ONTOLOGY DESIGN PATTERNS WITH APPLICATIONS TO SOFTWARE MEASUREMENT

A dissertation submitted
to Kent State University in partial
fulfillment of the requirements for the
The degree of Doctor of Philosophy

by
Mazen S. Al-Zyoud

December 2015
© Copyright
All right reserved
Except previously published material
Dissertation wrote by
Mazen S. Al-Zyoud
B.S., Al al-Bayt University, Jordan, 2003
M.S., Al al-Bayt University, Jordan, 2009
Ph.D., Kent State University, USA, 2015

Approved by

Professor Austin Melton, Chair, Doctoral Dissertation Committee
Professor Johnnie Baker, Members, Doctoral Dissertation Committee
Professor Angela Guercio.
Professor Alan Brandyberry.
Professor Jeffrey A. Ciesla.

Accepted by

Professor Javed I. Khan, Chair, Department of Computer Science
Professor James L. Blank, Dean, College of Arts and Sciences
Table of Contents

TABLE OF CONTENTS ........................................................................................................III
LIST OF FIGURES ............................................................................................................... VII
LIST OF TABLES ................................................................................................................ IX
DEDICATION ....................................................................................................................... X
ACKNOWLEDGEMENTS ...................................................................................................... XI
ABSTRACT .......................................................................................................................... XII

CHAPTER 1 INTRODUCTION ............................................................................................... 1
  1.1 Goals of the Research ................................................................................................. 1
  1.2 Contributions .............................................................................................................. 2
  1.3 Publication Notes ....................................................................................................... 3
  1.4 Organization of the Dissertation .............................................................................. 3

CHAPTER 2 BACKGROUND MATERIAL ON SEMANTIC WEB AND ONTOLOGY

  2.1 Introduction about Semantic Web ............................................................................. 5
  2.2 Software Design Patterns .......................................................................................... 7
    2.2.1 Structure of a design pattern in software engineering ......................................... 9
    2.2.2 What are the relationships among these patterns? ............................................. 11
  2.3 Introduction about Ontology .................................................................................... 11
    2.3.1 Importance of ontology ..................................................................................... 12
    2.3.2 Ontology languages .......................................................................................... 13
    2.3.3 Classifying ontology .......................................................................................... 14
    2.3.4 Methodology for development ontology .......................................................... 15
    2.3.5 Competency questions ..................................................................................... 16
    2.3.6 The web ontology language ............................................................................. 17
    2.3.7 Protégé ............................................................................................................. 17
2.4 ONTOLOGY ENGINEERING ........................................................................................................... 18
  2.4.1 Ontology development process ............................................................................................... 18
2.5 ONTOLOGY DESIGN PATTERNS ODPs ....................................................................................... 20
  2.5.1 Ontology design pattern benefits ........................................................................................... 21
  2.5.2 Classification of ontology design patterns ............................................................................... 21
  2.5.3 Types of ontology design patterns for semantic web .............................................................. 22
  2.5.4 Content ontology design patterns (CPs) ................................................................................. 24
  2.5.5 Architectural ODPs ..................................................................................................................... 27

CHAPTER 3 A CONTENT ONTOLOGY DESIGN PATTERNS FOR ONTOLOGY IN
ACADEMY/UNIVERSITY DOMAIN ................................................................................................... 28
  3.1 INTRODUCTION AND MOTIVATION ....................................................................................... 28
    3.1.1 Patterns design ....................................................................................................................... 29
    3.1.2 Conceptual setting ............................................................................................................... 30
    3.1.3 Domain space ....................................................................................................................... 30
    3.1.4 Summary of terms ............................................................................................................... 31
  3.2 STEPS TO SUBMIT A CPs IN ODPs PORTAL ........................................................................... 32
  3.3 COURSE CONTENT ONTOLOGY DESIGN PATTERN (COURSE CPs) ........................................ 32
    3.3.1 Description of the course CPs ............................................................................................. 32
  3.4 PROFESSOR CONTENT ONTOLOGY DESIGN PATTERN (PROFESSOR CPs) ....................... 41
    3.4.1 Description of the professor CPs ......................................................................................... 41
  3.5 A RECOMMENDED PROCEDURE FOR CREATING A CPs ...................................................... 50
  3.6 A RECOMMENDED GUIDELINE FOR SELECTING AN ONTOLOGY DESIGN PATTERN ....... 54

CHAPTER 4 BACKGROUND MATERIAL ON THE SOFTWARE DEVELOPMENT
LIFE CYCLE AND SOFTWARE MEASUREMENT ................................................................................. 58
  4.1 INTRODUCTION TO SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC) ............................... 58
  4.2 PHASES OF THE SOFTWARE DEVELOPMENT LIFE CYCLE ......................................................... 59
  4.3 BENEFITS OF THE SOFTWARE DEVELOPMENT LIFECYCLE PROCESS .............................. 60
  4.4 SOFTWARE DEVELOPMENT LIFE CYCLE MODELS ................................................................. 60
    4.4.1 Agile SDLC model ................................................................................................................. 61
List of Figures

Figure 2-1 Ontology classification based on domain scope or the domain ................................................. 15
Figure 2-2 Ontology development process steps .......................................................................................... 20
Figure 3-1 Course content ontology design pattern graphical diagram ....................................................... 35
Figure 3-2 Class hierarchies for course content ontology design pattern .................................................. 37
Figure 3-3 Course CPs axioms metrics ........................................................................................................ 40
Figure 3-4 Professor content ontology design pattern graphical diagram ................................................ 44
Figure 3-5 Class hierarchies for professor content ontology design pattern .............................................. 46
Figure 3-6 Professor CPs axioms metric ....................................................................................................... 49
Figure 3-7 Graphical diagram for a suggested procedure for creating a CPs ............................................. 50
Figure 3-8 Matches an LUC to a GUC ......................................................................................................... 56
Figure 3-9 Graphical diagram for how to select ontology design patterns ............................................... 57
Figure 4-1 Why software metrics ............................................................................................................... 64
Figure 4-2 Software measurement process architecture .............................................................................. 67
Figure 5-1 Architectural ontology design pattern diagram ......................................................................... 76
Figure 5-2 Class hierarchies ....................................................................................................................... 79
Figure 5-3 Case study how to use object property applied in ..................................................................... 82
Figure 5-4 Case study how to use object property cascade impact with object property ongoing improve .................................................................................................................. 83
Figure 6-1 Ontology construction steps as iterative approach ................................................................. 91
Figure 6-2 Steps involved in creating the of software measurement ontology ........................................... 97
Figure 6-3 The class and subclass (classes hierarchy) of software measurement ontology . 98
Figure 6-4 Object property of software measurement ontology SMO .................................................. 99
Figure 6-5 Individuals by class type of software measurement ontology SMO ................................. 100
Figure 6-6 A test case for software measurement ontology SMO ........................................ 101
Figure 6-7 A test case to retrieve information from the SMO .......................................... 102
Figure 6-8 A rules for qualified specification document in SMO .................................... 102
Figure 6-9 Asserted model for the software measurement ontology ............................. 103
Figure 6-10 Entire ontology diagram ............................................................................ 104
Figure 6-11 Ontology axioms metrics .......................................................................... 105
Figure 7-1 Domain space diagram ................................................................................ 107
Figure 7-2 Semantic Module for classes and properties ............................................... 108
List of Tables

Table 2-1 Pattern template ........................................................................................................... 10
Table 3-1 Summary terms of academic/university domain ......................................................... 31
Table 3-2 Class for course content ontology design pattern ..................................................... 36
Table 3-3 Object properties for course content ontology design pattern .................................. 38
Table 3-4 Data properties for course content ontology design pattern ...................................... 39
Table 3-5 Class dictionary for professor content ontology design pattern ............................... 45
Table 3-6 Object properties for professor content ontology design pattern ............................. 47
Table 3-7 Data properties for professor content ontology design pattern ............................... 47
Table 5-1 Summary of terms / classes ........................................................................................... 71
Table 5-2 Summary of terms / Object property ............................................................................ 72
Table 5-3 Class for the pattern / 1 .............................................................................................. 77
Table 5-4 Class for the pattern / 2 .............................................................................................. 78
Table 5-5 Object properties for course pattern .......................................................................... 80
Table 6-1 Summary of terms for product quality metrics ontology / data type property .......... 92
Table 6-2 Summary of terms for product quality metrics ontology / classes ............................. 93
Table 6-3 Summary of terms for product quality metrics ontology / object property ............... 94
Dedication

I dedicate my work especially to my wonderful parents, Salem and Fawziah. My wife, Thumaila Masadah. My brothers, and sisters. I have special feelings of gratitude toward my loving parents.
Acknowledgements

It has been a privilege for me to study at the Department of Computer Science at Kent State University where students and professors are always eager to learn new things and to make constant improvements.

I would never have been able to finish my dissertation without the guidance of my advisor, help from friends, and support from my family and wife.

I would like to express my deepest gratitude to my advisor, Professor Austin Melton, for his excellent guidance, caring, patience, and providing me with an excellent atmosphere for doing research.

Thank you, Dr. Johnnie Baker, Dr. Angela Guercio, Dr. Alan Brandyberry, and Dr. Jeffrey A. Ciesla for agreeing to serve on my committee.

I would like to express my deepest gratitude to my wonderful parents, Salem, and Fawziah, for their enormous support, infinite patience, and unwavering belief towards me, as always. My loving brothers and sisters have always been on my side, supporting me and encouraging me with their best wishes. There is a litany of family members and friends who are not individually mentioned here, but they certainly made a difference.

Finally, I would like to thank my wife Thumaila Masadah; she was always there cheering me up and stood by me through the good times and bad.
Abstract

MAZEN SALEM ALZYOUD, Ph.D., December 2015  COMPUTER SCIENCE

ONTOLOGY DESIGN PATTERNS WITH APPLICATIONS TO SOFTWARE MEASUREMENT (140 PP.)

Ontologies are a cornerstone of the semantic web. Ontologies are helpful in sharing and understanding domain concepts, and they also enable reusability of domain knowledge. Furthermore, ontologies enable logical inferencing by using existing logic reasoning mechanisms. This inferencing may be used to discover new knowledge and to check and remove inconsistent content.

Ontology design is a crucial research area for semantic technologies. Ontology Design Patterns (ODPs) aim to support the process of ontology engineering, and may also be used to improve the existing ontologies.

We design a procedure to help in creating content ontology design patterns, and we use this procedure to create new content ontology design patterns for Course and for Professor in the University/Academy domain, which work as a generic cases for developing ontology design patterns. Furthermore, we promote the process of reusability, by developing recommended guidelines for how to select ontology design patterns.
Ontologies are usually used to organize information about a domain. We want to extend the use of ontologies in the area of ontology reasoning, so that ontologies can be effectively used in organizing and directing processes. By using axioms, ontology reasoning can play a key role in answering queries. Also, the reasoning checks the consistency and finds the logical contradictions.

We take product quality metrics in the software measurement domain as an example to create a correct and consistent ontology, which may be used to improve the interaction between metrics in the software development process.

Architectural patterns describe how to organize an ontology. These types of patterns can be used in the conceptual modeling phase and at the beginning of the ontology design phase. We introduce a new architectural ontology design pattern to characterize the overall structure of a measurement program ontology in the software measurement domain, which may be used to implement an effective software measurement program for use during software development.
CHAPTER 1

Introduction

The semantic web has recently become a popular and interesting field of research with numerous scientific research papers published in the topic so far. Ontology is an important component of the semantic web, and many papers about applying ontology in specific domains have been published. Building ontology in a certain domain is an important issue. Any improvement on ontology engineering process must be documented.

Software metrics is an essential part of any software development process. Metrics are important for monitoring the software engineering process, and for understanding, controlling and improving the software process.

We use ontology design patterns to effectively and efficiently generate ontologies in two domains. We have the main goal to create ontologies easily in two selected domains; the two selected domains are Academy/ University domain and Software measurement domain.

1.1 Goals of the research

The dissertation focuses on pushing boundaries of creating a fundamental ontology (core, reference) to reduce the cost of the time-consuming process. Hence, a good generic case is introduced and stored in a warehouse for reusability.
The dissertation addresses the following main research questions:

- Our main goal to easily create ontologies in two selected domains; the two selected domains are Academy/ University domain and Software (measurement activity) domain. We create different types of ontology design patterns, and we help with ontology mappings in our domains.
- We introduce a general procedure to create content ontology design pattern (CPs) for different domains. We address the question of how to select an ontology design patterns (ODPs) for reuse.
- We create a arithuctural ODPs that may be used by software developers to create a robust specialized software measurement plan that will encompass the entire lifecycle of a software system.
- We create a semi-formal ontology for software measurement domain to help in get a common agreement among software measuerments researcher.

