must be established [3], [48].

Reusability is applied at different levels. In software engineering, reusability can be applied at following three levels [3]:

1. **Requirements Reuse:** It deals with the models of the domain or the generic model of the requirement domain.
2. **Design Reuse:** It deals with the models, data structures and algorithms.
3. **Software Component Reuse:** It deals with reuse of software classes and editable source code.

2.2.1   Ontology Reuse

The construction of ontology is time consuming and labor intensive. Reuse of an ontology from an ontological engineering perspective can be hard. This is even more when there are large ontologies to be reused [29]. According to the first viewpoint, most of the ontologies cannot at all be reused for any other application because of its high domain and application specificity. Hence, the development methodology of ontologies does not constitute the integration of existing ontologies. The second viewpoint, on the other hand presents the view that it is impossible to do ontological engineering from scratch. Thus the process of constructing an ontology involves the combination of ontological parts and other knowledge sources that already exist. Almost all the ontological scholars like to place themselves in a midway position of this scale so that they are able to find those ontologies that already exist and then make a 'fit' with the application case. This not only facilitates the combination, extension and adaptation of ontologies but also helps in building various parts of the ontology from the scratch [3].

Ontology library: For ontologies to be grouped and organized so that they may be reused further and for ontology integration, maintenance, mapping

and versioning, an important tool known as ontology library systems was developed. These systems must fulfill all ontological reuse needs and must be easily accessible. Also this library system can facilitate the reuse of existing ontologies and apply standards on them with the help of upper-level ontologies and ontology representation languages. An ontology library system should, therefore, contain a functional infrastructure so that it is able to provide storage and maintenance for ontologies, and provide an adaptation environment that is uncomplicated and facilitate addition, search and reasoning with ontologies [30].

Ontology Matching: Ontology matching is a process of finding correspondence between semantically related ontologies, solving the problem of semantic heterogeneity and can be used in ontology merging, query answering, data translation etc. Thus ontology matching enables interoperability between matched ontologies [9].

To facilitate such discoveries of correspondences, finding similarities and subclasses existing ontologies can be used. In addition external sources of information such as dictionaries, global ontologies, previously performed matchings and also user input can be used to support matching process. These supporting ontologies are (TBox) ontologies in the sense of knowledge collection and can act as a reference ontolog for information integration [9]. In [3], many approaches for ontology matching are presented. These approaches mostly match only some parts of two ontologies but in future these approaches can be applied in ontology reuse to search ontologies for reuse by exploring ontology libraries [3].

2.3 Patterns

A pattern is something re-occurring that can be applied from one time to another and also from one application to another. These concepts are in use in our daily life and also in our professional life. We use old solution as patterns. We search patterns in our surroundings that can be useful. Patterns are used to describe best practices, good designs, and capture experience in the way that is helpful for others to reuse that experience [3], [42].

Patterns can be perceived from two different angles. Knowledge reuse or quick implementation of parts is facilitated by the first kind of pattern. Similar to Software Engineering or architectural planning of a building, a pattern not only facilitates knowledge reusability but also a construction process. Using pattern recognition techniques, patterns of the second type are used in the representation of pattern-like structures that have been found in implementation that was in existence in existing implementations etc. The goal, in the latter case, is the patterns themselves. For instance,

finding out regularities in graph structures or common code usage so that the common code can be made more efficient [3].

When experts work on specific problems, they rarely tackle problems by inventing new solutions that have completely different attributes. In order to solve problem they make use of the essence of previous solutions. The previous solution is recalled and is made use of while solving this new problem. Such expert behavior is also found to be common in other domains. Many kinds of problems or social interactions can be naturally solved using the above method. Patterns evolve when specific problem-solution pairs are abstracted and when the common behaviors are distilled. The problem-solution pairs are categorized into similar problem-solution families [31].

A general definition of pattern is as follow:
"A pattern addresses a recurring design problem that arises in specific design situations, and presents a solution to it." [32]

2.3.1   Software Patterns

Currently in the computer science field the most common and popular patterns are software patterns. Most of the software development projects, where applying functional or object oriented design are conducted using patterns. The patterns are used for increasing reusability, product quality and for managing complexity of the system development process. According to the phase of the development process where they are used these patterns are divided into different kinds [3].
The most common categories are the following:

- Analysis Patterns (see [3], [50]).
- Architecture Patterns (see [3], [31]).
- Design Patterns (see [52], [32],[33],[42] and [51]).
- Programming Language Idioms (see [3], [31]).

Analysis Patterns
Analysis patterns are used to describe the conceptual structures of business processes for the different types of business domains like how to transform these processes into software. An example of an analysis pattern is the account pattern that can be use for the bank account. An analysis pattern is a set of classes and associations that have some meaning in the context of an application; that is, it is a conceptual model of a part of the application. However, the same structure may be valid for other applications, and this is the aspect that makes them very valuable for reuse and composition. This pattern belongs to the semantic analysis

patterns category. This is because it underlines "semantic aspects of the application model" and not the flexibility. This type of patterns is considered to be useful for starting the process of modeling from the requirements. For instance, in the process in which a few patterns in the requirements are identified, an initial model gets produced. This initial model can be applied so as to be used as a guiding post for the remaining design. Also, for developing frameworks and components, these same patterns can be helpful. Analysis patterns are used in the beginning stages of the software engineering process [3] [40], [50].

Architectural Patterns

 An overall structuring principle is used while constructing a viable software architecture. Architectural patterns are considered as templates for solid software architectures. System-wide structural properties of an application are described by the architectural patterns. These architectural patters influence the architecture of sub systems. A fundamental design decision in the development of a software system, therefore, is the process of architectural pattern selection [31]. The Architectural patterns are concerned with the overall structuring principles for software systems, such as how to divide them into sub systems, responsibilities of sub systems and their relations [3].

Design Patterns

 Design patterns provide the description of communicating objects and classes that provide a common solution of general design problem in a particular context. A design pattern indentifies the participating classes and their instances, their roles and interaction and distribution of responsibility. Every design pattern address one particular object oriented design problem and design pattern also provide sample code to illustrate the solution [52].

Software design phase is related to the design patterns, which are widely used [33]. Design patterns do not relate to things like linked lists and hash tables that can be encoded in classes and reused. Design patterns can be seen as describing communicating objects and classes, which, in a particular context, can be optimized to facilitate a general design problem solving [51]. Code is not generally specified by design pattern hence, reusing a design pattern should not be understood as code reuse. Rather, design patterns can be used to create code. Design Patterns which are object-oriented are typically found to be showing relations between classes or objects and they do so without specifying the final application classes or objects that are involved [51].

Programming Idioms

The lowest level of patterns is represented by idioms. The implementation of particular design issue is dealt with by the idioms. The aspects of both, the design and implementation, are treated by them [31]. Programming language idioms are patterns that are language specific and describe methods of implementing specific component aspects. Idioms, by using the specific features of the language, also describe the interactions among these components [3].

2.3.2   Software Pattern Templates

A template of a pattern is a standard way of representing a pattern. In a broad sense, a pattern template has four essential elements. These elements are: Name, Problem, Solution and Consequences [3] [51]. The different kinds of pattern templates are given below with their description and an example.

Design Pattern Template

This template proposed by Gamma, et. al. in their book "Design pattern Element of Reusable Object-Oriented Software" [51]. Table 1 shows the different parts of the template and their description.

Table 1 - Design Pattern Template [51]

Design Pattern Template	
Elements	**Description**
Pattern Name and Classification	Name is a short summary of the pattern. There are many design patterns, we need a way to organize them in a family. The section classification refers these families of design pattern.
Intent	It is a short statement that described the following. What does that design pattern do? What is the main goal of the pattern and what are the particular design issues or problem solve by the pattern.
Also Known As	Another name of the pattern, If the pattern has other name.
Motivation	It has a scenario that describes a design problem and how the class and object structures in the pattern solve this problem. The scenario will facilitate you to understand the more abstract description of the pattern.
Applicability	This section illustrates the situations in which the design pattern may apply.

Structure	It illustrates a detailed specification of the structural aspects of the pattern. It includes a graphical representation of the classes in the pattern using the notation of OMT (Object Modeling Technique). This section also has interaction diagrams to illustrate sequences of requests and collaboration diagram for description of collaboration between objects.
Participants	This section describes the different parts of the pattern and their relation. In design pattern the participants are classes and /or objects.
Collaborations	This section describes how the participants collaborate to carry out their responsibilities.
Implementation	Implementation gives guidelines for implementing the pattern. It gives hints and techniques which one should be aware before implementing the pattern. For example if there are language specific issues.
Sample Code	Sample code is a code fragment that illustrates how you might implement the pattern in a programming language.
Known Uses	Known Uses is the examples of the use of the pattern in real systems. It includes a minimum of two examples from different domains.
Related Patterns	Related patterns are described, i.e. what are the closely related patterns to this given pattern? What are important differences? With which other patterns should this one be used?

Example

The pattern description has been slightly abbreviated for readability issues.

Table 2 - Builder Design Pattern [34]

Builder Design Pattern	
Elements	**Description**
Pattern Name and Classification	*Builder, Creational Patterns*
Intent	*To split the construction of the complex object from its representation so that same construction process can create different representations.*

Also Known As	
Motivation	
Applicability	*When the builder pattern are used.* • *The algorithm for creating a complex object should be independent of the parts that make up the object and how they're assembled.* • *The construction process must allow different representations for the object that's constructed.*
Structure	
Participants	*Builder, ConcreteBuilder, Director, Product*
Collaborations	*The given interaction diagram describe how Builder and Director cooperate with a client.*

Implementation	
Sample Code	```/Abstract Builder
class abstract class TextConverter{
        abstract void convertCharacter(char c);
        abstract void convertParagraph();
}

.
.
.
.
        public static void main(String args[]){
                Client client=new Client();
                Document doc=new Document();
                client.createASCIIText(doc);
                system.out.println("This is an
example of Builder Pattern");
        }
}``` |
| *Known Uses* | *The RTF converter application is from ET++. Its text building block uses a builder to process text stored in the RTF format .* |
| *Related Patterns* | *Abstract Factory* |

Analysis Pattern Templates

Given below is the Analysis Pattern template described by Eduardo B. Fernandez and Ying Liu in their article "The Account Analysis Pattern" [35]. This template is also described in the book "Pattern-Oriented Software Architecture" [31].

Table 3 - Analysis Pattern Template [31]

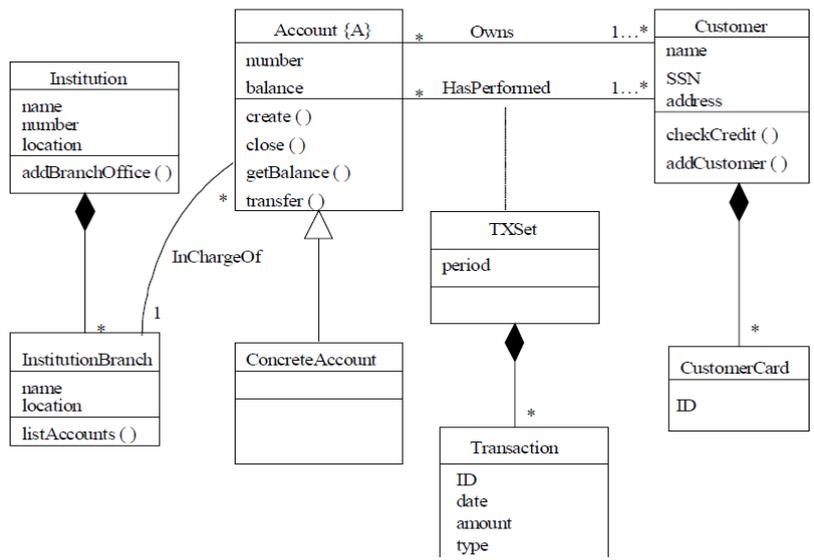| Analysis Pattern Template	
Elements	**Description**
Pattern Name	It describes the name for referring to the pattern and also other names if the pattern has another name.
Intent	It is a short statement that answers many

22

	questions like what does that design pattern do? What is the main goal of the pattern and what are the particular design issues or problems solved by the pattern.
Example	Provides a real world example, which shows an existing problem and exemplifies the need of the pattern.
Context	The section context is a fundamental component of a pattern. It provides an indication of the applicability of a pattern.
Problem	It defines the recurring problem that is solved by the general solution. Problem is a fundamental component of a pattern because it is the reason for the pattern. The problem which is addressed by the pattern is described in this section.
Solution	The solution details the participating entities in the solution, the collaborations between them and their behavior.
Example Resolved	Example Resolved gives the solution of the given example
Know Uses	Know Uses is the example of the use of the pattern in a real system. It includes a minimum of two examples from different domains.
Consequeses	It details the benefits that a pattern can offer and any possible restrictions.
Related Patterns	Related patterns are described what are the closely related patterns to this given pattern? What are important differences? With which other patterns should this one be used?

Example

The pattern description has been slightly abbreviated for readability issues.

Table 4 - Account Analysis Pattern [35]

Account Analysis Pattern Template	
Elements	**Description**
Pattern Name	*Account Analysis Pattern*
Intent	*The Account pattern keeps track of accounts of customers in institutions. These customers can perform transactions of different types against the accounts.*
Example	*Consider a banking institution, where customers have*

	accounts of different types, e.g., checking, savings, loan, mortgage, etc. For the convenience of their customers, the bank may have several branches or offices located in different places.
Content	*There are many institutions, e.g., banks, libraries, clubs, and others, that need to provide their customers or members with convenient ways to handle financial obligations, charge meals, buy articles, reserve and use materials, etc*
Problem	*Without the concept of account users need to carry large amounts of cash, may have trouble reserving items to buy or borrow, and would have serious problems sending funds to remote places.*
Solution	*Start from class Account and add relevant entities; in this case customers, cards, and transactions. Build an institution hierarchy describing the branches of the institution and relate accounts to the branches.*
Example Resolved	*An example for Bank accounts is shown in Figure. The classes contained in the model include **Bank**, **BranchOffice**, **Account,** **CheckingAccount**, **Customer**, **BankCard**, **TransactionSet** (TXSet), and **Transaction**, with their obvious meanings. Class **TXSet** collects all the transactions for a user on his account for a given period of time. There are, of course, other types of accounts.*

Know Uses	The following are examples of uses of this pattern: • Banks, where customers have financial accounts of different types. • Libraries, where patrons can borrow books and tapes. • Manufacturing accounts, where materials are charged.
Consequneses	The pattern has the following advantages: · It is clear that this model provides an effective description of the needs and can be used to drive the design and implementation of the software system. Not using a similar model would result in code that is hard to extend and probably incorrect. · One can easily add other use cases: freeze account and activate/deactivate account. The liabilities of this pattern come from the fact that to limit the size of the pattern and to make it more generic we have left out: Different types of customers. Each variety of customers could be handled in a special way.
Related Patterns	Accountability pattern

Architecture Pattern Templates

This template was described by Ayodele Oluyomi in his article "Patterns and Protocols for Agent-Oriented Software Development" for the Agent internal Architecture-Structure Patterns [36].

Table 5 -Architecture Pattern Template [36]

Architecture **Pattern Template**	
Elements	**Description**
Name	A brief summary of the pattern.
Problem	It defines the problem which a pattern can solve.
Context	Different kinds the circumstances in which a pattern can be applied.
Forces	It contains description of various forces and constraints that can affect the desired objectives.
Solution	This section describes the different part of the pattern and their relation.
Known Uses	Know Uses are examples of the use of the pattern in real system. We include minimum two examples from different domains.
Result Context	This section description of possible effects on the initial context when the solution is applied and also the resulting advantages and disadvantages.
Related Pattern	Related patterns are described; What are the closely related patterns to this given pattern? What are important differences? With which other patterns should this one be used?

Example

The pattern description has been slightly abbreviated for readability issues.

Table 6 -Agent as Delegate Pattern [36]

Elements	**Description**
Name	*Agent as Delegate*
Classification	*Multiagent System Architecture-Definitional*
Problem	*How should the role of a user be converted to an agent or agents in an agent based system while maintaining confidentiality of user information?*
Context	*a user role carries out activities in a system where confidentiality of user information is critical.*
Forces	***Goals**: to achieve optimum performance and maximize gains by taking decisions based on outcome of activities carried out.* ***Responsibilities**: the responsibilities of this role involve carrying out both non trivial operational tasks and making concluding decisions based on the*

	execution of the tasks carried out. User specific information is used in making the decisions. However, the user information should not be included in the execution of the operational tasks for security and confidentiality reasons.
Solution	*This pattern describes an approach for translating a role into agents. It prescribes translating a complex user with sensitive data into two types of agents which are User Agent and Task Agents. The pattern specifies the relationship and control that should exist between these two types of agents.*
Known Uses	
Result Context	*The interaction between the assistant agent and the task agents has to be analyzed, modeled and implemented.* ***Adaptation/Integration:*** *a user role can be translated into more than one assistant agents depending on the complexity and volume of the user information and decision making process.*
Related Pattern	*Agent as Mediator*

2.3.3   Data Model Patterns

Data Model Patterns help modelers to develop quality models by standardizing common and well-tested solutions for reuse [3]. The purpose of data model pattern is to provide a starting point for data modelers [37].

