POEM: Practical Ontology Engineering Model for Semantic Web Ontologies

Shaukat Ali and Shah Khusro

Cogent Engineering (2016), 3: 1193959
POEM: Practical ontology engineering model for semantic web ontologies

Shaukat Ali and Shah Khusro

Abstract: Sundown of the twentieth century saw the emergence of the World Wide Web. A decade later, semantic web (SW) envisioned enriching of web-accessible information and services with machine-processable semantics to solve the problems of information management and sharing which are aroused by the success of the web. However, success of the SW largely depends on the availability of formal ontologies for representing and structuring information in different domains. To facilitate web ontology engineering (OE), methodologies are proposed by the researchers since years. However, OE particularly web OE is still immature and none of the methodologies is standardized for web OE projects as yet. This paper presents an evolutionary SW ontology engineering methodology called POEM. POEM presents a complete methodology comprising of various phases for performing conceptualization, designing, implementation, evaluation, and project management tasks. POEM exploits and incorporates the large experiences from the widely used software engineering standards to make the methodology more realistic and provide greater functionality. Due to its broad nature, POEM has the potential applicability in a wider range of OE projects. Comparison of POEM with other OE methodologies has ascertained POEM as more advantageous and easy methodology for SW ontology engineering.
1. Introduction

Ontology is old genie in a new bottle. The term ontology was first used in Philosophy in the nineteenth century where it gave an organized justification of being (Breitmann, Casanova, & Truszkowski, 2007) and is now a most relevant word in the knowledge engineering community (Corcho, Fernández-López, & Gómez-Pérez, 2003). A widely quoted definition of ontology in computer science literature is of Gruber’s (Gruber, 1993): “An ontology is a formal, explicit specification of a shared conceptualization.” Ontology has gained increasing popularity in the computer science fields (i.e. knowledge management and sharing, e-commerce, intelligent information integration, and database integration), where researchers believe of its significant role in the foreseeable future (Noy & McGuiness, 2001; Su & Ilebrekke, 2002). Ontology popularity is due to its promise of a shared and common understanding of some domain of interest for enhancing people and machines communication. Ontology may presume different formalisms but corresponds to a knowledge area (domain), where it give hierarchical description of explicitly defined concepts, detailed descriptions of concepts’ attributes, and inter-relationships between the concepts that could be used for the domain interpretation.

Web is phenomenally a simple artifact providing new methods of information organization, retrieval, and sharing. Web simplicity is a great strength of the web and is an important factor of its popularity and growth (Horrocks, 2008). However, the exponential growth of the web resulting into information overload problem that is increasing the information interpretation burden of the human users. Semantic web (SW) envisions for making the web information more machine processable and understandable by mapping information into a rigorous structure using ontologies (Benslimane, Malki, Rahmouni, & Rahmoun, 2008). The term “ontology” is regularly quoted in SW literature, and it is believed that ontology will ease numerous types of information management tasks including information/knowledge engineering and management, information retrieval and integration, and information storage and sharing on the web. (Benslimane et al., 2008; Kapoor & Sharma, 2010). The quick and cheap development of formal domain-specific ontologies will serve as the major source of the SW success and proliferation (Benslimane et al., 2008; Vanitha et al., 2011). SW ontology allows for semantically enriching of web resources that is a pre-condition for developing novel advanced web services such as semantic search and web resources retrieval (De Nicola, Missikoff, & Navigli, 2009). Knowledge engineers (KEs) in cooperation with domain experts (DEs) builds SW ontologies which is a tedious, time-consuming, costly, and laborious task and might hinder the advancements of SW activities (Youn et al., 2004). To build a new ontology, one needs to answer a number of questions (related to contents, language, methodology, and tools) to guide and carry out the development process (Corcho et al., 2003). The automated SW ontology engineering tools are in their infancy, immature, and suffers with a number of shortcomings such as OWL 2 is designed to overcome some of the expressibility, syntax, and semantic problems of its prior version but it has lack of useful modeling constructs such as universal restriction (Rodrigues, Flores, & Rotta, 2015). Researchers have contributed a significant amount of research for analyzing and categorizing the available methodologies, tools, and languages for ontology building (Alam, Ali, Khan, Khusro, & Rauf, 2014; Corcho et al., 2003; Noy & McGuiness, 2001; Su & Ilebrekke, 2002).

An SW ontology building process is structurally and logically as complex as the production of a software artifact (De Nicola et al., 2009). Therefore, a cohesive and structured project development framework is needed for building SW ontologies which should ensure careful project planning, clear identification and definition of roles and tasks, and addressing of tones of crucial details for successful migration. In the past several years, several ontology engineering (OE) methodologies are proposed for helping researchers and organizations for successful development, implementation, and migration of ontologies. However, they are not suitable for SW OE where information are scattered, unknown,
and requires integration and mapping of information in real time. None of the methodologies is widely
acceptable due to a number of reasons including (Ashraf, Chang, Hussain, & Hussain, 2015; Corcho
et al., 2003; De Nicola et al., 2009; Ferreira et al., 2007; Karapiperis & Apostolou, 2006): (1) none
of them is accepted as a standard or not adopted widely to set as a standard in the foreseeable near
future, (2) lacking with guidelines to engineer ontologies reusing and reengineering of ontological and
non-ontological resources that are widely accepted in a domain, (3) do not describe OE process with
the same fashion and granularity as the methodologies used for software engineering, (4) immature
if compared with software engineering and knowledge engineering methodologies, (5) not unified
that is each group is following its own approach and having variable degree of dependency of a de-
veloped ontology on its applications, (6) mostly focus on the centralized development of ontologies by
employing KEs and DEs, however, methodologies are also proposed that advocates collaborative ap-
proaches for constructing ontologies, (7) targeted specifically for ontology developers and having no
provenance for software developers to enhance their capabilities and understandings about ontology
building for developing effective ontology-based applications, (8) primarily focusing on the core ontol-
ogy development activities, especially on the ontology implementation and do not consider OE as a
project by completely ignoring the project management-related activities such as project planning
and scheduling, risk analysis, and usage analysis, and (9) most of them are used in a single domain
and have not been used by external groups. A few of the methodologies are also proposed specifically
for web-specific ontology engineering, but they are very shallow and superficial and could not handle
the technical, development, and management tasks related with a web OE project. Web ontologies
produced by the proposed methodologies would be light-weighted and would lack with detailed se-
manics of information in a domain. They are majorly based on heuristics and do not consider using of
widely accepted scientific methods, necessary for ontology building as they are used in the software
engineering and knowledge engineering disciplines. Therefore, none of the approaches presented is
fully mature to meet the unique requirements and applications of SW ontologies.

Ontology-based software engineering is an emerging field where ontologies provide conceptual
basis for extending and automating software engineering process including modularization, reuse,
cross-cutting dependencies, standardization of domain knowledge, and distribution and integration
of software components. In this paper, we have presented a methodology for SW ontology building
called practical OE model for SW ontologies (POEM) that leverages the large valuable experiences
and concepts from the widely accepted standards of software engineering and other state-of-the-
art technologies. POEM is easy to conceptualize and use, and is aimed to provide greater functional-
ity and interoperability. POEM consists of a sequence of activities for building SW ontologies form
inception to deployment at any scale (i.e. lightweight or heavyweight). Goal of the POEM is similar to
the software engineering: turning the process of SW ontology building from an art into an engineer-
ing discipline. Main objectives of the POEM are: (1) general enough in the sense to provide greater
understandability and help to both software developers and ontology practitioners to build SW on-
tologies, (2) clear incorporation of software engineering processes or activities (i.e. requirements
specifications, evaluation, evolution, risk analysis, and management activities) in SW OE and clear
and precise definition of each process or activity stating its purpose, inputs and outputs, roles and
responsibilities of the experts, and execution, (3) reduction of time and costs in the production of
ontologies, (4) progressively testing of the incremental versions of the ontology for enhancing qual-
ity of the ontology, and (5) readily availability of the incremental version of the ontology for using in
ontology-based applications. In reality, POEM is focused for SW ontology development, but its pro-
cessing steps are broad and flexible enough to accommodate enterprise level ontology develop-
ment and deployment projects. In the context of this paper, POEM focuses on centralized OE.
However, POEM has the potential of supporting collaborative OE with certain extensions and
refinements.

2. Related work

Apparently, ontology design seems similar to object-oriented design but designing classes and rela-
tions in ontology building is different from object-oriented programming. Therefore, an object-ori-
ented program structure representing a domain is different from an ontology representing the same
domain (Noy & McGuiness, 2001). Knowledge from the fields of data modeling, object-oriented software analysis, and ontology modeling design patterns can be used for designing applicable principles for domain-specific ontology design patterns (Vujasinovic, Ivezic, & Kulvatunyou, 2015). Despite the emergence of several OE methodologies, none of them is succeeded in gaining significant marketplace and is accepted or going to be accepted as a standard in the foreseeable future. Therefore, OE is still much of an art (Ferreira et al., 2007). OE methodologies can be either centralized or collaborative. In centralized OE, all of the stockholders (i.e. ontology development team (ODT), DEs, and organization) work as a single group under a single roof at a single location with a single agenda. In collaborative OE, the stockholders work as decentralized groups at disperse geographical location with potentially divergent agendas and mutually established consensus (Simperl & Luczak-Rösch, 2013). These two schools of OE have support from the specifically designed and developed scenario processes, methodologies, and tools. POEM methodology could potentially support both of the paradigms but here mainly focuses on centralized OE. Scope of the paper includes centralized OE methodologies that are developed specifically for web ontology engineering and consisting of all the steps ranging from conceptualization to deployment. However, a slight review of the earlier centralized OE methodologies is also presented in the coming paragraphs.

2.1. General OE methodologies
Several OE methodologies are proposed by the researchers and organizations since years. However, research groups have no consensus and each group is using/applying his own methodology (Corcho et al., 2003; Noy & McGuiness, 2001). Some researchers believe that ontology development is an iterative process (Vanitha et al., 2011) starting with a rough initial version which would be progressively revised, evaluated, refined, and debugged using either application or problem-solving methods or arguing with DEs. The iterative process will continue throughout the life cycle of ontology (Kapoor & Sharma, 2010). Among the different alternative methodologies, the one to be selected largely depends on the application to be developed and the extensions that are anticipated (Noy & McGuiness, 2001). Generally, an ontology building methodology may include: identifying domain and scope of ontology, determining reuse of already available ontologies, identifying significant terms in the ontology, defining classes and the class hierarchies, identifying class properties (slots) and constraints, and defining instances of the classes (Kapoor & Sharma, 2010; Noy & McGuiness, 2001).

Uschold and King (1995) proposed a methodology for building Enterprise Ontology and is application independent. The methodology suggests four activities: (1) identifying purpose of ontology, (2) building ontology, (3) evaluating ontology, and (4) documenting ontology. Building activity encompasses knowledge acquisition, coding, and integrating other ontologies inside the current one. The methodology uses middle-out approach for identifying concepts and is application independent.

Grüninger and Fox (1995) proposed a methodology during the development of Toronto Virtual Enterprise (TOVE) project ontology for addressing business process and activities in a modeling domain. They proposed the use of first-order logic for the development of knowledge-based systems where informal description of ontological specifications needed to be defined earlier and then formalized. The methodology uses middle-out approach for identifying concepts and is semi-application dependents. To determine scope of the ontology, a set of competency questions (CQs) are identified. The CQs and their answers are used for the identification of main concepts, attributes, relations, and axioms in the ontology that can be properly encoded in first-order logic. It provides a good guideline for transforming informal scenarios into computable models.

SENSUS (Swartout, Patil, Knight, & Russ, 1996) methodology is developed at IST (Institute of Sciences Institute) which derives domain-specific ontology from a giant ontology using top-down approach and is semi-application dependent. A set of potential most relevant terms in a domain called “seed” terms are identified first. After identification, seed terms are manually located in broad-coverage ontology (i.e. SENSUS ontology that is containing more than 70,000 concepts). All the concepts coming up on the path from seed term to the top (root) of SENSUS are incorporated. A relevant term that is yet not appeared is manually added and this process is repeated until no term is missing.
A sub-tree entirely can be added if terms with large number paths are found or if majority of the
nodes in a sub-tree are found relevant to the other nodes in the tree/sub-tree. This methodology
prompts the usage of same base ontology for the development of many domain-specific ontologies.

KACTUS (Bernaras, Larestgoiti, & Corera, 1996) in KACTUS project proposed a methodology that
follows top-down approach for identifying concepts and is application-dependent. Ontology can be
built reusing existing relevant ontologies and the ontology can also be incorporated by later applica-
tions in their ontologies. Applying this criterion, ontology for applications can be built which would
contain the required agreed knowledge.

METHONTOLOGY (Lopez, Gomez-Perez, Sierra, & Sierra, 1999) methodology has been created in
the Artificial Intelligence Lab in the Polytechnic University of Madrid. This methodology proposes
ontology building in different ways either from the scratch, reusing other ontologies, or by reengi-
neering other existing ontologies. The METHONTOLOGY framework defines set of activities that
should be performed while building ontology such as scheduling, control quality assurance, specifi-
cation, knowledge acquisition, conceptualization, integration, formalization, implementation, evalua-
tion, maintenance, documentation, and configuration management. The life cycle of ontology
shows that ontology throughout of its life time should pass through which of the stages, and its in-
terdependency with the life cycles of other ontologies. This methodology also identifies that which
techniques have to be carried out in each activity, what should be the outputs of each activity, and
how they have to be evaluated. METHONTOLOGY uses middle-out approach for identifying concepts
and is application independent.