1.2 Contributions

We improve the process of building reference ontologies in selected domains by creating a suite of ontology design patterns. The contributions of this dissertation are outlined below.

- Find a generic use case to help in the mapping and composition of patterns. This generic case work as reference ontology, to check the
consistency of the composition, or to compare the sets of axioms that are to be mapped.

- Our ontologies are application ontologies. It is difficult to linking, mapping, and integrating. Thus, we create a core ontology (reference ontology) in the software measurement domain to help in reducing these issue. This is a valuable step for several stakeholders interested in this domain.

### 1.3 Publication Notes

The results in CHAPTER 3 have been submitted to an ontology design patterns portal [3] [4]. The guideline results are written up and will be submitted to a conference. CHAPTER 5’s results in architectural ontology design patterns has also submitted to an ontology design patterns portal. CHAPTER 6’s results for core (reference) ontology will be presented to a conference.

### 1.4 Organization of the Dissertation

The remainder of the dissertation is organized as follows: a review of the semantic web for general background, software design patterns, web ontology languages, ontology design patterns, and content ontology design patterns in CHAPTER 2. CHAPTER 3 presents a suite of content ontology design patterns for ontologies in academy/ university domain, provides guidelines to create content ontology designs pattern and addresses how
to select content ontology design patterns for reuse. CHAPTER 4 introduces the second part of this dissertation about ontology design patterns for software measurement plans. CHAPTER 5 present ontology design patterns for software measurement plans. CHAPTER 6 builds a core ontology for software product quality metrics. Finally, we conclude in CHAPTER 7 with a discussion on open issues and future directions.
CHAPTER 2

Background material on semantic web and ontology design patterns

In this chapter, we introduce the basic concepts associated to our work and that are used in later chapters. In section 2.1, we introduce concepts related to the semantic web. In the next section, we discuss design patterns. In the section 2.3, we say why we need an ontology. Section 2.4 is a brief introduction about ontology engineering. The last section is about ontology design patterns.

2.1 Introduction about semantic web

The web continues to grow, and above all, it continues to change in structural complexity and underlying architecture. Thus, the need for new technologies becomes urgent. [61]

The semantic web is the next generation web, which concerns the meaning of web documents. The semantic web has lately been a popular and prolific field of research with numerous research papers published on the topic. [6]

The word semantic itself denotes meaning or understanding. There are fundamental differences between semantic web technologies and other technologies related to data, such as relational databases or the world wide web itself. The semantic web is concerned with the meaning and not the structure of data. [61]
The semantic web initiative tries to establish better semantic connections between different resources on the web, and enable users to find, share and combine information more easily than they can now on the current web. [60]

The semantic web is a vision of information that can be readily interpreted by machines, so machines can do more of the tedious work involved in finding, combining, and acting upon information on the web. [42] [60]

The semantic web, as originally visualized, is a system that enables machines to "understand" and respond to complex human requests based on their meaning. Such an "understanding" requires that the relevant information sources be semantically structured. The aim of the semantic web can be explained in one sentence: to make the web more understandable to a computer. [55]

The semantic web has seen a growing popularity with standards institutions like W3C, academic communities like Stanford University, and individual websites, like Swoogle1. The semantic web envisions a world wide web in which data is described with rich semantics and applications can pose, and answer complex queries. [35]

The semantic web does use some ideas from artificial intelligence. Artificial intelligence project should use semantic web to be collectively more powerful. The combining of artificial neural networks with semantic web technologies adds more benefit to this field. [22] [55]

1 http://swoogle.umbc.edu/
2.2 Software design patterns

In software engineering, a design pattern is a general reusable solution to a commonly occurring problem within a given context in software design. A design pattern is not a finished design that can be transformed directly into source or machine code. Design patterns are a description or template for how to solve a problem that can be used in many different situations. [107]

Design patterns reside in the domain of modules and interconnections. At a higher level, there are architectural patterns that are larger in scope, usually describing an overall pattern for an entire system. [20]

Design patterns were first introduced in a book by Christopher Alexander et al., “A Pattern Language” which describes architectural solutions at municipal, building and construction levels. This book uses words to describe patterns, supported by drawings, photographs, and charts. It describes exact methods for constructing practical, safe, and attractive designs at every scale, from entire regions, through cities, neighborhoods, gardens, buildings, rooms, built-in furniture, down to the level of doorknobs. Alexander argued that a good (architectural) design can be achieved using a set of rules that are “packaged” in the form of patterns. Such as “courtyards that live”, “windows place”, or “entrance room”. Design patterns are then assumed as archetypal solutions to design problems in a certain context. [2]
Christopher Alexander said,

"Each pattern describes a problem that occurs over and over again in our environment. Then describes the core of the solution to that problem, in such a way that you can use this solution a million times over. Without ever doing it the same way twice" [2]

By borrowing the idea of architectural patterns, the notion has been endorsed by software engineering and database management system (DBMS) applications with so-called data model patterns. In these areas, the pattern is used as a general term for formatted guidelines in software reuse, and more recently has also appeared in requirement analysis and conceptual modeling.

Design patterns in architecture give us simple and elegant solutions to specific problems. They reflect the experience, knowledge and insight of architects who have successfully used these patterns, and they provide ready-made solutions.

Software engineering began to incorporate Alexander's principles into the creation of early design patterns documentation as a guide to novice developers. Then software engineers used design patterns to define simple solutions to specific problems in object-oriented programming, by providing common description templates for how to solve a problem that can be used in many different situations.

Design patterns seek to communicate these classic solutions in an easy to understand manner, and use shared guidelines that help solve design problems. Design
patterns provide a fundamental foundation for building maintainable and scalable software. Design patterns reflect the experience and knowledge of developers who have used patterns in their work successfully and give ready solutions that can be used in different problems. [107] [66] [67]

Design patterns have two major benefits. First, they provide a way to solve issues related to software development using a proven solution. The solution facilitates the development of highly cohesive modules with minimal coupling. They isolate the variability that may exist in the system requirements, making the overall system easier to understand and maintain. [82]

Second, design patterns make communication between designers more efficient. Software professionals can immediately picture the high-level design in their heads when they refer the name of the pattern used to solve a particular issue when discussing system design, so it can speed up the development process. [82] [81]

2.2.1 Structure of a design pattern in software engineering

Design patterns in software engineering are described using a common format that includes four essential elements: pattern name, problem, solution, and consequences. [39]

The patterns are documented from a template that identifies the information needed to understand the software problem and the solution in terms of the relationships between the classes and objects necessary to implement the solution. The template lends
a uniform structure to the information, making design patterns easier to learn, compare, and use. Software engineering describes design patterns using a consistent format. Each pattern is divided into sections according to the following template, table 2.1. This template captures the essential information required to understand the essence of the problem and the structure of the solution. [39] [82]

**Table 2-1 Pattern template**

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern Name</td>
<td>Describes the essence of the pattern in meaningful name</td>
</tr>
<tr>
<td>Intent</td>
<td>Describes what the pattern does</td>
</tr>
<tr>
<td>Also, Known As</td>
<td>List any synonyms for the pattern</td>
</tr>
<tr>
<td>Motivation</td>
<td>Give an example of a problem and how the pattern solves that problem</td>
</tr>
<tr>
<td>Applicability</td>
<td>Lists the situations where the pattern is applicable</td>
</tr>
<tr>
<td>Structure</td>
<td>Set of diagrams of the classes and objects that depict the pattern</td>
</tr>
<tr>
<td>Participants</td>
<td>Describes the classes and objects that participate in the design pattern and their responsibilities</td>
</tr>
<tr>
<td>Collaborations</td>
<td>Describes how the participants collaborate to carry out their responsibilities</td>
</tr>
<tr>
<td>Consequences</td>
<td>Describes the forces that exist with the pattern and the benefits, trade-offs, and the variable that is isolated by the pattern</td>
</tr>
<tr>
<td>Sample Code</td>
<td>Code fragments that illustrate how might implement the pattern</td>
</tr>
<tr>
<td>Known Uses</td>
<td>Examples of the pattern found in real systems. We include at least two examples from different domains.</td>
</tr>
<tr>
<td>Related Patterns</td>
<td>What design patterns are closely related, what are the important differences?</td>
</tr>
</tbody>
</table>
2.2.2 What are the relationships among these patterns?

Many patterns may need to work together when building a system. Different designers may use different patterns to solve the same problem. Usually, some patterns naturally fit together, or one pattern may lead to another, and patterns give hints to solve a problem effectively. [49] [82]

2.3 Introduction about ontology

The Merriam-Webster online dictionary defines the term ontology as:

1) A branch of metaphysics concerned with the nature and relations of being
2) A particular theory about the nature of being or the kinds of things that have existence.

Ontology in the original sense, is a branch of philosophy, and mean the study of what things exist and the relation between those things. Ontology has been used as a mechanism, to share common knowledge and understanding. Ontology is a popular term today, used in many areas and defined in many different ways. [27]

Ontologies are used in various fields of computer science, such as artificial intelligence, the semantic web, and system engineering, also software engineering, biomedical informatics, and others fields. [25] [17]

---

2 http://www.merriam-webster.com/dictionary/ontology
3 http://www.ontology.co/idx00.htm
Ontology in computer science and information science is a hierarchically structured set of concepts describing a specific domain of knowledge that can be used to create a knowledge base. [121]

Ontology is an important component of the semantic web, and many research papers about applying ontology in particular fields have been published. Ontology construction of a given domain is a significant step in the semantic web. An ontology contains concepts, a sub-assumption hierarchy, relations between concepts, and axioms. It may also contain other constraints and functions. [1] [30]

The more specific definition for ontology in the semantic web. Come from the W3C consortium:

”Ontology is a term borrowed from philosophy that refers to the science describing the kinds of entities in the world and how they are related”. [103]

2.3.1 Importance of ontology

Nowadays, the amount of knowledge is increasing at an accelerated pace. Moreover, become more complex, when we look to ontologies as a vital part of the solution to the problems in many fields like big systems and big data. Ontology, is formal representations of a set of concepts within a domain and the relationships between those concepts. There are some benefits of using the onology. [50] [72]
The general importance of ontology is shown by being helpful in sharing the knowledge over many users who have similar needs for knowledge representation in the same domain. Indeed, by using ontological analysis, we get clarify structured of knowledge in a given domain. [51]

Also, ontologies can make explicit assumptions about our systems, enable integration of systems and data through semantic interoperability. They also enable adaptability and reuse, reduce development and operational costs, enhance decision support systems and support knowledge management and discovery, and provide a basis for more adaptable systems. [50][49]

Also, to enable reuse of domain knowledge (to avoid “re-inventing the wheel.”). Also, to make domain assumptions explicit easier to change domain assumptions, easier to understand and update legacy data. [37] [53]

2.3.2 Ontology languages

In computer science, ontology languages are usually declarative formal languages used to construct, and encode ontologies. The numerous ontology languages are often classified by structure (logic type) or syntax (Traditional syntax, Markup). [24] [121]

Ontology description languages are specifically designed to define ontologies. These language are sometimes called web ontology languages, lightweight ontology languages or markup ontology languages. [44]
An ontology language allows users to write explicit, formal conceptualizations of domains models. The main requirements are a well-defined syntax, a well-defined semantics and efficient reasoning support, sufficient expressive power, and convenience of expression. [58] [90]

2.3.3 Classifying ontology

During the development of an ontology, it may be important to know which is the logic underlying that particular ontology. There is a different category for ontologies have been suggested. Each of them focused on different dimensions in which ontologies can be classified. Philosophers can classify ontologies in various ways using criteria such as the degree of abstraction and field of application. Introduce some of these kinds. [128] [73]

Guarino classification [54], Guarino proposed a classification based on the generality of the ontology. McGuinness classification [89], McGuinness proposed the classification based on the internal structure and the content of the ontologies (from the lightweight ontology to heavyweight ontology). [90]

Classification based on the scope of the ontology, or on the domain. This classification based on the context of the scope described by the ontology. [52] [106]

For instance, the scope of a local ontology is narrower than the scope of a domain ontology; domain ontologies have more specific concepts than core reference ontologies, which contains the fundamental concept of a domain. Foundational ontologies can be
viewed as meta ontologies that describe the top level concepts or primitives used to define others ontologies. [54]

Finally, general ontologies are not dedicated to a specific domain thus its concepts can be as general as those of core reference ontologies. The next figure 2-1 shows this category. [106]

![Ontology classification based on domain scope or the domain](image)

**Figure 2-1** Ontology classification based on domain scope or the domain

### 2.3.4 Methodology for development ontology

There are research groups are now proposing a series of steps and methods for developing ontologies. In this section, we state a short presenting of ontology development methodologies. [37]
Grüninger [52] [51] suggest a method to build the ontology using the competency question (question used to address the characteristics of ontology) to extract the concepts, axioms to define the terms in the ontology.