A data model pattern can be implemented by adding additional attribute to any entity in a model or by adding a new entity or a relationship to an existing model. David Hay presented a Universal Data Model in [38]. It is a theoretical model which explains the basic principles of a data model pattern. See Figure 5:
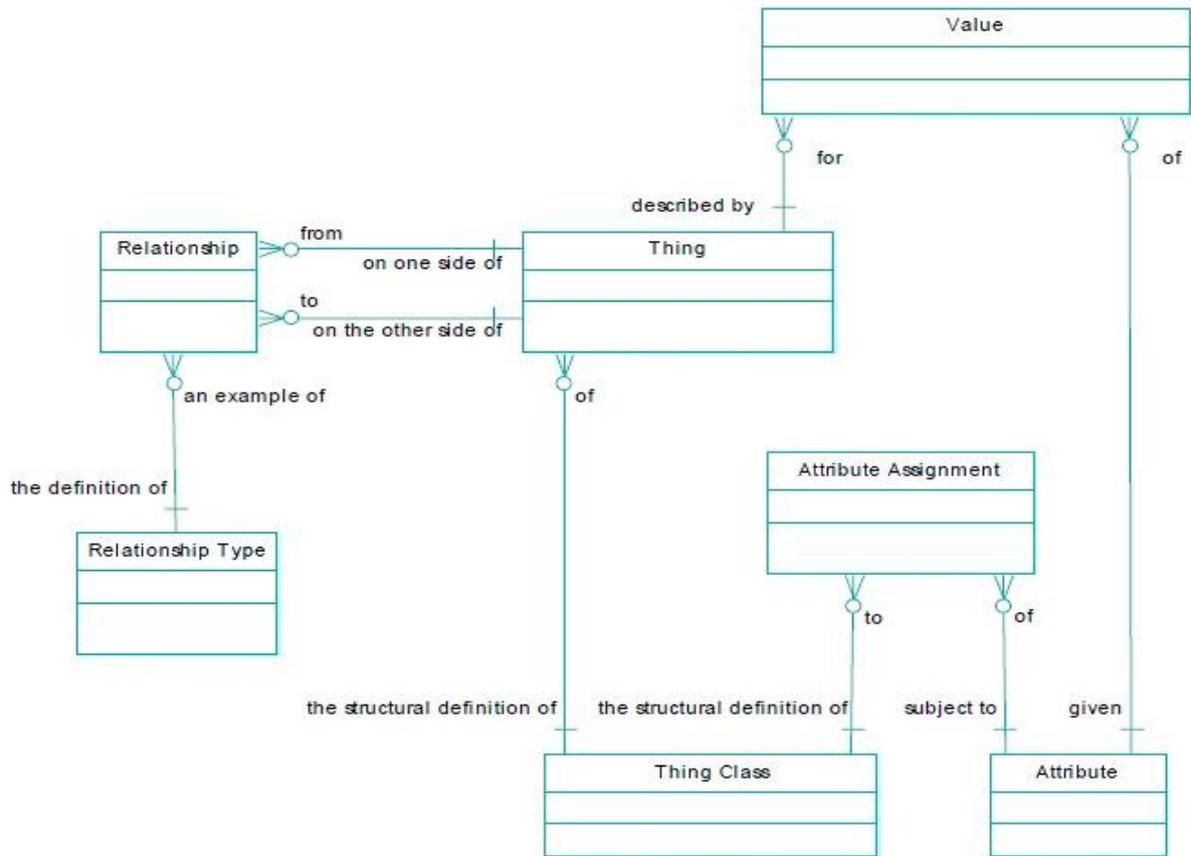
Figure 5 – Universal Data Model [38]

Figure 6 is a sample data model pattern. It depicts a purchase order, its vendors and the products being bought:
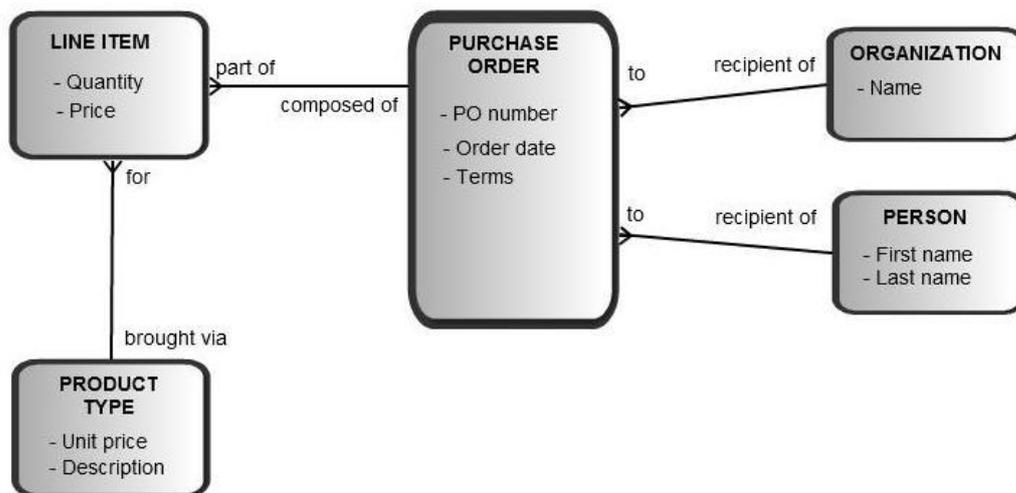
Figure 6 – A Purchase Orders data model pattern [39]

In [38], Hay mentioned conventions for building data models. These conventions are guidelines for creating new patterns. They help in establishing

a framework which data modelers can follow to reuse data model patterns. The modeling conventions are divided into three levels; Syntactic Conventions, Positional Conventions and Semantic Conventions.

Syntactic Conventions: This is the first type of conventions of modeling and deals with the symbols to be used. In the process of syntactic convention evaluation, the crucial point to remember is that there are two audiences in data modeling. The first audience is that community of users which use the models and their descriptions for the verifying whether or not the environment and requirements are actually understood by the analysts. The set of systems designers is the second audience. They make use of the rules of business as implied by the models to be the basis on which their design of computer systems is based [39].

Positional Conventions: This is the second type of Data modeling convention and dictates how entities of a model are laid out. They are concerned with the organization of elements and the overall structure of a model [39].

Semantic Conventions: Semantic conventions are those conventions that address the question of how can the meaning of a model be conveyed. These conventions help to represent common business scenarios in a standard way [39].

In an experiment, the knowledge from data model patterns was translated into ontology design patterns by mapping the parts using the KAON tool [4]. Table 7 shows the mapping between the different parts. The above mapping shows that the knowledge stored in the data model patterns can be translated into ODPs. This knowledge can be reused to create new ODPs from existing data model patterns.

Data Model Pattern	Ontology Design Pattern
Entity	Concept
Attribute	Relation to "attribute"
Subtype/Supertype	Subsumption hierarchy
Relationships	Relations
Mutually exclusive sets	Disjoint concepts

Table 7- Mapping of elements of Data Model Pattern and Ontology Design Pattern [4]

2.4   Ontology Patterns

The use of patterns in ontology engineering has gained importance in recent years. Ontology patterns assist knowledge reuse and provide solutions to ontology engineering problems. Ontology patterns also facilitate in semi-automatic ontology construction hence reducing time and effort. To improve the use of patterns in ontology engineering, the ontology patterns have been classified into different types. Each type addresses a specific level of ontology development process [16].

2.4.1   Classification of Ontology Pattern

The Study of the use of ontological patterns on broader perspective requires a general classification of patterns. Such a classification is presented below [41].

Application Patterns: These patterns deal with the purpose, scope, usage and context of the ontology that has been implemented. The ontology application patterns depend on the ontology usage area and domain [41].

Architecture Patterns: These patterns explain how implemented Design Pattern can be combined or arranged so that the overall objective of the ontology is achieved. [41].

Design Patterns: These patterns constitute small collections of semantic patterns which combine to form a generic construct during ontology development process [41].

Semantic Patterns: They provide description of certain concepts, axioms which are is not dependent on language [41].

Syntactic Patterns: These are ways of arranging representation symbols and creating certain concepts, relation or axioms. Syntactic Patterns are Language specific [41].

2.5   Ontology Design Pattern

"An ontology design pattern is a set of ontological elements, structures or construction principles that solve a clearly defined particular modeling problem [25]."

Ontology design patterns have been extended from software design patterns for knowledge acquirement in semantic web. The main concern of ontology design patterns is to how concepts, relations and axioms are established in a ontology using ontological elements. The extent of use of

these patterns can be divided into three levels, i.e. focusing on single logical elements such as a concept, relation or axiom, focusing on an individual module or focusing on the general structure of a complete ontology [25].

Problems that ODPs Address

Ontology design patterns aim to solve modeling problems in ontologies. To address a problem effectively, a pattern should have multiple facets and flexibility. ODPs provide user a flexibility to use pattern at different levels. An ODP can be implemented by either reusing an individual element or a module or using the logical structure of a complete ontology [25].

Another problem that ODPs address is the automatic construction of ontology by using a pattern. If a pattern is very generic then it can only be used for manual construction of ontology. ODPs are designed to be specific enough to be used in automatic or semi-automatic ontology construction and generic enough to be used in multiple ontologies in a specific domain [41].

2.5.1   Types of Ontology Design Patterns

To solve a recurrent ontology design problem, a modeling solution (ontology design pattern) can be used. Several types of ODPs have been identified and they have been categorized into six families: Structural ODPs, Correspondence ODPs, Content ODPs (CPs), Reasoning ODPs, Presentation ODPs, and Lexico- Syntactic ODPs [21].

Graphically the types of patterns are represented as:
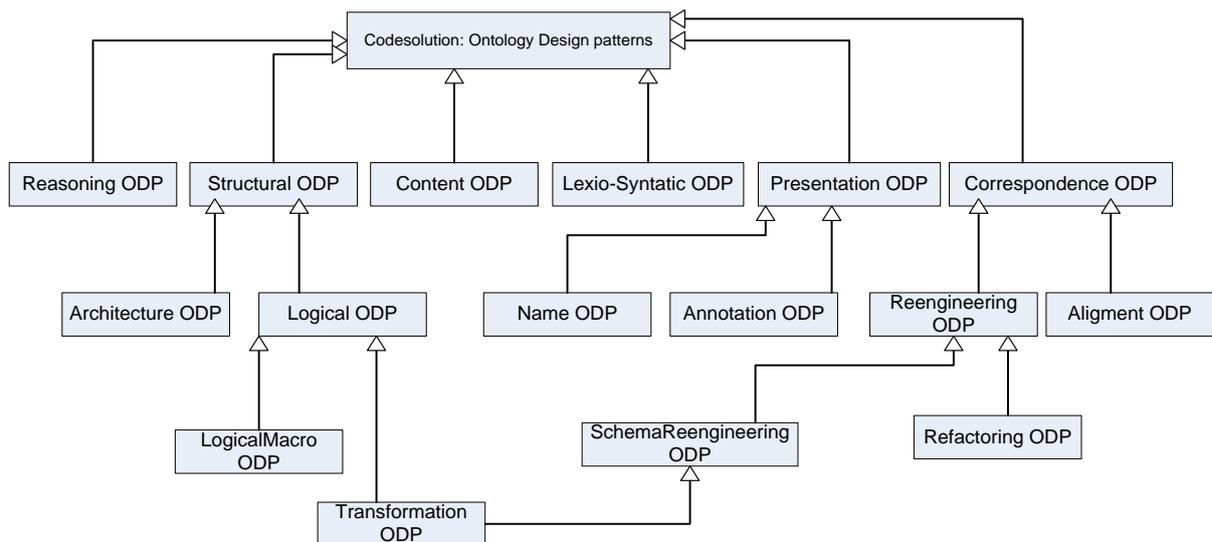
Figure 7 - Taxonomy of Ontology Design Pattern [5]

2.5.2   Structural Ontology Design Pattern

Structural Ontology design patterns contain Logical ODPs and Architectural ODPs. Logical ODPs are made of logic constructs that give the solution to a problem of expressivity. On the other hand Architectural ODPs define how the logical patterns are composed and how logical patterns are used to compose the overall shape of the ontology [5].

Logical Ontology Design Patterns: Logical ODPs contain only a logical vocabulary, because their signature is empty. In other words logical ODPs are not dependent on a specific area of interest. Therefore the logical ODPs are dependent on the expressivity of the logical formalism that is used for representation. The logical ODPs are used to solve design problems where the primitives of the representation language do not directly support the logical constructs. To solve a same modeling problem, a logical OP can be used several times in the same ontology [5].

Architectural Ontology Design Patterns: Architectural ODPs are concerned with the overall structure of an ontology. The main focus of architectural patterns is how the overall ontology should look like. They are used in the design phase of an ontology by giving a combination of logical ODPs. An example of Architectural ODP is Modular Architecture pattern which contains a network of ontologies. Each ontology in the network act as a module and each module is connected with the root ontology with the help of a *import* function [5].

2.5.3   Correspondence Ontology Design Patterns

Correspondence ODPs are divided into Re-engineering ODPs and Alignment ODPs. Re-engineering ODPs give the solution of a problem to designer by transform a conceptual model, which can be a non-ontological resource, into a new ontology. Alignment ODPs are patterns that create a semantic relation between two existing ontologies [5].

Re-engineering Ontology Design Patterns

Re-engineering ODPs are transformation rules that apply for creating a new ontology, starting from source element at a model. The target model is and ontology. In other words the source model is either an ontology or a non ontology, for example, the words of a thesaurus, a data model pattern or a UML model. Re-engineering ODPs are described in terms of meta-model transformation rules. There are two types of re-engineering ODPs [5].

1. **Schema Re-engineering Patterns:** Are used for transforming non-OWL DL meta-models into an OWL-DL ontology, for example used of SKOS for KOS (knowledge organization systems) re-engineering [5].

2. *Refactoring Patterns:* Gives the transforming rule to designers, for example it is refactoring the existing OWL DL source ontology into new target owl Ontology. The transformation rules has effect due to the changing of the type of ontology elements which take part in the refactoring [5].

Alignment ODPs

Ontology alignment means facilitating communication between ontologies modeling a same domain. Alignment ODPs represent most common types of alignments that can occur between ontologies. Each pattern represents a relationship between two or more entities of two ontologies [55].

2.5.4   Content Ontology Design Patterns

Content ontology design patterns solve recurring content modeling problems in the form of reusable solutions. Content ODPs are conceptual patterns rather than logical patterns. Content ODPs address design problems that are related to the conceptualization of a domain. Content Patterns are used to solve the design problem for the domain classes and their properties that compose an ontology, therefore content patterns address a content problem [5].

In correspondence to conceptual modeling and knowledge engineering, the solved modeling problems by content ODPs consist of two basic components: domain and requirements. A same domain can include multiple requirements and a same requirement can be found in different types of domains. Competency questions are the requirements for solving the problem. A competency question is a simple query that an ontology engineer can send to a knowledge base of its target domain to perform a specific task. Content ODPs are characterized as small and autonomous ontologies that help ontology engineer to easily cover the complexity of entire ontology [5]. Based on the above characteristics, content ODPs can be defined as

> *"CPs are distinguished ontologies. They address a specific set of competency questions, which represent the problem they provide a solution for. Furthermore, CPs show certain characteristics i.e., they are: computational, small and autonomous, hierarchical, cognitively relevant, linguistically relevant, and best practice [5]."*

2.5.5   Reasoning Ontology Design Patterns

Reasoning ODPs are an application of logical ODPs that obtain certain reasoning results, which are based on the behavior implemented in a reasoning engine. Examples of Reasoning ODPs are classification, subsumption, inheritance, materialization, de-anonymizing. These Reasoning ODPs allow a system to made decisions regarding what needs to be done with the ontology so that queries, evaluation, etc may be carried out [5] [21].

2.5.6   Presentation Ontology Design Patterns

Presentation ODPs describe the usability and readability of ontologies from a user perspective. Presentation ODPs are good practices to support the reuse of patterns, facilitating the process of evaluation and selection [5].

Naming: Naming ODPs describe how to create the names for name spaces, files and ontologies, including their classes and properties. Naming ODPs are used for ontology readability and understandability for humans. Naming procedures support homogeneity [5].

Annotation: Annotation ODPs describe annotation properties and annotation schemas that are to be use for improving the understandability of ontologies and their elements. Annotation ODPs include labels and comments [5].

2.5.7   Lexico-Syntactic Ontology Design Patterns

Lexicon-Syntactic ODPs are linguistic structures or schemas that contain particular types of words in a defined order, that allow generalizing and deriving few conclusions of the meaning that they convey. Lexico-Syntactic ODPs use some notations for formalizing and describing the syntax of a language. The elements described in the formalized patterns are recognized as essential for finding the relation of interest that are indicated by the pattern [5].