On-To-Knowledge (Staab, Studer, Schnurr, & Sure, 2001) methodology is application dependent,
uses top-down approach for concepts identification strategy, and specifies the goals to be achieved
by the management tools. The methodology includes steps: (1) kick-off: specify and capture ontol-
ogy requirements, identify CQs, identify and study potentially reusable ontologies, and built an initial
draft version of the ontology, (2) refinement: producing a mature and application-oriented ontology,
(3) evaluation: checking requirements and CQs and testing ontology in application environment, and
(4) ontology maintenance: carrying out activities related to ontology maintenance.

Detailed comparisons of the ontology building methodologies using IEEE 1074–1995 standard have
been quoted in several relevant literatures (Gómez-Pérez, Fernández-López, & Corcho, 2007), but none
of them is found fully mature if compared to software engineering methodologies. Each of the meth-
odologies has certain enhancements and shortcomings such as On-To-Knowledge describes more
activities but do not provide accurate descriptions of each activity (Gómez-Pérez et al., 2007). None of
the methodologies covers all of the processes involved in ontology building. However, METHONTOLOGY
is found as the most mature methodology which could be heavily used in for ontology building in dif-
f erent domains as recommended by FIPA for the ontology construction task (Casely-Hayford, 2003;
Corcho et al., 2003). A state-of-the-art survey of classification and principles for domain-specific ontol-
ogy design patterns development is presented in (Vujasinovic et al., 2015). Furthermore, several meth-
odologies are proposed for other tasks as well such as ontology learning, ontology reengineering,
ontology evolution, ontology evaluation, and ontology merging (Corcho et al., 2003). However, they
are not discussed here because they do not fall into the scope of the paper.

2.2. Web OE methodologies
Web ontology describes metadata about resources in web documents. Web ontology makes web
documents to be more processable by applications rather than only presented and interpreted by
humans. A web ontology building process is also treated as an iterative process which starts with a
rough first pass and is revised, refined, and more details are added in the subsequent iterations until
a complete ontology has been formed. However, web ontology building is different from traditional
ontology building due to non-standardized varying and scattered nature, and not readily availability
of information. Devising of a specific methodology for web ontologies that should meet the unique
nature, needs, and applications of web information has not received considerable response from the research community as yet.

Vanitha et al. (2011); Noy and McGuiness, (2001) proposes a simple iterative-based web ontology process model consisting of seven activities including: (1) determining the domain and scope of ontology, (2) reusing existing ontologies, (3) enumerating terms in the ontology, (4) defining classes and class hierarchy, (5) defining properties of classes, (6) defining facets (role restrictions) of the slots, and (7) creating instances. During the lifetime of ontology these activities would be repeated in circular fashion again and again until a functional ontological product has been obtained.

Kalyanpur, Hashmi, Golbeck, and Parsia (2004) proposes a life cycle for casual web ontology development process. The proposed methodology is similar to the standard web page development process, where users uses standard HTML editors (e.g. MS FrontPage) to quickly arrange and layout certain information they wish to deploy on the Web and creates links to the existing relevant information. The life cycle includes key stages including the use of shorthand key notation to draft ontology skeletons quickly, an effective ontology searching algorithm combining keyword with DL-based constructs to search-related concepts, an iterative copy–paste mechanism to borrow relevant fragments from a related existing ontology, and natural language-based presentation of terms to build and maintain ontologies effectively. An ontology development toolkit called SWOOPed is invented using these features and stages.

Some researchers have suggested using of semi-automatic approach of ontology extraction as the practical approach. Rules and methods have been suggested for ontology engineering by relational databases reverse engineering. (Stojanovic, Stojanovic, & Volz, 2002) approach suggests a set of rules for analysis of relations, keys, and inclusion dependencies for mapping relational databases constructs into semantically equivalent constructs in ontology. (Tijerino, Embley, Lonsdale, Ding, & Nagy, 2005) suggests understanding of HTML table structure and contents, constraints holding between the concepts, matching the recognized concepts with more general specifications of related concepts, and merging the resultant structure with other similar knowledge representation essential for web ontology building. (Astrova & Stantic, 2005) suggests construction of ontology by analysis of HTML forms for creating a model schema, transforming the schema into ontology, and creating ontological instances from data contained in the page. (Benslimane et al., 2008) suggests reverse engineering of data-intensive web sites into ontology based SW by analyzing the related HTML page for extracting the semantics of relational database, and merging the semantics with the captured relational schema to build ontology.

The proposed methodologies are valuable but are limited in scope due focusing mainly on the core web ontology building activities such as identifying classes, defining properties and relations between classes. They have no definitions of strategies for ontological constructs identifications and application dependency/independency, and have completely ignored management activities (i.e. schedule and quality assurance) as well as supporting activities (ontology integration, ontology evaluation, and ontology documentation, etc.) which are essential for web ontology building projects like any other software development project. Furthermore, the presented approaches are very shallow and providing no technical details of even prescribed engineering activities during a web ontology building process. Similarly, the reverse engineering approaches could also not produce an effective ontology for a domain modeling because the ontologies produced would not be semantically enriched due to absence of axioms, requirement of auxiliary information, and error-prone in analysis of tables' structures and schemas. Furthermore, they could also be very time consuming due to analysis of large data in a table or database. Thus, none of them is suitable for implication in a heavyweight web ontology building project.

3. An overview of POEM
A methodology has to answer what, who, and when questions about performing activities in a development process (Gómez-Pérez et al., 2007). IEEE defines methodology as “a comprehensive,
integrated series of techniques or methods creating a general system theory of how a class of thought intensive work ought to be performed” (IEEE, 1990). A methodology is composed of methods and techniques where methods are composed of processes, processes are composed of activities, and activities are composed of related tasks. Together, using the guidelines of IEEE and support from SW technologies can provide conceptualization for POEM. What distinguishes POEM from the other web OE methodologies is its software engineering compliance, evolutionary, incremental, and iterative nature. As an innovative aspect of the PEOM is the presentation of life cycle for web ontology building projects from their inceptions to live deployments which will help project team in focusing and achieving business values because no value can be realized until the solution is deployed and is operational. POEM believes in building ontologies in the same way of developing software and combines features from several of the software engineering industry standard methodologies into a single suit. Included features are: (1) stressing on the accomplishment of milestones from waterfall model, (2) coupling of the iterative nature of prototyping model with the systematic and controlled aspects of classical model from spiral model to accommodate changes as process progresses, (3) facilitating of system development through reusing of the existing prepackaged components from component-based development model, (4) supporting of system development using object-oriented concepts (e.g. classes, objects, and inheritance) from object-oriented process model, (5) emphasizing on verifications and validations from V-model, (6) rapid and continuous delivery of useful ontological product from agile model, and (7) using of UML for the preparation of all of ontology building blueprints.

SW ontology is also of evolutionary nature which like any complex system evolves over time with changes in the information domain. Moreover, the non-standardized and scattered nature, and readily unavailability of web information would make an SW ontology construction in a single go increasingly difficult. Therefore, a methodology making a straight line path to an end SW ontological product has unrealistic implication in the domain (Hesse, 2005). Furthermore, several other hidden factors (i.e. unpredictable in the early stages of a project) can also make completion of a comprehensive SW ontological product impossible; however, a limited version (i.e. based on a set of core product requirement) could be introduced to meet competitive and business needs. Therefore, a methodology is needed that has been explicitly designed to divide the development process into distinct and well-defined activities, and accommodate SW ontological products that evolve over the time. Methodologies with such characteristics are termed as evolutionary models in software engineering. Evolutionary approach is considered appropriate for ontology development (Hesse, 2005). Evolutionary models are iterative by nature which enables developers to develop increasingly more completed versions of a product in increasing number of iterations.

POEM is evolutionary by nature that would complete an SW ontology building project in a number of iterations as more and more information becomes available but is also incremental because at each iteration the ontology would be further detailed and extended in a stepwise fashion. POEM divides the ontology building methodology into several cycles (i.e. iterations), a cycle into phases (i.e. process), and a phase into activities for requirements specifications, project planning and scheduling, risk analysis, implementation, and testing. Each cycle ends with the release of a new incremental version of ontology. The activities can be divided into tasks (i.e. sub-activities), which will be smaller in number for a small/simple ontology project and larger in number for a large/complex ontology project. A sketchy representation of POEM model is presented in Figure 1. The ontology building process starts at the core of the model with the onetime processes of project identification and initiation, and feasibility analysis. The first iteration is of inception nature and majorly concerned with the identification/capturing of requirements in terms of the ontology requirements specifications (e.g. identifying domain, and scope of the ontology), concepts development and analysis, and initial planning and scheduling. This would not probably involve any implementation or testing processes and could take several other iterations as well, if required. The iteration would release ontology glossary that would contain terms from the domain of interest and their informal definitions. The subsequent iterations would include detailed conceptualization and designing, planning and scheduling, implementation and testing, feedback and roadmap for next iteration, and release of
incremental versions of ontology. The early iterations would end in the production of preliminary implementation to provide small skeletal blueprints of ontology. Stepwisely with enhancements and refinements, increasingly more sophisticated additive versions (increments) and eventually the final version of domain ontology would be released as the iterations proceeds further. Major versions of the ontology produced after the increasing iterations would have to be constantly aligned with the modeled reality. The later iterations will result increasingly in the ontology enhancements and maintenance. POEM is a milestone driven methodology where a milestone has to be produced at the end of every phase of each cycle containing the achievements and the criteria for accomplishing the phase. Each phase could have interim milestones which would help in achieving a phase’s final milestone. Likewise, deliverables must be completed and produced where critical questions related to the development must be satisfactorily answered. In parallel to all these activities, umbrella activities such as ontology configuration management, ontology quality assurance, ontology development project tracking and control, formal technical reviews, and documents preparation and productions are to be carried out as well.

To demonstrate the practicality of POEM, the methodology is supported with a running proof-of-concept ontology building example of a web page in the web domain called Web Page Ontology (WPO). In the example, the ontology is chiefly concerned with semantic modeling of a web page structure and content, and tracking of a user’s activities and interactions taking place while browsing a web page (e.g. downloading a picture from a web page by a user at an instant of time).

4. The POEM methodology

POEM is an evolutionary methodology, advocating for completion of ontology building process from inception to implementation in a number of iterations and incremental versions. Each of the iterations is composed of a number of phases. In the following subsections, detailed specific information about each of the phases is presented in reference to our WPO example.

4.1. Project identification and initiation

Identification of business needs yields in the identification of ontology building project. A business need may arise either due to some kid of “pain” within an organization (i.e. poor customer service levels or increase in competition) or an organization wants to take maximum advantages by leveraging the emerging IT technologies (e.g. cloud computing, radio frequency identification (RFID), and SW). Mostly, information system projects grow out of business process management (BPM) methodology used by organizations to improve their end-to-end business processes. Automating business process can bring several important benefits to organization by replacing ineffective information management process with technology components to gain cost efficiencies. Web has been proven as the rich information highway providing novel methods and opportunities for business processes improvements. Organization

Figure 1. Top-level representation of POEM methodology.
can exploit novel SW technologies (i.e. ontologies) to provide high-valued services to their customers using the Web such as music CDs transitioning on the Web to provide enhanced searching, purchasing, and downloading services to customers. Ontology development team ODT\(^1\) and business people needs to work conjunctively for finding ways to use ontologies to support business needs and bring ultimate real business objectives such as increasing sales, improving customer sales, and decreasing operating expenses.

The ontology project is properly initiated by issuing Ontology System Request Document (OSRD) describing the business reasons for building ontology and the values expected from the ontology to provide. An OSRD comprises five elements: project sponsor, business need, business requirements, business value, and special issue. Template of OSRD along with the description of each element is shown in Figure 2. Once the ontology system request got approval from the approval committee, the next step is the feasibility analysis.