Fernández [36] introduce the Methodology to build the ontology step by step guidelines which based on the idea of software engineering. Moreover, it’s help in building ontology from scratch and can be used for reuse an existing ontology.

Uschold [126] this methodology is based on the experience of developing the Enterprise Ontology, an ontology for enterprise modeling processes. This methodology provides guidelines for developing ontologies. [127]

Nanda [93] used formal concept analysis to develop a methodology for domain ontology, identify similarities among design artifacts with also using web ontology language (OWL). Natalya [95] suggest a life cycle for creating an ontology, in particular for the semantic web. These steps called ontology engineering.

2.3.5 Competency questions

Competency questions are the basis for a rigorous characterization of the problems that the pattern can solve. They are also a requirement for create ontology, that are used to characterize the ontology in a domain. Moreover, by solving these questions, we determine the necessary and sufficient set of axioms to represent the ontology. [52]
2.3.6 The web ontology language

The web ontology language (OWL) is one of the most significant ontology languages. Several extensions to semantic web ontology languages have been proposed during the last decade, and OWL it is more expressive than XML, RDF or RDFS by providing additional vocabulary along with formal semantics. [7]

A web ontology language is a markup language that designed for use by applications that need to process the content of information instead of just presenting information to humans. [5]

There are many benefits of using OWL language, describes the hierarchical organization of ideas in a domain, in a way that can be parsed and understood by software. Here is some of characteristic of OWL, has a well-defined syntax, efficient reasoning support, and formal semantics, and has efficient, expressive power of the expression. [68] [62]

There are two generation of the web ontology language OWL, OWL 1 started in 2002, and the final W3C recommendation was released in 2004. OWL 2, the new version, was released in 2012. [75] [24]

2.3.7 Protégé

A free, open source ontology editor and framework for building ontology. The Protégé platform supports modeling ontologies via a web client or a desktop client.
Protégé ontologies can be developed in a variety of formats including OWL, RDF(S), and XML Schema. [96]

Protégé provides graphical editors for class expressions, access to description logical reasoning, and a powerful platform for hooking in custom-tailored components. Protégé is based on Java technology, is extensible, and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development. The Protégé Programming Development Kit (PDK) contains a set of documents designed to describe and illustrate the manner in which a plug-in extension for Protégé should be developed and installed. [119]

2.4 Ontology engineering

Ontology engineering in computer science and information science is a new field. That is studying the methods and methodologies of building ontologies. [37][126]

2.4.1 Ontology development process

There are several steps that have been proposed to design the ontology and thus to facilitate it. [126]

1. Determine Domain and Scope

In this step, we need to define the domain (scope) the ontology will cover, mention the aim or where we use the ontology. The next important point in this step to write down the competency question. [106]
2. Consider Reuse

Since the reusability provides a bunch of benefits. If we can reuse an ontology validated in the application through using it. To save effort and interact with some tools that use other ontologies. To help at this point there are some repository portals provide a list of ontology for example Protégé ontology library\(^4\). For more repository see appendix A-1. [106]

3. Enumerate Important Terms

The main step for builds ontology. Now mentions the core concepts -terms- after that determine the properties of each term.

4. Define Classes and the Class Hierarchy

We represent each term as a class. Then define the relation between concepts -classes- by determining the hierarchy of classes. Here we need to document the classes by describing the classes in natural language, listing the synonyms.

5. Define Properties of Classes – Slots

In this step; definition –describe- the attributes of instances of a class and relations to other instances for other classes; by using the different types of property

6. Property Constraints

\(^4\) http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library
Declare the property constraints (facets) which mean to describe or limit the set of possible values for a properties (slot).

7. Create Instances

The last step is to create an instance for the classes.

The next figure 2-2, show the sequence of the steps.

![Ontology development process steps](image)

**Figure 2-2 Ontology development process steps**

2.5 Ontology design patterns ODPs

One of the most challenging and attentional areas of ontology design is reusability, which getting more and more important. To support and facilitate reuse, Gangemi introduced ontology Design Patterns (ODPs) [45] and Blomqvist& Sandkuhl [14] in 2005 (extending ideas by the W3C Semantic Web Best Practices and Deployment Working Group[^5]). Some ODPs can be found on the W3C website. [44]

An ontology design patterns (ODPs) is a modeling abstract solution to solve a recurrent ontology design problem in ontology engineering. ODPs are ready made


20
modeling solutions for creating and maintaining ontologies. ODPs are intended to guide non-experts in performing ontology engineering lifecycle successfully. [56] [122]

2.5.1 Ontology design pattern benefits

There are experiments prove that ODPs improve the ontology engineering lifecycle in different ways as, the reuse of ODPs makes the ontology development process easier, reduce mistakes in ontologies, provide a neat way of producing more modular and robust ontologies, and improves communication between ontology developers. [48] [59]

2.5.2 Classification of ontology design patterns

The semantic web contained many ontologies and expected to contain more of them in the future. Classification of ontology patterns is the first step toward a structured use of pattern in building ontology as accepted way to facilitate and support reuse to reduce time and effort. There is a different classification of ontology design patterns; we will introduce the most popular classification. [118]

2.5.2.1 Manchester’s ontology design patterns classification

This classification is focused on the biological knowledge domain. For bio-oncologists to be able to explore efficiently and retrieve ODPs, an online public classification of ODPs has been created. The classification based on the functionality of ODPs which help the bio-otologist in choosing the appropriate ODP. [8] [19]
2.5.2.2 Blomqvist et al. classification.

Blomqvist presents a different topology based on the level of abstraction and granularity of the reusable solution. According to this classification structure, there are five levels of ontology pattern abstraction, application patterns, architecture patterns, design patterns, semantic patterns, and syntactic patterns. [15] [14]

2.5.2.3 Ontology Design Patterns for semantic web content

Ontology design patterns are an emerging technology that favors the reuse of encoded experiences and good practices. Also are modeling solutions to solve recurrent ontology design problems. ODPs introduce by Gangemi [45] [42]. They familiarized in six different types. More in the next section.

2.5.3 Types of ontology design patterns for semantic web

There are mainly six types of ontology design patterns, and some of them have subtypes, structural ODPs, correspondence ODPs, content ODPs (CPs), reasoning ODPs, presentation ODPs, and lexico-syntactic ODPs. Our research focused on two types, CPs, Architectural. [42] [63]

1. Structural ODPs

Structural ODPs include two subtypes, Logical ODPs and Architectural ODPs. Logical ODPs are compositions of logical constructs that help to solve design problem,
where the ontology language do not support them. While Architectural ODPs affects the overall shape of the ontology either internally or externally. [122]

2. Reasoning ODPs

Reasoning ODPs are applications of logical ODPs oriented to obtain certain reasoning results, based on the behavior implemented in a reasoning engine. [104]

3. Correspondence ODPs

Correspondence ODPs includes two sub-types: Reengineering ODPs and Mapping ODPs. Reengineering ODPs provide designers with solutions to the problem of transforming a conceptual model, which can even be a non-ontological resource such as UML diagram, into a new ontology. Mapping ODPs are patterns for creating semantic associations between two existing ontologies. [17] [12]

4. Presentation ODPs

Presentation ODPs deal with usability and readability of ontologies from a user perspective. Presentation ODPs include two types: Naming ODPs and Annotation ODPs. Annotation ODPs provide annotation property or property schema to improve the understandability of ontology and his element. Naming ODPs related to namespace declared for an ontology. [29]

5. Lexico-Syntactic ODPs
Lexico-Syntactic ODPs are linguistic structures or schemas that consist of certain types of words following a specific order, and that permit the user to generalize and extract some conclusions about the meanings they express. They are useful for associating simple logical and Content ODPs with natural language sentences. [87]

6. Content ontology design patterns (CPs)

CPs provide solution to oriented-domain. CPs propose patterns for solving design problems for the domain classes and properties that populate an ontology. [45][76]

2.5.4 Content ontology design patterns (CPs)

In software design patterns, there are classes of problems that can be solved by common solutions (patterns). Under a similar assumption if we can describe small, motivated ontologies that can be used as building blocks in ontology design, we call this building blocks Content Ontology Design patterns (CPs). They provide a solution to the domain-oriented problem can be directly reusable. [101]

CPs encodes conceptual rather than logical design patterns. In other words, while logical ODPs solve design problems independently of a particular domain, CPs addressing content problems. [45][42]

Aldo and Valentina define Content Ontology Design Pattern (CP):

“CPs is distinguished ontologies. They address a specific set of competency questions, which represent the problem they provide a solution for it. Furthermore, CPs
shows certain characteristics, i.e., they are: computational, small and autonomous, hierarchical, cognitively relevant, linguistically relevant, and best practices”. [7]

2.5.4.1 Features of content ontology design patterns

Here are some characteristics of CPs that make them helpful to create. [45]

1. **Language expressive**

The CPs use computer language, language-independent. Moreover, should be encoded in a higher-order representation language. Be able to be represented in any ontology language. However, their (sample) representation in OWL is needed to (re)use them as building blocks over the Semantic Web.

2. **Small, autonomous components**

They are small, independent ontologies. The smallness mean from two to ten classes with relations between them. Independent of CPs means template to represent and solve modeling problems. Smallness also allows diagrammatical visualizations that are aesthetically acceptable and easily notable.

3. **Hierarchical components**

A hierarchy of CPs can be built by specializing or generalizing some of the elements (either classes or relations). A CPs can be taxonomy form. Where that at least one of the classes or properties in the pattern be specialized

4. **Inference-enabling components**
A CPs allows some form of inference e.g. taxonomy with two sibling disjoint classes, a property with explicit domain and range set, a property and class with a universal restriction on that property.

5. **Best practice components**

A CPs describe as “best practice” of modeling. Best practices come from the experts, emerging from real experience.

2.5.4.2 **Benefits of content ontology design patterns**

Content ontology patterns offer an advantage according to their nature and characteristics. It is possible to create rich and rigorous ontologies with less effort. The reuse of CPs makes the ontology development process easier. There are additional benefits, CPs solves design problems for the domain classes and properties that populate an ontology, address content problems, and easily reused because of domain specificity.

2.5.4.3 **Content ontology design pattern creation**

CPs can be the result of a re-engineering from different data model (conceptual model). Data model that can be re-engineered to produce a CPs are database schemas, knowledge organization systems (e.g. thesauri), and UML diagram. [12] [129]

A CPs can be created either by combining other CPs or by specialization of another CPs. Also, can be extracted from an existing ontology, which acts as the “source” ontology [101] [7]
2.5.5 Architectural ODPs

Architectural ODPs affect the overall shape of the ontology. Aim to constrain ‘how the ontology should look like’. In other words supposed to characterize the overall structure of the ontology. Architectural ODPs emerged as design choices motivated by particular needs, e.g. computational complexity constraints. They are useful as reference documentation for those initially approaching the design of an ontology. [43] [89]

Examples of Architectural ODPs (AODPs)

1) Lightweight ontology and taxonomy

It is an ontology or knowledge organization system in which general associations connect concepts with other features (taxonomy + other features), for example, a class can be related to other classes through the relation disjointWith, object and datatype properties can be defined and used to relate classes. [93] [91]

2) Modular architecture

Structuring an ontology as a configuration of components. Each is having its identity based on some design criteria. When an ontology is committed to a huge domain of knowledge, a good practice is to decompose the domain into smaller subdomains that address simpler tasks. Each subdomain can be then encoded in an ontology module, to provide the whole ontology with a modular architecture. [108]
CHAPTER 3

A Content ontology design patterns for ontology in academy/university domain

The work presented in this chapter introduces our two content ontology design patterns for the academic domain (throughout section 3.3 and section 3.4). Also introduce guideline for how to construct a content ontology design patterns and how to select a content ontology design patterns for reusability (section 3.5 and section 3-6).

In this chapter we presented two content ontology design patterns created for academic/ university domain, also presented recommended procedure for creating a CPs, and recommended guideline for selecting an ontology design pattern.

3.1 Introduction and motivation

The number of ontologies are continuous increase, and the available ontology on the web is also increasing the number, it becomes crucial to establish, as far as possible, a reference ontology for some knowledge domain. [88]

The use of reference ontology instead of application ontology has benefited for reusability. Reference ontology work as core ontology cover the central or core concepts for a particular domain, rich axiomatic. [130] [84]

The main contribute and significantly for build a reference ontology. Is to attenuate in a concrete way the many problems generated by information heterogeneity.
On other hand and to realize information share and reuse. Help in a way in ontology mapping, evaluation, and population. [131]

The idea for the design patterns that will introduce here was first developed as a research paper for a class that offered by computer science department at Kent State University in the fall of 2013. The class was named Semantic Web and was a graduate level course. Its, goal was to explain the fundamentals of Semantic Web knowledge.

