Received: October 29, 2018

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Agile Methodology for Ontology Development

Abdelghany Salah Abdelghany[1]* **Nagy Ramadan Darwish[1]** **Hesham Ahmed Hefni[2]**

[1]*Department of Information Systems and Technology, Institute of Statistical Studies and Research, Cairo University, Egypt*
[2]*Department of Computer Science, Institute of Statistical Studies and Research, Cairo University, Egypt*
* Corresponding author's Email: body_1986@live.com

Abstract: Ontology is defined as an explicit specification of a shared conceptualization. Currently, ontology is used in semantic web, information retrieval, artificial intelligence, information systems, knowledge management, etc. The development of ontology involves a structural and logical complexity that is comparable to the development of software artifacts. Therefore, ontology building requires a methodology to ensure its reliability. In this context, there are several methodologies proposed for building ontologies. However, most of the existing methodologies failed to provide sufficient details for the activities and techniques employed in them with a defined ontology lifecycle. To build ontologies that are reliable, long lived and continually adapted, the ontology engineering (OE) should be supported by the software engineering (OE). But, SE was not initially meant to support the development of software artifacts such as ontologies. There is a significant gap between them in terms of popularity and maturity level. The aim of this paper is to bridge this gap by proposing an Agile Methodology for Ontology Development (AMOD). AMOD adopts the agile principles and practices in the ontology development. The final framework of AMOD fits the various ontology activities into the phases of the Scrum agile methodology. It has three phases: pre-game, development and post-game. AMOD was applied to develop ontology for software project time management. Additionally, a compliance analysis of different ontology methodologies with respect to the IEEE Standard was made. Results showed that AMOD resulted in 56% satisfaction for IEEE standard processes. This resembles 22% enhancement in the satisfaction against the other methodologies.

Keywords: Ontology, Ontology development, Ontology engineering, Software engineering, Ontology methodologies, Knowledge engineering.

1. Introduction

Ontology is generally defined as an explicit and formal specification of a shared conceptualization [1]. A conceptualization refers to the concepts related to a domain of interest and the relationship exiting among them. Explicit indicates that the concepts used and the restrictions on their use should be defined clearly. Formal means that an ontology should be machine-readable. Shared means that an ontology must be accepted by a group or community [2]. The growth of ontology is becoming more popular in diverse fields such as semantic web search, artificial intelligence, natural language processing, information systems, bio-informatics, knowledge management, etc. the development of ontology is complex because of various factors that include the dynamic changes of business needs, the heterogeneous platforms, the need of advanced tool support, etc. [3]. Therefore, the development of ontology should follow a methodology to ensure its reliability.

Ontology engineering is an emerging field that concerns the principles methods, methodologies and tools for developing and managing ontologies. This field aims at making explicit knowledge included in software applications, enterprises and business procedures for a specific domain [4]. The ontology development methodology refers to the set of activities that need to be performed when building

ontologies. The goal of the ontology development methodology is to ensure the clarity, the coherence, extendibility, the reusability and the reliability of the ontology [5].

In this context, there are various methods and methodologies proposed for developing ontologies. The exciting methodologies have either been proposed initially or derived from experience during ontology development for different projects. Among the methodologies of developing ontologies are Cyc method, Uschold and King's method, Grüninger and Fox's methodology, Methontology, KACTUS, SENSUS and On-To-Knowledge [6]. Some methodologies are designed for developing ontologies form scratch while the others focus on reusing other ontologies.

However, most of the existing methodologies failed to provide sufficient details for the activities and techniques employed in them along with a defined lifecycle model [7]. Moreover, the field of ontology engineering still lacks standardized methodologies that can be adapted to different ontology settings. The main reason is that most of the methodologies were applied to develop ontologies for specific projects. So, the generalization of the methodology was not proposed for other contexts.