4.2. Feasibility analysis
A knowledge management system needs to be properly integrated into its operating organization for functioning satisfactorily (Staab et al., 2001). Ontology building project success and failure can be attributed to a number of factors. Feasibility analysis analyzes these factors by firstly identifying the problem/opportunity areas and secondly putting them together into a wider organizational perspective. Feasibility study identifies the potential risks associated with ontology building project that must be managed before proceeding into the project. The feasibility analysis acts as a decision support system for determining technical and economical feasibility of ontology building project by selecting potential focus areas and target solutions. Technical feasibility (also called technical risks analysis) answers the question “Can we build it” by determining the extent to which the ontology can be successfully designed, developed, and implemented by ODT. Ontology technical feasibility is actually technical risk analysis that can endanger successful ontology building project completion (Dennis, 2012). The life-threatening risk includes level of ODT and DEs familiarity with the application and technology, accurate estimation of ontology project size (i.e. development team, time, and features), and compatibility of ontology with the technologies already existing in an organization. Economic feasibility (also called cost–benefit analysis) answers the question “Should we build the ontology?” by identifying and analyzing ontology building project's cost and benefits, assigning values to them, calculating future cash flows, and measuring the financial worthiness of ontology (Dennis, 2012). The format and approach of feasibility study vary according to an organization’s business needs and processes. POEM incorporates feasibility analysis and uses the feasibility analysis approach proposed by CommonKADS methodology (Schreiber & Akkermans, 2000). The feasibility analysis activity needs to be performed before starting ontology building process and would serve as basis for the subsequent ontology project planning and scheduling, and risk analysis phases.

| Figure 2. Template of ontology system request document OSRD. |
|-------------------------------------------------------------|
| **Ontology System Request Document**                        |
| **1 Project Sponsor**                                       |
| The person who initiates the SW ontology project and who serves as the primary point of contact for the project on the business side. For example, IT Manager, CEO, and Vice President of Marketing etc. |
| **2 Business Need**                                         |
| The business-related reasons for initiating the SW ontology project. For example, increasing sales, improving access to information, and improving customer service etc. |
| **3 Business Requirements**                                 |
| The business capabilities that the SW ontology will provide. For example, providing online access to information semantically, capturing and semantically organizing customer information, and providing semantic search capabilities etc. |
| **4 Business Values**                                       |
| The benefits that the SW ontology will create for the business. For example, 5% increase in sales, and $100,000 cost saving from removal of existing system etc. |
| **5 Special Issues**                                        |
| Issues relevant to the development and implementation of SW ontology. For example, ontology needed to be implemented before the Christmas etc. |
4.3. Ontology requirements specifications and conceptualization

The goal of this phase is to capture knowledge to be modeled in ontology in such a way to describe its universe of discourse, purpose and goal, contents and ingredients, users, and uses, etc., and to model the captured knowledge semantically to increase its conceptualization. This phase requires agreement and commitment between ODT, DEs, and end-users for establishing effective communication mediums and methods. Guidelines for building ontology and objectives of ontology from a user’s point of view would be identified and laid down in the earliest meetings. However, detailed knowledge capturing needs to answer several questions related to the ontology to be developed including (Youn et al., 2004): (1) What is the domain of interest of ontology?, (2) What is the scope of ontology?, (3) What are the business purposes and economic benefits of ontology?, (4) Who are the intended users and what are the intended uses of ontology?, (5) Is there any other ontology existing in the domain?, (6) what are the questions to be answered by ontology?, (7) How would you describe “good” result that would be generated by a successful ontology?, and (8) Will any special performance issues or constrains affect the way the solution is approached?. During ontology building process, answers to these questions may vary, but, at any instant, of time they will help to constrain scope of ontology (Youn et al., 2004). Ontology Requirements Specification Document (ORSD) should be produced as output of this phase that would contain detailed description of requirements to be represented in ontology, facilitating searching of knowledge sources for reuse, intended uses and users of ontology, and ingredients for validation and verification of ontology, etc. Figure 3 shows template of the ORSD. The ORSC phase is divided into a number of activities to capture the required knowledge and answers the above questions. Figure 4 depicts the sequence of execution of the activities.

4.3.1. Ontology requirements specifications

Ontology requirements specifications (ORSs) summarizes the features and capabilities that the ontology would have to include, such as the ability to collect customer orders online or the ability for suppliers to receive inventory information as orders are placed and sales are made, etc. Users, DEs, and ODT work closely to carry out this activity by taking a set of ontological needs as input for identifying the ontological requirements. In this early stage, ODT would also look for already developed and potentially reusable ontologies as a starting point for the development of the domain ontology. Ontological requirements can be classified as non-functional requirements and functional

| Ontology Requirements Specifications Document |
|-----------------------------------------------|
| 1 | Purpose and Scope and Level of Formality |
| | Main goals and objectives of the ontology should be defined in terms that what are the main functions or roles that the ontology should provided in a domain of interest. The general coverage and the degree of detail of the ontology should be defined. The level of formality and possible description languages and tools for the ontology development should be determined. |
| 2 | Uses and End-Users |
| | The intended uses and intended users of the ontology should be outlined. The application use cases determining the existing or prospective applications that might use the ontology presently or in the future. |
| 4 | Relevant Ontologies and Components for Reuse |
| | Interfaces and references to other existing relevant ontologies and components (prior, ancestor, descendant, neighbor, competing, and conflicting ones etc.) are determined, investigated, and outlined. |
| 5 | Domain Lexicon |
| | The collection of terminology mostly used in the domain of interest. |
| 6 | Ontology Requirements |
| | A Non-Functional Requirements |
| | The general requirements or aspects that the ontology should fulfill, including optionally priorities for each requirement. Non-functional requirements are not related to the ontology content. |
| | B Functional Requirements |
| | Functional requirements are content specific requirements which refer to the particular knowledge to be represented in the ontology. |
| | A Groups of Competency Questions |
| | The content specific requirements that the ontology should fulfill in the form of groups of competency questions and their answers, including optional priorities for each group and for each competency question. |
| 7 | Domain Glossary |
| | The information definitions of the terms mostly used in the domain of interest that are taken from competency questions and their answers, and domain lexicon. |
| 8 | Prioritizing Requirements and Use Cases Modeling |
| | Assign priorities to competency questions and developing use cases for modeling requirements in competency questions. |
requirements (Suarez-Figueroa & Gomez-Perez, 2009). Non-functional requirements are general requirements that the ontology should fulfill and are not related to the contents of ontology. Functional requirements are content-specific requirements which refer to the particular knowledge to be represented in ontology. Standard software engineering requirements elicitation methods (e.g. interviews, case studies, and simulation) can be employed in this activity to gather ontological requirements from DEs, and end-users. The DEs can be asked to write storyboards (narratives for modeling contexts and situations) by sketching sequence of activities that take place in different scenarios in the domain of interest. Technically, a storyboard corresponds to a particular scenario that is realizing one or more business purposes. In the context of our WPO, a storyboard would sound as follows:

For downloading a research paper file, a researcher (user) sends a web page request to a web server using a URL. The web server processes the request and sends back the web page to the user browser's tab. The user analyzes the web page text, enter required information
(e.g., name and credit card number etc.) in the form elements (i.e., text boxes, and radio buttons etc.), and click the download button in the form or hyperlink from the menu in the header. The file is downloaded in the user’s computer default download folder and information about date and time, and location of download is recorded in the browser’s download log file.

The ontological requirements gathered in this activity would enable ODT to understand and specify the following aspects of ontology.

- **Domain of Interest, Purpose, Scope, and Level of Formality of Ontology:** Understanding and confining domain of interest is essential for concentrating focus of ODT to a specific fragment of reality to be modeled (De Nicola et al., 2009). A network of ontologies could be formed if the domain is large. In the context of this research paper, the domain used to validate POEM methodology is web page that is providing consensual semantic model of contents and actions for better understanding of the domain of interest by the semantic services (i.e., intelligent agents, and SW services) to serve its users. Purpose of ontology corresponds to the business needs and business values specified in the OSRD. Scope of ontology is laid out by identifying the most important concepts and their characteristics that will push the refinement to the suitable granularity. Scope determines coverage and granularity of ontology, and ontological commitments by bringing certain parts of domain into focus while neglecting the other parts. The degree of formality of ontology determines the implementation language to be used for codifying ontology.

- **Intended Users and Uses of Ontology:** Specifying reasons to have ontology and its potential uses and users corresponds to business process (De Nicola et al., 2009). DEs and ODT together carry out this activity and would use a set of ontological needs to identify potential users of ontology (Suarez-Figueroa & Gomez-Perez, 2009). The potential users of WPO could be both humans (e.g., web developers) and automatic systems (e.g., intelligent agents). Likewise, DEs and ODT would use a set of ontological needs to identify scenarios in which ontology would be used. Scenarios correspond to applications that will use ontology and describes the requirements that ontology should contain after being formally implemented (Suarez-Figueroa & Gomez-Perez, 2009). The scenarios can be expressed in natural language or modeled in UML use cases. The potential uses of WPO could be used in applications for web page modeling, user modeling, user adaptive interaction, and web page ranking and recommendations.

### 4.3.2. Validating ORSs

Once ontological requirements have been specified, they would have to validate for finding any conflict between requirements, missing requirements, and contradiction between requirements. POEM uses frameworks proposed by NeOn Methodology (Suarez-Figueroa & Gomez-Perez, 2009) for validating ontological requirements.

### 4.3.3. Creating domain lexicon

Domain lexicon (DL) is built by collecting the terminology used in the domain of interest (De Nicola et al., 2009). The specified and validated ontological requirements would serve as the main source for DL construction. However, additional domain-related resources (e.g., research papers, technical reports, glossaries, thesauri, standards, web resources, and relevant lexicons and ontologies) could also be contacted for gathering relevant terms into DL. The process of extracting and collecting of terms can be accomplished both manually and using automatic tools such as OntoLearn (Navigli & Velardi, 2004). In the WPO example, the DL is formed by extracting and collecting terms mainly from the relevant domain documents (i.e., research papers, books, technical reports, and web resources) and from DEs (i.e., web developers). Analysis of the information sources resulted in an extended set of DL that demanded for proper filtering. After consistently analyzing and filtering the collected terms using generic lexical database (e.g., WordNet), the DL is reduced to 293 most meaningful and specific terms. Figure 5 shows share of the different information sources in the construction of DL of WPO. An excerpt of WPO DL is reported in Table 1.
4.3.4. Create competency questions

Competency Questions (CQs) are informal (i.e. written in natural language) and conceptual-level (i.e. different from database query aiming at retrieving concepts rather than instances) questions defining ontology requirements that the ontology is needed to answer (Grüninger & Fox, 1995). CQs do not contribute to ontological commitments rather they are used to evaluate ontological commitments made by the ontology. Using the requirements gathered from DEs during interviews, end-users brainstorming, and terms gathered in DL, CQs can be outlined that could be used to evaluate the ontological commitments (i.e. coverage, consistency, and depth) during ontology testing process (De Nicola et al., 2009). To increase understandability, the knowledge workers can list the CQs into several groups either manually or automatically using tools such as MindMap and Cicero. CQs should be grouped in such a way that each category should contain questions specific to a particular feature of ontology. Once CQs are grouped, the next task is to validate the CQs to find any conflict between CQs, missing CQs, and continuation in CQs. This task is performed by end-users and DEs by taking a set of grouped CQs and answers at a time, and decides the validity of CQs using the gathered ontological requirements. Ontology Competency Questions Document (OCQD) should be produced containing the identified CQs and their respective answers for helping in further validations. In the WPO, total 53 CQs are identified and an excerpt of the CQs is presented in Table 2.

4.3.5. Creating domain glossary

Domain glossary (DG) is built by assigning informal definitions (i.e. natural language sentences) to the terms in a domain of interest for conceptualization purposes (De Nicola et al., 2009). The ODT carry out this task by taking DL, and CQs and their answers as input for identifying a list of the mostly

---

**Table 1. An excerpt of the WPO domain lexicon**

| WPO domain lexicon | Web page | Title | Header | Footer | Hyperlink |
|--------------------|----------|-------|--------|--------|-----------|
|                     | Form     | Button | Text box | Text | Breadcrumb |
|                     | Border   | Chat room | Comment | Web feed | Side bar |
|                     | Label    | Accordion | Tab | Audio | Video |
|                     | Frame    | Menu | Bookmark | Text | Metadata |
|                     | URL text | Header image | Header text | Hyperlink URL | Frame scrolling |
|                     | Label text | Element name | Footer menu | Footer text | Frame type |
used terms and assigning one or more definitions to each term (Suarez-Figueroa & Gomez-Perez, 2009). DL and CQs should be used for extracting terminology (i.e. nouns, verbs, and adjectives) that could be used for formally representing concepts, attributes, and relations in ontology. Terminology extracted from answers of the CQs would be representing objects in the universe of discourse that could represent instances in ontology. Definitions to terminology should be assigned using domain-specific lexical databases and general lexical databases (e.g. WordNet) and common understandings of DEs and users. The WPO contains 90 entries in its DG at the time of writing this paper and an excerpt of the DG is presented in Table 3.

4.3.6. Prioritize requirements
The goal of this activity is to assign different priority levels to the different requirements identified in the different CQs groups (Suarez-Figueroa & Gomez-Perez, 2009). Users, DEs, and ODT carry out this task by taking group of CQs as input and assigning priorities to each group of CQs and to each CQ within a group. Prioritizing CQs will help ODT in identifying that which of the requirements should be addressed during the early iterations and which ones can be postponed. This specification would further guide the planning and development activities. If not prioritized, the team will model all of the requirements at the same time. UML use cases can be used to model CQs specifying the expected use of the ontology. The application of UML in OE is already shown in (Guizzardi, Herre, & Wagner, 2002) and found useful. In the ontologies, use cases would correspond to knowledge paths in ontology that would be followed for achieving business operations and addressing CQs. Use cases corresponding to particular CQs would be prioritized accordingly. Figure 6 shows a use case model and its corresponding CQ from the WPO.