The Professor had been explaining the approach to teaching this subject, which essentially followed, in this approach. Topics covered in this class progressed from light-weight to heavy-weight, starting with RDF and RDFS, their semantics and completion algorithms, the OWL syntax and intuitive semantics, followed by the introduction of description logics. The class included a brief introduction to ontology design patterns ODPs with primary examples.

3.1.1 Patterns design

In academic/ university domain, we intend to build a content ontology design patterns representing the skeleton of the conceptual structure of the knowledge domain of educational institute. These design pattern contains major concepts, for example; “Course”, “Instructor”, “Student”, and ”Professor”. It serves to make explicit the relationships between the concepts of the designed patterns.

For the modeling session, our plan to look at some popular educational institute websites to developing our CPs terms list, general properties for each term. Then
formulate in natural language, for instance, what are the terms used to define the course. Then we develop a graph structure as the basis for a pattern and check that their queries work with these. Such as using Unified Modeling Language UML diagram⁶. [23]

Our goal was as follows: “Looking for different educational websites to matching the terms representing the same concepts for ontology mapping. Design an ontology design pattern that can be used as a part of educational institute semantically website. The pattern shall be set up such that content from existing educational websites can "in principle" be mapped to the pattern (i.e., the pattern gets populated with data from the educational websites). Also, these patterns may be used for creating a new ontology or websites”.

3.1.2 Conceptual setting

The main concepts -terms- derived from the competence questions are the course, professor. These concepts aim to address the domain space.

3.1.3 Domain space

The primary contributions to the development of an ontology design patterns for academic domain ontology. We intend to a well-established list of terms in our domain space. Thereafter for each term determines the general properties. Which help in describe the internal ontology structure. [48]

⁶ http://www.uml-diagrams.org/
### 3.1.4 Summary of terms

Now mention the terms defined in our domain, which define ten classes.

**Table 3-1 Summary terms of academic/university domain**

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>Class</td>
<td>A set of attributes that represent a person.</td>
</tr>
<tr>
<td>Instructor</td>
<td>Class</td>
<td>A set of attributes that represent the instructor.</td>
</tr>
<tr>
<td>Student</td>
<td>Class</td>
<td>A set of attributes that represent a student.</td>
</tr>
<tr>
<td>Course</td>
<td>Class</td>
<td>A set of attributes that represent a course</td>
</tr>
<tr>
<td>Employee</td>
<td>Class</td>
<td>A set of attributes that represent an employee at university</td>
</tr>
<tr>
<td>Academic</td>
<td>Class</td>
<td>A set of attributes that represent a person work in academic field</td>
</tr>
<tr>
<td>Administrative</td>
<td>Class</td>
<td>A set of attributes that represent a person works in the administrative field.</td>
</tr>
<tr>
<td>Instructor</td>
<td>Class</td>
<td>Represent an instructor</td>
</tr>
<tr>
<td>Professor</td>
<td>Class</td>
<td>Represent a professor</td>
</tr>
<tr>
<td>PartTime_Instructor</td>
<td>Class</td>
<td>A set of attributes that represent a part time instructor</td>
</tr>
</tbody>
</table>
3.2 Steps to submit a CPs in ODPs portal

To submit the proposal content ontology design patterns into the ontology design patterns portal. There is a recommended-procedure for the posting defined by the portal. We follow this procedure when submitting the patterns (course and professor CPs). [65]

3.3 Course content ontology design pattern (Course CPs)

Here present our new content ontology design pattern to represent the fundamental attributes and relationships of a course in an educational institution\(^7\). This design pattern is named course, and it is extracted mainly by using the re-engineering technique.

The name of the pattern described in this section is course CPs. This content ontology design pattern for a course in an educational institution. Otologist who use the content ontology design pattern. Could use it as part of build an ontology for an educational institution. Also help in ontology mapping.

3.3.1 Description of the course CPs

In the description of our content ontology design pattern, we will follow the format for describing patterns that were used in describing software design patterns, additionally we suggest to add the data dictionary to give more explanation. [39]

\(^7\) http://ontologydesignpatterns.org/wiki/Community:University
Intent

The aim of this content ontology design pattern (Course Pattern) is to model the core attributes of a course and the basic relationships of the course in an educational institution.

Motivation

We consider context where we have a properly designed ontology for an educational institution. That is maybe reuse the course content ontology design pattern. Proposing a new content ontology design pattern called Course. All proposed CPs are submitted in a semantic web portal called OntologyDesignPatterns.org\(^8\). This portal contains a collection of such content ontology design patterns CPs that were designed for many domains. There are no CPs for Academy, University domain. This domain is promising research area to build an ontology for an educational institute. [41]

Domain

The domain of applicability of the Course content ontology design pattern is University, Academy.

Applicability

\(^8\) http://ontologydesignpatterns.org/wiki/Main_Page
This content ontology design pattern has promising applicability for building an ontology to represent coursework for the academic degree.

**Competency questions**

From modeling session, our goal, we determine what the design pattern should contain (terms, relation...) and identify the competency questions as following:

- What is the course name?
- What is the course number?
- Who is the instructor of the course?
- Who is taking the course?

**Solution description**

This pattern represents the fundamental identifying attributes of a given course in academic institute, for example; has-Name, taughtBy and teaches. These attributes allows us to know the name, number, and level of the course. Also, we can know the instructor who teaches the course and the student.

**Graphical diagram**

Figure 3-1 represents the diagram of the Course content ontology design pattern. Additionally, table 3-2, table 3-3, and table 3-4 provides some of the elements that do not appear on the diagram.
Figure 3-1 Course content ontology design pattern graphical diagram
Data dictionary for course content ontology design patterns

After we determine the main terms, properties and the relation between the terms, we define the conceptual components of the course content ontology design pattern classes, data properties, and object properties.

Data dictionary for classes

The following table shows the classes that is defined in the pattern

Table 3-2 Class for course content ontology design pattern

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
<th>Subclass of</th>
<th>Object Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>A set of attributes that represent a person.</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>Instructor</td>
<td>A set of attributes that represent an instructor.</td>
<td>Person</td>
<td>isPerson</td>
</tr>
<tr>
<td>Student</td>
<td>A set of attributes that represent a student.</td>
<td>Person</td>
<td>isPerson</td>
</tr>
<tr>
<td>Under-Grad-Student</td>
<td>Represent an undergraduate student</td>
<td>Student</td>
<td>___</td>
</tr>
<tr>
<td>Grad-Student</td>
<td>Represent a graduate student</td>
<td>Student</td>
<td>___</td>
</tr>
<tr>
<td>Course</td>
<td>A set of attributes that represent a course</td>
<td>___</td>
<td>CourseTaughtBY</td>
</tr>
<tr>
<td>Grad-Course</td>
<td>Represent a graduate course</td>
<td>Course</td>
<td>Inherited from its super class Course</td>
</tr>
<tr>
<td>Under-Grad-Student</td>
<td>Represent a graduate student</td>
<td>course</td>
<td>Inherited from its super class Course</td>
</tr>
</tbody>
</table>
Class hierarchy

In the following figure 3-2, represents the hierarchy diagram for classes of course content ontology design pattern. The arrow “is-a” means a subclass of the superclass. For example; Instructor class (subclass): is-a subclass of class Person (superclass).

Figure 3-2 Class hierarchies for course content ontology design pattern
**Data dictionary for object properties**

The table 3-3 shows the object properties that characterize the course ontology design pattern as follow:

**Table 3-3 Object properties for course content ontology design pattern**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Domain</th>
<th>Range</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasCourseLevel</td>
<td>Course</td>
<td>Course</td>
<td>____</td>
</tr>
<tr>
<td>taughtBy</td>
<td>Course</td>
<td>Instructor</td>
<td>teaches</td>
</tr>
<tr>
<td>takenBy</td>
<td>Course</td>
<td>Student</td>
<td>____</td>
</tr>
<tr>
<td>isPerson</td>
<td>Student/Course</td>
<td>Person</td>
<td>____</td>
</tr>
<tr>
<td>teaches</td>
<td>Instructor</td>
<td>Course</td>
<td>taughtBy</td>
</tr>
<tr>
<td>Register</td>
<td>Student</td>
<td>Course</td>
<td>Register by</td>
</tr>
<tr>
<td>Register by</td>
<td>Course</td>
<td>Student</td>
<td>Register</td>
</tr>
</tbody>
</table>

**Data dictionary for data properties**

In table 3-4 shows the data properties that are specific for course ontology design pattern as follow:
### Table 3-4 Data properties for course content ontology design pattern

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasName</td>
<td>Course</td>
<td>String</td>
</tr>
<tr>
<td>hasNumber</td>
<td>Course</td>
<td>Integer</td>
</tr>
<tr>
<td>hasFirstName</td>
<td>Person</td>
<td>String</td>
</tr>
<tr>
<td>hasLastName</td>
<td>Person</td>
<td>String</td>
</tr>
<tr>
<td>hasDateOfBirth</td>
<td>Person</td>
<td>DateTime</td>
</tr>
<tr>
<td>hasID</td>
<td>Instructor, Student</td>
<td>Integer</td>
</tr>
</tbody>
</table>

### Consequences

The course content ontology design pattern helps ontology engineers to design Academic, University ontology easily and in an integrated way.

### Implementation

It is essential for a content ontology design pattern to be implemented in a particular ontology language. When an ontology is created, the content pattern can be imported. This content ontology design pattern was implemented in the Web Ontology Language (OWL) [58], using the ontology editor and knowledge acquisition system Protégé [119]. This implementation of the Course content ontology design pattern can be imported in Protégé and used in designing Academic, University ontologies.
We also made an available description of Course CPs online in the patterns collection from the ODP portal at [3]. Next figure 3-3, shows the content ontology design patterns axioms metrics.

![Course CPs axioms metrics](image)

Figure 3-3 Course CPs axioms metrics
3.4 Professor content ontology design pattern (professor CPs)

Now present our second new content ontology design pattern to represent the fundamental attributes and relationships of a professor in an educational institution\(^9\). This design pattern is named Professor, and it is extracted mainly by using the re-engineering technique.

The name of the pattern described in this section is Professor. This content ontology design pattern for a professor in an educational institution. Otologist who use our content ontology design pattern. Could use it as part of build an ontology for an educational institution.

3.4.1 Description of the professor CPs

Similar to when we defined the course content ontology design pattern, we will again use the format from software design patterns, such as intent, motivation, applicability, structure, participants, collaboration, consequences and implementation. Additionally we suggest to add the data dictionary to give more explanation. [39]

**Intent**

The aim of this content ontology design pattern (Professor Pattern) is to model the core attributes of a professor and the basic relationships of the professor in an educational institution.

\(^9\) http://ontologydesignpatterns.org/wiki/Community:University
Motivation

We consider context where we have a properly designed ontology for an educational institution. That is maybe reuse the professor content ontology design pattern. Proposing a new content ontology design pattern called Professor. There are no CPs Academy, University domain. This domain is promising research area to build an ontology for an educational institute. [41]

Domain

The domain of applicability of the professor content ontology design pattern is University, Academy.

Applicability

This content ontology design pattern has promising applicability for building an ontology for academic institute.

Competency questions

From modeling session, our goal, we determine what the design pattern should contain (terms, relation...) and identify the competency questions as following, which help in figure out the main parts of the pattern:

• What is the professor name?
• What is the academic rank?
• What is the professor university ID?
• What is the interest research area?
• How can teach the course?

Solution description

This pattern represents the fundamental identifying attributes of a given professor at the academic institute. For example; has-Name, academic rank. These attributes allows us to know the name, number, and email of the professor.

Structure, Participants, and Collaboration

Figure 3-4 represents the diagram of the professor content ontology design pattern. Additionally, table 3-5, table 3-6 and table 3-7, provides some of the elements that do not appear on the diagram.
After we determine the main terms, properties and the relation between the terms, we define the conceptual components of the professor content ontology design pattern classes, data properties, and object properties.
Data dictionary for classes

The following table 3-5 shows the classes that is defined in the professor content ontology design pattern.

Table 3-5 Class dictionary for professor content ontology design pattern

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
<th>Subclass of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td>A set of attributes that represent an employee at university</td>
<td>___</td>
</tr>
<tr>
<td>Academic</td>
<td>A set of attributes that represent a person work in academic field.</td>
<td>Employee</td>
</tr>
<tr>
<td>Administrative</td>
<td>A set of attributes that represent a person works in the administrative field.</td>
<td>Employee</td>
</tr>
<tr>
<td>Instructor</td>
<td>Represent an instructor</td>
<td>Academic</td>
</tr>
<tr>
<td>Professor</td>
<td>Represent a professor</td>
<td>Academic</td>
</tr>
<tr>
<td>Part Time Instructor</td>
<td>A set of attributes that represent a part time instructor</td>
<td>Academic and Administrative</td>
</tr>
<tr>
<td>Full Prof</td>
<td>Represent a full professor</td>
<td>Professor</td>
</tr>
<tr>
<td>Associate Prof</td>
<td>Represent an associate professor</td>
<td>Professor</td>
</tr>
<tr>
<td>Assistant Pro</td>
<td>Represent an assistant professor</td>
<td>Professor</td>
</tr>
</tbody>
</table>
Class hierarchy

In the following figure 3-5 represents the hierarchy diagram for classes of professor content ontology design pattern. The arrow “is-a” means a subclass of the superclass.