2.6 Template of Content Ontology Design Patterns

Ontology design patterns are similar to software design patterns. The core idea of describing software design patterns is to use a template and collect them by means of a catalogue. In order to describe ODPs we can use a similar approach as used in software engineering but the difference is that, the template used for the presentation has been optimize for the web and defined in an OWL annotation schema. It is the same used on the semantic

web portal http://www.ontologydesignpatterns.org. This part contains a template of Content ODPs which is composed of the following information fields, defined in the annotation schema [5], [2], [53]:

Table 8 – Content Ontology Design Pattern Template [2], [5]

Elements	Description
Name	It contains the name of the pattern. The names of patterns should be descriptive and unique names that help in identifying and referring to the patterns.
Submitted by	This part of the template includes author names. In the portal it gives the link to the author page.
Also Known as	It gives the alternative names for the ODP, since it might be possible that the pattern has some other name but this part is not compulsory.
Intent	This part of the template describes the goal of the ODP. Intent is a description of the goal behind the pattern and the reason for using it.
Domain	This part of the template concerned with the area, domain and where the ODP is applicable.
Competency Question	It contains a list of competency questions expressed in natural language that are covered by the pattern. A competency question is a classical way of capturing a use case. A competency question is a simple query which an ontology engineer can submit to knowledge base to perform a certain task.
Solution description	It describes how the given pattern provides the solution to a design problem in a certain context.
Reusable OWL Building Block	It is a reusable representation of the pattern. This part is basically the implementation of the design pattern. It contains the URI of the OWL implementation of the content pattern, i.e. the reusable component available for download.
Consequences:	This part of the template contains a description of the benefits and/ or possible trade-offs when using the ODPs.

Scenarios	Giving examples or scenarios where the given pattern implemented.
Known Uses	This part of the template gives examples of real ontologies where the ODP is used. This part of template is the example of real usages of the pattern.
Other References	This part of the template contains references to resources (e.g. papers, theories, and blogs) that are related to the knowledge encoded in the ODPs.
Examples	This field contains a link of an example owl file which is reusable. The example owl file presents a possible scenario which may sometime also include a UML diagram of classes and their relationships.
Extracted From	Contains the URI (if any) of the ontology from which the pattern has been extracted.
Reengineered from	It contains the name of the reference ontology which has been used reused in the pattern.
Has Components	This field refers to components of the Content ODP which are in turn ODPs themselves.
Specialization Of	This part of the template refers to ontology elements or ODPs. The specialization relation between ontology elements of ODPs consists of creating subclass of some ODP class and/or sub properties of some ODP properties.
Related ODP	This part contains the names of the patterns which related to the current pattern based on generalization, specialization or composition. It also mentions other patterns that are used in corporation with the current pattern.
Elements	This part of the template describes the elements (classes and properties) included in the ODP, and their role within the ODP.
Diagram Representation	This part of the template depicts a graphical representation of the ODP.
Additional Information	In Additional information authors provided that informatuion which is not avalible in the rest of the template.

Example

Table 9 – Classification Pattern [56]

Elements	Description
Name	Classification
Submitted by	ValentinaPresutti
Also Known as	
Intent	To represent the relations between concepts (roles, task, parameters) and entities (person, events, values), which concepts can be assigned to. To formalize the application (e.g. tagging) of informal knowledge organization systems such as lexica, thesauri, subject directories, folksonomies, etc., where concepts are first-order elements.
Domain	General
Competency Question	▪ What concept is assigned to this entity? ▪ Which category does this entity belong to?
Solution description	
Reusable OWL Building Block	http://www.ontologydesignpatterns.org/cp/owl/classification.owl
Consequences:	It is possible to make assertions about e.g., categories, types, roles, which are typically considered at the meta-level of an ontology. Instances of Concept reify such elements, which are therefore put in the ordinary domain of an ontology. It is not possible to parametrize the classification over different dimensions e.g., time, space, etc.
Scenarios	Mac OSX 10.5 is classified as an operating system in the Fujitsu-Siemens product catalog.
Known Uses	
Web References	
Other References	
Examples	

Extracted From	http://www.loa-cnr.it/ontologies/DUL.owl
Reengineered from	
Has Components	
Specialization Of	
Related ODP	
Elements	**Concept** (owl:Class) A concept is a Social Object. The classifies relation relates concepts to entities at some time. **Entity** (owl:Class) Anything: real, possible, or imaginary, which some modeller wants to talk about for some purpose. **classifies** (owl:ObjectProperty) A relation between a Concept and an Entity, e.g. the Role 'student' classifies a Person 'John'. **is classified by** (owl:ObjectProperty) A relation between a Concept and an Entity, e.g. 'John is considered a typical rude man'; your last concert constitutes the achievement of a lifetime; '20-year-old means she's mature enough'.
Diagram Representation	
Additional Information	

3 Research Methodology

3.1 Research Method

The approach of performing research and collecting data for analysis can be qualitative or quantitative. The choice of approach depends upon the problem at hand. A quantitative research approach is associated with the positivist research tradition and applies scientific methods to social science. This research is concerned with the collection of quantitative data in the form of numbers, collected by techniques such as questionnaires and other measurement instruments. This approach considers that knowledge can be based on what can be objectively observed and experienced [43]. On the other hand qualitative research is concerned with developing explanations of social phenomena and is concerned with data in the form of words, collected by techniques such as interviews and observation. The qualitative research paradigm is associated with the interpretivist research tradition and concerned with the opinions, experiences and feelings of individuals producing subjective data [43],[44].

There are some research questions that cannot easily be answered by a quantitative research design but can be answered by qualitative research methods instead. Qualitative research allows the researcher to get much richer answers to questions and give valuable insights which might have been missed by other methods. It not only provides valuable information on its own but can be used to complement quantitative research methods [45].

The value of data depends on its trustworthiness that is validity and reliability of the data. The trustworthiness is high if quantitative and qualitative approaches of data collection and analysis can be combined rather than being used separately [46].

In our research the focus is on the collection of both qualitative and quantitative data, so we will use both kinds of research methods.

3.2 Outline of Thesis Work

The work starts with the literature review. Our first task was to study the current patterns that exist in other fields and compare them to ontology design patterns' templates to analyze the difference. In the next phase, we conducted two different types of online surveys.

3.2.1 First Survey

The aim of the first survey was to determine how well the current ODP template supports the understanding and usage of Content ODPs. The participants of the online survey were the students who attended the Information Logistics course at Jönköping University, Sweden. The reason for their selection was because they were familiar with ODPs and used them for ontology development during the course. The questionnaire was sent to forty students but we got response from seventeen of them.

In the survey, a description was given of one of the ontology design patterns, called AgentRole (see Appendix 8.1.1). The pattern and its description were copied from an online portal, i.e. the ODP portal at http://ontologydesignpatterns.org. The participants were required to read the description and understand different parts of the pattern and then answer a questionnaire based on their observations. The questionnaire of the survey was designed to collect wide range of opinion in the form of both open-ended and closed-ended questions. The first four questions of the survey were closed-ended which were designed to specifically cover each part of template while the purpose of next three questions was to allow participants to provide suggestions for the improvement of current template.

3.2.2 Second Survey

The aim of second survey was to get experts opinions on the current structure of the Ontology design pattern template. The participants of the online survey were the members of the ODP portal quality committee. In total, nine members participated in the survey. The survey consisted of two parts (see Appendix 8.1.2):

In Part 1, the participants were asked to select a set of patterns for an ontology engineering problem, based on seeing only a part of the complete template for content ODPs. The results from the first survey showed that the Graphical Representation, General Description and Scenario were the three most important parts in the current template of ODP. The purpose of this section was to identify the most important part among them.

The questionnaire in Part 2 was the same as in the first survey where participants were asked to give their opinion with respect to the content and structure of the current template. The reason for conducting the same survey was to observe the difference in opinion between novice and expert users in understanding ODPs.

3.3 Data collection

The techniques used for data collection are literature review and survey. The literature review stage can be considered as a qualitative technique as it develops a conceptual framework before collection of further data [47]. Surveys were set up after a period of examining the relevant literature, i.e. conducting a literature review.

Survey research involves the collection of primary data from all or part of a population. There are different types of surveys, each encompasses a variety of data collection techniques e.g. questionnaires, interviews and observation. We have used web surveys to conduct our research and questionnaire as a data collection technique. A questionnaire is a combination of close ended or open ended questions or both. A closed ended question requires the respondent to select an answer from a number of options while an open ended question asks the respondent to devise his own answer [57]. The questionnaires used in both surveys were a combination of open ended and closed ended questions.

The responses from the closed ended questions were measured by ranking the answers. To cover whole range of responses, the importance was measured by categorizing the answers in the form of *Most Important, Important, Neither Important nor Unimportant, Not Important* and *Not Important at all*.

In the Part 1 of the second survey, the participants were to select a correct set of patterns for the four competency questions based on seeing only one part of the pattern. The survey randomly generated a single part (i-e Graphical Representation, General Description or Scenario) for each participant. There were total nine participants and each part of the pattern received three answers.

3.4 Evaluation methods

Evaluation methods are used to validate the research results. In our study, evaluation was done on the results of the comparison of different patterns and the survey results. Templates of the patterns were compared to identify the difference and similarities in their presentation. Each part of the templates was studied with respect to its objective and the content provided in that part.

Evaluation of the survey results was done by analyzing the responses of both open ended and closed ended questions. The closed ended questions were analyzed by calculating the frequency of responses for each question to identify the most important part. For open ended questions, all the comments from the participants were read through and categories for
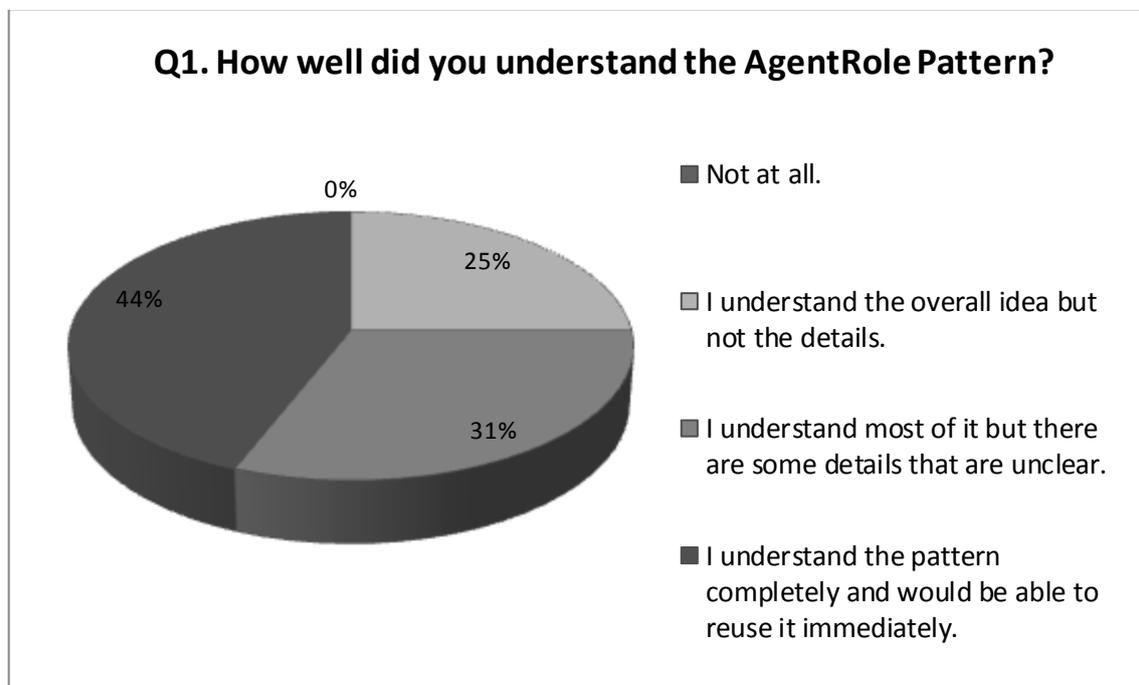
responses were developed. After studying the data, trends were identified and the results were summarized to represent general opinion. Finally, we used the notions of external and internal validity to analyze the results of the study.

4 Results

4.1 First Survey

Below are the summarized results of each question of the first survey. Refer to Appendix 8.3.1 for detailed population numbers and percentages.

The first survey was conducted to get the opinion of novice users. They had little experience of working on ontology projects. The first question was how well the participants understood the AgentRole Pattern. After reviewing the results, it was evident that 56% of the population was unclear about some details in the pattern while the rest thought they have understood the pattern and can use it immediately. See Figure 7.

Q1. How well did you understand the AgentRole Pattern?

0%

25%

44%

31%

■ Not at all.

■ I understand the overall idea but not the details.

■ I understand most of it but there are some details that are unclear.

■ I understand the pattern completely and would be able to reuse it immediately.

Figure 7 – Graph representing percentage of user understanding of pattern

The current ODP template consists of five parts: Graphical Representation, General Description, Element, Scenario and Additional Information. Participants were asked which part they considered most important in understanding the pattern and in their opinion the 'General Description' part was considered the most important while the 'Additional Information' part was considered least important. See Figure 8.
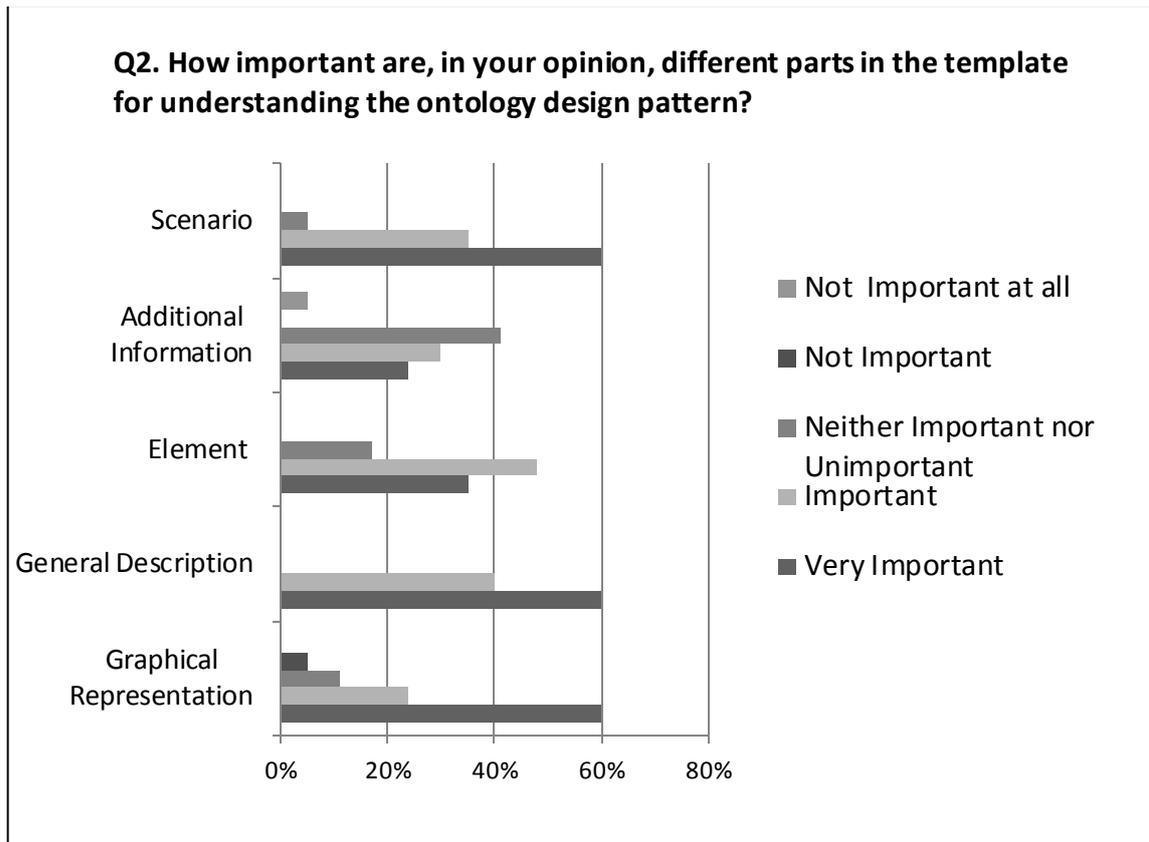
Figure 8 – Graph representing importance of different parts of the template

The third question was to determine if there are any parts in the template that the participants think are not needed. A majority of the population thought that all parts must be in the template and there is no need to exclude them. Only a few of the participants suggested that the Additional Information and Elements sections are not needed. See Figure 9.
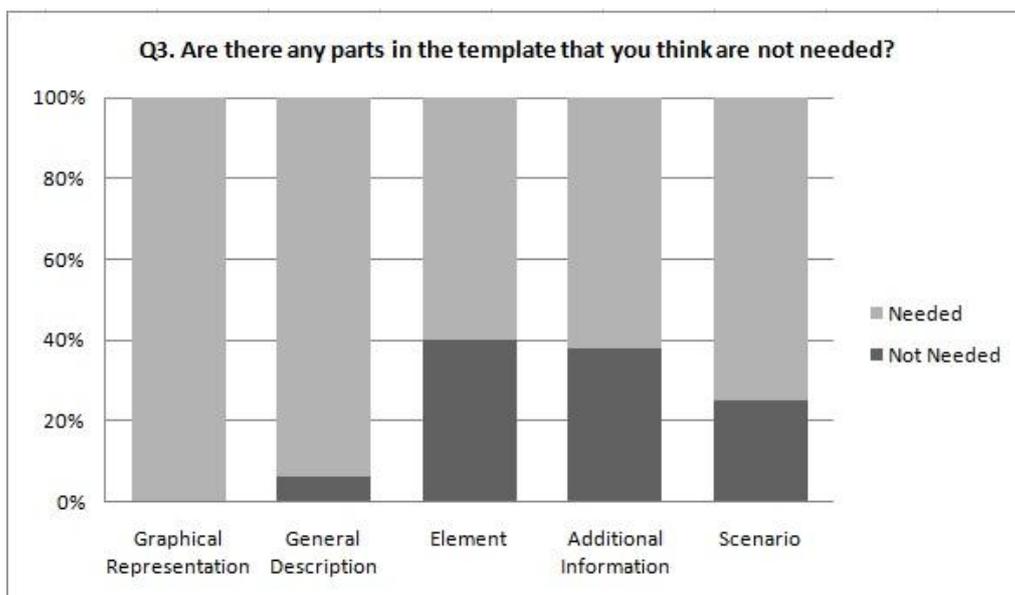
Figure 9 – Graph representing the needed and not needed percentage of each part