4.4. Ontology project planning and scheduling
A successful ontology building project would deliver ontology according to the requirements of organization within the specified time and cost constraints. To properly manage OE project, ontology project planning and scheduling (OPPS) are related activities to be carried before formally starting ontology development. ORSD and other potential knowledge resources (i.e. historical data used for project planning and scheduling) serve as input to the OPPS phase. OPPS is the process of scoping, planning, staffing, organizing, directing, and controlling the development of an acceptable ontology at a minimum cost and within a specified time frame. A top-level view of OPPS process is shown in Figure 7 in algorithmic format. Application developers and ODT in collaboration with DEs and end-users carry out the task of defining an effective OPPS framework for OE project. The output of the phase is Ontology Project Plan Document (OPPD) describing an overall ontology building plan and a

| No. | WPO competency question                        |
|-----|-----------------------------------------------|
| CQ1 | What is web page about?                       |
| CQ2 | What is the title of web page?                |
| CQ3 | What is the URL of web page?                  |
| CQ4 | What is visit time and date of a web page?    |
| CQ5 | How much time a user has spent on a web page? |
| CQ6 | What type of multimedia a web page has?       |
| CQ7 | From which page picture X has been downloaded?|
| CQ8 | Is web page is bookmarked?                    |
| CQ9 | What type of menu is used in web page?        |
| CQ10| What is the target URL of a hyperlink?         |
| CQ11| Which activity would have to be performed before a user bookmarking web page? |
| CQ12| Is web page has breadcrumb?                   |
| CQ13| What are the tags associated with web page?    |
schedule strategy for monitoring and controlling the development and integration processes. The template of OPPD is shown in Figure 8. The OPPS phase can be mainly divided into two activities: ontology project planning and ontology project scheduling. Figure 9 shows the sequence of execution of the activities.

4.4.1. Determine ontology project planning

Objective of the ontology project planning (OPP) is to provide a framework for ODT to make reasonable estimates of resources, costs, and schedule. These estimations are identified within limited time frame at the beginning of ontology building project and should be updated regularly as the process progresses. Firstly, OPP proceeds by understanding ontology scope which would be already defined and documented in the feasibility analysis and ORSs phases. Secondly, the OPP estimates the resources required to accomplish ontology development efforts. Figure 10 illustrates the development resources as a pyramid. Each resource is described by providing description of the resource, statement of availability, time when the resource will be required, and duration of time that the resource will be used. Human is the primary resource and the number and skill of human resources required depends on the scope of the ontology building project. Integration and reusing of

### Table 3. An excerpt of WPO domain glossary

| Term        | Description                                                                                                                                 |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Web page    | A web document written in HTML that is connected to the world wide web and viewable by anyone connected to the internet who has a web browser |
| Title       | A heading that names a web page and may give a brief summary of the contents of the web page                                               |
| Header      | Header is the top portion of a web page that contains the company name and logos. Header may also contain a menu having hyperlinks for helping users to navigate other web pages |
| Hyperlink   | A highlighted text or picture in a web page that user can click on with a mouse to go to another place in the same or a different web page         |
| Bookmark    | A bookmark is the record of a web page containing uniform resource locator (URL) of the web page that is stored for later retrieval               |
| Text box    | Text box is a rectangle shaped graphical control element intended to enable a user to input text information to be used by the program. A text box can be either single-line text box or multi-line text box |
| URL         | URL stands for uniform resource locator and is the address of a web page. Each page has its own unique web address (URL). URL is used by the internet to locate the web page |
| Clicking    | The process of placing mouse cursor on an object in a web page and pressing the mouse button                                              |
| Browsing    | The process of exploring the world wide web by following one interesting link to another, usually with a definite objective but without a planned search strategy. Browsing is the opposite of surfing |
| Web feed    | A web feed (or news feed) is a data format used for providing users with frequently updated content. Content distributors syndicate a web feed, thereby allowing users to subscribe to it. A web feed is also sometimes referred to as a syndicated feed |

Figure 6. A use case model along with the corresponding CQ.
ontological and non-ontological resources in ontology development can significantly reduce time and effort costs. However, OPP must clearly reflect the level of accuracy and confidence of the reusable resources, the amount of modification required, and the nature of associated risks. Environmental resources refer to the hardware and software tools that would be required for the development and implementation of ontology. Thirdly, the OPP estimates the cost required for ontology building project. Cost estimation includes both money and efforts estimates. Ontology cost and efforts can never be exact and many variables (i.e. human, technical, environmental, and political) can affect the ultimate cost and efforts estimations of ontology building project. However, OPP can use empirical estimation models and automatic estimation tools (CASE tools) for effective estimations of cost and efforts (Pressman, 2010).

4.4.2. Determining ontology project schedule
Ontology project scheduling (OPS) is similar to software engineering project scheduling and concerned with determining the timelines to be met by ontology development processes. OPS divides ontology building process into separate tasks and determines that how these tasks would be executed. OPS estimates the calendar time required, the efforts required, and the resources required for completing each task (Sommerville, 2006). However, the tasks should not be either too small or too large. OPS schedule tasks to be executed either in sequence or in parallel in such away to optimally

---

**Figure 7. Top-level view of OPPS process.**

**Figure 8. Template of ontology project plan document OPPD.**

| **Ontology Project Plan Document** |
|-----------------------------------|
| **1 Introduction**                |
| Overview of the ontology project is noted here by specifying scope of the project, functional decomposition of the ontology, performance or behavioral issues, and management and technical constraints. |
| **2 Project Estimates**           |
| The ontology project cost, effort, and time (duration) estimates are noted here. People, hardware, software, tools, and other resources required to build the ontology is also presented here. |
| **3 Processes and Activities**    |
| Overview of selecting processes and activities, mapping of processes and activities into the development model phases, setting of order of processes and activities, and establishing of resources assignments to process and activities is presented here. |
| **4 Risk Management**             |
| The ontology project risk and the approaches to manage them are presented here. Name of risk, probability, impact, and RM3 (risk mitigation, monitoring, management) is noted here. |
| **5 Project Schedule**            |
| A complete ontology project schedule comprising of framework activities and task set, functional decomposition, tasks and their dependencies, and timeline chart is presented here. |
| **6 Staff Organization**          |
| Detailed description of staff which is organized, team structure, and mechanisms for progress reporting and inter/intra team communication is outlined here. |
| **7 Tracking and Control Mechanisms** |
| Techniques to be used for ontology project tracking and control are identified here. Overview of quality assurance and control, and change management and control is presented here. |
| **8 Appendix**                    |
| Supplementary information is provided here. |
use the workforce and resources. Figure 11 illustrates the top-level view of OPS process. OPS provide mechanism for ODT to track and control a project progress by scheduling dates to which milestones and deliverable should be produced. CASE tools are developed that will take ontology building project information and produce OPS automatically in graphical notations in the shape of Gantt charts and activity networks (Sommerville, 2006). Bar chart (also called Gantt charts) shows that who is responsible for a task, the expected elapsed time for a task, and when a task is scheduled to begin and end. Activity network is a network diagram showing the dependencies between the different tasks making up a project. The initial OPS is created at the project startup phase and is refined and modified as the project progresses.

4.5. Risk analysis and management

Risk analysis and management (RAM) is a sequence of steps that helps ODT to identify and manage risks by drawing plans to minimize their effect on ontology building project. From software engineering perspectives, proper risk analysis can ensure product quality that is reasonably free of defects, delivered on time and within budget, meet requirements and expectations, and is maintainable. POEM uses proactive risk management approach where potential risks, their probability, and impacts are identified, assessed, and classified according to their importance long before any of the technical work is initiated. ODT team should try to avoid risks by actively accessing, monitoring, and managing potential risks until they are either handled or a contingency plan is prepared to handle unavoidable risks in a controlled and effective manner. POEM proposes an effective risk analysis and management framework (shown in Figure 12), which is inspired from MSF (Microsoft Solution Framework) risk management model and is a six-step logical model advocating managing current risk, planning and executing risk management strategies, and capturing knowledge for the enterprise. POEM has advantage over the other methodologies in risk management because of evaluating
risks much earlier and providing opportunity to decide that which risk is to be handled with each cycle. POEM enables to deal with major obstacles and select alternatives in the earlier stages of the project, which is less expensive. ODT, DEs, and users work closely together to carry out risk analysis and management phase. Output of the phase will be a risk table that would be produced as an integral part of the OPPD as shown in Figure 9. Broadly, the activities to be carried out during this phase are risk identification, risk analysis, risk planning, and risk monitoring. Figure 13 illustrates the flow of execution of the activities.

4.5.1. Risk identification
Risk identification is a systematic process to specify threats to the OPP (i.e. estimates, schedule, and resource loading) (Pressman, 2010). Generic risks are easy to identify due to being potential threat to every ontology building project. Ontology-specific risks are hard to identify and can only be identified by those having clear understandings of the technology, the people, and the environment specific to the project. One way of identifying ontology-specific risks is to create a risk item checklist comprising of questions related to the categories: (1) ontology project size—overall size of the ontology to be built or modified, (2) business impact—due to constraints imposed by management or the marketplace, (3) user/DE characteristics—sophistication of the user/DE and ability of ODT to communicate with user/DE in a timely manner, (4) process definition—the degree of ontology development process definitions and being following by the ODT, (5) development environment—the availability and quality of tools to be used for building ontology, (6) technology to be built—the complexity and newness of ontology and system to be built, and (7) staff size and experience—overall technical and project experiences of ODT who do the work. Answer to these questions would help in identifying potential risks and the number of negative answers that would be directly proportional to the degree at which the project is at risk. Table 4 illustrates an excerpt of the risk identified along with corresponding categories for the WPO project.

4.5.2. Risk projection
Risk projection, also called risk estimation, attempts to determine likelihood or probability of a risk to occur and estimate the impacts and consequences associated with the risk (Pressman, 2010). Probability of a risk can be rated as very low, low, moderate, high, and very high. Impact of a risk can be rated as catastrophic, serious, tolerable, and insignificant. ODT in collaboration with technical staff workout risk projection by establishing a scale for perceiving probability of a risk, drawing
consequences of a risk, delineating risk impacts on ontology building project and the resulting ontological product, and estimating accuracy of a risk projection to avoid any ambiguity. ODT can simplify risk projection process by developing a risk table with columns: risk ID, risk description, category, probability, impact, risk exposure, and risk planning. Probability and impact can be scaled numerically where one might represent the lowest possible rate and n might represent the highest possible rate. Risk exposure is the product of these two rankings. Once the table is formed and sorted using risk exposure from high to low, the ODT can draw a cutoff line signifying the risks lying above the line are of the highest risk exposure (i.e. highest priority) and should be treated first. Risks coming below the line would be re-evaluated and treated in the second-order priority. Table 4 illustrates an excerpt of the risk table showing risk projection for the potential risk identified in the WPO project.

4.5.3. Risk planning
Risk management is concerned with developing strategies for dealing with a risk (Pressman, 2010). An effective strategy would include risk avoidance or risk management and contingency planning. For a proactive approach to risk management, risk avoidance is the best strategy by developing a risk monitoring plan to monitor factors that would provide information whether the risk is becoming more or less likely. A risk mitigation plan would be developed to either conquer or reduce the net effect of a risk. If risk mitigation plan is failed, the risk management and contingency plan is to be developed that would define strategy to deal with a risk and handle the risk smoothly and efficiently. Table 4 illustrates an excerpt of the risk planning strategies for the risks identified in the WPO project.

4.6. Ontology design and implementation
The goal of this phase is to give ontological structure to the ontological requirements gathered in the ORSD under the constraints and guidelines of OPPD. In essence, this phase takes the development process from linguistic dimension into conceptual dimension. Ontology design and ontology implementation are closely related activities and the issues of one could severely affects the working of another such as implementation issues guides the decisions to be taken at the design development time. For simplicity to explain and understand, we divided this phase into two sub-phases: ontology design and ontology implementation.