Figure 3-5 Class hierarchies for professor content ontology design pattern
Data dictionary for object properties

The table 3-6 shows the object properties that characterize the professor ontology design pattern as follow:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Domain</th>
<th>Range</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is-Employee</td>
<td>Academic and Administrative</td>
<td>Employee</td>
<td>___</td>
</tr>
</tbody>
</table>

Data dictionary for data properties

In table 3-7 shows the data properties that are specific to professor ontology design pattern as follow:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasName</td>
<td>Employee</td>
<td>String</td>
</tr>
<tr>
<td>hasU-ID</td>
<td>Employee</td>
<td>Integer</td>
</tr>
<tr>
<td>HomeCampus</td>
<td>Employee</td>
<td>String</td>
</tr>
<tr>
<td>hasEmail</td>
<td>Academic</td>
<td>String</td>
</tr>
<tr>
<td>hasDateOfBirth</td>
<td>Employee</td>
<td>DateTime</td>
</tr>
<tr>
<td>Department-ID</td>
<td>Academic</td>
<td>Integer</td>
</tr>
<tr>
<td>AreaOfresearch</td>
<td>Academic</td>
<td>String</td>
</tr>
</tbody>
</table>
Consequences

The professor content ontology design pattern helps ontology engineers to design Academic, University ontology easily and in an integrated way.

Implementation

It is essential for a content ontology design pattern to be implemented in a particular ontology language. When an ontology is created, the content pattern can be imported. This content ontology design pattern was applied in the Web Ontology Language (OWL) [58], using the ontology editor and knowledge acquisition system Protégé [119]. This implementation of the professor content ontology design pattern can be imported in Protégé and used in designing Academic, University ontologies.

We also made an available description of professor CPs online in the patterns collection from the ODP portal at [4]. Next figure 3-6, shows the content ontology design patterns axioms metrics.
Figure 3-6 Professor CPs axioms metric
3.5 A recommended procedure for creating a CPs

To help in creating content ontology design patterns, we recommend a procedure for developing a content ontology design patterns, also can guide you to create other patterns.

Our recommended procedure provide help, guideline to create content ontology design patterns. The recommended procedure has mainly three phases, each phase has steps to perform it. As illustrative example, we will show our works for each phase/step as following. [69]

In a brief manner, the main three phases and sub-steps can picture in the following figure 3-7:

![Graphical diagram for a suggested procedure for creating a CPs](image)

Figure 3-7 Graphical diagram for a suggested procedure for creating a CPs
Phase #1: Chose the domain for the content ontology design pattern “scope.”

In this phase we need to determine the domain, write the competency questions and consider reuse.

**Step1: determine the proper domain**

By trying to answer the following questions, we can determine the proper domain, and we can use the portal OntologyDesignPatterns.org to define the domain.

- What is the domain that an ontology will cover?
- Why is ontology used for?

For our CPs, after we answer these questions, we determine our domain is “Academy, University”. Moreover, this CPs will be used to create ontologies in our domain, so there are no CPs for this domain at the portal OntologyDesignPatterns.org.

**Step2: Formulate and identified competency questions**

Competency questions types of questions should the information in the ontology provide answers. With a note that the responses to these questions may change through the life cycle of creating CPs.

---

10 [http://ontologydesignpatterns.org/wiki/Community:Domain](http://ontologydesignpatterns.org/wiki/Community:Domain)
After decided the questions, which are the basis for a rigorous characterization of the problems that the pattern can solve. The competency questions are mention on page 39.

**Step3: Considering the reusability**

To save the effort, use ontology patterns that have been validated and interacted with the tools that use other ontologies. Here are several different approaches to finding the ontology design pattern that's right for a particular problem. Consider classifications of patterns and their intent. Also, consider how design patterns solve design problems.

Moreover, at this step we can use our guideline to selecting the appropriate ontology design patterns, described in the next section 3.6.

**Phase#2: Determine the conceptual design and documentation**

In this phase, enumerate the major concepts, important terms. Moreover, unify terms informally to find the relation between them. The use of rewording and clarifying terms to produce informal concepts definitions, draw a draft diagram, and we can make assumptions to regard to the terms.

Moreover, documentation describes the terms, a list of the terms and concepts with relevant domains, and a list of the synonyms for terms. As we defined our data dictionary for the classes, object property, and data type property.
Course design pattern contains three major concepts: “Course”, “Instructor”, and “Student”. An instructor can teach a course, student register course, and the course is taught by an instructor, taken by the student, more information about the concepts (classes) in table 3-2.

**Phase#3: Owl formalization**

The content ontology design pattern need to implemented using web ontology language. The three parts of the OWL content ontology design patterns are classes, properties and constraints. During this important phase, we need to define the core parts for the content ontology design patterns.

**Step1: Define classes, class hierarchy, and property restriction**

There are three general approaches to developing class hierarchy from the terms;

- Top-down, first define the most general concepts and then specialize them
- Bottom-up, first define the most specific concepts then organize them in more general classes
- The combination, first defines the most salient concepts, then generalizes and specialize the concepts. [40]

Since there are three main concepts, each one represented by class; for that, there three classes, “Course”, “Instructor”, and “Student”. Top-down approaches use, for the
nature of the molding to look at some educational institute websites. As see in figure 3-2 shows the class hierarchy for course content ontology design pattern.

**Step2: Define properties**

There are three kinds of properties for OWL patterns, object property relates Individuals to each other’s, data type property relate individuals to data types and annotation property for attaching metadata to classes, individuals or properties.

Here; defined the data dictionary for an object property and data type property. Tables 3-3, 4-4 shows these properties for Course ontology design patterns.

**Step3: Define constraints.**

Here; determined the right constraints for class and properties, depending on the previous two steps. For example, as allValuesFrom. Figure 3-3, figure 3-6, are show the axioms defined in our content ontology design patterns.

**3.6 A recommended guideline for selecting an ontology design pattern**

Design patterns have long been well known as a tool for effective software engineering. Also, they are a tool for effective communication. Design patterns are common solutions to common problems. ODPs are not just common solutions. However, they are struggle tested proven, performance and considered the best solution. [57] [56]
Since design patterns are so useful, by supporting the process of ontology engineering and improve the existing ontology. The numbers of patterns are increasing as they evolve. Make the reusability of patterns more difficult to choose one of them. [31] [124]

When starting build a new ontology; never think in terms to design new patterns. Often think in terms of using a ready ontology design patterns, rather than having to reinvent the wheel. Allow speeding up the ontology development process, saving time, cost, and more efficiency, promoting the application of good practice. [102] [128]

Since there are numerous works, suggest that are many benefits of knowing and reusing design patterns in the development process. We recommended a guideline aim to help in some way to select the right ontology design patterns. The guideline as some steps (phase), after each phase we can narrow the search field. [81] [122]

**Step#1 answer the question “what the domain I need an ontology for?”**

After this step, determine the appropriate domain, this the first shrink for the search field. Also in this phase need to write down the main requirements, and then we can make a filtering by these requirements. Now filter the ODPs by the domain. [128]
Step#2 answer the question “There are ontology design pattern solve a similar problem?”

The answer for this question can do by looking at the definition of ontology requirement which determined by the competence questions to define the local use case (LUC). Looking for patterns intent, patterns purpose and scope that called generic use case (GUC) for ontology design patterns.

By match the LUC to a GUC. We can schematize this phase by using the following figure 3-8. After the matching step, we are trying to select the most appropriate ontology design patterns to be reuse. In this phase, we may be named more than one ontology design patterns to be a reuse.

Figure 3-8 Matches an LUC to a GUC
**Step#3 Understand how a particular pattern works.**

Once understand how a particular pattern works. Will be able to see what fits that pattern and know whether to apply or not. This phase helps in figure out how we can work with the selected patterns together.

**Step#4 understand how to integrate selected patterns**

In this stage, we need to know how to reuse these patterns by using a different technique. For example, import or composition. [102]

In a brief manner, the main four phases and sub-steps can picture in the following figure 3-9:

![Figure 3-9 Graphical diagram for how to select ontology design patterns](image-url)
CHAPTER 4

Background material on the software development life cycle and software measurement

In this chapter, we introduce the basic concepts associated to our work in the second part of this dissertation (the second domain) that are used in later chapters. In section 4.1 we cover the concept that relates to software development life cycle. In the sections 4.2 through 4.4 we give more information about software development life cycle and introduce one of the software development models (agile model). The last section of the chapter is about software measurements, classification of software metrics and the advantage.

4.1 Introduction to software development life cycle (SDLC)

A software development life cycle is a collection of steps, or phases that provide a model for the development and lifecycle management of an application or software product. A software process model is basic framework that has guidelines for successful planning, organization, to finishing the software product by a sequence of stages. There are many different techniques and methods used to develop software products. [114] [117]
4.2 Phases of the Software Development Life Cycle

ISO/IEC 12207 \(^{11}\) is an international standard for software lifecycle processes that aims to be the standard that defines all the tasks required for developing and maintaining software. There are six major phases in most software development life cycle models.

1. **Requirement gathering and analysis**

   This phase is a primary focus for the project managers and stakeholders. Meetings with managers, stakeholders, and users are held to determine the requirements. The requirement specification document serves as a guideline for the next phase of the model.

2. **Design**

   In this step, the system and software design is prepared from the requirement specifications that is developed in the previous phase. System design specifications satisfy hardware and system requirements and also help in defining the overall system architecture. The output of this stage serves as input for the next phase of the model.

3. **Implementation / Coding**

   After system design specifications, the work is divided into modules/units, and actual code is written. Since in this phase the code is created, it is the main focus point for the developer. This is the longest period of the software development life cycle.

4. Testing

After developing the code, it is tested according to the requirements in order to help ensure that the product is solving the needs addressed and gathered during the requirements phase. Throughout this stage, unit testing, integration testing, system testing, and acceptance testing are done.

5. Deployment

After successful testing, the product is delivered (deployed) to the customer for their use.

6. Maintenance

When the customers start using the developed system, actual problems come up which need to be solved.

4.3 Benefits of the software development lifecycle process

The main goal of an SDLC process is to help produce a high-quality software product that is cost-efficient, effective and meets or exceeds customer expectations, and reaches completion within time and cost estimates. [112]

4.4 Software Development Life Cycle models

There are various software development process models which are defined, designed and which are followed during the software development process. Actually each
process model has a series of steps which create the software development process. There are some examples of SDLC models followed by industry, waterfall model, iterative model, spiral model, v-shaped model, and agile model. [92] [114] [117] [112]

4.4.1 Agile SDLC model

The agile SDLC model is an iterative and incremental process model with a focus on process adaptability and customer satisfaction by rapid delivery of a working software product. [86] [123]

Agile methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross-functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing. At the end of the iteration, a working product is deployed to the customer and important stakeholders. [28] [92]

The agile software development model offers advantages that other SDLC models cannot address. The agile model focuses on adaptability and response time of changing requirements. Also, the agile model gives us a real ability to make a modification of any phase. We chose this model as our model to build the information system (Skeleton ontology for software metrics plan). Agile is one software development life cycle model
which has some practices that give the agile model the ability to make modifications of any phase. [9] [123]

The main advantage in agile is based on the adaptive software development methods whereas the traditional SDLC models like waterfall model are based on the predictive approach. There are error checks after each iteration. Further, the customer interaction gives the opportunity to the requirements deeply during each iteration until the valuable software is delivered. [77] [120]

4.5 Introduction to software measurement

“When you can measure what you are speaking about and express it in numbers, you know something about it; but when you cannot measure, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of science.” [74]

LORD WILLIAM KELVIN (1824 – 1907)

There are many benefits of software metrics including giving knowledge of the status of software attribute and helping to evaluate the software attributes in an objective way. Software metrics also help to make plans for modifications that need to be implemented in the future. [71]

12 http://www.agilemanifesto.org/
A software metric is a measure of some property of a piece of software or its specifications. Since quantitative measurements are essential in all sciences, there is a continuous effort by computer science practitioners and theoreticians to bring similar approaches to software development. [74]

The goal of software measurement is to obtain objective, reproducible and quantifiable measurements, which may have numerous valuable applications in schedule and budget planning, cost estimation, quality assurance testing, software debugging, software performance optimization, and optimal personnel task assignments. [26]

Software measurement has an increasingly important role in the software engineering field. Software metrics prove to be very effective for building high-quality prediction systems, understanding and improving software development and maintenance projects, and assessing and maintaining the system. [100]

4.5.1 Software metrics

Software metrics are important tools to help assess software process improvement in software-intensive organizations. Software measurement provides information which is helpful in decision making for the several levels that impact the performance of software development life cycle. [10] [11]

Software measurement can help determine the size of an attribute of a software product and evaluate it. Measurement is a mapping from the empirical world to the formal, relational world. Consequently, a measure is the number or symbol assigned to an
entity by this mapping to characterize an attribute. This assignment of numbers or symbols to an entity is made according to unambiguous rules. [26] [34]

The purpose of a software measurement plan is to add value to all life cycle phases of development by effectively improving software engineering methodologies. Actually without metrics at every stage of the software development lifecycle (SDLC), it is difficult to know if the process is providing quality software. [11] [34]

The next figure 4-1 shows the need to apply software measurement. To improve the development process, evaluate the outcome of each phase, and make a prediction about the timeline. [78]

![Figure 4-1 Why software metrics](image)

There are many benefits encouraging the software developer and project managers to use the metric during the entire life cycle phases for software. Use these metrics can improve the quality of the software product. Increase predictability and reliability. Also, they help in the timeline for the product. On another hand can decrease
risks and cost. Metrics helps in understanding the type of improvement required and helps in taking decisions on process or technology change. [18]

4.5.2 Classification of software metrics

Software metrics can be classified into different categories although same metrics may belong to more than one category. Several books present different classification of software metrics, most of them agree on three types as following [105] [111]:

1. Software product metrics

These metrics measure the software product in any stage of its development. They are often classified according to the size, complexity, quality, and data dependency.