Furthermore, there are challenges in the area of ontology engineering that need to be addressed. These challenges include working with people, gathering requirements form a diverse set of users, prioritizing these requirements, keeping domain experts engaged and responding to changing knowledge [8]. Some of these challenges are similar to those existed in the field of software engineering. To build ontologies that are reliable, long lived and continually adapted, the ontology development should be supported by the software engineering methods and practices. But, software engineering was not initially meant to support the development of software artifacts such as ontologies. Thus, it should be adapted to meet the needs of ontology development.

According to the Forrester's report, 65% of mid-sized companies has reported that 100% of their teams adopts agile in their software development. Additionally, 85% of the agile adaptors use Scrum as their agile method [9]. The principles and values of agile methods are defined in a document called agile manifest [10]. Agile methods include Scrum, Extreme Programming, Dynamic System Development, Adaptive Software Development, Feature Driven Development, Crystal, etc. [11]. Some of advantages of agile methods include better software quality, adapting to changing requirements, improved communication, process adaptability and higher customer satisfaction [12].

The aim of this paper is to bridge the gap between ontology engineering and software engineering. This paper proposes an Agile Methodology for Ontology Development (AMOD) by integrating the ontology engineering activities and the agile practices. The final framework of AMOD fits the various ontology activities into the phases of the Scrum agile methodology. Furthermore, the aim is also is to make the ontology activities easily applied during the development of ontology-based software.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 describes the proposed methodology and its phases and activities. Section 4 discusses the application of the proposed methodology in the domain of software project management. Section 5 discusses the evaluation of the proposed methodology. The last section gives conclusion and future research directions.

2. Related work

A range of methodologies has been proposed in the literature for building ontologies. However, the most known methodologies only have been considered. Uschold and King [13] proposed a method based on the experience for building the Enterprise Ontology. The Enterprise Ontology is of a set of terms and definitions related to business enterprises. This method includes four main stages: identifying purpose, building the ontology, evaluation and documentation. Although it was the first method proposed for building ontologies, it does not describe any techniques for performing each of the above four stages.

Grüninger and Fox [14] proposed a methodology for ontology building related to the domain of business. This methodology is based on the Toronto Virtual Enterprise (TOVE) project. It consists of the following steps: motivating scenarios, informal competency questions, formal terminology, formal competency questions, formal axioms and completeness theorem. But, this methodology does not describe a lifecycle model for building ontologies. Although there are ontologies developed according to this methodology, the domain is limited to business.

Methontology [15] was designed at the Polytechnic University of Madrid. It is used for creating ontologies at the knowledge level. Its framework includes the following development-oriented activities: specification, conceptualization,

formalization, implementation and maintenance. In parallel to these activities, support activities are performed simultaneously. They include knowledge acquisition, documentation, integration, evaluation and integration. However, the methodology does not include any techniques to perform these activities in a formal manner.

Staab et al. [16] proposed the On-To-Knowledge methodology for developing ontology-based Knowledge Management (KM) systems. On-To-Knowledge consists of six activities as follows: feasibility study, kickoff, refinement, evolution and maintenance. It ranges from the early stage of starting a KM project to the final version of the ontology-based KM application. However, this methodology does not consider ontology documentation and configuration management.

DOGMA (Developing Ontology-Grounded Methods and Applications) was also proposed for ontology engineering [17]. DOGMA consists of two main steps: preparatory stages and ontology engineering stages. Preparatory stages include the following: vision statement, feasibility study, project management and preparation and scoping. Ontology engineering stages consist of domain conceptualisation and application specification. However, DOGMA does not address ontology evaluation and integration.

Nicola et al. [18] proposed the Unified Process for ONtolog (UPON) building. UPON is based on the widely used software engineering process: Unified Process (UP). It also tacks the advantage of the Unified Modelling Language (UML). UPON is composed of five main workflows: requirements, analysis, design, implementation and test. One of the drawbacks of UPON is that it does not consider the development of generic ontologies. Moreover, UPON neglects the collaborative ontology construction aspect.