4.6.1. Ontology design
Ontology design (OD) transforms the conceptual model of ontological requirements into a design model that would serve as a blueprint for ontology construction. The conceptualization phase figure
out that what are the business needs and the design phase decides that how to build it. This is accomplished by refining the terms (representing entities, relationships, and processes) captured in the conceptualization phase, organizing the terms into conceptual hierarchies, and structuring them with attributes and axioms (De Nicola et al., 2009). Apart from devising structures and hierarchies (that could be possibly sub-ontologies) and interfaces as well as rules for commitment, mappings and translations of these conceptual constructs into “dialects” (i.e. Unified Modeling Language (UML)) are also provided in the ontology designing phase. UML has been used in several ontology building methodologies for ontology modeling (D’Antonio, Missikoff, & Taglino, 2007; De Nicola et al., 2009; Missikoff & Taglino, 2002). Researchers have demonstrated that how UML could be used to model lightweight ontologies and how UML models could be enriched by adding Object Constraint Language (OCL) expression to give explicit descriptions of specifications (Gómez-Pérez et al., 2007). UML class diagrams would represent concepts (and their attributes), relations between concepts (both taxonomic relationships and ad hoc ones), and axioms in OCL (e.g. cardinalities on object and datatype properties, and disjointness). Furthermore, UML object diagrams could be used to represent instances. Ontology development tools (e.g. Protege-2000 and WebODE) support exporting and importing of ontologies in UML format. ODT plays the key role in this phase by bringing concepts into taxonomic hierarchy but DEs can provide assistance for the required application and domain

Table 4. An excerpt of the risks identified, projected, and planned in the WPO project

| Risk ID | Risk description | Category | Probability | Impact | Risk exposure | Risk management planning |
|---------|------------------|-----------|-------------|--------|--------------|--------------------------|
| 1       | Ontology project scope and size is not stable | PS        | 4           | 3      | 12           | Arrange efficient meetings with DEs and consult relevant domain literature |
| 2       | Possibility of recruiting staff with the skill for the project | ST        | 4           | 4      | 16           | Arrange training sessions and developments courses for the available staff for improving their skills |
| 3       | Not sure about the potential applicability of the ontology | TC        | 3           | 4      | 12           | Develop prototype to demonstrate feasibility |
| 4       | Reusable ontological and non-ontological components contain defects | PD        | 3           | 3      | 9            | Replace defective components by either developing new components or make moderate changes to the available components or bought in new components of known reliability |
| 5       | Financial problems may force reduction in the ontology development project | BU        | 2           | 4      | 12           | Prepare a brief document for the volunteer organizations showing how the project is making a very important contribution to the goals of the business domain |
| 6       | DE may propose changes in requirements that would require major design rework | PS        | 3           | 3      | 9            | Derive traceability information to access requirements change impact and maximize information hiding in the design |
| 7       | Key and skilled staff may get ill at critical stage of the project | ST        | 3           | 3      | 12           | Reorganize staff so that there is more overlap of work and people therefore understand each others’ job |
| 8       | Lack of staff training on development tools | DE        | 4           | 3      | 12           | Arrange training sessions and developments courses for the available staff for improving their skills |

Notes: Category: PS—ontology project size, BU—business impact, CU—customer characteristic, PD—process definition, DE—development environment, TC—technology to be built, ST—staff size and skill.

Probability: 1—very low, 2—low, 3—moderate, 4—high, 5—very high.

Impact: 1—insignificant, 2—tolerable, 3—serious, 4—catastrophic.
knowledge. The tasks in this phase are classified into concepts classification and modeling, modeling classes hierarchies, mapping reusing and alignment, modeling axioms and relationships, and ontology merging integration and alignment. Figure 14 shows the sequence of execution of the tasks. Output of the phase will be ontology semantic network in the UML format.

- **Concepts Classification and Modeling:** This activity starts by identifying concepts from glossary and UML use cases diagrams in the ORSD and works by classifying the concepts into different categories depending on their nature. Concepts can be classified using sameness that they have a common structure or exhibit a common structure. OPAL (Object, Process, and Actor modeling Language) (D’Antonio et al., 2007) is a UML- and an OWL-based ontology representation methodology that has provided common guidelines for classifying concepts into different categories that are deeply inspired from the primary modeling constructs of UML. These categories are business actor, business object, business process, messages, and attributes. A detailed discussion about these categories can be found at (D’Antonio et al., 2007). Before classifying concepts into categories, concepts are needed to be divided into four basis constructs (i.e. classes, object properties, datatype properties, and objects) that are necessary to complete a rich ontological representation of an observed reality.

Classes (also called concepts) in ontology are similar to UML classes that represent collections of similar entities (objects) serving the roles of business agents, business object, or business processes in a domain of discourse. Classes in a domain can be people, places, roles, things, organizational units, and occurrences or events. Classes in a domain are identified from the terms belonging to the noun or noun phrase classes of parts of speech. Classes in a domain will be related with each other due to either sharing or having some kind of semantic connection. Table 5 illustrates an excerpt of potential classes in WPO. Objects (also called instances) are occurrence of classes that are tangible and can be comprehended intellectually. In OE, classes are the logical modeling of a domain and objects are physical representation of classes. An ontology constituting classes together with a set of object forms a knowledgebase (Noy & McGuiness, 2001).

Properties represent characteristics of classes and will be used for relating classes with other classes or literals in the format of domain–range relationships pattern. Properties in a domain are identified from the terms belonging to the verb or verb phrase classes of parts of speech. In OE, properties can be either object properties or datatype properties. Object properties are similar to UML messages in a domain of discourse that represent information exchange during interactions between classes and representing relationships between instances of classes. Datatype properties are similar to UML attributes, characterizing the information structure of classes and representing
relationships between instances of classes with instances of literals. A datatype property can be either atomic for modeling elementary information (e.g. URLText) or complex for modeling structured information (e.g. UserName is composed of first name, middle name, and last name). Table 5 illustrates an excerpt of object properties and datatype properties of WPO.

- **Modeling Classes Hierarchies:** In this stage, classes are organized into hierarchies that are conceptual clusters of classes using the criteria of sameness in their properties. Classes are classified into categories by first creating conceptual descriptions of categories and then classifying the classes according to the descriptions. A class hierarchy shows the existence of classes and their relationships in a logical view. A class hierarchy is created by relating classes in a category hierarchically using the generalization–specification (i.e. is-a) relations. Classes in a hierarchy are arranged in superclass–subclass pattern where superclass is general and subclass is specific. There are three ways to arrange classes in a hierarchy (Gómez-Pérez et al., 2007): (1) top-down (from general to specific that is from most abstract to most concrete), (2) bottom-up (from specific to general that is from the most concrete to the most abstract), and (3) middle-out (from the most relevant to the most abstract and most concrete). Middle-out approach tends to be more common and effective because concepts of the middle level are more informative about a domain. POEM uses middle-out approach by first identifying the salient concepts and then generalizing and specializing them. POEM encourages reusing of existing ontological and non-ontological resources. Therefore, once a candidate class is identified while constructing a class hierarchy, then mapping would be performed by searching the ontology library to find similar class in any of the existing ontologies. If yes, it will be reused via importing it in the developing ontology otherwise it would have to be engineered in the developing ontology. Alignment is not physical reusing of classes (i.e. importing of classes) rather ontology is connected with required classes in the existing ontologies via developing explicit relationships. POEM carries out this activity in accordance to NeOn Methodology (Suarez-Figueroa & Gomez-Perez, 2009) which provides a comprehensive framework for reusing ontological and non-ontological resources. The resulting class hierarchy can be extended with other UML relations such as part-of relations between classes using aggregation and composition. Conclusively, this task will convert the informal ontology representation into diagrammatic representation in the form of a set of UML class diagrams. Figure 15 illustrates an excerpt of the WPO UML class hierarchy.

- **Modeling Axioms and Relations:** The level and ability of ontologies to describe axioms and relations signifies their power. Axioms are logical assertions/constraints that are imposed by modeling reality and represent the complete theory that ontology is describing in its domain of discourse. Axioms are statement in the if-then (antecedent–consequent) format that constitutes the logical inferences which can be drawn from an assertion in a specific format. A simple example of axioms is disjointness that is classes in a category belonging to the same level are mutually disjointed (i.e. the object of one class in a disjointed set cannot be the object of another class in the same set). Properties of a class represent relationships/links of a class with other classes or literals in ontology. Linking classes using relations in an ontology exhibits that how object are related and how they interact with each other in the domain of discourse. Property restrictions (i.e. quantified, and cardinality) delegates business rules that defines rules needs to be followed while defining objects of classes in ontology (e.g. a person can be a student if he has exactly one occurrence of age property which value is greater than or equal to 5 years). In this stage, classes in a domain that are identified in concepts classification and modeling stage are needed to be enriched semantically by imposing different axioms and property restrictions from the domain of discourse. Figure 16 depicts an excerpt of relationships between the classes in WPO.

- **Ontology Merging Integration and Alignment:** POEM is iterative methodology and each iteration would produce a new incremental version of the ontology. The ontological model produced in a recent iteration needs to be merged and integrated with the ontological model produced in the previous iterations to obtain a single unified ontology that would have/include all the knowledge obtained from the domain so far. POEM supports the hypothesis of METHONTOLOGY that
integration should be carried out at knowledge level (i.e. in conceptualization) not at the symbol level (i.e. in formalization, when selecting the representation language) or implementation level (i.e. when the ontology is codified in the target language). At this stage, the ontology merging integration and alignment is performed by either simply defining relationships between the classes in new and previous versions of ontology or restructuring the class hierarchies at certain level and then defining relationships between the classes. Figure 17 illustrates the conceptual-level understanding of ontology merging integration and alignment in WPO.

4.6.2. Ontology implementation
At the ontology implementation (OI) phase, the ontology has to be codified in a formal ontology development language. This stage is concerned with transforming the conceptual ontological structures created in the ontology designing phase into a formal and explicit format to be understandable by both machines and humans and directly usable by the applications. ODT plays the key role in this phase by bringing the conceptual models and designs into physical reality with minimal assistance of DEs for the required application and domain requirements. The activities in this phase are classified into selection of representation language and tools, and codifying ontology. Figure 18 illustrate the sequence of execution of the tasks in this phase. Output of the activity is actual ontology code in a formal ontology representation language.

- **Selection of Representation Language and Tools:** In this task, appropriate formal representation language and tools are to be selected for the ontology development. However, certain factors related to ontology representation languages (i.e. expressiveness power, computational complexity of the associated reasoning, and level of acceptance within the community) as well as business requirements are needed to be considered while in selection of formal language for ontology. A number of formal languages are available for ontology developments varying in expressiveness power, computability, and decidability. Similarly, a number of automatic tools are also available that are providing features for the automatic and quick development and visualization of ontologies. Detailed surveys of ontology modeling languages, and OE tools and APIs are presented in (Alam et al., 2014; Corcho et al., 2003). Similarly, a number of serialization formats are also available for ontology developments and a detailed discuss about serialization formats can be found at (Alam, Khan, Ali, & Khusro, 2014). However, Web Ontology Language (OWL) is the W3C standard and is the most commonly used and accepted ontology.
development language in the OE community. Similarly, Protégé is the most commonly used ontology editing tool.

- **Codifying Ontology**: In this task, ontology is codified using the selected formal ontology language and tools in accordance to the requirements gathered and designs produced. Results from sub-development cycles (regarding sub-ontologies, and previous versions) are integrated and a new version of ontology is created. The WPO is developed in OWL-DL dialect of the OWL in RDF/XML serialization format using Protégé 4.3 ontology editor. Figure 19 shows an excerpt of the WPO code in OWL.

### 4.7. Ontology quality testing and release

The architecture of POME results in a series of layered sub-ontologies (i.e. incremental versions) that encapsulates collaborating classes. Each of the ontological elements (i.e. sub-ontologies and classes) represents information that helps in achieving ontology requirement and business requirements indirectly. The objective of ontology quality testing is to uncover the greatest possible number of errors (i.e. arise due to classes collaborations within a sub-ontology) with a manageable amount of efforts applied over a realistic time span. Ontology quality testing is essential before releasing
ontology to customer for reducing the chances of experiencing frustrations associated with poor quality ontology. A set of quality matrices for ontology quality testing has been proposed in (Burton-Jones, Storey, Sugumaran, & Ahluwalia, 2005). The activities in this phase are classified into syntactic quality testing, semantic quality testing, pragmatic quality testing, and ontology release. Figure 20 illustrate the sequence of execution of the activities in this phase. Output of the phase is a tested incremental version of ontology and ontology release notes.

4.7.1. Syntactic quality testing
Syntactic quality testing employs measures to estimate the quality of ontology according to how it is written (Burton-Jones et al., 2005). Syntactic quality is measured in lawfulness and richness. Lawfulness is the degree to which ontology language syntactic rules have been followed. Richness is the degree to which features in ontology language are exploited for ontology development, the more the better. A set of quality matrices has been proposed and reused for syntactic quality testing from both software engineering and OE communities (Duque-Ramos et al., 2014). Matrices from software engineering community include response for a class, number of ancestor class, and number of children per class. Matrices from OE include lack of cohesion in method, property richness, attribute richness, and inheritance relationship richness. OntoQA (Tartir & Arpinar, 2007) is a

| Classes          | Header | Footer | Menu | Logo |
|------------------|--------|--------|------|------|
| Web page         |        |        |      |      |
| Sidebar          | Breadcrumb | Bookmark | Image | Audio |
| Form             | Button | Textbox | Textarea | Frame |
| Chatroom         | Accordion | Comment | Feed | Border |
| URL              | Text | Map | Tab | Popup |
| Hyperlink        | Video | Submit | Column | Dynamic |

| Object properties | hasBreadcrumb | hasMenu | hasTab | hasURL | Accordion | text
|-------------------|--------------|---------|--------|--------|-----------|------
| Footerimage       | Headerimage  | HyperlinkURL | Hyperlinktarget | Tabmap |
| Logoimage         | URLtext      | Popupform | Sidebarmenu | hasForm |
| Popupform         | hasTextarea  | hasMap   | hasComment | Commenttext |
| Feedfrom          | hasChartroom | Tabform  | Viewimage | Headertext |

| Datatype properties | hasTitle | hasText | hasVisittime | isSaved | Textcolor |
|---------------------|----------|--------|-------------|--------|-----------|
| hasName             | hasValue | Menutype | isChecked   | Titletext |
| ButtonType          | isPrinted | hasColumns | Elementname | Formname |
| FormAction          | Downloadtime | Downloadpath | Bookmarkname | Framename |

Table 5. An excerpt of classes, object properties, and datatype properties of WPO

Figure 18. Sequence of execution of activities in OI tasks.
4.7.2. Semantic quality testing

Semantic quality testing evaluates the meaning of terms in the ontology in terms of interoperability, consistency, and clarity (Burton-Jones et al., 2005). Interoperability measure estimates the meaning of terms (i.e. classes and properties) in ontology and is achieved by checking the terms in the ontology in another independent semantic source (i.e. generic lexical database such as WordNet) using the formula

$$EI = \frac{W}{C}$$

where $C$ is the total number of terms used in the ontology and $W$ is the number of terms having senses in WordNet. Clarity measures the context clarity of the terms in ontology and is estimated using the formula

$$CL = \frac{\sum 1/Ai}{W}$$

where $Ai$ shows the number of meanings of every interoperable term in WordNet. Consistency measures whether terms have consistent meaning in ontology by finding any contradiction (i.e. declaring of an object of two disjointed classes) and correct use of molding constructs. To calculate consistency, the ontology has to pass through two major tests: sub-sumption test to check whether a class is a sub-class of another class or not, and the logical consistency check to see whether a class can have any instance or not. Consistency checking of the ontology can be achieved using a reasoner (e.g. Fact++, HerMiT, Pellet, and RacerPro). A reasoner works by taking a manually created class hierarchy (called asserted ontology) to automatically compute an inferred class hierarchy (called inferred ontology) using the descriptions of classes and relationships. Figure 21 depicts a snippet of the WPO asserted and inferred class hierarchies after reasoning using Fact++ in Protégé 4.3.