2. Software process metrics

These metrics measure the process in regards to the time that the project will take, cost, methodology followed and how the experience of the team members can affect these values. They can be classified as empirical, statistical, theory base and composite models.

3. Software project metrics:

Project metrics is used to avoid development schedule delays, to mitigate potential risks, and to assess product quality on an on-going basis. A software team can use software project metrics to adapt project workflow and technical activities.
4. Software resource metrics

These metrics measure the resource for software product, software development activities, like hardware, knowledge, and people.

Metrics should be simple and computable; it should be relatively easy to learn how to derive the metric, and its computation should not demand inordinate effort or time. Meaningful, quantified measures, and independent of programming language should be based on the analysis model or the structure of the program itself. Effective mechanisms for quality feedback; lead to a higher-quality end product. Enable strategic planning to achieve strategic objectives. [50] [94]

4.5.3 Software measurement process

During all the software development life cycle, it is crucial to apply metrics. Because metrics set expectations, the purpose of software measurement life cycle is to provide a suite of matrices used during the lifecycle of a project. There are different metrics can be determined in each phase of the software development life cycle. For example, if need to draw a metric roadmap for quality assurance process, there are various metrics can be named during requirement definition, design code and so on. Moreover, if another roadmap is draw for a secure software system, then have another sets of metrics. [94] [38]

To have effective software measurements plan, which adds value to all phases of software development lifecycle, there are necessary to presents guidelines for
establishing a measurement process. This section describes an architecture model for designing a software measurement process. The architecture model is pictured in figure 4-2. The first phase is planning, and then implementing, finally improving. [88] [38] [10]

To apply software measurement, the organization must plan for it. Based on its goals, the organization has to determine the processes, and products properties, as size, cost, and time that should be measured. After that, define the measures are to be used to quantify those elements. For each measure, define how the measure must be collected and analyzed. Finally, the measurement process and its products should be evaluated to identify potential improvements. [10] [33]

![Figure 4-2 Software measurement process architecture](image-url)
CHAPTER 5

An ontology design pattern for ontology in software measurement activity

The work presented in this chapter introduces our architectural ontology design patterns for software measurement domain (throughout section 5.3).

In this chapter we presented an architectural ontology design pattern for software measurements domain. That may be used by software developers to create a robust specialized software measurement plan that will encompass the entire lifecycle of a software system. Architectural patterns describe how to arrange an ontology, these types of patterns can be used in conceptual molding phase and at the beginning of the ontology design phase. [83]

5.1 Introduction and motivation

In software measurement domain, we need to develop an information system about the metrics usage during the software development process and the activities during the process. Also, to find the relations and interactions between the metrics and software activities, this information system will be the domain space. [41]

A software development methodology or system development methodology in software engineering is a framework that is used to structure, plan, and control the process of developing an information system.
The metrics can serve as a significant pointer for efficiency and effectiveness to all software development life cycle, which helps in an improvement the process.

5.1.1 Patterns design

In this domain, we intend to build an architectural ontology design patterns representing the skeleton of the conceptual structure of the knowledge domain of software measurements. This design pattern contains two main concepts; “SDLC”, “Software Metrics”, and other minor concepts as sub-classes. It serves to make explicit the relationships between the concepts of the designed patterns.

For the modeling session, our plan to look at some literature studies. Then formulate in natural language, for instance, what are the terms used to define the software measurement plan. Then we develop a graph structure as the basis for a pattern and check that their queries work with these. Such as using Unified Modeling Language UML diagram\textsuperscript{13}.

Our goal was as follows: “Looking for different literature studies and different organization for standardization software metrics to matching the terms representing the main concepts for ontology mapping, design an ontology design pattern which can be used as part of an ontology in software measurements.”

\textsuperscript{13} http://www.uml-diagrams.org/
5.1.2 Conceptual setting

The main concepts – terms- derived from the competence questions for each designed patterns. These concepts aim to address the domain space.

5.1.3 Domain space

The primary contributions to the development of an ontology design patterns for software measurement domain ontology. We intend to a well-established list of terms in our domain space. Thereafter, for each term determines the general properties, which helps in the description of the internal ontology structure. [48]

5.1.4 Summary of terms

Now mention the vocabulary used in software measurement domain, which define thirteen class shown in table 5-1, and four object properties presented in table 5-2.
<table>
<thead>
<tr>
<th>Term Name</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measurement program</td>
<td>Class</td>
<td>A set of attributes that represent the main part of software measurement plan.</td>
</tr>
<tr>
<td>Metrics</td>
<td>Class</td>
<td>A set of attributes that represent the types of metrics.</td>
</tr>
<tr>
<td>SDLC</td>
<td>Class</td>
<td>A set of attributes that represent the phases of software development phases.</td>
</tr>
<tr>
<td>Process metrics</td>
<td>Class</td>
<td>A set of attributes that represent process metrics</td>
</tr>
<tr>
<td>Product metrics</td>
<td>Class</td>
<td>A set of attributes that represent product metrics</td>
</tr>
<tr>
<td>Project Performance Metrics</td>
<td>Class</td>
<td>A set of attributes that represent project performance metrics</td>
</tr>
<tr>
<td>Resources Metrics</td>
<td>Class</td>
<td>A set of attributes that represent resources metrics</td>
</tr>
<tr>
<td>Software Test Metrics</td>
<td>Class</td>
<td>A set of attributes that represent software test metrics</td>
</tr>
<tr>
<td>Analysis phase</td>
<td>Class</td>
<td>Represent an analysis phase in SDLC</td>
</tr>
<tr>
<td>Coding phases</td>
<td>Class</td>
<td>Represent a coding phase in SDLC</td>
</tr>
<tr>
<td>Design phases</td>
<td>Class</td>
<td>Represent an design phase in SDLC</td>
</tr>
<tr>
<td>Testing phases</td>
<td>Class</td>
<td>Represent a test phase in SDLC</td>
</tr>
<tr>
<td>Development phases</td>
<td>Class</td>
<td>Represent an development phase in SDLC</td>
</tr>
<tr>
<td>Maintenances phases</td>
<td>Class</td>
<td>Represent an maintain phase in SDLC</td>
</tr>
</tbody>
</table>
Table 5-2 Summary of terms / Object property

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applied in</td>
<td>Object property</td>
<td>Used to applied a metrics in a phase</td>
</tr>
<tr>
<td>Cascade Impact</td>
<td>Object property</td>
<td>To show the effects of one metrics in other metrics</td>
</tr>
<tr>
<td>Input For</td>
<td>Object property</td>
<td>To display the input for each phase in SDLC</td>
</tr>
<tr>
<td>Ongoing Improve</td>
<td>Object property</td>
<td>Used to apply a performance metric in one of other metrics</td>
</tr>
</tbody>
</table>

5.2 Steps to submit an architectural ODPs in ODPs portal

To submit the proposal architectural ontology design patterns in the ontology design patterns portal there are recommended procedure for the posting defined by the portal. We follow this procedure when submitting the pattern. [64]

5.3 Description of the skeleton measurement program architectural ontology design patterns

Here, present a new architectural ontology design pattern supposed to characterize the overall structure of measurement program ontology in software measurement domain\(^{14}\), we create this as a new domain in the ODPs portal. This design pattern is

\(^{14}\) http://ontologydesignpatterns.org/wiki/Community:Software_measurement
named skeleton measurement program, and it is extracted mainly by defining the main parts of measurement plan.

In the description of our architectural ontology design pattern, we will follow the format for describing patterns that were used in describing software design patterns, additionally we suggest to add the data dictionary to give more explanation. [39]

**Intent**

The aim of this architectural ontology design pattern (skeleton of measurement program) is to show the overall shape of the software measurements ontology.

**Motivation**

We consider context where we have a properly designed a new ontology for a software measurement plan that the architectural ontology design pattern help to figure out the main concepts and the relation between them and the overall shape.

Proposing a new architectural ontology design pattern called the skeleton of measurement program architectural ontology design patterns. The proposed architectural ontology design pattern are submitted in a semantic web portal called OntologyDesignPatterns.org\(^{15}\). This portal contains a collection of such architectural ontology design pattern were designed for many domains.

\(^{15}\) [http://ontologydesignpatterns.org/wiki/Main_Page](http://ontologydesignpatterns.org/wiki/Main_Page)
There is no architectural ontology design pattern for Software measurement domain. This domain is promising for research to build a semantic web page for Software measurement plan.

**Domain**

The domain of applicability of the skeleton of measurement program architectural ontology design pattern is software measurements.

**Applicability**

This architectural ontology design pattern has very good applicability for building an ontology to have a good picture of the shape and main parts for the ontology.

**Competency questions**

Mainly, we have identified the following competency questions, which help in some sense to figure out the main parts of an ontology for software measurement plan, and characterize the main property:

- What are the main part of measurements plan?
- What SDLC consist of?
- What the input and out for each phase of SDLC
- What are types of metrics used?
• What is the relation between SDLC and metrics?

• How one metric affect another metric?

Solution description

This pattern represents the fundamental identifying attributes of a given software measurement plan for example; an input for, ongoing improve, that allows to know the interacting between the metrics.

Structure, Participants, and Collaboration

Figure 5-1 represents the diagram of the skeleton of measurement program architectural ontology design pattern.
Figure 5-1 Architectural ontology design pattern diagram
Data dictionary for architectural ontology design patterns

The conceptual elements of the architectural ontology design pattern are classes, data properties, and object properties. The definition of classes, object property in the next section.

Data dictionary for classes

The following tables (table 5-3, table 5-4) show the classes that is defined in the architectural ontology design pattern.