NeOn methodology [19] is a collection of pathways for building ontologies. It includes nine scenarios, a glossary of processes and activities, two ontology life cycle models (iterative and waterfall) and a set of methodological guidelines for different processes and activities. NeOn is intended for the traditional ontology engineer who can be a software developer or ontology practitioner involved in the ontology development. Thus, it does not include any guidelines for non-experienced domain experts to build ontologies.

The Enterprise Strength Ontology Engineering (EsOE) was proposed by Annamalai and Rosli [20]. This methodology adopts a set of value-added activities from the Rational Unified Process (RUP) model, Agile model, Capability Maturity Model Integration (CMMI) model and IEEE 1074-1995 standard. It is structured into three levels: engineering, project-focus and organization-focus. However, a defined ontology lifecycle is missing in EsOE.

Falbo [21] proposed a Systematic Approach for Building Ontologies (SABiO). SABiO development process is composed of five main phases: identifying purpose and collecting requirements, capturing and formalizing ontology, design, implementation and test. SABiO supports the development of both reference and operational domain ontologies. However, SABiO does not address some important activities of ontology development such as planning, configuration management and maintenance.

Rani et al. [22] proposed an ontology methodology that is derived from both traditional waterfall model and Rational Unified Process (RUP). The stages proposed by the methodology are based on the Methontology. The framework of the proposed methodology fits the different phases and stages of ontology development into the RUP phases: inception, elaboration, construction and transition. However, the stages and the techniques are not described in detail.

The Lightweight Methodology for Rapid Ontology Engineering (UPON Lite) was proposed by Nicola and Missikoff [23]. It is derived from the Unified Process for ONtology building (UPON). In UPON Lite, ontologies are developed by domain experts, while ontology engineers only intervene to deliver formal ontology. UPON Lite includes the following steps: domain terminology, domain glossary, taxonomy, predication, parthood and ontology. However, the techniques for performing these steps are not explained in detail.

John et al. [24] proposed an Incremental and Iterative Agile Methodology (IIAM) for ontology development for the education domain. The stages of IIAM are domain vocabulary acquisition, enumeration of concepts and properties, taxonomy identification, adhoc binary relationships, concepts attributes and relationships, restrictions and vocabulary linking with data. These stages are fitted into RUP phases: inception, elaboration, construction and transition. However, IIAM stages for building ontology are described in a general way.

The Software Centric Innovative Methodology (SCIM) was proposed for ontology development by extending the process models of software engineering [25]. SCIM has five ontology development workflows: requirements analysis, domain analysis, conceptual design, implementation and evaluation. These workflows are mapped to the

disciplines of the RUP model. However, there are not ontologies developed according to this methodology in order to assess its accuracy and applicability.

The exiting ontology engineering methodologies isolate the activities of ontology development from the mainstream development of the software. So, they cannot be easily used during the development of ontology-based software. Furthermore, they do not consider the agile principles and practices in their activities and techniques. Thus, the main difference between the proposed methodology and the other methodologies is that it adopts the agile principle and practices in the ontology development. Therefore, it can be easily understood and followed by software developers during the development of ontology-based software.

3. The agile methodology for ontology development (AMOD)

The aim of the proposed methodology is to adapt the agile principles and practices from software engineering into the development of ontologies. As shown in Fig. 1, AMOD classifies the ontology development into three phases: pre-game, development and post-game. It also identifies some support activities that occur in parallel with other activities. The primary roles considered in AMOD are ontology owner, ontology engineer and ontology user. Ontology owner is responsible for representing customer needs to the ontology engineers. Ontology engineer is responsible for implementing the ontology. Ontology user intends to use the ontology for a specific purpose. Fig. 2 presents the final framework of AMOD.

3.1 Pre-game phase

The pre-game phase is the first phase of ontology development. This phase includes the identification of the ontology goal and scope, tools and techniques, competency questions and available sources. All of these activities are described below in detail.

- *Ontology Goal and Scope*: The definition of ontology goal and scope is the first step in building ontologies. This activity states why the ontology is being created and what its intended uses are and who the users of ontology are [13]. The scope specifies what should be included in the ontology and what should not. It limits the amount of concepts to be analyzed [26].