4.7.3. Pragmatic quality testing

Pragmatic quality testing evaluates the usefulness of ontology for its users irrespective of syntax and semantics (Burton-Jones et al., 2005). Pragmatic quality of ontology is measured in its accuracy, comprehensiveness, and relevancy. Accuracy checking determines that the asserted knowledge in ontology agrees with the experts’ knowledge in the domain. Ontology with correct definitions and
descriptions of classes, properties, and individuals will result into high accuracy. Recall and precision rate are the two primary measures used for evaluating the accuracy of ontology (Maynard, Peters, & Li, 2006). The accuracy of ontology can be measured by defining use cases. Use cases are concerned with CQs that are used for creating SPARQL queries to query ontology for retrieving relevant information. For example, a user might be interested in finding the URL of a particular web page which is the CQ3 of the WPO. A SPARQL query can be created on the basis of this use case and issued to WPO to answer with its content as shown in Figure 22. Relevance refers to coverage verifies that whether ontology satisfies a user’s specific requirements. Relevancy checking, however, requires prior knowledge about a user’s needs. Relevance can be checked in two ways: (1) semantically annotation and mapping of UML diagrams with the ontology contents by the DEs, and (2) using CQs to retrieve specific answers from the ontology contents. The second method is similar to querying ontology using SPARQL queries. Comprehensiveness refers to completeness and is a measure of the size of the ontology that whether the ontology satisfies the requirements and constraints of the problem it was meant to solve. Larger ontologies are more likely complete representation of their domains and provide more knowledge to their users. Comprehensiveness of the ontology can be evaluated using the domain goals, coverage of the ontology, and the degree of response to CQs.

4.7.4. Ontology release
Once ontology is thoroughly tested and approved against the specification provided in the requirements, it could be released for applications’ uses. Like software engineering, ontology release notes documents should be produced at this stage that is shared with users, customers, and organizations. There is no standard format for release notes to be followed throughout the industry. However, ontology release notes details the general information and key issues about the current ontological version. The general information includes purpose, version number, type of release (e.g. final or not final release), release date, several statistics (e.g. number of classes, number of object properties,
and number of datatype properties) about the total and newly added content, refactoring, license type, tests results and tests procedures, and potential applications about ontology. Key issues include the problems and errors identified in the testing process, bugs fixed, and novel methods of operations (e.g. how to perform a search process) about ontology. Ontology release notes are to be written by the ODT.

4.8. Ontology evaluation and feedback

After cold tests, the incremental version of ontology is provided for users’ experiences using in different applications and problem solving methods. Although, ontology would have been tested in the previous phase, there could be still the possibilities of bugs, usability issues, and room for improvements which could only be identified if ontology is used in real-world scenarios. The activities to be carried out at this phase can be broadly classified into ontology evaluation and customer feedback. Figure 23 show the sequence of execution of activities at this phase.

4.8.1. Ontology evaluation

Ontology evaluation is the process of using ontology by the users with the intention of uncovering hidden errors and uncertainties. Ontology evaluation information will be helpful for ODT in more understandings about coverage, quality, reliability, and performance of ontology. Ontology evaluation will also provide information about ontology usage analysis that would help ontology developers in determining which parts of ontology could be used the most and which are not at all, and which parts of ontology are needed to be updated, deleted, or improved. A detailed review on the importance of ontology usage analysis is presented in (Ashraf et al., 2015). Ontology evaluation can be either open evaluation or closed evaluation. Open evaluation is uncontrolled where users are allowed to use ontology for all possible use cases and application scenarios. Close evaluation is controlled evaluation where some specific use cases would be outlined for evaluation of ontology by the ODT, DEs, and end-users. POEM uses close approach due to producing ontological products in
4.8.2. User feedback
User feedback is the process of collecting information about users’ experiences of using ontology after the evaluation process. User feedback can be either evaluation-specific or unfolding aspects of ontology which has not yet been discovered. Different techniques have been proposed for taking and valuing user feedbacks in the field of software engineering. Criteria-based assessment is a quantitative assessment technique that is derived from ISO/IEC 9126 Software Engineering—Product Quality standard providing information about usability, sustainability and maintainability areas of a product. POEM encourages the using of the same technique for ontology evaluation and taking users’ feedbacks. All that is required in the methods is the creation of questionnaire consisting of questions from each area of ontology and rating of the answer options. The summary score of each criterion will indicate clear understanding of the strength or weakness of ontology in an area.

4.9. Plan the next iteration
Testing and evaluation of incremental versions of ontology will not only determine the degree of completeness of ontology but will also highlight a number of issues that will require immediate attention from the ODT. Planning next iteration needs to detail what is to be done in the next iteration in a fine-grained way. This would be a fine-grained plan for a next single iteration and would be developed under the ontology project plan. However, it could affect the ontology project plan.
ontology project plan plans out the entire project into detailed iterations in advance but the project unfolds as iterations progresses and plans changes using the feedbacks obtained after delivery of the incremental versions of ontology. For next iteration planning, POEM uses the idea of iteration or sprint plan from agile software engineering methodology that emphasizes on changing and articulating new plan as per situations. Like agile process, POEM makes honest plans based on the users’ feedbacks. However, plans should not be of short duration because that could affect the ontology project plan. Plan for the next iteration will determine the current status of the ontology project, results of the current iteration, list of scenarios and use cases that must be completed by end of the next iteration, a list of risk that must be addressed by end of the next iteration, and list of changes (i.e. bug fixes, and changes in requirements) that must be incorporated in the current version of the ontology. Output of the phase is Next Iteration Plan Document (NIPD) comprising of different sections about needs of the next iteration, tasks to be achieved in the next iteration, and objective achieved in the current iteration. Template of NIPD is shown Figure 24.

5. POEM assessment
The focus is on assessing the comprehensiveness and effectiveness of POEM. We have divided the quantitative assessments of POEM into two sessions: (1) experiment for comparing the features offered by POEM with other OE methodologies using a criterion extending IEEE standard, and (2) experiment for evaluating effectiveness of POEM by how POEM improves the quality of the result (i.e. ontologies). The sessions are separated in time and each session involves a different criteria, objective, and participants. Below we describe the experiments settings, the participants, and the criterions we have applied in order to assess POEM.

5.1. First session experiment settings and results
POEM exploits the large experiences drawn from the widely used standards in the software engineering for building SW ontologies. In order to compare POEM with different ontology building methodologies generally and web ontology building methodologies specifically, a comparison framework has been established that is majorly using the framework presented by Fernández-López and Gómez-Pérez (Gómez-Pérez et al., 2007) based on the IEEE 1074–1955 standard for software development life cycle. Two different criterions are used for comparing POEM with other sister proposals and are shown in Tables 7 and 8 respectively. The criterion included in Table 7 is of subjective features whose values are determined by contacting and analyzing the mother or relevant literature sources. The criterion included in Table 8 is empirical and an empirical study is conducted for the determination of values of the features. The subjects of the study are twenty (20) participants who are voluntarily selected from the junior researchers (i.e. master students) specializing in areas of Web Semantics and Web Engineering in the Department of Computer Science, University of Peshawar. The criterion for selection of the participants is their familiarity, understanding, and having much experience of the ontologies and OE phenomenon. Anyhow, for accurate estimations, first the students are trained with the available ontology building methodologies (including POEM) for a week and then they are asked to fill a questionnaire consisting of questions regarding the criterion features of each methodology. The questionnaire is close-ended where each question is given with four potential answer options. The options were: (1) not proposed—the criterion feature is not referenced at all, (2) proposed—the criterion feature is referenced only, (3) described—the criterion feature is included but no detailed technique is proposed, and (4) fully described—the criterion feature is include and detailed technique is proposed, respectively to cover all of the possible features in a methodology. For numerical analysis the options are weighted with 0, 1, 2, and 3 ranks respectively. The value of a criterion feature is computed by taking arithmetic mean of the corresponding answers reported by the participants using the formula shown in Equation (1) where \( n \) is the sample size \( (n = 20) \) and \( A_i \) is answer of the \( i \)th participant. The overall maturity of a methodology is shown in summary score that is computed by taking sum of all of criterion feature values using the formula shown in Equation (2) where \( m \) is the total count of criterions and \( C_i \) is criterion value of the \( i \)th criterion feature.
Table 7 compares the methodologies using the construction strategy used for ontologies by the methodologies. Comparatively, POEM provides satisfactory results by following approaches that are most commonly supported by the researchers and practitioners for the ontology engineering. For example, POEM is using the prevailing evolving life cycle model, using the most commonly used middle-out approach for identifying concepts in ontology taxonomy, and supporting incremental-based development which grows a developing ontology by layers and new definitions would be included when a new version is planned, etc. Rest of the information in the table are self-explanatory.

Table 8 empirically compares the methodologies using an extended version of the framework proposed by the IEEE 1074–1995 standard. The framework is extended with certain additional features especially project management (e.g. project planning and scheduling, risk analysis, and merging and alignment) to provide an extended comparison. POEM realizes the complexity of an SW ontology building project and incorporates pre/post project management activities equally to core development activities. POEM suggest the proper ontology project initiation, feasibility analysis, project planning and scheduling, risk analysis and management, taking users’ feedback, laying plan for the next incremental version, installation, maintenance, and retirement that makes POEM advantageous over its competitors especially METHONTOLOGY and UPON. Particularly, the risk analysis and management provides a measuring tool to both ODT and DEs that earlier methodologies do not have. Another advantage is, like UPON, POEM has strong roots in UML and suggests using of UML tools (e.g. Rational Rose, and Microsoft Visio) for diagramming and documentation for ontology requirements and designs, etc. Another broad aspect of POEM is its milestone- and deliverable-oriented nature emphasizing on the production of enough documentations at each phase and activity (properly approved by the ontology quality assurance group) for helping ODT and DEs in understanding, guiding, and control the ontology building process.
| Methodologies   | Details of methodology | Recommendation of formalization | Strategy for building applications | Strategy for identifying concepts | Recommended life cycle | Reusing existing ontologies | Collaborative development | Use of core ontology | Interoperability support |
|----------------|------------------------|---------------------------------|------------------------------------|----------------------------------|------------------------|-----------------------------|---------------------------|------------------------|-------------------------|
| Uschold        | Low                    | None                            | Application independent            | Middle-out                       | Non-proposed           | Yes                         | No                        | No                     | No                      |
| TOVE           | Medium                 | Yes (Logic)                     | Application semi-independent       | Middle-out                       | Evolving prototypes    | No                          | No                        | Depends on the available resources | No                      |
| METH-ONTOLOGY  | High                   | None                            | Application independent            | Middle-out                       | Evolving Incremental   | Yes                         | No                        | No                     | No                      |
| KACTUS         | Low                    | None                            | Application dependent              | Top-down                         | Evolving prototypes    | Yes                         | No                        | No                     | No                      |
| SENSUS         | Low                    | None                            | Application semi-independent       | Top-down                         | Non-proposed           | No                          | Yes                       | Yes                    | Yes                     |
| On-to-Knowledge| Low                    | None                            | Application dependent              | Top-down Bottom-up Middle-out    | Incremental            | Yes                         | No                        | Depends on the available resources | No                      |
| UPON           | High                   | None                            | Application dependent              | Middle-out                       | Incremental            | No                          | No                        | No                     | No                      |
| Vanitha        | Low                    | None                            | Application independent            | Top-down                         | Evolving               | Yes                         | No                        | Depends on the available resources | No                      |
| Noy            | Low                    | None                            | Application independent            | Top-down                         | Non-proposed           | Yes                         | No                        | No                     | No                      |
| Kalyanpur      | Low                    | Yes (Logic)                     | Application dependent              | Top-down                         | Non-proposed           | Yes                         | No                        | No                     | No                      |
| POEM           | High                   | None                            | Application independent            | Middle-out                       | Incremental            | Yes                         | No                        | Depends on the available resources | Yes                     |
Table 8. Comparison of POEM with earlier methodologies using extended IEEE 1074–1955 framework