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
<th>Subclass of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measurement program</td>
<td>A set of attributes that represent the main part of software measurement plan.</td>
<td>—</td>
</tr>
<tr>
<td>Metrics</td>
<td>A set of attributes that represent the types of metrics.</td>
<td>Measurement program</td>
</tr>
<tr>
<td>SDLC</td>
<td>A set of attributes that represent the phases of software development phases.</td>
<td>Measurement program</td>
</tr>
<tr>
<td>Process metrics</td>
<td>A set of attributes that represent process metrics</td>
<td>Metrics</td>
</tr>
<tr>
<td>Product metrics</td>
<td>A set of attributes that represent product metrics</td>
<td>Metrics</td>
</tr>
<tr>
<td>Project Performance</td>
<td>A set of attributes that represent project performance metrics</td>
<td>Metrics</td>
</tr>
</tbody>
</table>
Table 5-4 Class for the pattern / 2

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
<th>Subclass of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Test Metrics</td>
<td>A set of attributes that represent software test metrics</td>
<td>Metrics</td>
</tr>
<tr>
<td>Resources Metrics</td>
<td>A set of attributes that represent resources metrics</td>
<td>Metrics</td>
</tr>
<tr>
<td>Analysis phase</td>
<td>Represent an analysis phase in SDLC</td>
<td>SDLC</td>
</tr>
<tr>
<td>Coding phases</td>
<td>Represent a coding phase in SDLC</td>
<td>SDLC</td>
</tr>
<tr>
<td>Design phases</td>
<td>Represent the design phase in SDLC</td>
<td>SDLC</td>
</tr>
<tr>
<td>Testing phases</td>
<td>Represent the test phase in SDLC</td>
<td>SDLC</td>
</tr>
<tr>
<td>Development phases</td>
<td>Represent the development phase in SDLC</td>
<td>SDLC</td>
</tr>
<tr>
<td>Maintenances phases</td>
<td>Represent the maintain phase in SDLC</td>
<td>SDLC</td>
</tr>
</tbody>
</table>

**Class hierarchy**

In the following figure 5-2 represents the hierarchy diagram for the classes of skeleton measurement program architectural ontology design pattern. The arrow “is-a” means a subclass of the superclass.
Figure 5-2 Class hierarchies
Data dictionary for object properties

The table 5-5 shows the object properties that characterize the architectural ontology design pattern as follow:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applied in</td>
<td>Metric</td>
<td>SDLC</td>
</tr>
<tr>
<td>Cascade Impact</td>
<td>Project performance metric</td>
<td>Resource metrics</td>
</tr>
<tr>
<td>Input For</td>
<td>SDLC</td>
<td>SDLC</td>
</tr>
</tbody>
</table>
| Ongoing Improve        | Project performance metric | Union of  \{  
                                         |                    | Resource metrics  
                                         |                    | project metrics  
                                         |                    | process metrics  
                                         |                    | }                                     |

Consequences

The skeleton measurement program architectural ontology design pattern helps ontology engineers to design software measurements plan ontology easily and in an integrated way. Help in figure out the shape of the ontology, reorganize the ontology modules, and splitting the ontology into smaller parts, allowing to finish the problems on its time.
Implementation

It is essential for an ontology design pattern to be implemented in a particular ontology language. When an ontology is created, the content pattern can be imported. This content ontology design pattern was implemented in the Web Ontology Language (OWL) [58]. Moreover using the ontology editor and knowledge acquisition system Protégé [119]. This implementation of the skeleton measurement program architectural ontology design pattern can be imported in Protégé and used in designing software measurements plan ontologies.

5.3.1 Case study to use the architectural ontology design pattern

In this section, a case study will be presented regarding how to use the architectural ontology design pattern in build a new ontology. By using this architectural ontology design pattern, showing the usability the metrics during software deployment phases.

Case number one

In this first instance study; to demonstrate how this pattern help in applied one metric in a phase of software development lifecycle.

Now applied metric functionality delivered as a type of product metrics in the analysis phase of software deployment phases which is the specification document, by using the object property applied in.
Next figure 5-3 shows this case study, the arrow number 1 shows the domain and range for this object property (as subject, predicate, object). Arrow number 2 shows the example how to use this object property.

Figure 5-3 Case study how to use object property applied in

Case number two

This case study address the competency question *how one metric affect other metric*, there are project performance metric used to improve other types of metrics, for example, resource metrics. As kind of performance metric *change request* metric effect another metric.
The *software usability* metric as a type of resource metric which is applied in analysis phase; in the *specification document*, to know the effectiveness of using the software [100]. During the SDLC if there are a changing. There are metric that is used to see the effect of this change by using the object property *ongoing improve* arrow number 3. This change has cascade impact; we use metric *change request* to applied in specification document arrow number 2. Next figure 5- 4 shows this case study.

![Diagram](image)

**Figure 5-4** Case study how to use object property cascade impact with object property ongoing improve.
CHAPTER 6

Toward a core ontology for software measurements plan

The work presented in this chapter introduces our semi-formal ontology for software measurement domain to help in get a common agreement among software measurements researcher in this domain throughout section 6.3.

In this chapter we presented a semi-formal ontology for software measurements domain. We select product quality metrics as an oriented-domain.

6.1 Introduction

Unfortunately, the terms (concepts) used in the software measurements domain are varied to represent the same meaning. In different standards the same concept is designated by various terms, the same term refers to different concepts. To solve this issue, is an important to establish a common conceptualization regarding to the software measurement domain. [46]

To have software measurements plan well establish, defined, to help the researchers in this domain. Creating a core ontology that is one of a key to create a building block which necessary to provide model formalize the vocabulary and concepts used in the software measurements. [32] [71]

The benefits obtained from ontology as define the concepts in an unambiguous and explicit way with a precise definition of all the terms and clarify the relation between
them. For that, the development of an ontology will facilitate the effort in building a new ontology in software measurements domain. [13] [46]

For brevity, there is no enter into details in all in software measurement terminology. We select software quality metrics as a specific domain to build the semi-formal ontology. [13] [71]

6.2 Software quality metrics

“If you cannot measure it, you cannot manage it.” [98]

Tom DeMarco, 1982

Roger Pressman define in his book the software quality metrics: [105]

“Conformance to explicitly stated functional and performance requirements explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.”

In software projects, it is the most important to measure the quality, cost and effectiveness of the project and the processes. Without measuring these, the project cannot be completed successfully. [113]

Software quality metrics is one of aspect key to the right software product, which need to be planned and controlled by a project manager during the entire project lifecycle. To plan and control it, first of all, has to understand how to measure it. [47]
Software quality metrics is a subset of software metrics that focus on the quality aspects of the product, process, and project. Software quality metrics is more closely associated with process and product metrics than with project metrics. [70]

6.2.1 The benefit of quality metrics

Software quality metrics has advantage that provide valuable information for the software development life cycle overall. Such as common language to gauge progress around quality, potential useful to reduce shortages, increases in efficiency, reductions in complaints, greater profits and better savings. [79] [38]

6.2.2 Standard for the evaluation of software quality

ISO/IEC 9126 as an international standard for the evaluation (assessment) of software quality classifies software quality in a structured set of characteristics. [16]

6.2.3 Types of software quality metrics

1. Product Quality Metrics

The quality of software product it is all characteristic that satisfied the requirements, end-product quality. [105]

2. Process Quality Metrics

Metrics used for measure a process from various dimensions, time, cost and process effectiveness to enhancing the process performance. [85]
3. **Software Maintenance Metrics**

Software maintenance metrics characterize and measure the aspects of products and processes that seem to affect cost, schedule, quality, and functionality of a software maintenance delivery. [115]

4. **Software Testing Metrics**

The testing of software is an important component of software quality assurance. Software testing metrics is essential for evaluating the software to check the system after the change, and to determine the quality. [113]

5. **Project quality metrics**

Used metrics to describe the project characteristics and execution, such as the number of developers, cost and schedule, productivity. [85]

6.2.4 **Software product quality metrics**

Software product metrics helps software engineers to understand the attributes and the quality of the software products based on a set of rules. Some of these attribute related to each phase of software development lifecycle. [109] [110]

6.2.5 **The taxonomy of software product quality metrics**

The quality taxonomy is part of a framework for software quality metrics. [70]
1. Metrics for the Analysis Model

2. Metrics for the Design Model

3. Metrics for Source Code

4. Metrics for Testing

**6.2.6 Importance of assessing product quality**

There are many benefits or advantages to apply software product metrics. To measure how well products conform to “requirements”. Track maturity and progress, ascertain whether products are used properly, and to take necessary corrective actions or improvements. [72] [110]

**6.3 Development an ontology for product quality metrics**

Here present our semi-ontology to represent the fundamental attributes and relationships of a software measurements plan. This semi-ontology is named *product quality metrics ontology*.

**6.3.1 Introduction and motivation**

Metrics has always been used to help guide managers with decisions about their organizations. Software metrics has a significant importance on the quality aspect of a software product. Various metrics should be grouped during the phases of software development lifecycle. [98]
Quality Assurance (QA) metrics is an important part of the SDLC. They provide granular information that helps engineers accurately assess the quality of software builds and figure out what is and isn't working. Metrics is the building blocks of progress and improvement in software creation. [109]

This ontology is used to reach an understandable unified semantic framework application for quality measurements that their concepts and terminologies are an inconsistency among the current studies and literature studies. [131] [47]

6.3.2 Ontology design

There are many techniques exist for development ontology, we have used OWL ontology [97], which not applied in software measurements. The reasons behind use OWL ontology are allowed to define clear terminology with restriction, guarantees a consistent ontology since consistency criteria must be fulfilled, and define deferent types of property with constraints. [84] [43]

We build our ontology using Protégé-5.0.0. As standard ontology languages from the World Wide Web Consortium (W3C OWL ontologies have similar components to Protégé-5.0.0 frame-based ontologies.). [97] [125]

We built an ontology for product quality metrics as quality assurance metrics as the domain for this ontology. We reuse the architectural ontology design patterns we designed to help in figure out the overall shape for our ontology.
For the modeling session, we look at some literature studies. Then formulate in natural language, for instance, what are the terms used to define the software measurement plan. Then we develop a graph structure as the basis for a pattern and check that their queries work with these. Such as using Unified Modeling Language UML diagram\textsuperscript{16}.

Our goal was as follows: “\textit{build an ontology assist in determining what program level product quality metrics would make sense. Moreover, to be meaningful, clear and concise, and be practical to collect and report.}”

As mentioned in chapter four there are several models for software development life cycle. From these models, agile select as a model to represent the software development life cycle. The agile model has four main phases of software development lifecycle, these main four phases are, requirement analysis (with planning) phase, design phase, coding (building) phase, and testing phase. [120]

\textbf{6.3.3 Ontology creation steps}

When an ontology becomes growing, it needs to be handled as a project, for ontology design as development process. For that, there is an iterative approach for building the ontology, will follow in creating this ontology. Next diagram shows the steps for constructing the ontology; figure 6-1. [83] [130]

\textsuperscript{16} http://www.uml-diagrams.org/
6.3.4 Conceptual setting

The main concepts – terms- derived from the competence questions. These concepts aim to address the domain space.

6.3.5 Domain space

The primary contributions to developing an ontology for software measurements domain ontology. We intend to a well-established list of terms in our domain space. Thereafter, for each term determines the general properties, which help in describe the internal ontology structure. [48] [116]
6.3.6 Summary of terms

Now mention the vocabulary that is used in software measurement domain. That is define one data type metrics shows in table 6-1, fourteen class shown in table 6-2, and six object property presented in table 6-3.

Table 6-1 Summary of terms for product quality metrics ontology / data type property

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Has Value</td>
<td>Integer</td>
<td>Used to handle the value of a metric</td>
</tr>
</tbody>
</table>
**Table 6-2 Summary of terms for product quality metrics ontology / classes**

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMO</td>
<td>Class</td>
<td>A set of attributes that represent the main part of the software product quality metrics plan.</td>
</tr>
<tr>
<td>Metrics</td>
<td>Class</td>
<td>A set of attributes that represent the types of metrics.</td>
</tr>
<tr>
<td>Agile-SDLC</td>
<td>Class</td>
<td>A set of attributes that represent the phases of Agile software development phases.</td>
</tr>
<tr>
<td>Product quality metrics</td>
<td>Class</td>
<td>A quantitative measure that represent the product quality metrics</td>
</tr>
<tr>
<td>Metrics for Analysis</td>
<td>Class</td>
<td>A metrics that represent the analysis model metrics</td>
</tr>
<tr>
<td>Metrics for Testing</td>
<td>Class</td>
<td>A metrics that represent the testing model metrics</td>
</tr>
<tr>
<td>Metrics for Source Code</td>
<td>Class</td>
<td>A metrics that represent the source code model metrics</td>
</tr>
<tr>
<td>Metrics for the Design</td>
<td>Class</td>
<td>A metrics that represent the design model metrics</td>
</tr>
<tr>
<td>Coding phases (building)</td>
<td>Class</td>
<td>Represent a coding phase in Agile SDLC</td>
</tr>
<tr>
<td>Design phases</td>
<td>Class</td>
<td>Represent a design phase in Agile SDLC</td>
</tr>
<tr>
<td>Testing phases</td>
<td>Class</td>
<td>Represent a test phase in Agile SDLC</td>
</tr>
<tr>
<td>Requirement analysis</td>
<td>Class</td>
<td>Represent an analysis phase in Agile SDLC</td>
</tr>
<tr>
<td>Agile metrics</td>
<td>Class</td>
<td>Represent the metrics used to improve/test agile model</td>
</tr>
<tr>
<td>Qualified</td>
<td>Class</td>
<td>Represent the rule used to qualify the Agile model</td>
</tr>
</tbody>
</table>
The next table 6-3 define the object property used in the ontology. These object property import from the architectural ontology design patterns.