- *Tools and Techniques*: Techniques for knowledge capture should be identified. The language and tools that will be used for implementing the ontology should be selected. Ontology building tools allow users to visually edit, browse, inspect, and code ontologies. Examples of ontology building tools include Apollo, Protégé 3.4, IsaViz and SWOOP [27]. The most popular languages for representing ontologies include knowledge Interchange Format (KIF), Web Ontology Language (OWL), Resource Description Framework (RDF) and DARPA Agent Markup Language and Ontology Inference Layer (DAML+OIL), etc. [28].

- *Ontology Requirements*: After identifying the ontology goal and scope, the requirements of ontology should be gathered. The requirements gathered can be stated as a set of competency questions (CQs). CQs are the questions that the ontology must be able to give answers [21]. The set of CQs is stored in a product backlog that is ranked by business value and risk [29]. Moreover, the CQs provide a way for evaluating the ontology.

- *Source Selection*: This activity aims to select sources that can be used for eliciting domain knowledge. The main source for knowledge acquisition is domain experts. Other sources include international standards, books, technical reports glossaries, classification schemes and reference models [30].

3.2 Development

The development phase incorporates multiple and iterative cycles that are called sprints. Sprints are typically 1-4 weeks in length. Each sprint includes the following activities:

- *Sprint Planning*: In sprint planning, the ontology owner and ontology engineers select the high-priority product backlog items that will be implemented during the sprint. Then, the ontology engineers decide how they will implement these items [31].

- *Knowledge Acquisition*: In this step, the techniques for knowledge acquisition are applied in order to capture all the relevant terms related to domain of interest (concepts, attributes, relations, etc.). These techniques

include interviewing, brainstorming, protocol analysis and Delphi method, etc. [21].

- *Conceptualization*: The goal of this activity is to organize the gathered knowledge into a semi-formal specification based on a set of intermediate representations (IRs). IRs includes terms glossary, concept dictionary, concept classification trees, binary relationship diagrams, etc. The main output of this activity is the ontology conceptual model [15].

- *Formalization*: The formalization activity transforms the conceptual model into a formal model by coding it using the chosen language and tool.

- *Integration*: The ontology implemented in the sprint must be integrated with the ontologies developed in the previous sprints [30]. To do this, integration operations and integration oriented design criteria are needed [32].

- *Sprint Review*: This meeting is held at the end of the sprint. In this meeting, the ontology engineer and ontology owner review what was done in the sprint [31].

3.3 Post-game phase

The purpose of the final phase is to prepare for a final ontology. It includes the following activities:

- *Evaluation*: Ontology evaluation can be defined in in the view of two perspectives:

  verification and validation. Ontology verification ensures that the ontology is being built correctly, while ontology validation ensures that the correct ontology is being built [33]. Ontology should be evaluated in respect to three different characteristics:
  - Ontology consistency: Checking the consistency of the ontology by using a reasoner such as FaCT++ or Racer [34].
  - Answering CQs: Verifying the ontology against its competency questions.
  - Ontology content: The content of the ontology is evaluated based on some quality criteria such as correctness and completeness [35].