| Feature                                | Uschold | TOVE  | KACTUS | METH-ONTOLOGY | SENSUS | On-To-Knowledge | UPON | Vanitha | Noy | Kalyanpur | POEM |
|----------------------------------------|---------|-------|--------|---------------|--------|-----------------|------|---------|-----|-----------|------|
| Project management processes           |         |       |        |               |        |                 |      |         |     |           |      |
| Project initiation                     | 0       | 0     | 0      | 0             | 0      | 0               | 0    | 0       | 0   | 0         | 1    |
| Project monitoring and control         | 0       | 0     | 0      | 0             | 0      | 0               | 0    | 0       | 0   | 0         | 2    |
| Ontology quality management and assurance | 0 | 0     | 0      | 0             | 0      | 0               | 0    | 0       | 0   | 0         | 1    |
| Project planning and scheduling        | 0       | 0     | 0      | 0             | 0      | 0               | 0    | 0       | 0   | 0         | 2    |
| Risk analysis                          | 0       | 0     | 0      | 0             | 0      | 0               | 0    | 0       | 0   | 0         | 2    |
| Ontology development-oriented processes|         |       |        |               |        |                 |      |         |     |           |      |
| Pre-development processes              |         |       |        |               |        |                 |      |         |     |           |      |
| Feasibility study                      | 0       | 0     | 0      | 0             | 0      | 2               | 0    | 0       | 0   | 0         | 0    |
| Environment study                      | 0       | 0     | 0      | 0             | 0      | 1               | 1    | 0       | 0   | 0         | 1    |
| System (resources) allocation          | 0       | 0     | 0      | 0             | 0      | 0               | 0    | 0       | 0   | 0         | 1    |
| Development processes                  |         |       |        |               |        |                 |      |         |     |           |      |
| Requirements specification             | 1       | 3     | 1      | 3             | 1      | 3               | 3    | 0       | 0   | 1         | 3    |
| Design                                 | 0       | 3     | 1      | 3             | 0      | 1               | 3    | 1       | 1   | 1         | 3    |
| Implementation                         | 1       | 2     | 1      | 3             | 2      | 2               | 3    | 0       | 0   | 0         | 3    |
| Post-development processes             |         |       |        |               |        |                 |      |         |     |           |      |
| Installation                           | 0       | 0     | 0      | 0             | 0      | 0               | 0    | 0       | 0   | 0         | 1    |
| Operation                              | 0       | 0     | 0      | 0             | 0      | 1               | 0    | 0       | 0   | 0         | 0    |
| Support/feedback                       | 0       | 0     | 0      | 0             | 0      | 0               | 0    | 0       | 0   | 0         | 2    |
| Maintenance                            | 0       | 0     | 0      | 0             | 0      | 1               | 1    | 0       | 0   | 0         | 1    |
| Retirement                             | 0       | 0     | 0      | 0             | 0      | 0               | 0    | 0       | 0   | 0         | 0    |
| Ontology support activities            |         |       |        |               |        |                 |      |         |     |           |      |
| Knowledge acquisition                  | 1       | 1     | 0      | 3             | 0      | 2               | 3    | 0       | 0   | 0         | 3    |
| Evaluation                             | 1       | 3     | 0      | 3             | 0      | 1               | 2    | 0       | 0   | 0         | 3    |
| Ontology configuration management      | 0       | 0     | 0      | 2             | 0      | 1               | 0    | 0       | 0   | 0         | 2    |
| Documentation                          | 1       | 1     | 0      | 3             | 0      | 2               | 1    | 0       | 0   | 0         | 3    |
| Training                               | 0       | 0     | 0      | 0             | 0      | 0               | 0    | 0       | 0   | 0         | 1    |
| Integration                            | 1       | 1     | 1      | 1             | 0      | 1               | 0    | 1       | 1   | 1         | 2    |
| Merging and alignment                  | 1       | 0     | 1      | 1             | 0      | 1               | 1    | 1       | 1   | 1         | 2    |
| Summary score                          | 7       | 17    | 7      | 28            | 3      | 26              | 20   | 3       | 3   | 4         | 43   |

Notes: Ranks: 0—Not proposed, 1—Proposed, 2—Described, 3—Fully described.
5.2. Second session experiment settings and results

For effectiveness assessment of POEM, another empirical experiment is designed for the second session. The subjects of the second session are 15 participants who are either having substantial experience or directly involved in developing ontologies using POEM including ontology developers mainly PhD students, DEs, and end-users. The data for the empirical experiment is collected from the participants using a questionnaire consisting of 30 questions. Questions in the questionnaire are propositions whose answers are selected from a 5-point Likert-scale, ranging from “strongly disagree” to “strongly agree.” The questions describing the methods and measures for assessing effectiveness of the POEM are included in the questionnaire to exploit experiences of the personals and characteristics of the ontologies developed using POEM. The questions analyzes POEM with respect to its neutrality, coverage of problem, usability, usefulness, solving tasks faster, modeling mistakes, results quality, and using of software engineering experiences for OE. Table 9 summarizes statistics of the participants’ responses to the questions. About 73.78% of the participants (31.56% strongly agree and 42.22% agree) agreed in effectiveness of POEM and showed their confidence, whereas, only 17.11% of the participants (6.89% strongly disagree and 10.22% disagree) did not.

The data collected using Likert-scale is of ordinal scale. With ordinal scale, the order of the values is important and significant but the difference between each one is not really known. For analyzing Likert-scale data with descriptive statistics, we combined the five response categories (i.e. strongly disagree, disagree, neutral, agree, and strongly agree) into two nominal categories (i.e. disagree, and agree) for allowing to carry out Chi-Square test. Table 10 shows the distribution of response categories into nominal categories and their percentage values in the nominal categories and in the total. To check whether POEM provides an effective approach for developing SW ontologies, we have performed a Chi-Square test on the data collected in the second session using SPSS 16.0. The null and alternative hypotheses are respectively:

\[ H_0: \text{POEM is not an effective methodology for SW ontology engineering.} \]
\[ H_1: \text{POEM is an effective methodology for SW ontology engineering.} \]

Table 11 shows results of Chi-Square test, where Pearson Chi-Square statistic \( \chi^2 = 71.20 \), and \( p < 0.001 \). The null hypothesis is rejected, since \( p < 0.05 \) (i.e. in fact \( p < 0.001 \)). Therefore, the alternative hypothesis is significant, which signifies that POEM is an effective methodology for SW ontology engineering.

5.3. Additional comments

Apart from the frameworks comparisons, several related valuable characteristics of POEM are observed in the process. These includes: (1) flexibility for empowering ODT in determining the development phases according to nature and complexity of ontology building project, (2) flexibility for accommodating changes at any stage of the ontology building life cycle, (3) supporting reuse of ontological and non-ontological resources to accelerate the ontology building process, increase probability of producing high quality ontological product, save time, and reduce costs, (4) blurring the difference between ontology development and ontology maintenance results into the production of a highly customizable ontological product, and (5) applicability in complex domains where information would be difficult to extract and understand, and may be not readily available. Despite potential advantages, POEM also has several weaknesses/challenges including: (1) requiring active and timely involvement and commitment from the ODT, DEs, and end-users to finish ontology building project within the defined timeframe and constraints, (2) communication and coordination skills plays major role in a project progress, (3) a confusion may arise due to informal demands of improvements after each phase—controlled mechanisms need to be develop to handle subsequent requests, (4) success of a project is highly dependent on the proper project planning and scheduling, and risk analysis and management—requiring high level of expertise to carry out these tasks, (5) correct division and estimation of a project into increments requires prior clear and complete definition of the whole requirements and applications which might not be readily available, (6) applicable
but not suitable for small ontology building projects, and (7) requiring strong technological support that tools should extend either full or partial to the methodology.

6. POEM and collaborative OE

Collaborative OE is a consensus building process that involves collaboration and participation of geographically distributed stockholders to develop useful and economically feasible ontologies in an incremental and asynchronous fashion (Simperl & Luczak-Rösch, 2013). Collaborative OE is the result of intensive theoretical and research within the fields of SW and supporting technologies (e.g. Web 2.0). Stakeholders in collaborative OE use Wikis and similar communication platforms as the most important technological components to exchange ideas and discuss modeling decisions. OE methodologies varies for centralized and collaborative OE. A collaborative OE methodology has the phases (Karapiperis & Apostolou, 2006): (1) preparatory and analysis phase that defines the criteria of analysis of the domain to be captured in the ontology by agreeing stockholders on the requirements and their priorities for guiding and assessing its success, (2) anchoring phase to develop initial version of ontology that will seed next phase, (3) iterative development phase for revising ontology

Table 9. Statistics of the participants’ responses for the questions in second session

| Questions * sample | Likert-scale | Frequency | Percent | Valid percent | Cumulative percent |
|--------------------|--------------|-----------|---------|---------------|-------------------|
| 30 *15             | Strongly disagree | 31 | 6.89 | 6.89 | 6.89 |
|                    | Disagree     | 46 | 10.22 | 10.22 | 17.11 |
|                    | Neutral      | 41 | 9.11  | 9.11  | 26.22 |
|                    | Agree        | 190 | 42.22 | 42.22 | 68.44 |
|                    | Strongly agree | 142 | 31.56 | 31.56 | 100  |
|                    | Total        | 450 | 100  | 100  |         |

Table 10. Distribution of response categories into nominal categories and their percentage values

| Groups            | Group wise distribution | Likert-scale responses | Total |
|-------------------|-------------------------|------------------------|-------|
|                   | Strongly disagree | Disagree | Neutral | Agree | Strongly agree |
| Disagree          | Count | 31 | 46 | 41 | 0 | 0 | 118 |
|                   | (%) Group disagree | 26.3 | 39.0 | 34.7 | 0 | 0 | 100.0 |
| Agree             | Count | 0 | 0 | 0 | 190 | 142 | 332 |
|                   | (%) Group agree | 0 | 0 | 0 | 57.2 | 42.8 | 100.0 |
| Total             | Count | 31 | 46 | 41 | 190 | 142 | 450 |
|                   | (%) Groups | 6.9 | 10.2 | 9.1 | 42.2 | 31.6 | 100.0 |

Table 11. Results of the Chi-Square test using SPSS

| Descriptive statistic | Value | df | p-Value |
|-----------------------|-------|----|---------|
| Pearson chi-square    | 4.500E+2 | 4 | .000    |
| Likelihood ratio      | 257.832 | 4 | .000    |
| Linear-by-linear association | 340.427 | 1 | .000    |
| N of valid cases      | 450    |    |         |

*a0 cells (0%) have expected count less than 5. The minimum expected count is 8.13.
until consensus is reached through a consensus building technique, and (4) application phase to apply the final ontology structure to demonstrate its uses. The two fundamental roles in collaborative OE are ontology editors and ontology contributors where the former have the authority to perform changes in the ontology and the later give feedback and suggest changes based on new and evolving requirements of their particular settings (Simperl & Luczak-Rösch, 2013). POEM, by nature, is a comprehensive methodology proposing an extended list of activities that have the potential to fulfill the phases of collaborative OE methodology, as described above. However, in the context of this paper, POEM is experimented and used for centralized OE. Furthermore, the ontologies developed using POEM are all based on centralized approach. As collaborative OE has emerged as an independent research topic in the OE community, therefore, POEM is to be experimented in the collaborative OE paradigm and may be enhanced or restructured accordingly to provide full potential support.

7. Conclusion and future work

A number of OE methodologies are reported in the literature for OE generally and web OE specifically. However, none of them has been accepted as a standard nor used commonly to be adopted as a standard in the near foreseeable future. Therefore, OE is still much of an art and not an engineering discipline. Building ontology is not the same as developing a software system, but the steps and phase used for software systems development can be equally applied to ontologies development.

In this paper, we have presented a methodology for SW ontology building called practical ontology engineering model (POEM) that leverages the large valuable experiences and concepts from the widely accepted standards of software engineering and other state-of-the-art technologies. POEM is an evolutionary incremental methodology that consists of sequence of phases and activities for guiding SW ontology building projects from inception to deployment. The strength of the methodology lies in the clear incorporation of software engineering experiences for SW ontology engineering and its vibrant support for ontology project management activities equally to the core ontology building activities within a single suit. The phases and activities within the phases are comprehensive enough to provide enough opportunities to project the need and feasibility of the project, to understand domain of interest and scope and complexity of the ontology, to properly plan and schedule the project for effective utilization of resources and skill of the experts under the project’s constraints, to effectively anticipate the risk which could hinder the project and ultimately the quality of the ontology to be developed, to correctly transform conceptual models into semantic networks for achieving extreme accuracy at implementation, and to design effective testing and feedback strategies to fuel plan for the next iterations. The methodology is flexible enough allowing ODT to scale and customize the phases and activities according to the natures and contexts of projects. Theoretical and empirical evaluations of POEM have shown that POEM has all of the features of the best solutions and in some cases outclass them by providing features which they do not have. In addition to SW ontology building, POEM has the potential application for a wide range of ontology building projects at enterprise level. Practicality of POEM has been demonstrated by developing WPO as proof-of-concept in this paper. Apart from it, POEM has been experimented in a few of the ontology building activities (e.g. used in the ontology building project for supporting the automation of hospital management systems of KP, Pakistan) of the WISE (Web Information and System Engineering) research group projects at the University of Peshawar. POEM, however, is a pragmatic approach and do not claim to be an orthogonal, complete, and universally acceptable one. It, therefore, needs attention from the academia and industry for its usage in technically high and large scale ontology building projects to effectively evaluate its strengths and weaknesses.