**Table 6-3 Summary of terms for product quality metrics ontology / object property**

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applied in</td>
<td>Object property</td>
<td>Used to applied a metrics in a phase</td>
</tr>
<tr>
<td>Cascade Impact</td>
<td>Object property</td>
<td>To show the effects of one metrics in other metrics</td>
</tr>
<tr>
<td>Input For</td>
<td>Object property</td>
<td>To show the input for each phase in SDLC</td>
</tr>
<tr>
<td>Ongoing Improve</td>
<td>Object property</td>
<td>Used to apply a performance metric in one of other metrics.</td>
</tr>
<tr>
<td>Affected by</td>
<td>Object property</td>
<td>Used to show a metric is affected by another metric</td>
</tr>
<tr>
<td>Is associated with</td>
<td>Object property</td>
<td>To show there are some metrics associated with other</td>
</tr>
</tbody>
</table>

**6.3.7 Components of OWL ontologies**

An OWL ontology consists of Individuals, Properties, and Classes, which roughly correspond to Protégé-5.0.0 frames Instances, Slots, and Classes. [62] [68]

**Individuals**

Individuals represent objects in the domain in which we are interested. [80]
Properties

Properties are binary relations\textsuperscript{17} on individuals - i.e. properties link two individuals together. [80]

Classes

OWL classes are interpreted as sets that contain individuals. [80]

6.3.8 Defining a software product quality metrics ontology as OWL ontology

In this section, providing a description of our ontology, as OWL format description. Such as intent, motivation, competency questions, applicability. Moreover, go through the steps of development the ontology; as described in chapter two.

The intent

This proposal core ontology aims to enable software engineers, researchers, stakeholders, and ontologies developers to share the common concepts and terms for describing software quality measures and clarify the relationship between them. In common sense, help in shrink the gaps, inconsistencies, and terminology conflicts that can affect the software measurements to reach integrational domain.

\textsuperscript{17} A binary relation is a relation between two things
**Competency questions**

The main purpose of competency questions is to identify the requirement specification that clearly define the ontology purpose and its proposed use.

CQ1. What is the metrics used for product quality metrics?

CQ2. What are the types of product quality metrics?

CQ3. What are the rules that control each phase in SDLC?

CQ4. What is the interaction between metrics?

CQ5. What are the rules used to govern the phases of Agile SDLC?

**6.3.9 Proposed work for implemented the ontology**

To illustrating building the ontology for product quality metrics plan, and called it software measurements ontology (SMO). Protégé 5.0.0 used to create the ontology.

**6.3.10 Steps to build the ontology**

This section represents the steps that is followed to build the ontology. In designing the ontology, there are steps need to follow it. These steps structured in the next figure
Step #1 Identify the classes and class hierarchy

The first step was to define the concepts or classes. All concepts represent as a class. The main class and subclasses are shown in the flowing figure 6-3.
Step #2 Identify the properties of ontology

In this phase defined three types of properties. The three properties can be defined in OWL are object property, datatype property, and annotation property. Object
properties defined according to the relationship between the classes, figure 6-4 shows the object properties.

![Object property hierarchy]

**Figure 6-4 Object property of software measurement ontology SMO**

**Step #3 Create restrictions**

Here define the necessary condition to check the consistency of ontology (restrictions) in class, properties.

**Step #4 Create the instance (Individuals)**

To define the instance (individual) for a class, need to select the class then create the instance for it. Next code shows the code snippet to create an instance. Below the display of individuals for the ontology. Figure 6-5 show that.

```xml
<Class Assertion>
  <Class IRI="#Architectural_DM"/>
  <Named Individual IRI="#Structural_Complexity"/>
</Class Assertion>
```
Figure 6-5 Individuals by class type of software measurement ontology SMO
Step #5 Identify the reasoning of ontology

To create a correct and consistent ontology, the reasoning is the most important part. For that need to check the ontology consistency. Reasoner as plugin in protégé, checks the consistency and finds the logic contradiction implicit in the definitions. [125]

For example, the object property ongoing improve as shown as triple

Agile metrics $\rightarrow$ ongoing improve $\rightarrow$ product quality metrics.

In the ontology define the relation between the instance Complexity_Metrics and DRE instance which is not consistent with rules defined. By starting the reasoner, he finds the inconsistent. As shown in the next figure 6-6. In figure 6-7 another example shows we can use the DL query to retrieve information from the ontology. [59]

Figure 6-6 A test case for of software measurement ontology SMO
Figure 6-7 A test case to retrieve information from the SMO

Step #6 Define the rules of ontology

In this step it is shown clearly the rules that is used to determine if the phases of SDLC is qualified or not, depending on the values of the metrics that is used in each phase. Also to determine if the product is qualified or not. For example, defining rules for qualified specification document; shown in the next figure 6-8. [99] [21]

Figure 6-8 A rules for qualified specification document in SMO
6.3.11 Visualization view

The visualization shows the classes and subclasses of the ontology, as a graph. By using the OWL Viz, which is visualization plug-in of protégé tool. Next figure 6-9 the asserted view of the classes. Figure 6-10, show the whole software measurements ontology, by using the Onto Graf plug-in of protégé tool. Figure 6-11, shows the ontology axioms metrics.

Figure 6-9 Asserted model for the software measurement ontology
Figure 6-10 Entire ontology diagram
### Ontology metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axiom</td>
<td>130</td>
</tr>
<tr>
<td>Logical axiom count</td>
<td>68</td>
</tr>
<tr>
<td>Class count</td>
<td>25</td>
</tr>
<tr>
<td>Object property count</td>
<td>7</td>
</tr>
<tr>
<td>Data property count</td>
<td>1</td>
</tr>
<tr>
<td>Individual count</td>
<td>12</td>
</tr>
<tr>
<td>DL expressivity</td>
<td>ALIN(D)</td>
</tr>
</tbody>
</table>

#### Class axioms

<table>
<thead>
<tr>
<th>Metric</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubClassOf axioms count</td>
<td>21</td>
</tr>
<tr>
<td>EquivalentClasses axioms count</td>
<td>2</td>
</tr>
<tr>
<td>DisjointClasses axioms count</td>
<td>0</td>
</tr>
<tr>
<td>GCI count</td>
<td>0</td>
</tr>
<tr>
<td>Hidden GCI Count</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Object property axioms

<table>
<thead>
<tr>
<th>Metric</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubObjectPropertyOf axioms count</td>
<td>0</td>
</tr>
<tr>
<td>EquivalentObjectProperties axioms count</td>
<td>0</td>
</tr>
<tr>
<td>InverseObjectProperties axioms count</td>
<td>1</td>
</tr>
<tr>
<td>DisjointObjectProperties axioms count</td>
<td>0</td>
</tr>
<tr>
<td>FunctionalObjectProperty axioms count</td>
<td>0</td>
</tr>
<tr>
<td>InverseFunctionalObjectProperty axioms count</td>
<td>1</td>
</tr>
<tr>
<td>TransitiveObjectProperty axioms count</td>
<td>0</td>
</tr>
<tr>
<td>SymmetricObjectProperty axioms count</td>
<td>0</td>
</tr>
<tr>
<td>AsymmetricObjectProperty axioms count</td>
<td>0</td>
</tr>
<tr>
<td>ReflexiveObjectProperty axioms count</td>
<td>0</td>
</tr>
<tr>
<td>IrreflexiveObjectProperty axioms count</td>
<td>0</td>
</tr>
<tr>
<td>ObjectPropertyDomain axioms count</td>
<td>6</td>
</tr>
<tr>
<td>ObjectPropertyRange axioms count</td>
<td>6</td>
</tr>
<tr>
<td>SubPropertyChainOf axioms count</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Data property axioms

<table>
<thead>
<tr>
<th>Metric</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubDataPropertyOf axioms count</td>
<td>0</td>
</tr>
<tr>
<td>EquivalentDataProperties axioms count</td>
<td>0</td>
</tr>
<tr>
<td>DisjointDataProperties axioms count</td>
<td>0</td>
</tr>
<tr>
<td>FunctionalDataProperty axioms count</td>
<td>0</td>
</tr>
<tr>
<td>DataPropertyDomain axioms count</td>
<td>0</td>
</tr>
<tr>
<td>DataPropertyRange axioms count</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Individual axioms

<table>
<thead>
<tr>
<th>Metric</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassAssertion axioms count</td>
<td>15</td>
</tr>
<tr>
<td>ObjectPropertyAssertion axioms count</td>
<td>9</td>
</tr>
<tr>
<td>DataPropertyAssertion axioms count</td>
<td>4</td>
</tr>
<tr>
<td>NegativeObjectPropertyAssertion axioms count</td>
<td>0</td>
</tr>
<tr>
<td>NegativeDataPropertyAssertion axioms count</td>
<td>0</td>
</tr>
<tr>
<td>SameIndividual axioms count</td>
<td>0</td>
</tr>
<tr>
<td>DifferentIndividuals axioms count</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Annotation axioms

<table>
<thead>
<tr>
<th>Metric</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnnotationAssertion axioms count</td>
<td>17</td>
</tr>
<tr>
<td>AnnotationPropertyDomain axioms count</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 6-11 Ontology axioms metrics**
CHAPTER 7

Conclusion and future work

The dissertation addresses several practical research issues that is related to ontology design patents and creating an ontology faced by almost all developer work in the semantic web field. Specifically, work with content ontology design patterns and creating an ontology for the oriented domain.

7.1 Conclusion

The first issue deals with is to create a content ontology design patterns for selected domain. The selected domain is Academic / University domain, in this domain create two content ontology design patterns.

In the second domain; the main contribution is facilitated, and extendable create core ontology in software measurements.

7.2 Future Work

The work presented in this dissertation forms the basis for some avenues of research in ontology design patterns, and we plan to extend our work in a few directions.

The goal of a core (reference) ontology is to provide key building blocks for a specific domain, and specialization, express the core concepts, relations. That is a help in ontology integration and mapping. [32]
Regarding to the future work on this issue. Intend to list the core concepts -terms- in oriented domain space, as attempts to provide consistent terminology. Moreover, general design diagram for these concepts using in ontology in the oriented domain. Thereafter for each term determines the general properties. This diagram called domain space diagram, as comparison framework to solve the problem of completeness and consistency. As in the following diagram 7-1.

![Domain Space Diagram](image)

**Figure 7-1 Domain space diagram**

Then each concept represents a class, also find the relation among the classes. In addition to, the classes and properties for our module are formally encoded using The web Ontology Language (OWL). All semantics of the modules and how they interrelate is also captured schematically by using the semantic module as in the following figure 7-2.
In software measurement domain, we need to develop an information system about the metrics usage during the software development process and the activities during the process. Also to find the relations and interactions between the metrics and software activities, this information system will be our domain space.

Furthermore, the core ontology in software measurements, intend to make this ontology as exemplary ontology.
Appendix A

Appendix A-1 Ontology Repository

Ontology libraries

- DAML ontology library (www.daml.org/ontologies)
- Ontolingua ontology library (www.ksl.stanford.edu/software/ontolingua/)
- Protégé ontology library (http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library)

Upper ontologies

- IEEE Standard Upper Ontology (suo.ieee.org)
- Cyc (www.cyc.com)

General ontologies

- DMOZ (www.dmoz.org)
- WordNet (www.cogsci.princeton.edu/~wn/)
- Linked Open Vocabularies (LOV) (http://lov.okfn.org/dataset/lov/)

Domain-specific ontologies

- UMLS Semantic Net
- GO (Gene Ontology) (www.geneontology.org)
- Bio-ontologies (http://bioportal.bioontology.org/)
Appendix A-2 Definition of the Academic/University ontology domain

**Definition 1:** An ontology module is an ontology that exposes elements to be reused by other ontology modules (defined via an export interface and in turn imports elements from other ontology modules (defined via import interfaces). Additionally, an ontology module may import mappings to define the correspondences with elements in the imported ontology modules. A set of modular ontologies with their interrelationships and mappings make up a modular ontology space.

**Definition 2:** modular architecture refers to the design of any system composed of separate components that can be connected. The beauty of modular architecture is that you can replace or add any one component (module) without affecting the rest of the system.

**Definition 3:** a lightweight ontology is an ontology or knowledge organization system in which concepts are connected by rather general associations than strict formal connections. Examples of lightweight ontologies include associative network and multilingual classifications, but the term is not used consistently.

**Definition 4:** an exemplary ontology is one that can serve as a model that can be imitated or leveraged by ontology engineers in the future. The most fundamental property of an exemplary ontology is well designed for its intended purpose.
REFERENCES


3. Alzyoud, Mazen. Submissions: Course
   http://ontologydesignpatterns.org/wiki/Submissions:Course

4. Alzyoud, Mazen. Submissions: Professor
   http://ontologydesignpatterns.org/wiki/Submissions:Professor


56. Hammar, Karl. "*Ontology Design Patterns: Adoption Challenges and Solutions.*"


58. Hefflin, Jeff. "*An Introduction to the OWL Web Ontology Language.*" Lehigh University (2007).


70. Kan, Stephen H. “Metrics and models in software quality engineering”.


72. Kayed, Ahmad, Nael Hirzalla, Ahmad Samhan, and Mohammed Alfayoumi. "Towards an ontology for software product quality attributes." In Internet and


74. KLASKY, HILDA B. "A STUDY OF SOFTWARE METRICS." PhD diss., Rutgers, the State University of New Jersey, 2003.


InWorkshop on Ontology Patterns, p. 35. 2010.