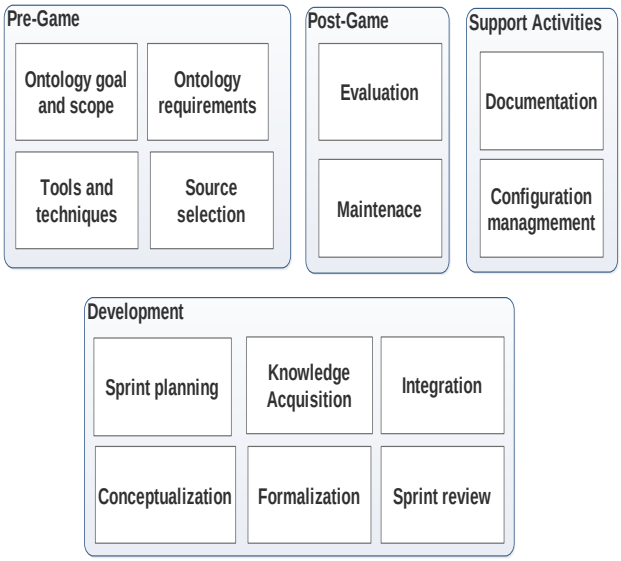

Figure. 1 Abstract View of AMOD

- *Maintenance*: The resulting ontology needs to be updated and corrected to reflect the changes of the domain of interest that it describes. New concepts or relations may be added to extend the resulting ontology to ensure its reliability [36].

3.4 Support activities

The supporting activities are performed at the same time as the other ontology development activities. They include the following activities:

- *Documentation*: Results of the ontology development activities and evaluation process must be documented [21]. Ontology documentation includes three main aspects. The first aspect is to create a human-readable representation of the ontology content. The second aspect is to create machine-readable annotations of documentation metadata. The third aspect is to make the documentation files available as a web resource [37].

- *Configuration Management (CM)*: The goal of this activity is to record all the versions of the documentation and ontology code and to control the changes [38]. Generally, CM of ontology includes four activities derived from the software engineering: configuration identification, configuration control, configuration control and configuration audits [39].

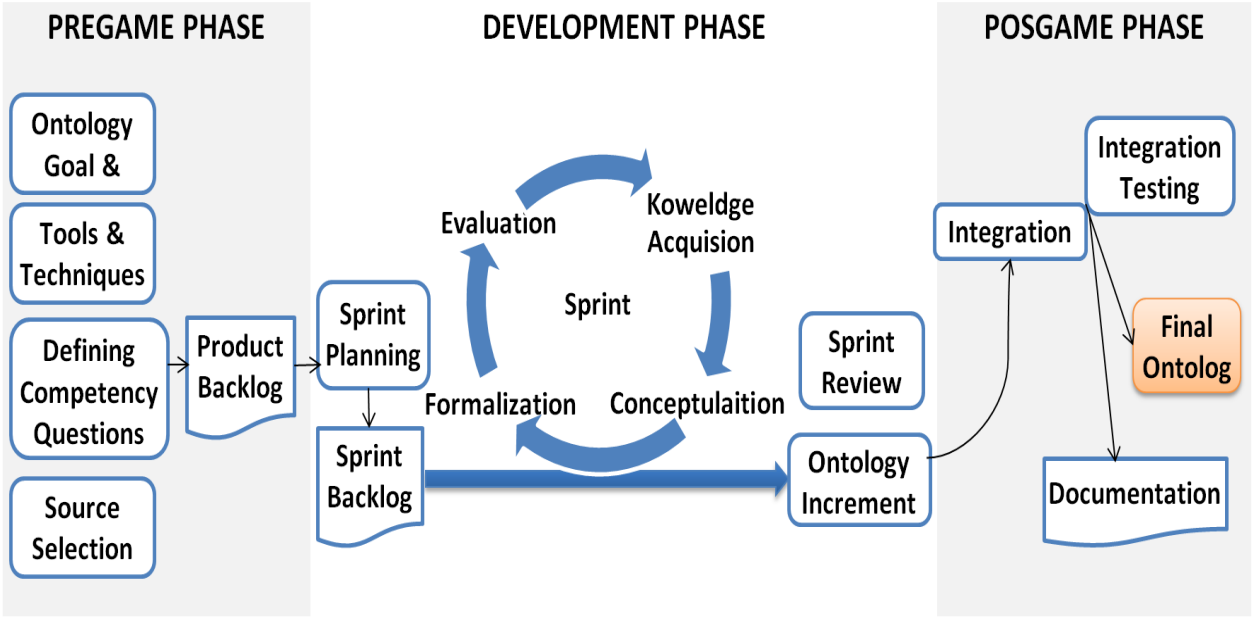

Figure. 2 Final framework of AMOD

4. Application of the proposed methodology

The domain selected for the application of the proposed methodology is software project management where knowledge sharing and reusability is vital. Many tools have been developed to support this process. However, semantic conflicts occur when integrating these tools to exchange data and services. The reason is that these tools do not share a common conceptualization. In this context, ontologies can be useful to solve this problem. The execution of all the phases and activities of the proposed methodology is described in the following sections.