The 'General Description' section itself contains further sub-parts. The next question was to determine the most important part in this section. The results showed that 'Domain' and 'Intent' were considered most important while 'Reusable Owl Building Block' and 'Example Owl File' were given least importance. See Figure 10.
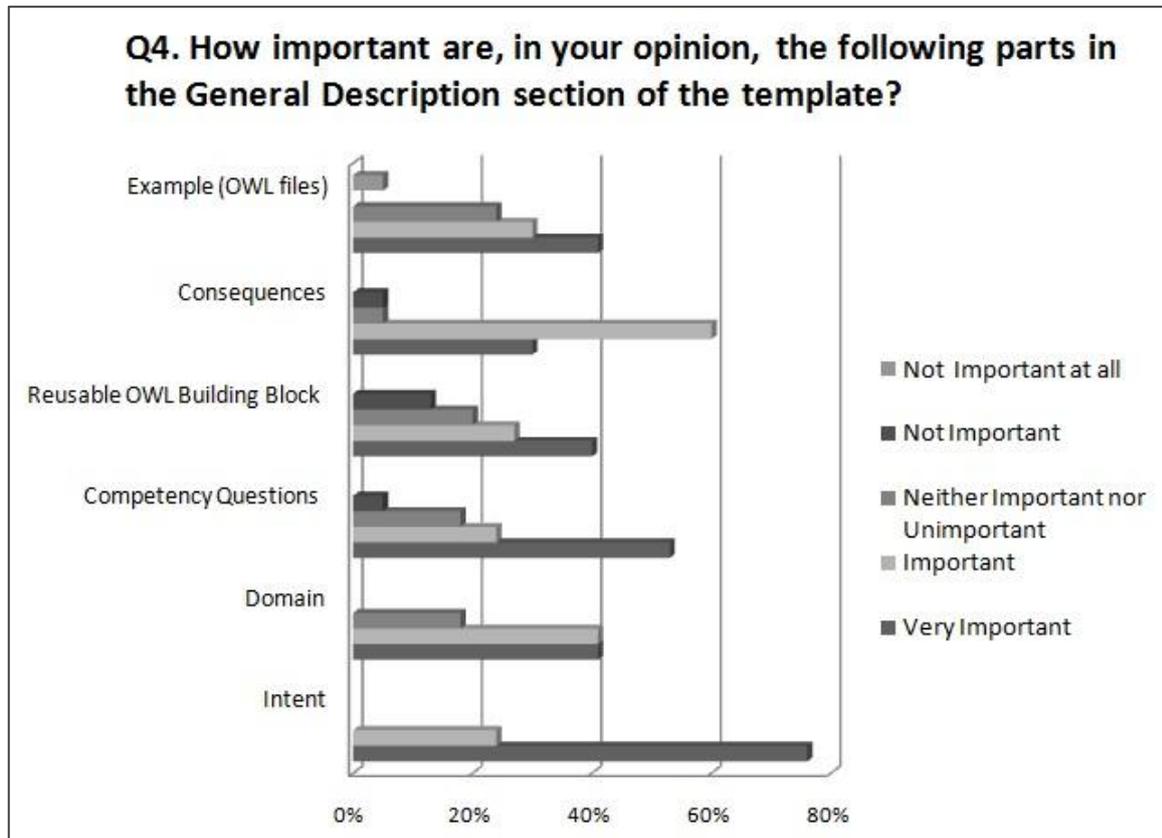
Figure 10 – Graph representing the importance of General Description parts

Further, the participants were asked if there is any part in the General Description section of the template that they think is definitely not needed? Most of the participants suggested that 'Example Owl File' was the part in General Description that can be excluded. Some participants also suggested excluding 'Solution Description' part.

Next question was to determine if there anything missing in the overall template of the design pattern and whether the participant need some additional information to reuse the pattern and in that case what information? In general, the participants did not point at anything missing in the current template, however they gave different kinds of suggestions how to improve the ODPs presentation. Some suggested changes in the structure of the template while others suggested extra functionality in the tools for implementing ODPs. For example, there was one suggestion to add a GUI Plug-in in the Protégé tool which can provide suggestions to

user to use a particular pattern based on the intent and competency questions.

In the last question, participants were asked if they have any suggestions or proposals for improvement of the current design pattern template and presentation. Most of the participants gave similar suggestions as in previous question however the general opinion was to improve graphical illustration of the pattern, modify general description and scenario sections. Some of the suggested modifications include adding more domains in the domain part, describing the limitations of the usage of the pattern in general description and adding more relevant patterns in the Relevant CPs part.

4.2 Second Survey

Below are the results of the second survey. Refer to Appendix 8.3.2 for detailed population numbers and percentages. The survey was answered by nine participants.

4.2.1 Part 1

The survey randomly generated one of the three parts (Graphical Representation, General Description, and Scenario) to each participant. The criteria for correct answer required participant to select the right set of patterns for each competency question. Even if the participant had selected some extra patterns apart from the correct set of patterns for a competency question, it was considered a correct answer.

In total, each part received three answers. After calculating the results, it was clear that all the participants, who solved the problem with the help of General Description part, selected the correct patterns. This was followed by the Scenario part, where two out of three participants were able to select correct patterns while the participants with the Graphical Representation part provided least correct answers. Thus the results from Part1 indicate that the General Description is the most important part of content ODP.

4.2.2 Part 2

Participants of the second survey answered the same questions as of the first survey. As in the first survey, the first question was how well the participants understood the AgentRole The majority of expert users (66%) were also unclear about some details of the pattern while 34% understood the pattern completely. See Figure 11.
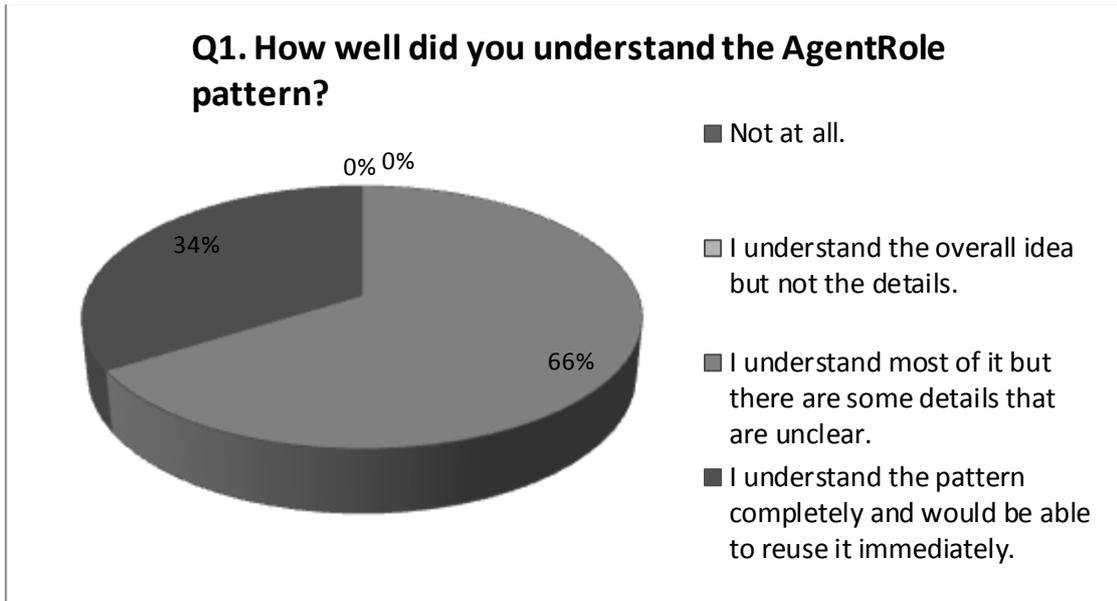
Figure 11 – Graph representing percentage of expert users' understanding of pattern

The second questions was how important are, in participant opinion, different parts in the template for understanding the ontology design pattern? Similar to the opinion of the participants of the first survey, most of the participants suggested Graphical Representation, General Description and Scenario as more important parts than Additional Information and Elements. See Figure 12.
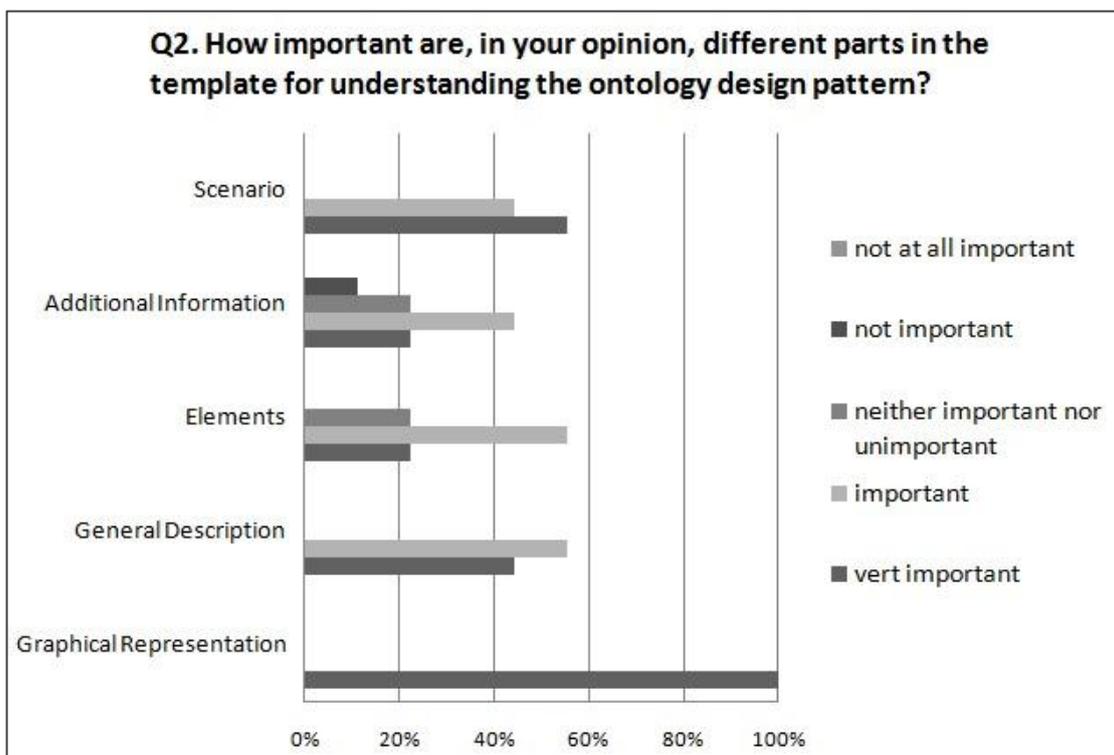
Figure 12 - Graph representing importance of different parts of the template

Regarding third question about determining if there are any parts in the template that the participants think are not needed, all the participants thought that Graphical Representation, General Description and Scenario parts must be in the content ODP template however 63% of the population was also in favor of the Elements and Additional Information parts. See Figure 13:
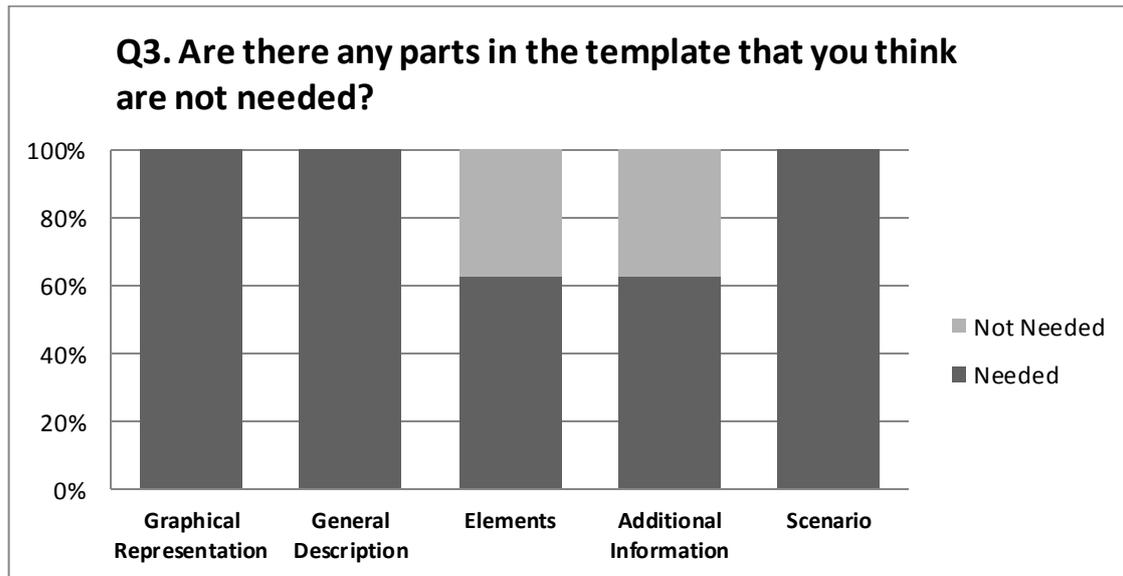
Figure 13 - Graph representing the needed and not needed percentage of each part

The next question was how important are different parts in the General Description section of the template in their opinion? Contrary to the results of the first survey, there was a difference in opinion when the participants were asked to select the most important part in General Description section. The results indicated that 'Reusable OWL Building Block' and 'Example OWL Files' were the most important parts while 'Intent' and 'Domain' were the least important parts in the General Description. See Figure 14
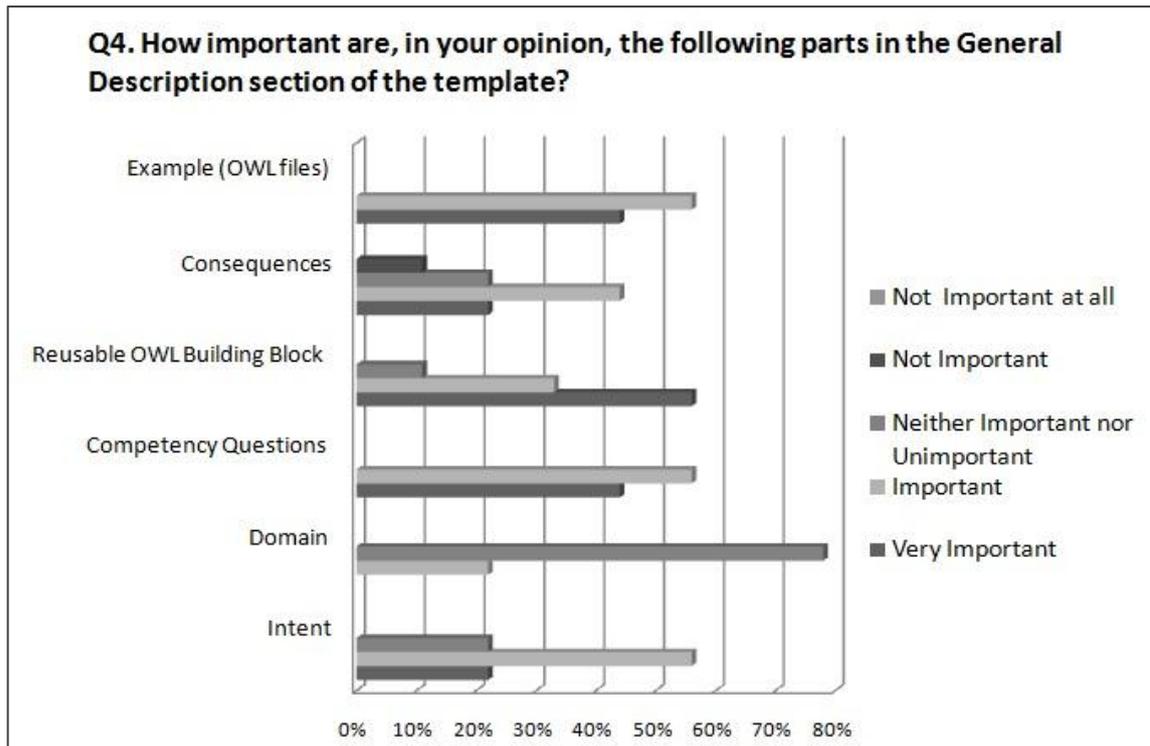
Figure 14 – Graph representing the importance of General Description parts

The fifth question was if there is any part in the General Description section of the template that they think is definitely not needed? None of the participants recommended removing any part from the General Description section however there were suggestions for improvement. One suggestion was to make scenarios more explicit to improve understanding. Another suggestion was about the Consequences part in which it was suggested that its description should be focused on the Graphical Representation and General Description.

The sixth question was to determine if there anything missing in the overall template of the design pattern and whether the participant need some additional information to reuse the pattern and in that case what information? Participants pointed out some changes in several parts of the pattern. In the Graphical Representation section, it was suggested that concepts and relations should be spelled out clearly. Scenario should be more complex and it should contain information about the binding from the concrete elements to the pattern elements. Also more information can be included about the imported patterns and a distinction of imported parts in the diagram as well as more references to other patterns.

