A Comparative Study for Ontology and Software Design Patterns

Zaheer Ahmed, Muhammad Arif(✉), Muhammad Sami Ullah, Adeel Ahmed, and Muhammad Jabbar

Department of Computer Science, University of Gujrat, Gujrat, Pakistan
{zaheer,m.arif,msamiullah,adeel.ahmed, jabbar.ahmed}@uog.edu.pk

Abstract. 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. Ontology design pattern provide the solution of quality modeling for ontologies. In user prospective the presentation of ODP play a central role for the reusability of ontologies. This research determine the 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 the template of different software engineering patterns (SEP) and ODP. On the basis of this analysis we suggest possible changes in current template 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. The ontology design pattern templates were compared with existing templates of other patterns for determine the improvement areas. The template of an ODP 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.

Keywords: Ontology design pattern · Templates · Content ontology design pattern · Software engineering pattern

1 Introduction

ODPs are semantic patterns that provide quality modeling solutions for ontologies. ODPs play an important role in learning and teaching ontology engineering [1]. They facilitate the automatic and semi-automatic ontologies construction and provide a base for creating ontologies in different domains [2]. In user prospective the reusability of ontology depends on presentation of ODP. 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, review and suggest changes in patterns [4].

In computer and information science ontology is defined as a "*formal, explicit specification of a shared conceptualization*" [3]. 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. Due to unfamiliar logical structure the existing ontologies provide limited support. Must be learned the good practices form literature. This problem is solved by implementing common solution as we learn in software engineering [4]. 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 [5]. 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 development process. In SE the common practices to build software through design and architecture patterns. This practice also follow in ontology engineering [2].

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 [4]. 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 [6]. This thesis deals with the presentation of content ontology design pattern. It describes the design issues of the presentation of ontology 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 made suggestions for the potential improvement in the current templates of ODP presentation.

2 Related Work

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 concept of program components related to the reusability of design. To achieve reusability, number of design technique available in object oriented software development model for more reusable building block. 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 expedite design by reuse, first the design for reuse process must be established [5, 7].

Reusability is applied at different levels. In software engineering, reusability can be applied at following three levels [5]:

Requirements Reuse: It deals with the models of the domain or the generic model of the requirement domain.

Design Reuse: It deals with the models, data structures and algorithms.

Software Component Reuse: It deals with reuse of software classes and editable source code.

The development process of ontology is time consuming and more effort required form developer. Reuse of an ontology from an ontological engineering perspective can be hard. This is even more when there are large ontologies to be reused [8].

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 [9].

Ontology Matching: Ontology matching is a process of judging correspondence between semantically related ontologies, solving the problem of semantic heterogeneity and can be used in ontology merging, query answering, data translation etc. So ontology matching facilitates interoperability between matched ontologies [10].

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. In reuse the pattern are best practices that provide the good design [5, 11].

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 [5].

The most common categories are the following:

- Analysis Patterns [5, 12].
- Architecture Patterns [5, 13].
- Design Patterns (see [11, 14–17]).
- Programming Language Idioms (see [3, 13]).

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 [3, 12, 18].

An overall structuring principle is used while constructing a viable software architecture. Architectural patterns are considered as templates for solid software architectures [3].

Design pattern describe the common solution of overall design problem in certain context through the depiction object and class communication. Design pattern provide the description of participation classes, their instance, their roles and interaction, distribution of responsibility. One object oriented design problem address by the certain design pattern that also provided sample code to describe the partial solution of the problem [14].

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 [13].

2.1 Problem Statement

The ontology design patterns (ODPs) have divided in grouped of six different families: Structural ODPs, Content ODPs, Reasoning ODPs, Presentation ODPs, Correspondence ODPs and Lexico Syntactic ODPs [6].