4.1 Pre-game phase

The proposed methodology was used to develop the Software Project Time Management (SPTM) ontology. The goal of building SPTM ontology is to establish a common understanding of the meaning of the terms used by the tools that support this process. The SPTM ontology could be used as intimidator to map concepts and services used by these tools. The scope of the SPTM ontology focuses on the following process as defined in the PMBOK: plan schedule, define activities, sequence activities, estimate activity resources, estimate activity durations, develop schedule and control schedule. After identifying the ontology goal and scope, the requirements of the ontology were gathered. The ontology requirements were formulated as competency questions. The set of competency questions were prioritized by adopting the Planning Poker agile technique. Some examples of them are presented in Table 1.

Table 1. Examples of competency questions

CQ1	What are the types of project processes?
CQ2	What is a project process composed of?
CQ3	What are the types of project activities?
CQ4	What are the inputs and outputs of each activity?
CQ5	Who is responsible for performing project activities?

After a review of available ontology tools, protégé was selected to formalize SPTM ontology. The selection was based on some criteria including tool's customizability, flexibility, usability and extensibility. Protégé is an open source editor that uses different formats and plugins to facilitate ontology development. OWL was selected as the implementation language. The sources chosen for knowledge acquisition include interviews with experts, international standards such as Project Management Body of Knowledge (PMBOK) Guide and technical reports. Table 2 summarizes the outcomes of the main activities included in the pre-game phase.

4.2 Development

The SPTM ontology was developed in three sprints. The development of each increment of the ontology was done based on the activities presented

Table 2. Outcomes of the pre-game phase

Domain	Software project time management
Goal	Used as a reference to map concepts used by the tools supporting the project time management process.
Scope	The following processes: plan schedule, define activities, sequence activities, estimate activity resources, estimate activity durations, develop schedule and control schedule.
Tools and techniques	Protégé, OWL language
Knowledge sources	Interviews with the experts PMBOK Guide Technical reports

in Fig. 1. The first increment followed the following activities:

- Sprint planning: The sprint planning meeting resulted in a set of features that the ontology owner and ontology engineer decided to implement in the sprint.

- Knowledge acquisition: The knowledge acquisition techniques were used to capture all potential relevant terms in the domain. The primary outcome of this activity was a glossary of terms. A part of the SPTM ontology glossary is presented in Table 3.

- Conceptualization: After building the glossary, the concepts and the relationships exiting among them were defined. The captured knowledge was stored in a concept dictionary table. A part of the SPTM ontology concept dictionary is presented in Table 4.

- Formalization: Based on the conceptual model the formal ontology was build using Protégé ontology editor. It was represented in OWL.

- Sprint review: At the end of the sprint the ontology engineers presented the results of the sprints to the ontology owner and other stakeholders.

The other two sprints also followed the activities described above. The ontologies created in these two sprints were integrated with the ontology produced in the first sprints. The general hierarchy of the SPTM ontology is shown in Fig. 3. A Project has

Table 3. Glossary of terms and concepts of SPTM

Concept	**Synonyms**	**Acronyms**	**Description**	**Type**
Project	-	-	A temporary endeavor undertaken to create a unique product or result.	class
Process	-	-	A set of activities that interact to achieve a result.	class
Activity	Task	-	A piece of work that forms one logical step within a process.	class
Simple activity	-	-	An activity that consists of a single step or action.	Sub-class
Human Resource	Contact	HR	The people who work for a company or organization.	class