In the last question, experts were asked if they have any suggestions for improvement of the current design pattern template and presentation. In Graphical Representation, they suggested to provide more uniform graphical notations and all its elements should be fully defined in the Element section. The Scenario for a given pattern should also be more explicit. Namespaces should be made more explicit in the additional information section.

5 Analysis and Discussion

5.1 Comparison of Pattern Templates

There are many types of patterns but we have limited our comparison to those of software engineering and data models patterns, since these are the most used and well known patterns.

5.1.1 Comparison of Software Patterns and Ontology Design Patterns

Software patterns have been compared to ontologies by Vladan Devedzic in one of his article [33] where the author argued that there is a significant overlap between the two concepts and that it is the aim, while generality and practical usage of these concepts differ. The concepts of patterns and ontologies have some common goals, e.g. sharing and reusing of knowledge. Both these concepts necessitate hierarchies of concepts, relationships, vocabularies and constraints. Moreover, both of them can be seen as using an object-oriented paradigm [3].

The basic knowledge of problems is in software engineering modeled by conceptual models that are known as Analysis patterns. The patterns could be illustrated using a UML notation [3]. For Example the Account Analysis Pattern is implemented by using a UML Class diagram, Sequence Diagram and State Diagram which were described in section 2.3.2. Looking at an example of an analysis pattern, one can easily gauge the above relationship between ontologies and Software Patterns, and other patterns used in Computer Science. This similarity is evident even in, for example, graphical representations of an Analysis Pattern and an ontology pattern [3].

5.1.2 Comparison of Templates of Software Patterns and Content ODPs

This comparison is between the templates of software patterns and content ontology design patterns. We have described the elements of both software pattern templates and content ODPs in the background chapter in sections 2.3.2 and 2.5, which provide us with a good starting point for comparison. There are several types of software patterns that are available and several techniques to present them but we selected Analysis Patterns, Architecture Patterns and Design Patterns. The template of a pattern is a standard way of representing a pattern. In a broad sense, a pattern template has four essential elements. These elements are: Name, Problem, Solution and Consequences as describe in section 2.3.2.

This comparison is based on similarities and differences between the Content ODP template and software pattern templates. We compare each element of the ODP template with software pattern templates. Comparison is based on the names, content and the overall presentation of the template.

The parts described in Table 10 are considered as basic parts of a pattern. The description of these parts is stated in background chapter in sections 2.3.2 and 2.5. The table shows how these basic parts are described in software patterns and ODP templates.

Pattern Type	Context	Problem	Solution	Consequence	Example
Ontology Design Pattern	Domain	Competency Question	OWL Building Block, Element, Solution description and Graphical Representation	Consequence	Scenario, Example (OWL file)
Analysis Pattern	Context	Problem	Solution	Consequence	Example, Example resolved
Design Pattern	Applicability	Motivation	Structure, Participant Collaboration, implementation and sample code		Sample Code
Architecture Pattern	Context	Problem	Solution		

Table 10 - Representing basic elements of other patterns

Context: In Software Patterns, the section context describes the situations where the given pattern is applied. Every pattern has a context based on its application area. In content ODPS, context is defined in the form of domains and scenarios where the pattern is applicable.

Problem: For Analysis Patterns, Design Patterns and Architectural Patterns, the section Problem or Motivation describes the problem that can be solved by implementing these patterns. In analysis patterns, the section problem describes some generic use cases. In Design Patterns, some of the patterns use scenarios for giving more description of patterns. Such description should be able to clarify the details of the problem. In ODPs, the Problem is described by Competency Questions. Competency Questions consists of a list of competency questions expressed in natural language that are covered by the pattern. The section Competency Question describes the problem which is to be solved by the ODP. Competency questions are the requirements that an ontology should fulfill.

Solution: In Analysis Patterns, Architecture Patterns and Design Patterns, the section Solution gives a fundamental solution principle to be used in the pattern for solving a problem. In Analysis Patterns, the solution is described by using UML diagram and also a brief description of different parts of the pattern is given. In Design Pattern, the sections structure, participants and collaboration describe the solution. The section Structure is a graphical representation of the classes in the pattern which use the notation of the object modeling technique (OMT). It also uses interaction diagrams to illustrate the sequence and collaboration between the objects. The section participants give a detailed description of the classes and objects and their responsibilities. The section collaboration describes how the participants collaborate to carry out their responsibilities. In Architecture pattern, the section solution gives the description, using text and a graphical representation, as to how we can achieve the intended goals and objectives. It also describes the variants and specializations of the solution. It gives the description of people and computing actors and their collaboration. In ODPs, the section solution consists of four parts: OWL Building Block, Elements, Solution description and Graphical Representation. The section Solution description describes how a given pattern can solve the problem in the context. The section OWL Building Block contains an OWL ontology with reusable classes and reusable properties. The section Elements briefly describes Classes and Properties of the pattern implementation and the role of these classes and properties within the ODP. The Graphical representation gives a visual presentation of the pattern classes and their relations.

Consequence: In both software patterns and content ODPs, the section consequence describes the possible benefits and limitations on the solution after using the pattern. What are the results of using the pattern? In some ODPs it is more about unexpected consequences and limitations.

Example: For Analysis Patterns, Design Patterns and Architecture Patterns, the section Example gives the real world example, which shows the existence of problems and needs for the patterns. For Analysis Patterns and Design Patterns the section example consists of two parts. In the first part the problem and in second part the implementation. In content ODPs, example is given in the form of a scenario and an example OWL file which is the OWL implementation of the scenario.

Table 11 describes the common elements of the template of content ontology design patterns and software patterns (analysis patterns, design patterns and architecture patterns). These elements are described in the background chapter with details in section 2.3.2. The section motivation of design pattern is similar to the section problem of other software patterns.

Template Heading	Ontology Design pattern	Analysis Pattern	Design Pattern	Architecture pattern
Name	×	×	×	×
Known Uses	×	×	×	×
Intent	×	×	×	×
Consequences	×	×	×	
Also Known as	×		×	
Classification			×	×
Related Pattern	×	×	×	×

Table 11 - Representing common elements of other patterns

The template of content ODPs has developed by following the template of software design patterns. The comparison of both patterns reveals that both patterns have a lot of similarities and the current template of content ODPs includes all the elements of software patterns except 'forces'. Forces define constraints and problems that can affect the solution. In content ODPs, only the benefits and/ or possible trade-offs when using the ODPs on the initial problem are mentioned. Apart from the software patterns, the content ODPs has some additional parts which have been added according to the pattern requirements.

Unique Sections: Content ODPs have some unique sections, which are not described in other Software Pattern Templates. These sections are: EXTRACTED FROM, REENGINEERING FROM and HAS COMPONENTS, as mentioned in the background chapter in section 2.5.

5.1.3 Comparison of Data Model Patterns and Ontology Design Patterns

Data model patterns and ODPs are different to each other because data model patterns are presented only in graphical form while ODPs have much more detailed graphical and textual description. While content ODPs have an official catalogue where users can select from a list of patterns, there is no official catalogue for data model patterns.

The template of an ODP has a description of different parts of the pattern which is presented in the graphical and textual form. To implement an ODP, an ontology engineer has to study the description to understand a pattern. A data model pattern is presented in the form of an UML diagram. The diagram is built by following conventions which are a set of rules. These conventions standardize a data model hence makes it easier for a data modeler to reuse a pattern.

The reusability of both patterns depends on different factors. Content ODPs can also be used directly as they are, just like data models, although they are usually specialized but this depends on their generality. In data model patterns, it is up to the data modeler to decide whether to use a real

model to make some minor adjustments or to use a completely abstract model of a problem.

The similarities of data model patterns and ODPs lie in the graphical representation of a problem. Also as we saw in Table 7, different parts of both patterns can be mapped to each other hence knowledge from data model patterns can be translated into ODPs. The use of conventions in data model patterns makes it easier to understand the diagram. These conventions and practices can be translated into guidelines for creating more uniform diagrammatic notations for ODPs.

Overall, the comparison of software patterns and data model patterns with content ODPs shows that there are a lot of similarities in the templates of software patterns and content ODPs. The difference lies in the presentation of the content. The graphical representation of content ODPs can be made uniform by defining conventions as it is done in data model patterns.

5.2 Survey Analysis

5.2.1 First Survey

The participants of the first survey were the students who worked with content ODPs in the lab work of their course. The tasks in the survey were designed to be similar to tasks they had performed during the labs (see Appendix). Results of the survey show that the participants considered the graphical representation as an important part in the template of a content ODP so it can be assumed that a novice user can understand a pattern better by visual description.

Another observation which was made from results indicates that novice users tend to concentrate on information which describes the purpose and objectives of a pattern rather than its reusable components. For example, in the general description section, Intent and Domain were considered more important than reusable OWL building block.

The AgentRole pattern used in the survey had many parts which were not completely described. This lack of information may have influenced the participants to ignore the importance of those parts. For example, Elements and the Solution Description parts were not fully described in AgentRole pattern so they were considered the least important parts. Some participants even suggested removing these parts from the template.

In general, the evaluation of the first survey shows that a novice user does not go deep into details when understanding an ODP, rather they focus on the intent, the graphical representation of the classes and their relations and a given scenario of a pattern. The opinion from the participants also suggests improvements in these areas (see section 4.1).

5.2.2 Second Survey

The second survey was conducted with experience users to get expert opinion about the ODPs. The survey was divided into two parts. As analysis from the first survey revealed Graphical Representation, General Description and Scenario as the important sections in the pattern, the purpose of Part 1 was to determine the most important section among them. The Part 2 of the survey was same as in the first survey. The reason to ask same questionnaire was to determine the difference in opinion between a novice and an expert user.

The results from the Part1 indicate that the General Description is the most important part of a content ODP. It indicates that the General Description part contains the most vital information in understanding a pattern. The reason participants were not able to select right patterns from Graphical Representation and Scenario was they were unclear about them. This fact was evident from the results of Part2 of the survey.

In Part2, the participants had a similar opinion with the participants of the first survey about main parts of the pattern. For example, Elements and Additional Information section were considered least important. However, the major difference between novice and expert ontology users was that while novice users gave importance to those parts which help in understanding a pattern, the ontology engineers gave importance to the information regarding implementation of a pattern. For example Reusable Owl Building Block and Example Owl File were considered more important than other parts of the content ODP.

Most participants gave suggestions for the improvement in graphical representation and scenario of the content ODPs. One suggestion was to make scenario more explicit to make it more understandable for new user. For graphical representation, one suggestion was to use uniform graphical notations for all patterns because in ODP portal, there are many content ODPs which have different graphical representations from others. For example, the Time Indexed Participation and the CatchRecord pattern. Another suggestion was to highlight distinction between local and imported parts of other pattern in graphical diagram.

Overall, the evaluation of both surveys shows that General Description is the most important part in understanding content ODPs. The major difference in opinion between novice user and expert ontology engineer is about the descriptive and implementable parts of the pattern. There is a scope for improvement in graphical representation and scenario sections.

5.3 Evaluation of the Study

Validity helps in determining the quality of a study. Validity requires that the results are relevant to the objectives of the research. The objective of our research was to improve the understanding and usage of content ODPs and to observe the difference in opinion between novice and expert ontology engineers.

5.3.1 Internal Validity

Internal validity is concerned with the validation of the conclusions of the research. There must be sufficient evidence to support the claims such as checking if the results we got were really the effect of the variables we were studying or could they have been affected by something else?

Online surveys were conducted as an experiment. The surveys consisted of questionnaires which were designed to cover each part of the template to get precise answers from the participants. However, the amount of background knowledge can affect the ability of the participant to answer a questionnaire. For example, participants of the second survey were required to select correct set patterns for a problem without looking at the complete description of the patterns. As the participants were the members of the quality committee of ODP web portal so it can be assumed that most of them had previously gone through all the patterns available on the portal which makes it easier to understand patterns without having to study the whole template of a pattern.

5.3.2 External Validity

External validity is used to measure the generalisability of the results. If external validity is low then the results are valid for only specific area or group of people.

The participants of the surveys were the people who have worked on ODPs so the conclusions are mainly generalizable to people involved in the ontology development. However, some of the participants were quite novice in using patterns, so they are most likely similar in background to people who encounter and learn about Content ODPs for the first time. The results of the comparison are only valid for ODPs because we didn't compare all types of patterns. The scope of the comparison was limited to software patterns and data model patterns.

The conclusion drawn from the survey results reflect an opinion of a small population so it is important not to make strong conclusions about the subject of the research. However, the results provide a base to conduct a much larger study and explore further improvement areas in the presentation of content ODPs.

6 Conclusions

The purpose of this study was to improve the presentation of content ontology design patterns. To achieve this purpose, we have suggested some improvements areas in the current ODP's template. Some suggestions are based on the results from the surveys while the rest were identified through comparison of the different patterns.

In chapter 1, four research questions were mentioned to direct the study. Below are the answers related to each research question:

1. *What is the most important information, present in current templates, for understanding and reusing content ontology design patterns?*

   Results from the surveys show that both novice and expert users consider General Description as the most important part in an ODP. It contains the most vital information for understanding a pattern which includes the objectives and limitations of a pattern, the reusable components, example owl files and references of related patterns.

2. *Is there a difference, with respect to question1, between novice and expert ontology engineers?*

   The results of the surveys shows that a novice user does not concentrate on deep details while selecting a pattern, rather they focus on the intent and the graphical representation of the classes and their relations pattern. On the other hand, an expert ontology engineer focuses on the OWL implementation of a pattern for reusability and understandability.

3. *What is missing in the current template for content ontology design patterns?*

   Comparison of ontology design patterns with other patterns revealed that the content ODPs lack uniformity in representation of graphical notations. Also, there is incomplete information in many parts of several patterns at ODP portal.

   Participants who answered the surveys did not suggest that anything was missing in the current template; rather they suggested improvements in the current structure of the pattern template.

4. *How can the current template and the presentation of current content ontology design pattern be improved?*

   Based on the feedback from the surveys and the literature review, we propose following improvements in the current template for content ODP:

- The Graphical Representation should have a more uniform diagrammatic notation. This can be done by defining standards or conventions which ontology engineers can follow while creating patterns. As guidance, the conventions used in data model patterns can be studied to define similar conventions for content ODPs. Also, namespaces provided in the Graphical Representation section should be made explicit in the Additional Information section.

- Scenarios should be more explicit. While scenarios presented in the General Description section are simple, they can be more complex in general. Further, the scenario section should describe the binding from concrete elements to the pattern elements. At ODP portal, scenarios of several patterns are missing. They must be included in each pattern because they were considered as important parts during the surveys.

- Elements of the pattern should be fully defined. This is important because at present the content ODPs lack a standard way of representing graphical notations so it is vital that each element of the diagram should be defined in detail.

- Some of the information in the template which was considered least important during the surveys can be hidden by collapsing it in its section. The user can expand the information by using a 'plus' button.

This research is part of an effort to solve the larger problem of engineering high quality ontologies. One possible solution to this problem is to introduce reuse in ontology engineering which can standardize the ontology development process and reduce the time and effort involved in it. ODPs are considered best practice to achieve this objective. To encourage the use of ODPs for ontology development, their presentation must be explicit and precise. The results of this research will help to improve the presentation of the ODPs template. Future work in this area

could be to implement the suggested changes in the current template of content ODPs and collect opinion from the much larger population on the updated template.

7 References

[1] T.R. Gruber, "A Translation Approach to Portable Ontology Specification", Appeared in Knowledge Acquisition knowledge systems laboratory, Stanford, Academic Press Ltd. London, 1993, pp. *1*99- 220

[2] V. Presutti, A. Gangemi, "Content Ontology Design Patterns as Practical Building Blocks for Web Ontologies", Proceedings of ER 2008, Springer Verlag, 2008.

[3] E. Blomqvist, "State of the Art: Patterns in Ontology Engineering", Research Report 04:8, ISSN 1404, Jönköping University, 2004.

[4] E. Blomqvist, "Fully Automatic Construction of Enterprise Ontologies Using Design Patterns: Initial Method and First Experiences", Proceedings of OTM 2005 Conferences, Ontologies, Databases, and Applications of Semantics (ODBASE), Springer, 2005.

[5] A. Gangemi et al., "NeOn: A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies", NeOn Integrated Project EU-IST-027595, 2006 to 2008.

[6] N. Guarino, "Formal Ontology in Information Systems", Proceedings of FOIS'98, IOS Press, 1998.

[7] J.A. Bateman, "On the Relationship Between Ontology Construction and Natural Language: A Socio-Semiotic View", International Journal of Human-Computer Studies, Volume 43, Issue 5-6, Academic Press, 1995, pp 929 – 944.

[8] R. Studer, V.R. Benjamins and D. Fensel, "Knowledge Engineering: Principles and Methods", Data & Knowledge Engineering, Volume 25, Issue 1-2, Elsevier Science Publishers B.V., 1998, pp. 161– 197.

[9] M. Rebstock, J. Fengel and H. Paulheim, "Ontologies- Based Business Integration", Springer-Verlag, 2008.

[10] M. Uschold and M. Gruninger, "Ontologies and Semantics for Seamless Connectivity", Vol. 33, No. 4, ACM SIGMOD Record, 2004.

[11] D. Allemang, R. Hodgson and I. Polikoff. (2005), "FEA Reference Model Ontology" Available at: http://semanticcommunity.info/Build_TOGAF_in_the_Cloud/FEA_Referenc e_Model_Ontology_(FEA_RMO) (Access Date: 02/05/2010).