Graphically the types of patterns are represented as (Fig. 1):

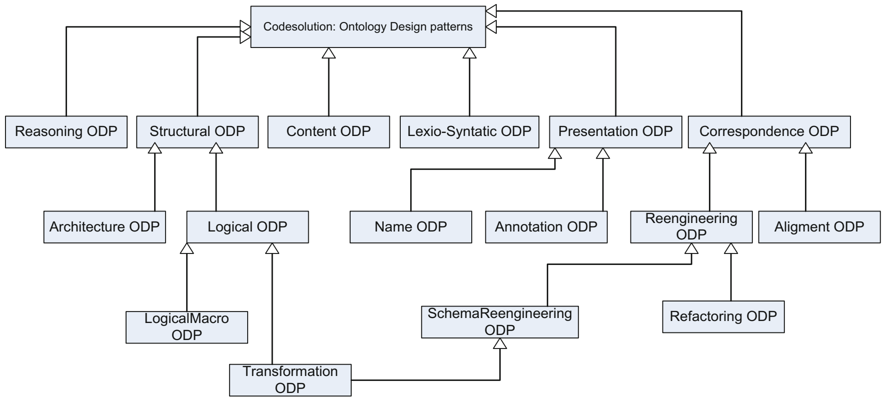


Fig. 1. Group of ontology design pattern [19]

In this paper, 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 [19].

This research guide the improvement areas in the presentation of content ODPs. Improvement in presentation can ultimately improve the understandability of a pattern from user perspective. The focus of research is to analyze different content ODPs and provide some possible recommendation in current templates of the ODP 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.

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. 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.

3 Methodologies

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 well know and common use patterns.

Software patterns have been compared to ontologies by Devedzic in one of his article [16] 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].

First, we analyze what current templates exist in other fields and then compare them to ontology design pattern templates to analyze the difference. In our study, evaluation was done on the results of the evaluation of different patterns. 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 [19].

A template of a pattern is a standard way of representing a pattern. In a broad sense, a pattern template has four important elements. These elements are: Name, Problem, Solution and Consequences [3, 11, 19].

The different kinds of pattern templates are given below with their description.

3.1 Design Pattern Template

This template proposed by Guarino [11] in their book "Design pattern Element of Reusable Object-Oriented Software" [11]. Table 1 shows the different parts of the template and their description (Table 2).

Table 1. Design pattern template [11, 19]

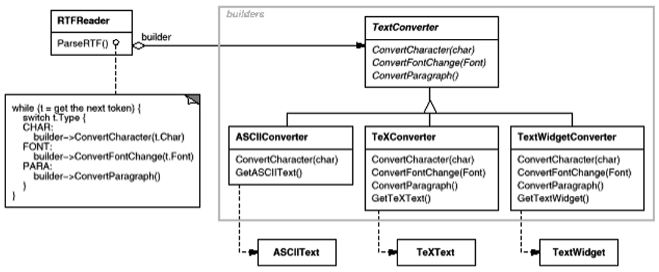
Design pattern template	
Elements	Description
Pattern name and classification	Name is a short summary of the pattern. There are several DP, we need a way to classified them in a family. The section classification refers these families of design pattern
Intent	It is a brief description that explain the following. How the design pattern work? 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 explain the class and object structures in the pattern describe the problem solution. The problem solution will facilitate you to understand the more abstract description of the pattern
Applicability	This section discus the situations in which the design pattern applicable
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?

Table 2. Builder design pattern [19]

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	Fig. 2
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	Fig. 3
Participants	Builder, ConcreteBuilder, Director, Product
Collaborations	The given interaction diagram describe how Builder and Director cooperate with a client Fig. 4
Implementation	Fig. 5

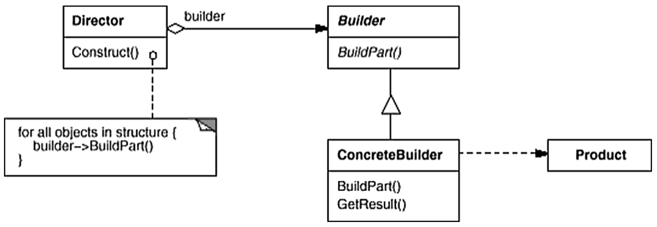
Sample code	/Abstract Builder
	class abstract class TextConverter{
	        abstract void convertCharacter(char c);
	        abstract void convertParagraph();
	}
	.
	.
	.
	.
	        public static void main(String args[]){
	            Client client = new Client();
	            Document doc = new Document();
	            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



```
while (t = get the next token) {
  switch t.Type {
  CHAR:
    builder->ConvertCharacter(t.Char)
  FONT:
    builder->ConvertFontChange(t.Font)
  PARA:
    builder->ConvertParagraph()
  }
}
```