POEM has almost all of the phases and activities that are needed for collaborative OE, but focused on centralized OE in this paper. In the future, we are considering to use POEM within the collaborative OE and may integrating the missing collaborative OE activities within the POEM to provide full potential support.
Acknowledgment
This research work has been undertaken by the first author as a partial fulfillment of PhD degree with support of the Higher Education Commission (HEC) of Pakistan.

Funding
The authors received no direct funding for this research.

Address details
Shaukat Ali
E-mail: shoonikhan@upesh.edu.pk
Shah Khusro
E-mail: khusro@upesh.edu.pk
ORCID ID: http://orcid.org/0000-0002-7734-7243
1 Department of Computer Science, University of Peshawar, Peshawar 25210, Pakistan.

Citation information
Cite this article as: POEM: Practical ontology engineering model for semantic web ontologies. Shaukat Ali & Shah Khusro, Cogent Engineering (2016), 3: 1193959.

Note
1. ODT would consist of personals belonging to the different groups such as KEs, ontology developers, and project managers

Cover image
Source: http://www.bosch-india-software.com/media/rbei_image_database/new_internet/engineering/product_engineering/embedded_software/engineering/images_38/RI_EMBEDDED_SOFTWARE_ENGINEERING_w734.jpg; http://www.extensions.in.th/amitiae/04_2012_1/images/insleepe_d3.png; http://singlegranimage/wp-content/uploads/2010/03/ontology.png

References
Alam, F., Ali, S., Khan, M. A., Khusro, S., & Rauf, A. (2014). A comparative study of RDF and topic maps development tools and APIs. BUJICT Journal, 7(1), 1-12.
Alam, F., Khan, M. A., Ali, S., & Khusro, S. (2014). The jigsaw of resource description framework (RDF) and topic maps serialization formats: A survey. Proceedings of the Pakistan Academy of Sciences, 51, 101-114.
Ashraf, J., Chang, E., Hussain, O. K., & Hussain, F. K. (2015). Ontology usage analysis in the ontology lifecycle: A state-of-the-art review. Knowledge-Based Systems, 80, 34-47. http://dx.doi.org/10.1016/j.knosys.2015.02.026
Astrava, I., & Stantic, B. (2005). An HTML-form-driven approach to reverse engineering of relational databases to ontologies. Paper Presented at the Databases and Applications. Retrieved from http://dblp.uni-trier.de/db/conf/db/conf/dbconf01005.html#AstravaS05
Benslimane, S., Malik, M., Rahmouni, M., & Rahmoun, A. (2008). Toward ontology extraction from data-intensive web sites: An HTML forms-based reverse engineering approach. The International Arab Journal of Information Technology, 5, 34-44.
Bernaras, A., Laregoiti, I., & Corea, J. M. (1996). Building and reusing ontologies for electrical network applications. Paper presented at the 12th European Conference on Artificial Intelligence ECAI’96. Retrieved from http://dblp.uni-trier.de/db/conf/ecai/ecai96.html#BernarasLC96
Breitmann, K. K., Casanova, M. A., & Truszkowski, W. (2007). Semantic web: Concepts. London: Springer-Verlag.
Burton-Jones, A., Storey, V. C., Sugumaran, V., & Aihluwalia, P. (2005). A semiotic metrics suite for assessing the quality of ontologies. Data & Knowledge Engineering, 55, 84–102. http://dx.doi.org/10.1016/j.datak.2004.11.010
Casesly-Hayford, L. (2003). A comparative analysis of methodologies, tools and languages used for building ontologies. Warrington: CCLRC Daresbury Laboratories.
Corcho, G., Fernández-Lópe, M., & Gómez-Pérez, A. (2003). Methodologies, tools and languages for building ontologies, Where is their meeting point? Data & Knowledge Engineering, 46, 41-64. http://dx.doi.org/10.1016/S0169-023X(02)00195-7
D’Antonio, F., Missikoff, M., & Taglino, F. (2007). Formalizing the OPL eBusiness ontology design pattern with OWL. Paper presented at the IESA. Retrieved from http://dblp.uni-trier.de/db/conf/iesa/iesa2007.html#DAntonioMT07
De Nicola, A., Missikoff, M., & Navigli, R. (2009). A software engineering approach to ontology building. Information Systems, 34, 258–275. http://dx.doi.org/10.1016/j.is.2008.07.002
Dennis, A. (2012). Systems Analysis and Design (5th ed.). Hoboken, New Jersey: Wiley Publishing.
Duque-Ramos, A., Boeker, M., Jansen, L., Schulz, S., Iniesta, M., & Ferna ndez-Breis, J. T. S. (2014). Evaluating the good ontology design guideline (GoodOD) with the ontology quality requirements and evaluation method and metrics (QOUnet). PLoS ONE, 9(8), 1-14.
Ferreira, C. R., Marques, P., Martins, A. L., Rita, S., Grilo, B., Araujo, R., ... Pinto, H. S. (2007). Ontology design risk analysis. In R. Meersman, Z. Tari, & P. Herrero (Eds.), On the move to meaningful internet systems 2007: OTM 2007 workshops: OTM federated international workshops and posters, AWESoMe, CAMS, OTM academy doctoral consortium, MONET, OntoContent, ORM, PerSys, PPN, RDDs, SWS, and SWWS 2007, Vilamoura, Portugal, November 25-30, 2007, Proceedings, Part I (pp. 522-533). Berlin: Springer, Berlin.
Gómez-Pérez, A., Fernández-López, M., & Corcho, O. (2007). Ontological Engineering: With examples from the areas of knowledge management, e-Commerce and the semantic web. (Advanced information and knowledge processing). New York, NY: Springer-Verlag.
Gruber, T. R. (1993). A translation approach to portable ontology specifications. Knowledge Acquisition, 5, 199–220. http://dx.doi.org/10.1016/0952-813X(93)90007-E
Grüninger, M., & Fox, M. S. (1995). Methodology for the design and evaluation of ontologies. Paper presented at the International Joint Conference on Artificial Intelligence (IJCAI95), Workshop on Basic Ontological Issues in Knowledge Sharing. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.44.8723
Guizzardi, G., Herre, H., & Wagner, G. (2002). Towards ontological foundations for UML conceptual models. In R. Meersman & Z. Tari (Eds.), On the move to meaningful internet systems 2002: CoopIS, DOA, and ODBASE: Confederated International Conferences CoopIS, DOA, and ODBASE 2002 Proceedings (pp. 1100–1117). Berlin: Springer, Berlin.
Hesse, W. (2005). Ontologies in the software engineering process. Paper presented at the Enterprise Application Integration (EAI’05), Marburg, Germany.
Horrocks, I. (2008). Ontologies and the semantic web. Communications of the ACM, 51, 58–67. http://dx.doi.org/10.1145/1405936
IEEE. (1990, December 31). IEEE standard glossary of software engineering terminology. IEEE Std 610-1990, 1-84. doi: http://dx.doi.org/10.1109/IEESTD.1990.101064. Retrieved August 25, 2015 from www.mit.jyu.fi/ope/kurssit/TIES462/Materiali/IEEE_SoftwareEngGlossary.pdf
Kalyanpur, A., Hashmi, N., Golbeck, J., & Parsia, B. (2004). Lifecycle of a casual web ontology development process. Paper presented at the WWW Workshop on Application Design, Development and Implementation Issues in the Semantic Web. Retrieved from http://dblp.uni-trier.de/db/conf/www/wwws2004.html#KalyanpurHGP04
Kapoor, B., & Sharma, S. (2010). A Comparative study of ontology building tools in semantic web applications. *International Journal of Web & Semantic Technology*, 1, 1–13.

Karapiperis, S., & Apostolou, D. (2006). Consensus building in collaborative ontology engineering processes. *Journal of Universal Knowledge Management*, 1, 199–216.

Lopez, M. F., Gomez-Perez, A., Sierra, J. P., & Sierra, A. P. (1999). Building a chemical ontology using methontology and the ontology design environment. *IEEE Intelligent Systems*, 14, 37–46. http://dx.doi.org/10.1109/5254.74904

Maynard, D., Peters, W., & Li, Y. (2006). Metrics for evaluation of ontology-based information extraction. Paper presented at the WWW Workshop on Evaluation of Ontologies for the Web (EON). Edinburgh.

Missikoff, M., & Taglino, F. (2002). Business and enterprise ontology management with SymOntOx. In L. Horrocks & J. Hendler (Eds.), The semantic web—ISWC 2002: First international semantic web conference Sardinia, Italy, June 9–12, 2002 proceedings (pp. 442–447). Berlin: Springer. http://dx.doi.org/10.1007/3-540-48005-6

Navigli, R., & Velardi, P. (2004). Learning domain ontologies from document warehouses and dedicated web sites. *Computational Linguistics*, 30, 151–179. http://dx.doi.org/10.1162/089120104323091276

Noy, N. F., & McGunness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Stanford, CA: Stanford University Knowledge Systems Laboratory Technical Report KSL-01-05.

Pressman, R. (2010). *Software engineering: A practitioner’s approach*. New York, NY: McGraw-Hill.

Rodrigues, F. H., Flores, C. D., & Rotta, L. N. (2015, February 7–9). An ontology design pattern to represent universal relationships. Paper presented at the Semantic Computing (ICSC), 2015 IEEE International Conference.

Schreiber, G. T., & Akkermans, H. (2000). Knowledge engineering and management: the CommonKADS methodology. London: MIT Press, Cambridge.

Simperl, E., & Lucchak-Rösch, M. (2013). Collaborative ontology engineering: A survey. The Knowledge Engineering Review, 29, 101–131.

Sommerville, I. (2006). *Software engineering: (Update)* International Computer Science (8th ed.). Boston, MA: Addison-Wesley Longman.

Staab, S., Studer, R., Schnurr, H. P., & Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16, 26–34. http://dx.doi.org/10.1109/5254.912382

Stojanovic, L., Stojanovic, N., & Volz, R. (2002). Migrating data-intensive web sites into the semantic web (pp. 1100–1107). Paper presented at the Proceedings of the 2002 ACM symposium on Applied computing, Madrid, Spain.

Su, X., & Ilebrekke, L. (2002). A Comparative Study of Ontology Languages and Tools. Paper presented at the Proceedings of the 14th International Conference on Advanced Information Systems Engineering, Toronto, Ontario, Canada.

Suarez-Figueras, M. C., & Gomez-Perez, A. (2009). NeOn methodology for building ontology networks: A scenario-based methodology. Paper presented at the Proceedings of the International Conference on Software, Services & Semantic Technologies (S3T 2009), Sofia, Bulgaria.

Swartout, B., Patil, R., Knight, K., & Russ, T. (1996). Towards distributed use of large-scale ontologies. Paper presented at the Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop. Retrieved from http://ki.cs.psc.ualgary.ca/KAOW/KAW.html

Tartir, S., & Arpinar, I. B. (2007). Ontology evaluation and ranking using OntoQA. Paper presented at the Proceedings of the International Conference on Semantic Computing, Irvine, California, CA.

Tijerino, Y. A., Embley, D. W., Lonsdale, D. W., Ding, Y., & Nagy, G. (2005). Towards ontology generation from tables. *World Wide Web*, 8, 261–285. http://dx.doi.org/10.1007/s11280-005-0360-8

Uschold, M., & King, M. (1995, August 20–25). Towards a methodology for building ontologies. Paper presented at the Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95, Montreal.

Vanitha, K., Yasudha, K., Venkatesh, M. S., Ravindra, K., Lakshmi, S. V., & Soujanya, K. N. (2011). The development process of the semantic web and web ontology. *IJACSA. International Journal of Advanced Computer Science and Applications*, 2, 122–125.

Vujasinovic, M., Ivezic, N., & Kulvatunyou, B. (2015). A survey and classification of principles for domain-specific ontology design patterns development. *Applied Ontology*, 10, 41–69.

Youn, S., Arora, A., Chandrasekhar, P., Joyanty, P., Mestry, A., & Sethi, S. (2004). Survey about ontology development tools for ontology-based knowledge management. Retrieved 2012, November 20 from www.scf.usc.edu/~csci586/projects/ontology-survey.doc

© 2016 The Author(s). This open access article is distributed under a Creative Commons Attribution (CC-BY) 4.0 license.

You are free to:
- Share — copy and redistribute the material in any medium or format
- Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:
- Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made.
- You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- No additional restrictions

You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.