[12] O. Corcho, M. F. López and A. G. Pérez. "Ontologies for Software Engineering and Software Technology", Springer Verlag, 2006, Ch. 1, PP. 1 – 39.

[13] N. Guarino, "Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration", in Proc. SCIE, Springer Verlag, 1997, PP. 139-170.

[14] N. Guarino. (1996, Oct 5), Understanding, Building and Using Ontologies. Available at: http://ksi.cpsc.ucalgary.ca/KAW/KAW96/guarino/guarino.html (Access Date: 08/11/2010).

[15] Ó. Corcho, A. Gómez-Pérez, R. González-Cabero, and M.D.C. Suárez-Figueroa, "ODEVAL: A Tool for Evaluating RDF(S), DAML+OIL and OWL Concept Taxonomies", in First IFIP Conference on Artificial Intelligence Aplicaciones and Innovaciones, Springer Verlag, 2004, pp.369-382.

[16] E. Blomqvist, "Ontology Patterns - Typology and Experiences from Design Development," The 26th annual workshop of the Swedish Artificial Intelligence Society, Linköping University Electronic Press, 2010, PP. 15-24.

[17] H. Pérez-Urbina, I. Horrocks, and B. Motik, "Efficient Query Answering for OWL 2", in Proc. International Semantic Web Conference, Association for Computer Machinery, 2009, pp.489-504.

[18] D. Jones, T.B. Capon and P. Visser, "Methodologies For Ontology Development", In Proc. IT&KNOWS Conference of the 15th IFIP World Computer Congress, Chapman-Hall, 1998.

[19] M.S. Fox and M. Grüninger, "Enterprise Modeling", presented at AI Magazine, AAAI Press, 1998, pp.109-121.

[20] M.D.C. Suárez-Figueroa, A. Gómez-Pérez and B. Villazón-Terrazas, "How to Write and Use the Ontology Requirements Specification Document", in Proc. OTM Conferences (2), Springer-Verlag, 2009, pp.966-982.

[21] H. S. Pinto, C. Tempich and S. Staab, "Handbook on Ontologies", Second Edition, Springer-Verlag, 2009.

[22] M. C. S. Figueroa, K. Dellschaft, E. M. Ponsoda, B. V. Terrazas, Z. Yufei, G. A. de Cea, A. García, M. F. López, A. G. Pérez, M. Espinoza and M. Sabou, "D5.4.1. NeOn Methodology for Building Contextualized Ontology Networks", NeOn Project, available at http://www.neon-project.org, February, 2008.

[23] A. Kalyanpur, B. Parsia and J. Hendler, "A Tool for Working with Web Ontologies", International Journal on Semantic Web and Information Systems, IGI Publishing, 2005, pp. 36-49.

[24] http://kaon2.semanticweb.org/ (Access date: 29/1/2011).

[25] E. Blomqvist, "Semi-automatic Ontology Construction based on Patterns", Proceedings of ISWC 2007, Springer, 2007.

[26] TopBraid Suite™ Semantic Web Solutions Platform (2007), Available at: http://topquadrant.com/products/TB_Suite.html (Access Date 30/01/2011).

[27] M. Erdmann and W. Waterfeld, "NeOn Toolkit Description", Available at http://neon-project.org, 2005. (Access Date: 03/12/2010)

[28] Protégé platform (2011), available at: http://protege.stanford.edu/ (Access Date 30/01/2011).

[29] P.Doran, "Ontology Reuse via Ontology Modularisation", Proceedings of KnowledgeWeb, PhD Symposium, L3S Research Center, 2006.

[30] V. Prabhu, S. Kummara and M. Kamath, "Scalable Enterprise Systems", Kluwer Academic Publication, MA, 2003.

[31] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad and M. Stal, "Pattern-Oriented Software Architecture, A System of Patterns", Chichester, UK, John Wiley & Sons, 1996.

[32] James W Cooper, "The Design Patterns Java companion", Addison-Wesley, 1998.

[33] V. Devedzic, "Ontologies: borrowing from software patterns", presented at Intelligence, Elsevier B.V, 1999, pp.14-24.

[34] Object Oriented Design Pattern: Available at http://www.oodesign.com (Access Date 12/01/2011).

[35] E. B. Fernandez and Y. Liu, "The Account Analysis Pattern", Proceedings of the 7th European Conference on Pattern Languages of Programs, 2002.

[36] A. Oluyomi, "Patterns and Protocols for Agent-Oriented Software Development," Ph.d Thesis, The University of Melbourne, Intelligent Agent Laboratory, November, 2006.

[37] L. Silverston "The Data Model Resource Book", Vol. 1, Wiley & Sons, 2009.

[38] D. C. Hay, "Data Model Patterns - Conventions of Thought", Dorset House Publishing, 1996.

[39] David C. Hay., "Comparison of Techniques: A Comparison of Data Modeling Techniques", Essential Strategies Inc, 1999.

[40] E. B. Fernandez, "Building Systems Using Analysis Patterns", Proceedings of the 3rd International Software Architecture Workshop (ISAW3), Association for Computing Machinery, Inc., 1998.

[41] E. Blomqvist and K. Sandkuhl, "Patterns in Ontology Engineering: Classification of Ontology Patterns", Proceedings of the Seventh International Conference on Enterprise Information Systems, ISBN 972-8865-19-8, 2005.

[42] R. C. Martin, (2000) "Design Principles and Design Patterns", www.objectmentor.com. (Access Date: 04/03/2010)

[43] K. Williamson, "Research Methods for Students and Professionals", 2nd Edition, Centre for Information Studies, 2002.

[44] B. Hancock, "Trent Focus for Research and Development in Primary Health Care: An Introduction to Qualitative Research", Trent Focus, 1998.

[45] Qualitative Research Methods, Available at: http://projects.exeter.ac.uk/prdsu/helpsheets/Helpsheet09-May03-Unlocked.pdf. (Access Date: 05/08/2010)

[46] N .Marsland, I.Wilson, S. Abeyasekera and U. Kleth, "A methodological framework for combining quantitative and qualitative survey methods", Statistical Services Centre, University of Reading, 1999.

[47] W. Olsen, "Developments in Sociology", Causeway Press, 2004, p. 300.

[48] R. E. Johnson, B. Foote and V. F. Russo, "Reusable Object-Oriented Design", Technical Report, Purdue University, 1991.

[49] V. Presutti, E. Daga, A. Gangemi and E. Blomqvist. "eXtreme Design with Content Ontology Design Patterns", Proceedings of the Workshop on Ontology Patterns WOP, collocated with the 8th International Semantic Web Conference (ISWC-2009), Springer-Verlag, 2009.

[50] E. B. Fernandez and X. Yuan, "An Analysis Pattern for Reservation and Use of Reusable Entities", Proceedings of PLoP'99, Springer, 1999.

[51] E. Gamma, R. Helm, R. Johnson and J. Vlissides, "Design Patterns – Elements of Reusable Object-Oriented Software", Addison-Wesley Professional Computing Series, Addison-Wesley, 1995.

[52] H. Kampffmeyer and S. Zschaler, "Finding the Pattern You Need, The Design Pattern Intent Ontology", Proceedings of MoDELS, Springer, 2007. pp. 211-225.

[53] A. Gangemi, "Ontology Design Patterns for Semantic Web Content", Proceedings of International Semantic Web Conference, Springer, 2005. pp.262-276.

[54] http://www.sics.se/ktm/kads.html (Access date: 29/1/2011).

[55] http://ontologydesignpatterns.org/wiki/Category:AlignmentOP (Access date: 15/02/2011).

[56] http://ontologydesignpatterns.org/wiki/Submissions:Classification (Access date: 31/1/2011).

[57] Urša Reja, Katja Lozar Manfreda, Valentina Hlebec, and Vasja Vehovar, "Open-ended vs. Close-ended Questions in Web Questionnaires", Developments in Applied Statistics, Ljubljana, Slovenia, 2003.

8 Appendix

Appendix A: Questionnaire of Surveys

To help the reader understand our thesis, in Appendix A we have included the questionnaire of the two surveys that were conducted for the research. The first survey was hosted on the *Ping Pong* system of the Jonkoping University. The second survey had three different variants. The survey randomly generated a single variant (i-e Graphical Representation, General Description or Scenario) for each participant. Participants of the second survey were sent the emails with the link of survey.

First Survey

This survey is about Ontology Design Patterns. Below is a description of one of the ontology design patterns, called AgentRole. The description is structured according to a template, with different headings containing different information. The AgentRole ontology pattern is a content ontology pattern. It is used to represent agents and the roles they play. The pattern and its description you see below have been copied from an online portal, i.e. the ODP portal at http://ontologydesignpatterns.org.

Now, imagine that you are going to build an ontology about people at the university, and that you are looking for ODPs to reuse. In a university, a person can have multiple roles, such as being a teacher as well as a researcher.

Your first task is to read the description carefully and understand different parts of the AgentRole pattern. The second task will be to answer a questionnaire based on your observations.

AgentRole

1 Graphical representation
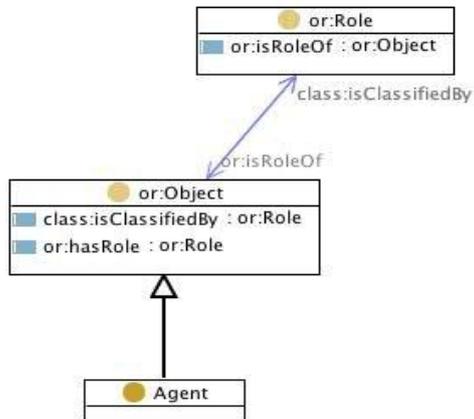
General Description

Name:	agent role
Submitted by:	ValentinaPresutti
Also Known As:	
Intent:	To represent agents and the roles they play.
Domains:	Management, Organization, Scheduling
Competency Questions:	▪ Which agent does play this role?
	▪ What is the role that played by that agent?
Solution description:	Stub
Reusable OWL Building Block:	http://www.ontologydesignpatterns.org/cp/owl/agentrole.owl (76)
Consequences:	This CP allows designers to make assertions on roles played by agents without involving the agents that play that role and vice versa. It does not allow to express temporariness of roles.
Scenarios:	She greeted us all in her various roles of mother, friend, and daughter.
Known Uses:	
Web References:	
Other References:	
Examples (OWL files):	http://www.ontologydesignpatterns.org/cp/examples/agentrole/ex1.owl
Extracted From:	http://www.loa-cnr.it/ontologies/DUL.owl
Reengineered From:	
Has Components:	
Specialization Of:	Submissions:Objectrole
Related CPs:	

Elements

*The **AgentRole** Content OP locally defines the following ontology elements:*

Agent (owl:Class) Any agentive Object, either physical, or social.

66

 Agent page
Additional information

The time indexed person role CP allow s to represent temporariness of roles played by persons. It can be generalized

for including objects or, alternatively the n-ary classification CP can be specialized in order to obtain the same

expressivity.

The elements of this Content OP are added w ith the elements of its components and/or the elements of the Content

OPs it is a specialization of.

Scenarios

Scenarios about AgentRole

■   Aldo Gangemi is a senior researcher. He is also father and a saxophonist.
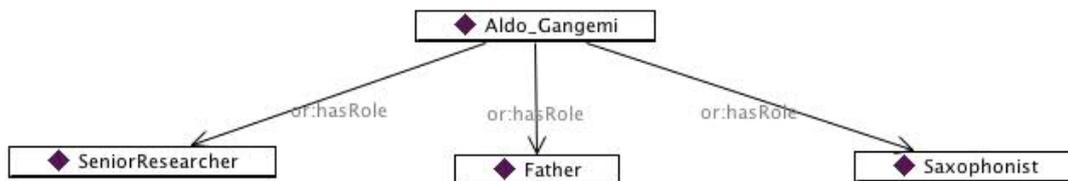
Questionnaire

Q1. How well did you understand the Agent Role pattern?

- o   Not at all
- o   I understand the overall idea but not the details
- o   I understand most of it but there are some details that are unclear
- o   I understand the pattern completely and could reuse it immediately

Q2. How important are, in your opinion, different parts (the sections containing information) in the template for understanding the ontology design pattern?

	Very Important	Important	Neither important nor unimportant	Not important	Not at all important
Graphical Representation	□	□	□	□	□
General Description	□	□	□	□	□
Element	□	□	□	□	□
Additional Information	□	□	□	□	□
Scenario	□	□	□	□	□

 Agent page

Q3. Are there any parts in the template that you think are not needed?

	Not Needed	Needed
Graphical Representation	☐	☐
General Description	☐	☐
Element	☐	☐
Additional Information	☐	☐
Scenario	☐	☐

Q4. How important are, in your opinion, the following parts (the single entries with information) in the General Description section of the template?

	Very Important	Important	Neither important nor unimportant	Not important	Not at all important
Intent	☐	☐	☐	☐	☐
Domain☐	☐	☐	☐	☐	☐
Competency Questions	☐	☐	☐	☐	☐
Reusable OWL Building Block	☐	☐	☐	☐	☐
Consequences	☐	☐	☐	☐	☐
Example (OWL files)	☐	☐	☐	☐	☐

Q5. Is there any particular part (the single entries with information) in the General Description section of the template that you think is definitely not needed?

Q6. Is there anything missing in the overall template of the design pattern? Would you need some additional information to reuse the pattern, and in that case what information?

Q7. Do you have any suggestions or proposals for improvement of the current design pattern template and presentation?

Second Survey

The purpose of this section is to identify the most important information for understanding Ontology Design Pattern.

Below, there is a scenario and a list of competency questions which is followed by a list of patterns. Competency questions are commonly used to specify the requirements of an ontology. A "Scenario" of each pattern can viewed in a separate window by clicking on the pattern. Your task is to read the competency questions (CQs), representing the requirements of an imagined ontology, carefully and then answer the questionnaire by selecting one or more correct patterns for each CQ using checkboxes.

The context gives you a setting for the task, i.e. why you are creating this ontology, and the story gives you some example information that we might want to store, while the CQs represent the actual queries the ontology should be able to answer.

Context
The national association for promotion of theater in Italy wants to set up a web-based system for keeping track of details about theater productions and the actors at different theaters. In order to support reasoning about the productions, the system should be based on an ontology. Below are some typical situations that should be representable in the ontology, and requirements in the form of competency questions.

Story
During each year a number of theatre festivals are held in cities around Italy. In January 2007 a festival called "Roma Loves Shakespeare" took place in Rome. Two different productions of "The Merchant of Venice" participated, one from a theatre in Pisa and the other from a theatre institute in Venice, featuring an ensemble of university art students. Other plays were Othello and a Midsummer Night's Dream.
Fabio Bianchi is an Italian actor employed at the theatre since May 2004, he is a part of the ensemble setting up the Merchant of Venice and he plays the Duke of Venice but also a servant in one of the scenes. During the second and third week of September the role of Shylock is played by Arnold Schwarzenegger as a special guest actor.

Competency Questions

1. When did a certain theatre festival take place?
2. Where did a certain festival take place?
3. What productions could be seen during a certain theatre festival?

4. What roles does a certain person have within a certain production during a certain time?

Scenarios for Patterns

Click on the pattern so see its scenario. This is a part of the complete pattern presentation template of the ODP portal, hence, not the complete description, but it has been selected to test the understandability of this particular style of description.

- ActingFor Pattern
- AgentRole Pattern
- Classification Pattern
- Collection Pattern
- Information Realization
- Situation Pattern
- Place Pattern
- Time Interval Pattern
- Type of Entities Pattern

Questionnaire

Q1. When did a certain theatre festival take place?
Q2. Where did a certain festival take place?
Q3. What productions could be seen during a certain theatre festival?

☐ ActingFor

☐ AgentRole

☐ Classification

☐ Collection

☐ Information Realization

☐ Situation

☐ Place

☐ Time Interval

☐ Type of Entities

Q4. What roles does a certain person have within a certain production during a certain time?

☐ ActingFor

☐ AgentRole

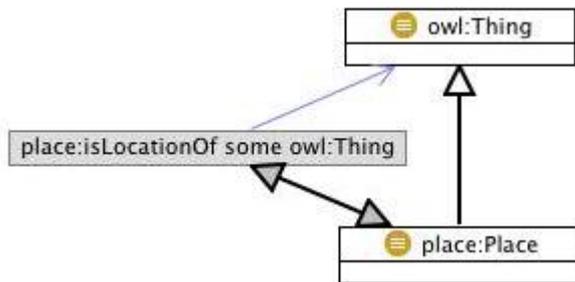☐ Classification

☐ Collection

☐ Information Realization

 Situation

 Place

 Time Interval

 Type of Entities

Appendix B: Second Survey - Patterns Sections

Here is the description of the each variant of the second survey. First are the graphical representations of each pattern included in the survey, next are the scenarios of the patterns and the last section contains the general description part of each pattern. All these parts were imported from the web portal http://ontologydesignpatterns.org.