Fig. 2. Builder design pattern



```
for all objects in structure {
    builder->BuildPart()
}
```

Fig. 3. Builder design pattern

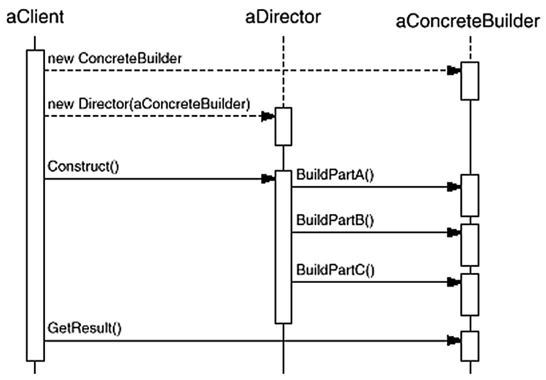


Fig. 4. Sequence diagram

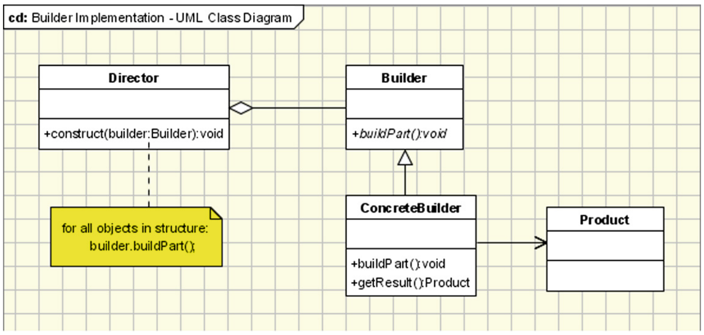


Fig. 5. Class diagram

3.2 Analysis Pattern Templates

Given below is the Analysis Pattern template described by Fernandez and Liu in their article "The Account Analysis Pattern" [20]. This template is also described in the book "Pattern-Oriented Software Architecture" [21] (Tables 3 and 4).

Table 3. Analysis pattern template [19, 21]

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 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?

Table 4. Account analysis pattern [19, 22]

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 Fig. 6
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 Fig. 7
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

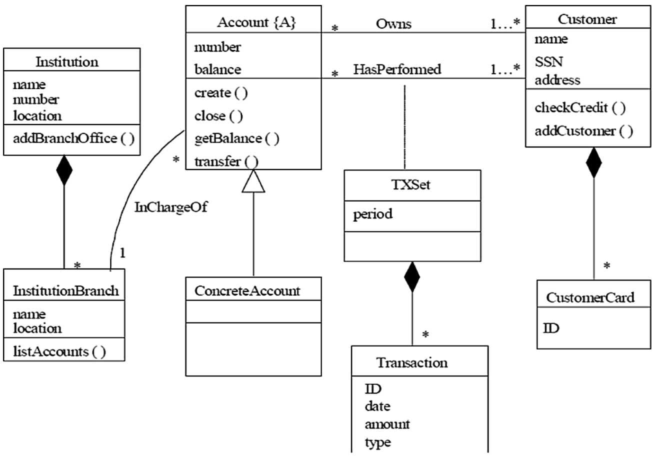


Fig. 6. Class diagram

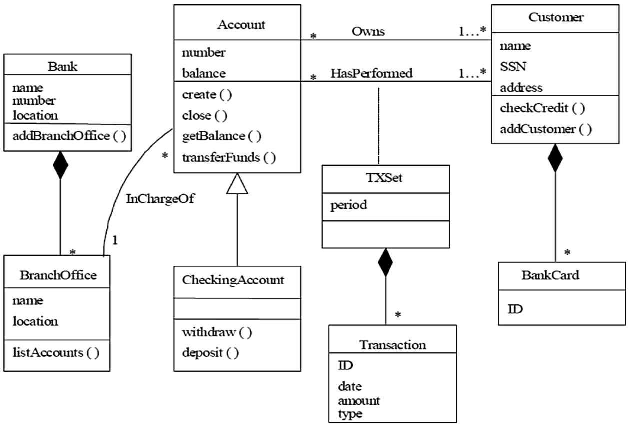


Fig. 7. Class diagram

3.3 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 [22] (Table 5).

Example
The pattern description has been slightly abbreviated for readability issues (Table 6).

Table 5. Architecture pattern template [19, 21]

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?

Table 6. Agent as delegate pattern [19, 22]

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	Know Uses section mentions the use of the pattern in various scenarios. It includes minimum two examples from different domains
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

3.4 Data Model Patterns

Data Model Patterns help modelers to develop quality models by standardizing common and well-tested solutions for reuse [3]. The objective of data model pattern is to explain a starting point for data modelers [23].

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 [24]. It is a theoretical model which explains the basic principles of a data model pattern. See Fig. 8:

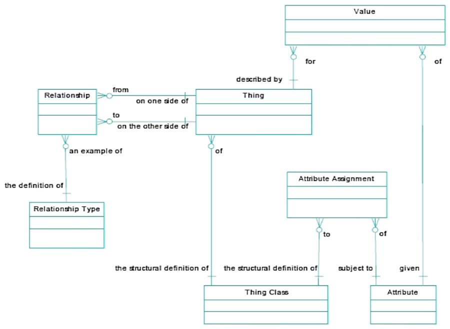


Fig. 8. Universal data model [19, 23]

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 [19].

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 [25].

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 [25].

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 [25].

3.5 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 [2, 5, 26] (Table 7):

Table 7. Content ontology design pattern template [2, 5, 19]

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

(*continued*)

Table 7. (*continued*)

Elements	Description
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

(See Table 8).

Table 8. Classification pattern [4, 19]

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?

(*continued*)

Table 8. (*continued*)

Elements	Description
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	Fig. 9
Additional information	

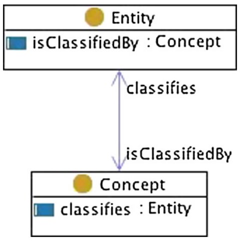


Fig. 9. Class diagram

4 Study Analysis and Results

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.

4.1 Comparison of Pattern Templates

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 Analysis Pattern is implemented by using a UML Class diagram, Sequence Diagram and State Diagram which were described in section. 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].

4.2 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 this chapter in Sects. 3.1 to 3.5, 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 Sect. 3.

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 9 are considered as basic parts of a pattern. The description of these parts is stated. The table shows how these basic parts are described in software patterns and ODP templates.

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

Table 9. Representing basic elements of other patterns [19]

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	Not provided	Sample code
Architecture pattern	Context	Problem	Solution	Not provided	Not provided

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.

5Consequence: 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 10 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 Sect. 2.3.2. The section motivation of design pattern is similar to the section problem of other software patterns.

In Table 10 the vertical line had different columns that shows the label of different pattern. The horizontal line shows the template heading and symbol X describes the presence of common element.

Table 10. Representing common elements of other patterns [19]

Template heading	Ontology design pattern	Analysis pattern	Design pattern	Architecture pattern
Name	×	×	×	×
Known uses	×	×	×	×
Intent	×	×	×	×
Consequences	×	×	×	
Also known as	×		×	
Classification			×	×

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 Sect. 2.5.

4.3 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 11, 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.

Table 11. Mapping of elements of data model pattern and ontology design pattern [19, 22]

Data model pattern	Ontology design pattern
Entity	Concept
Attribute	Relation to “attribute”
Subtype/supertype	Subsumption hierarchy
Relationships	Relations
Mutually exclusive sets	Disjoint concepts

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.

In an experiment, the knowledge from data model patterns was translated into ontology design patterns by mapping the parts using the KAON tool [2]. Table 11 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.

5 Conclusions

The purpose of this study was to improve the presentation of content ontology design patterns. 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.