Graphical Representation of Patterns
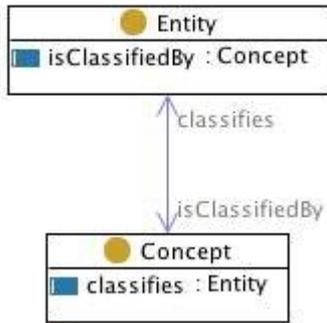
Place Pattern

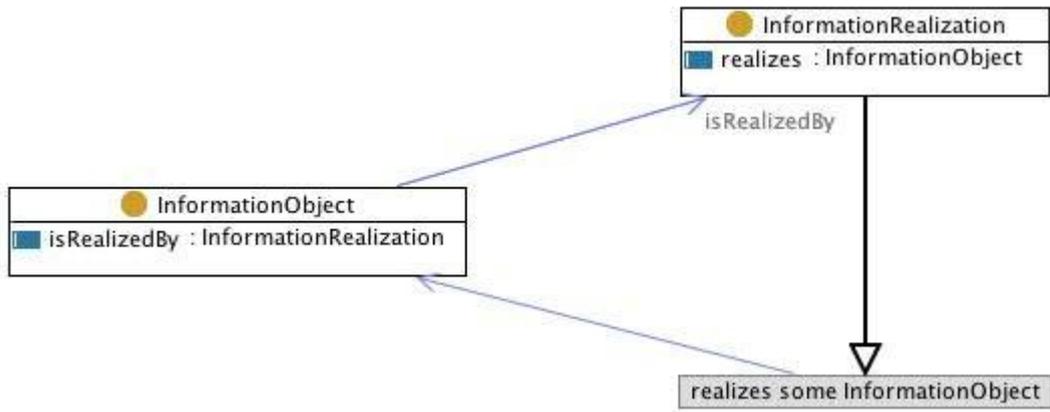
Agent Role Pattern
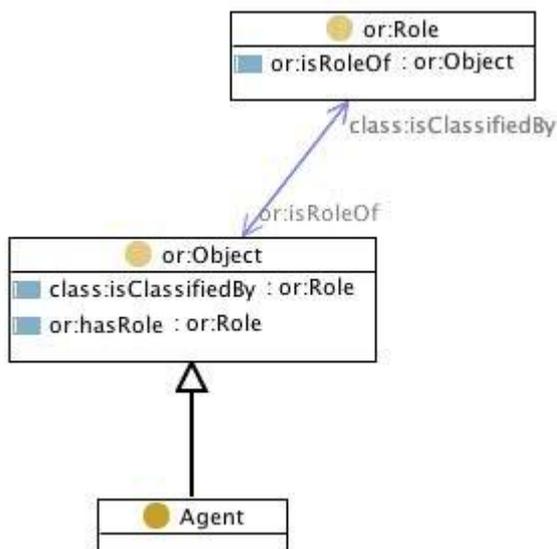
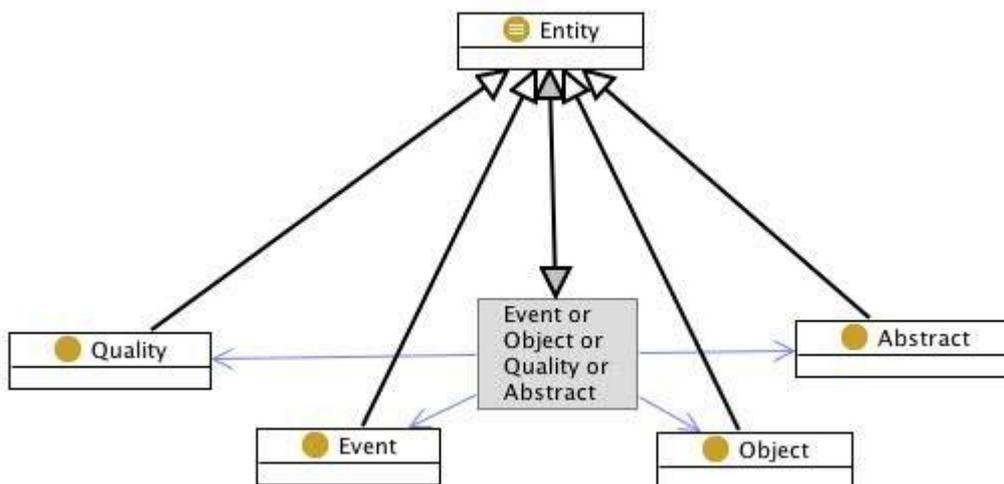## Classification Pattern
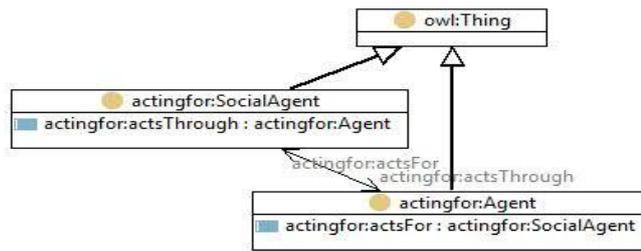

Information Realization Pattern

Type of Entity Pattern

Acting For Pattern

Collection Pattern
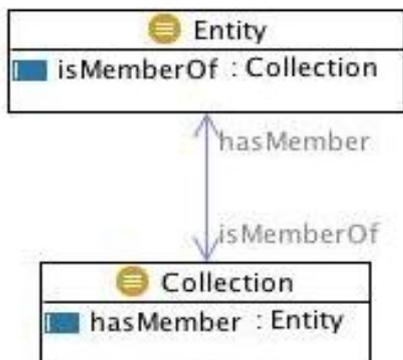
Situation Pattern
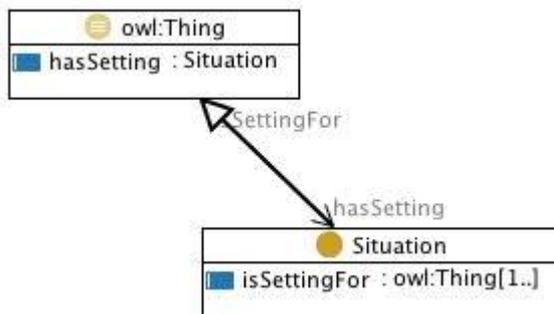
Time Interval Pattern

Scenario of Patterns

Place Pattern

The Colosseum is located in Rome
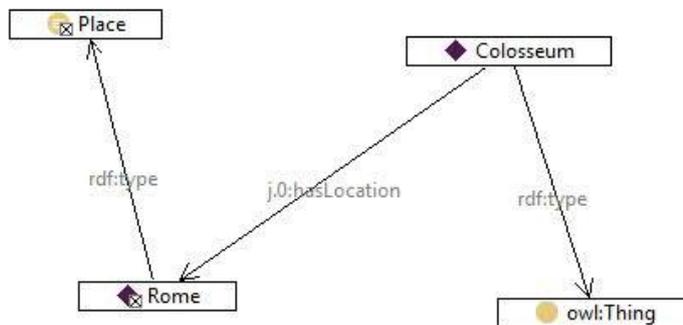
Agent Role Patterns
Aldo Gangemi is a senior researcher. He is also father and a saxophonist.
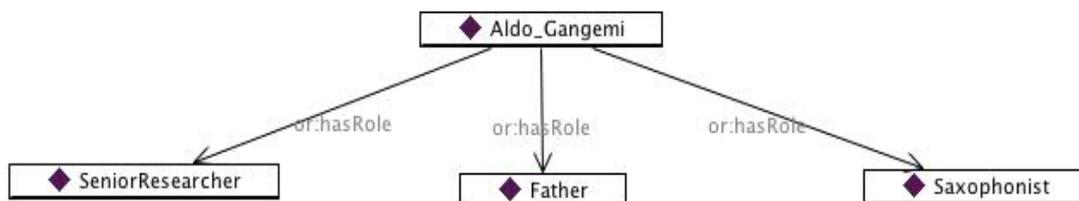
Classification Pattern

Mac OSX 10.5 is classified as an operating system.
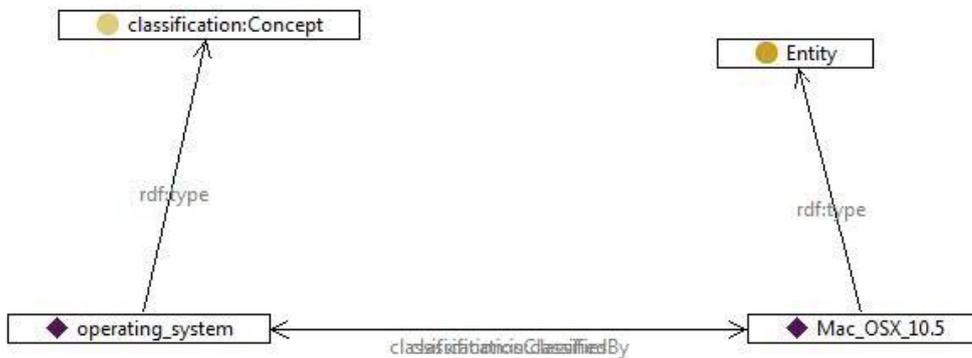
Information Realization Pattern

The book of the "Divina Commedia".
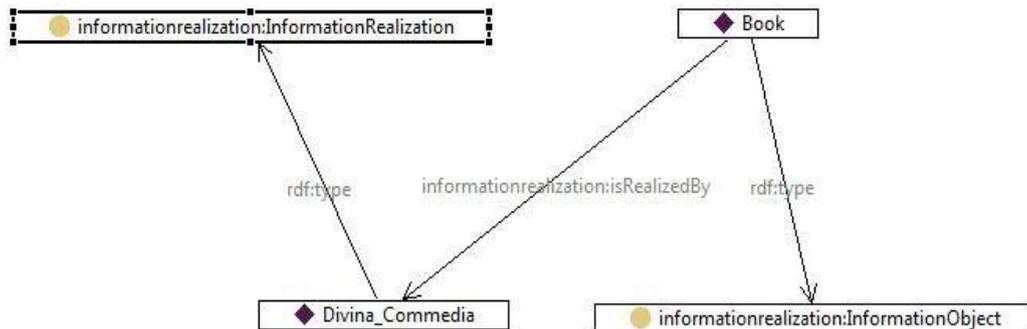
Type of Entity Pattern

That copy of Divina Commedia is a book (Object), the porosity (Quality) of the paper used is 5 mls/min (Abstract). The Rock Music Festival (Event) is organized by a friend of mine (Object).
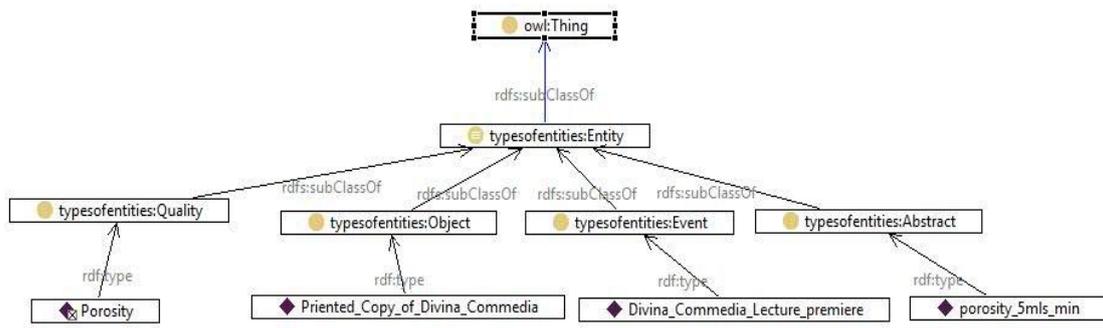
Acting For Pattern
Matteo Sanvitale is working as an officer for CEMA s.r.l
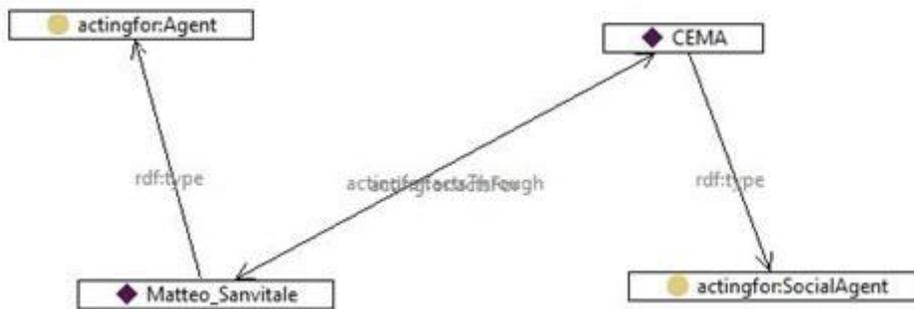
Collection Pattern

My saxophone collection includes a Mark VI tenor, a Balanced Action alto, and a Conn Transitional bari.
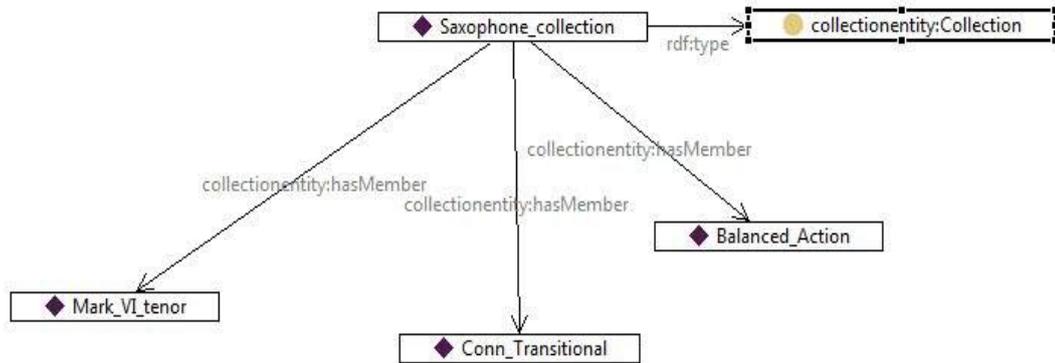
Situation Pattern

I prepared a coffee with my heater, 300 ml of water, and an Arabica coffee mix.

Time Interval Pattern

The time interval "January 2008" starts at 2008–01–01 and ends at and ends at 2008–01–31.

General Description of Patterns

Agent Role Patterns

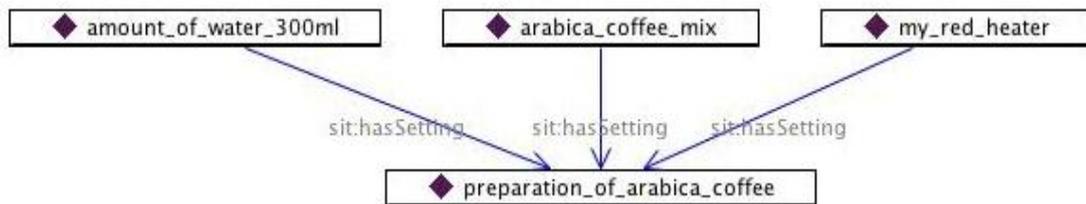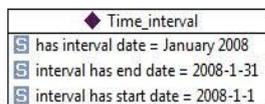|	
Name:	agent role
Submitted by:	ValentinaPresutti
Also Known As:	
Intent:	To represent agents and the roles they play.
Domains:	
	Management, Organization, Scheduling
Competency Questions:	
	▪ which agent does play this role?
	▪ what is the role that played by that agent?
Solution description:	stub
Reusable OWL Building Block:	http://www.ontologydesignpatterns.org/cp/owl/agentrole.owl (117)
Consequences:	This CP allows designers to make assertions on roles played by agents without involving the agents that play that roles, and vice versa. It does not allow to express temporariness of roles.
Scenarios:	She greeted us all in her various roles of mother, friend, and daughter.
Known Uses:	
Web References:	
Other References:	
Examples (OWL files):	
	http://www.ontologydesignpatterns.org/cp/examples/agentrole/ex1.owl
Extracted From:	
	http://www.loa-cnr.it/ontologies/DUL.owl
Reengineered From:	
Has Components:	
Specialization Of:	
	Submissions:Objectrole
Related CPs:	

Place Pattern

Name:	Place
Submitted by:	AldoGangemi
Also Known As:	
Intent:	To talk about places of things.
Domains:	
	General
Competency Questions:	
	Where is a certain thing located? What is located at this place?
Solution description:	This is a basic pattern, useful to represent generic locations for anything, which becomes a place when is assumed as a reference location.
Reusable OWL Building Block:	http://www.ontologydesignpatterns.org/cp/owl/place.owl (83)
Consequences:	We can represent, transitively, where something is located. It remains unspecified what kind of location relation we are trying to represent: reference location, partial location, physical location, social or metaphoric location, etc.
	Moreover, temporal location is not caught with this pattern

(you need a placement situation for that).