Suggestions are based on the comparison of the different patterns with the ontology design patterns. Based on 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. Software patterns include the implementation section with the solution of complex problem. Implementation help user to understand the pattern.

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.

6 Future Works

This study will be improved through experiment. Determine how well the current ODP template supports the understanding and usage of Content ODPs. To get experts opinions on the current structure of the Ontology design pattern template. There may be certain information that an ontology user need to understand a pattern but it is not available in the description. This research will improve by examine the missing information in the current ontology design pattern templates.

References

1. Uschold, M., Gruninger, M.: Ontologies and semantics for seamless connectivity. ACM SIGMOD Record **33**(4), 58–64 (2004)
2. Blomqvist, E.: Fully automatic construction of enterprise ontologies using design patterns: initial method and first experiences. In: OTM Confederated International Conferences On the Move to Meaningful Internet Systems. Springer (2005)
3. Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. **5**(2), 199–220 (1993)

4. Presutti, V., Gangemi, A.: Content ontology design patterns as practical building blocks for web ontologies. In: International Conference on Conceptual Modeling. Springer (2008)
5. Blomqvist, E.: State of the Art: Patterns in Ontology Engineering (2004)
6. Gangemi, A., Presutti, V.: Ontology design patterns. In: Handbook on Ontologies, pp. 221–243. Springer (2009)
7. Pree, W.: Meta patterns—a means for capturing the essentials of reusable object-oriented design. In: European Conference on Object-Oriented Programming. Springer (1994)
8. Doran, P.: Ontology reuse via ontology modularisation. In: KnowledgeWeb Ph.D. Symposium. Citeseer (2006)
9. Prabhu, V., Kumara, S., Kamath, M.: Scalable Enterprise Systems: An Introduction to Recent Advances, vol. 3. Springer Science & Business Media, New York (2012)
10. Rebstock, M., Janina, F., Paulheim, H.: Ontologies-Based Business Integration. Springer Science & Business Media, Heidelberg (2008)
11. Guarino, N.: Semantic matching: formal ontological distinctions for information organization, extraction, and integration. In: Information Extraction a Multidisciplinary Approach to an Emerging Information Technology, pp. 139–170. Springer (1997)
12. Fernandez, E.B., Yuan, X.: An analysis pattern for reservation and use of reusable entities. In: Proceedings of PLoP (1999)
13. Buschmann, F., et al.: Pattern-oriented software architecture: a system of patterns (1996). Part II, 2001
14. Kampffmeyer, H., Zschaler, S.: Finding the pattern you need: the design pattern intent ontology. In: International Conference on Model Driven Engineering Languages and Systems. Springer (2007)
15. Cooper, J.W.: The Design Patterns Java Companion, vol. 218. Addison-Wesley, Upper Saddle River (1998)
16. Devedzic, V.: Ontologies: borrowing from software patterns. Intelligence **10**(3), 14–24 (1999)
17. Gamma, E.: Design Patterns: Elements of Reusable Object-Oriented Software. Pearson Education India, Delhi (1995)
18. Fernandez, E.B.: Building systems using analysis patterns. In: Proceedings of the Third International Workshop on Software Architecture. ACM (1998)
19. Lodhi, S., Ahmed, Z.: Content Ontology Design Pattern Presentation (2011)
20. Fernandez, E.B., Liu, Y.: The account analysis pattern. In: EuroPLoP (2002)
21. Buschmann, F., et al.: Pattern-Oriented System Architecture: A System of Patterns, pp. 99–122. Wiley, Chichester (1996)
22. Oluyomi, A.O.: Patterns and protocols for agent-oriented software development (2006)
23. Silverston, L., Inmon, W.H., Graziano, K.: The Data Model Resource Book: A Library of Logical Data Models and Data Warehouse Designs. Wiley, Hoboken (1997)
24. Hay, D.: Data Model Patterns Conventions of Thought. Dorset House, New York (1996)
25. Hay, D.C.: A comparison of data modeling techniques (1999)
26. Gangemi, A.: Ontology design patterns for semantic web content. In: International Semantic Web Conference. Springer (2005)