Scenarios:	The Colosseum is located in Rome.
Known Uses:	
Web References:	
Other References:	
Examples (OWL files):	
Extracted From:	http://www.ontologydesignpatterns.org/ont/dul/DUL.owl
Reengineered From:	
Has Components:	
Specialization Of:	
Related CPs:	

Classification Pattern

Name:	Classification
Submitted by:	ValentinaPresutti
Also Known As:	
Intent:	To represent the relations between concepts (roles, task, parameters) and entities (person, events, values), which concepts can be assigned to. To formalize the application (e.g. tagging) of informal knowledge organization systems such as lexica, thesauri, subject directories, folksonomies, etc., where concepts are first-order elements.
Domains:	General
Competency Questions:	▪ What concept is assigned to this entity? ▪ Which category does this entity belong to?
Solution description:	-
Reusable OWL Building Block:	http://www.ontologydesignpatterns.org/cp/owl/classification.owl (80)
Consequences:	It is possible to make assertions about e.g., categories, types, roles, which are typically considered at the meta-level of an ontology. Instances of Concept reify such elements, which are therefore put in the ordinary domain of an ontology. It is not possible to parametrize the classification over different dimensions e.g., time, space, etc.
Scenarios:	Mac OSX 10.5 is classified as an operating system in the Fujitsu-Siemens product catalog.
Known Uses:	
Web References:	
Other References:	
Examples (OWL files):	
Extracted From:	▪ http://www.loa-cnr.it/ontologies/DUL.owl
Reengineered From:	
Has Components:	
Specialization Of:	
Related CPs:	

Information Realization Pattern

Name:	information realization
Submitted by:	ValentinaPresutti
Also Known As:	
Intent:	To represent information objects and their physical realization.
Domains:	
	Semiotics
Competency Questions:	
	▪ what are the physical realizations of this information object?
	▪ what information objects are realized by this physical object?
Solution description:	This is a basic patterns, representing the difference between abstract and realized (manifested, concrete, etc.) information.
Reusable OWL Building Block:	http://www.ontologydesignpatterns.org/cp/owl/informationrealization.o
Consequences:	This pattern allows to distinguish information objects from their concrete realizations.
Scenarios:	The book of the "Divina Commedia".
Known Uses:	
Web References:	
Other References:	
Examples (OWL files):	
	http://www.ontologydesignpatterns.org/cp/examples/informationrealizat
Extracted From:	
	http://www.ontologydesignpatterns.org/ont/dul/ontologies/DUL.owl
Reengineered From:	
Has Components:	
Specialization Of:	
Related CPs:	

Type of Entity Pattern

Name:	Types of entities
Submitted by:	ValentinaPresutti
Also Known As:	
Intent:	To identify and categorize the most general types of things in the domain of discourse.
Domains:	
	General
Competency Questions:	
	▪ What kind of entity is that?
	▪ Is this an event or an object?
	▪ Is this an abstract value or a quality of an entity?
Solution description:	---
Reusable OWL Building Block:	http://www.ontologydesignpatterns.org/cp/owl/typesofentities.owl (22)
Consequences:	The type of any element of the knowledge base is always known.
Scenarios:	
Known Uses:	
Web References:	

Other References:
Examples (OWL files):
Extracted From:

http://www.loa-cnr.it/ontologies/DUL.owl

Reengineered From:
Has Components:
Specialization Of:
Related CPs:

Acting For Pattern

Name:	ActingFor
Submitted by:	AldoGangemi
Also Known As:	
Intent:	To represent that some agent is acting in order to forward the action of a social (non-physical) agent.
Domains:	
Competency Questions:	

- Who is working for which organization?
- Who is representing the company?

Solution description:	-
Reusable OWL Building Block:	http://www.ontologydesignpatterns.org/cp/owl/actingfor.owl (18)
Consequences:	An ontology designer is able to express relations like delegation, working for, etc. It is not possible to express either time indexing (the situation pattern should be specialized to that purpose), nor the role or task, under which the social action is carried out by the physical agent (the description and situation pattern should be used instead).
Scenarios:	Matteo Sanvitale is working as an officer for CEMA s.r.l.
Known Uses:	
Web References:	
Other References:	
Examples (OWL files):	
Extracted From:	
Reengineered From:	

http://www.ontologydesignpatterns.org/ont/dul/DUL.owl

Has Components:
Specialization Of:
Related CPs:

Collection Pattern

Name:	Collection
Submitted by:	AldoGangemi
Also Known As:	membership, collection entity
Intent:	To represent domain (not set theory) membership.
Domains:	

General

Competency Questions:	

- What things are contained in this collection?
- What collections this thing is member of?

Solution description:	A class collection represents the concept of a set of entities

(things). Things are members of the collection.

Reusable OWL Building Block:	http://www.ontologydesignpatterns.org/cp/owl/collectionentity.owl (69)
Consequences:	Collections and their members can be associated. Time-indexed membership cannot be represented though (you need a situation-based pattern).
Scenarios:	My saxophone collection includes a Mark VI tenor, a Balanced Action alto, and a Conn Transitional bari.
Known Uses:	
Web References:	
Other References:	
Examples (OWL files):	
Extracted From:	http://www.ontologydesignpatterns.org/ont/dul/DUL.owl
Reengineered From:	
Has Components:	
Specialization Of:	
Related CPs:	Submissions:TimeIndexedMembership

Situation Pattern

Name:	Situation
Submitted by:	AldoGangemi
Also Known As:	situation
Intent:	To represent contexts or situations, and the thing
Domains:	
Competency Questions:	▪ What is the context or situation of something?
	▪ What are the things present in this context or situation?
Solution description:	-
Reusable OWL Building Block:	http://www.ontologydesignpatterns.org/cp/owl/situation.owl
Consequences:	We can contextualize things that have something in common, or are associated: a same place, time, view, causal link, systemic dependence, etc.
	We can also reify n-ary relations as situations.
Scenarios:	The lecture was held in January 1921 by Bela Fleck, with some physicians in the audience making questions, in a very relaxed athmosphere.
Known Uses:	
Web References:	
Other References:	
Examples (OWL files):	
Extracted From:	http://www.ontologydesignpatterns.org/ont/dul/DUL.owl
Reengineered From:	
Has Components:	
Specialization Of:	
Related CPs:	

Time Interval Pattern

Name:	time interval
Submitted by:	ValentinaPresutti
Also Known As:	

Intent:	To represent time intervals.
Domains:	
	Time
Competency Questions:	▪ What is the end time of this interval?
	▪ What is the starting time of this interval?
	▪ What is the date of this time interval?
Solution description:	--
Reusable OWL Building Block:	http://www.ontologydesignpatterns.org/cp/owl/timeinterval.owl (73)
Consequences:	The dates of the time interval are not part of the domain of discourse, they are datatype values. If there is the need of reasoning about dates this Content OP should be used in composition with the region Content OP.
Scenarios:	The time interval "January 2008" starts at 2008−01−01 and
	ends at and ends at 2008−01−31.
Known Uses:	
Web References:	
Other References:	
Examples (OWL files):	http://www.ontologydesignpatterns.org/cp/examples/timeinterval/january2008.owl
Extracted From:	
Reengineered From:	
Has Components:	
Specialization Of:	
Related CPs:	

Appendix C: Survey Results

The Appendix C includes all the answers from the two surveys. The first survey was conducted with the students who studied Information Logistics course at Jonkoping University during session 2008 and 2009. The survey was sent to 45 people. A total 17 students answered the first survey.

The second survey was conducted with the quality committee of the ontologydesignpatterns.org portal. Emails about survey were sent to the 40 members. Nine members answered the second survey.

The results of the multiple choice questions are presented in the tables with percentages of responses. The answers of the open ended questions are presented in the form of lists.

Fist Survey Results

Q1. How well did you understand the AgentRole Pattern?

Not at all.	0%
I understand the overall idea but not the details.	25%
I understand most of it but there are some details that are unclear.	31%
I understand the pattern completely and would be able to reuse it immediately.	44%

Q2. How important are, in your opinion, different parts (the sections containing information) in the template for understanding the ontology design pattern?

	Very Important	Important	Neither Important nor Unimportant	Not Important	Not Important at all
Graphical Representation	60%	24%	11%	5%	0%
General Description	60%	40%	0%	0%	0%
Element	35%	48%	17%	0%	0%
Additional Information	24%	30%	41%	0%	5%
Scenario	60%	35%	5%	0%	0%

Q3. Are there any parts in the template that you think are not needed?

	Not Needed	Needed
Graphical Representation	0%	100%
General Description	6%	94%
Element	40%	60%
Additional Information	38%	62%
Scenario	25%	75%

Q4. How important are, in your opinion, the following parts (the single entries with information) in the General Description section of the template?

	Very Important	Important	Neither Important nor Unimportant	Not Important	Not Important at all
Intent	76%	24%	0%	0%	0%
Domain	41%	41%	18%		0%
Competency Questions	53%	24%	18%	5%	0%
Reusable OWL Building Block	40%	27%	20%	13%	0%
Consequences	30%	60%	5%	5%	0%
Example (OWL files)	41%	30%	24%	0%	5%

Q5. Is there any particular part (the single entries with information) in the General Description section of the template that you think is definitely not needed?

- Name
  Submitted by
  Also Known As
  Solution description
  Reusable OWL Building Block
  Scenarios
  Known Uses
  Web References
  Other References
  Extracted From
  Reengineered From
  Has Components
  Specialization Of
  Related CPs
- The known uses can be skipped as it can be covered in the scenarios as they will describe the use of pattern according to the situations so they can be considered as known uses.
- I would additional information and elements do play a vital role during when a new user considered using a ODP. The best way to grasp what ODP is about is the scenario, the diagram and also the competency questions.

- competency question
- According to my understanding and being a ontology pattern student I personally think that example (owl files) are not somehow important because you are trying for the reuse not utilizing any already file in the ontology development.
- All the sections mention by the authors are important.
- Solution description
- solution description
  scenarios
  web references
  others references
- EXAMPLE OF OWL IS NOT NEED
- consequence
- I think all the selected parts are important but if we can ignore the competency questions then we can work well because competency questions are mostly based on scenarios and may be different from one scenario to another so we can ignore the competency questions part in it.
- I think there no part is irrelevant. Description should be comprehensive.

Q6. Is there anything missing in the overall template of the design pattern? Would you need some additional information to reuse the pattern, and in that case what information?

- Template should have terms part as text like concept, relations that are going to be used in pattern. That might be good idea to present, what exactly need to reuse, and for ontology learning point of view as well.
- Annotations description
- We can add the properties information in the general description part because it can helps us a lot while reusing the pattern and its understandability.
- It is better to give explanation about scenario and diagram in a way that user can understand the diagram by looking at scenario.
- YES I THING THE PATTERN HIERARCHICAL CHART IS MISSING, WHICH SHOW THE DIFFERENT PART OF THE PATTERNS
- There is no more information needed for reusing patterns
- in the elements section, the description of the class "role" will be beneficial
- I would suggest a Plug-in GUI with protege that would automatically give suggestions to users to use a particular pattern based on intent and competency question.
- It should be better if the first description of overall description give

more extra information to reader, Since so many people who are involving to this survey want to understand the area much in detail,.

- An example of OWL if added in the template then it will be helpful for the user for understanding the pattern.
- I think so there should be included one more part of this discussion which is "Effort" because i would like to support that if you have two pattern to utilize in your work and both are very suitable for your work but we are unable to provide information about the "Utilizing effort" for the user in the discussion.

Q7. Do you have any suggestions or proposals for improvement of the current design pattern template and presentation?

- It is okay because it has graphical representation as well as text description.
- Include effort as participant in the general description. Also mention that either it is domain oriented or generic for usage in any domain.
- I think if we can add some information with the diagram to describe it that how this pattern will work.
- Yes general description is too long. We can make it short.
- AS an information management student, and as a student who study the information logistic course, I would like to ask authors to write down in their report the all step of their work and implementation really in the detail. Hence reader of the report with no idea of the domain would get enough knowledge.
- I would suggest that the scenario used in the general description is illustrated with a diagram instead of having two examples. The subtitle 'Known uses' should be also as much as possible filled in because it could help also in getting used to the ODP.
- I think some information in the "general description" section should be hidden (solution, description, scenarios, web references, others references) and the reader will show them up if needed.
- I personally think that the whole point of making ODP presentable is to make it easier or the users to understand patterns and eventually use them. However I also feel that the user needs to search for the pattern and look into it and read it to understand it. Therefore a suggestion would be an Plug-in with protege.
- AS I EXPLAIN UP THERE MUST SOME HIERARCHICAL VIEW OF THE PATTERN INCLUDE THE PRESENTATION OF PATTERNS
- There are following sections that might have more information.

Domains: Need to put more domains to have clear understanding, where we can reuse this pattern.

Consequences: In Consequences, more information required about the context, where pattern can be applied. For example, describing the concepts and relations, forces elements of this pattern, that leads to have solution of design problem. There is need to describe the limitation bit more, where exactly, this pattern is not usable or by describing different kind of roles, which are not going to be handled by this pattern.

Scenarios: I think, scenarios should not be general, need to put more specific domain scenarios.

Related CPs: Somehow, others CPs, which are more relevant or that can be attached to this pattern to have small domain ontology, that CPS should be added.

- As mentioned above a field from general description can be omitted and the Element section can be incorporated in the general description. The consequences and Scenario sections should not be placed together as it creates some problem while reading. In the scenario section a little bit more complex examples can be added as well as in the real life scenarios are not that much simple. Also link to some ontologies which are using the particular pattern can be provided so user can see its implementation in a complete scenario.
- Giving detailed explanation about the diagram can be a help for the user. This will help in understanding the pattern deeply.

Second Survey Results

Part 1

Below are the results of the Part 1 of the second survey. The tables below contain the responses by participants and the results after evaluation. Each response was evaluated by comparing it to the correct set of patterns. The correct set of patterns for the first three questions was: Place and Situation pattern. The correct set of patterns for fourth question was: AgentRole, Time Interval and Situation pattern. If a participant has selected any extra patterns apart from the correct set of patterns, the answer was still considered correct.

Q1. When did a certain theatre festival take place?
Q2. Where did a certain festival take place?
Q3. What productions could be seen during a certain theatre festival?

Part Name	Answers from Participants
Graphical Representation	Collection, Place, Time Interval, Type of Entities
	Information Realization, Place
	Information Realization, Situation, Place, Time Interval
General Description	Classification, Collection, Information Realization, Situation, Place, Time Interval
	Situation, Place, Time Interval
	Collection, Situation, Place, Time Interval
Scenario	Information Realization, Situation, Place, Time Interval
	Classification, Collection, Situation, Place, Time Interval
	AgentRole, Information Realization, Place

Part Name	Correct Answers	Wrong Answers
Graphical Representation	1	2
General Description	3	0
Scenario	2	1

Q4. What roles does a certain person have within a certain production during a certain time?

Part Name	Answers from Participants
Graphical Representation	ActingFor, AgentRole, Time Interval, Type of Entities
	ActingFor
	AgentRole, Situation, Time Interval
General Description	AgentRole, Classification, Information Realization, Situation, Time Interval
	ActingFor, AgentRole, Information Realization, Situation, Time Interval

	AgentRole, Situation, Time Interval, Type of Entities
Scenario	AgentRole, Classification, Situation, Time Interval
	AgentRole, Situation, Time Interval
	AgentRole

Part Name	Correct Answers	Wrong Answers
Graphical Representation	1	2
General Description	3	0
Scenario	2	1

Part 2

Below are the results from Part 2 of the survey:

Q1. How well did you understand the AgentRole pattern?

Not at all.	0%
I understand the overall idea but not the details.	0%
I understand most of it but there are some details that are unclear	66%
I understand the pattern completely and would be able to reuse it immediately	34%

Q2. How important are, in your opinion, different parts (the sections containing information) in the template for understanding the ontology design pattern?

	Very Important	Important	Neither Important nor Unimportant	Not Important	Not Important at all
Graphical Representation	100%	0%	0%	0%	0%
General Description	44%	56%	0%	0%	0%
Element	22%	56%	22%	0%	0%
Additional Information	22%	44%	22%	12%	0%
Scenario	56%	44%	0%	0%	0%

Q3. Are there any parts in the template that you think are not needed?

	Not Needed	Needed
Graphical Representation	0%	100%
General Description	0%	100%
Element	37%	63%
Additional Information	37%	63%
Scenario	0%	100%

Q4. How important are, in your opinion, the following parts (the single entries with information) in the General Description section of the template?

	Very Important	Important	Neither Important nor Unimportant	Not Important	Not Important at all
Intent	22%	56%	22%	0%	0%
Domain		22%	78%	0%	0%
Competency Questions	44%	56%	0%	0%	0%
Reusable OWL Building Block	56%	33%	11%	0%	0%
Consequences	22%	44%	22%	11%	0%
Example (OWL files)	44%	56%		0%	0%

Q5. Is there any particular part (the single entries with information) in the General Description section of the template that you think is definitely not needed?

- No, but I think the scenario should be as explicit as possible. This is crucial for the first reusability assessment by the user.

- Consequences: this description should be really focused on the graphical representation and general description without mentioning further aspects like in the example the missing temporal descriptions which I would not expect after reading the description and the diagram

Q6. Is there anything missing in the overall template of the design pattern? Would you need some additional information to reuse the pattern, and in that case what information?

- The configuration of the DUL (DOLCE+ Ultralite) top level structure.
  Eg in the "Time indexed person role" pattern", it is difficult to understand how Classification and Concept related to each other.
  These concepts and relations should be spelled out more clearly in the portal (or if this is already available, more explicit links should be

provided), so that users can grasp the conceptual framework they provide without having to look for the ontologies they originate from. Only then will it be possible to fully re-use the pattern.

- Maybe we could mention in the scenario somehow the binding from concrete elements to the pattern elements. While it is quite clear in this example, it could be more complex in general

- * More info on the imported patterns, and more instructive distinction of the local and imported parts
  * More rigorous and ampler set of references to other patterns, including other than direct specializations
  * Links not just to CPs but also to logical patterns

Q7. Do you have any suggestions or proposals for improvement of the current design pattern template and presentation?

- In the graphical representation, quite often namespaces are provided such as sit:Entity; nclas:Classifies. These should be made explicit in the additional information (imports) section. Also, all pattern elements should be fully defined.

- More uniform diagrammatic notation.

- Scenarios can be more clearly defined and more references should be provided to other patterns.