Table 4. Concepts dictionary of SPTM

Class name	**Class attribute**	**Instance attribute**	**Relation**
Project	-	Name, description	Has
Project Process	-	Name, description	Defined for
Project Activity	-	Name, startDate, endDate	Is performed by, depends on
Human role	-	Name	Is to perform

Processes. Processes have two types: SpecificProcess and GeneralProcess. The GeneralProcess is composed of SpecificProcesses. The SpecificProcess consists of Activities. An Activity may be a SingleActivity or CompositeActivity. These activities are performed by HumanRoles. When assigning the start and end dates of processes and activities, the ScheduledProcesses and ScheduledActivities appears. A HRAllocation assigns a ScheduledActivity to a HumanResource playing a certain HumanRole. It depends on a TeamAllocation that allocates the HumanResource to the Team. When executing scheduled processes and activities, ProcessOccurrence and ActivityOccurrence are generated. Finally, ActivityOccurrence includes several HRParticipations that refers to the participation of a HumanResource.

4.3 Post-game phase

The FaCT++ reasoner supported by Protégé was used to ensure the SPTM ontology consistency. The

inconsistent classes are highlighted in red in a class called Nothing. The next test concerns the possibility to answer the CQs by the SPTM ontology content. This test gave a good result, since it is possible to answer all the listed CQs. For instance, CQ1:" What are the types of project processes?" is elaborated using the concepts "General Project Process" and "Specific Project Process". Finally, the SPTM ontology content was evaluated by the domain experts based on the following quality criteria: consistency, completeness and clarity. The results showed that the consistency of the ontology is 63%, the completeness is 45% and finally the clarity is 41%.

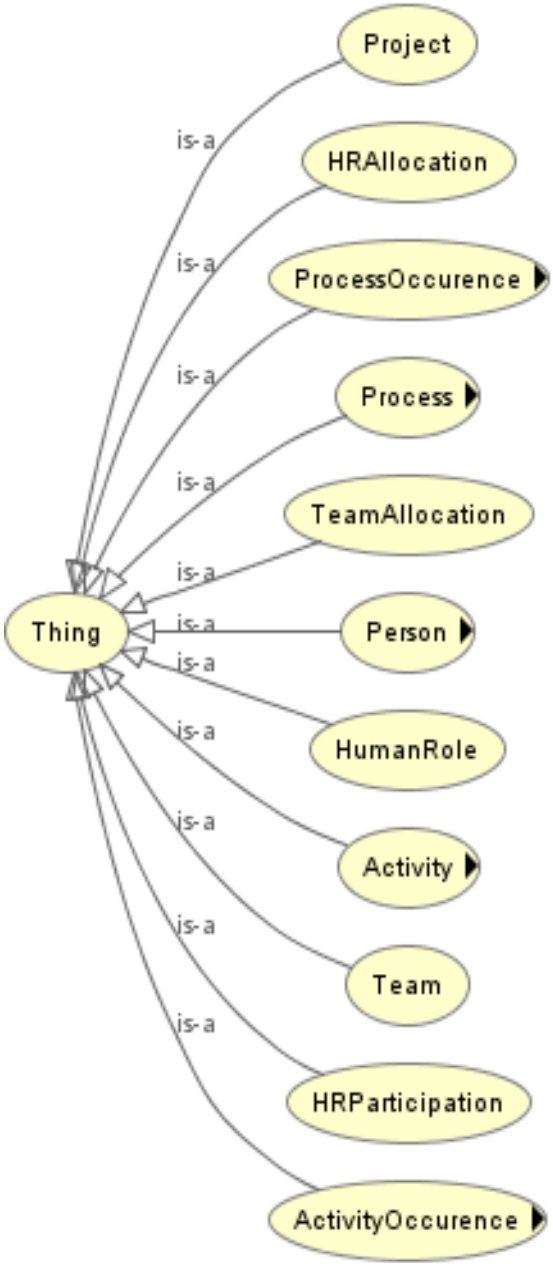

Figure. 3 SPTM general hierarchy

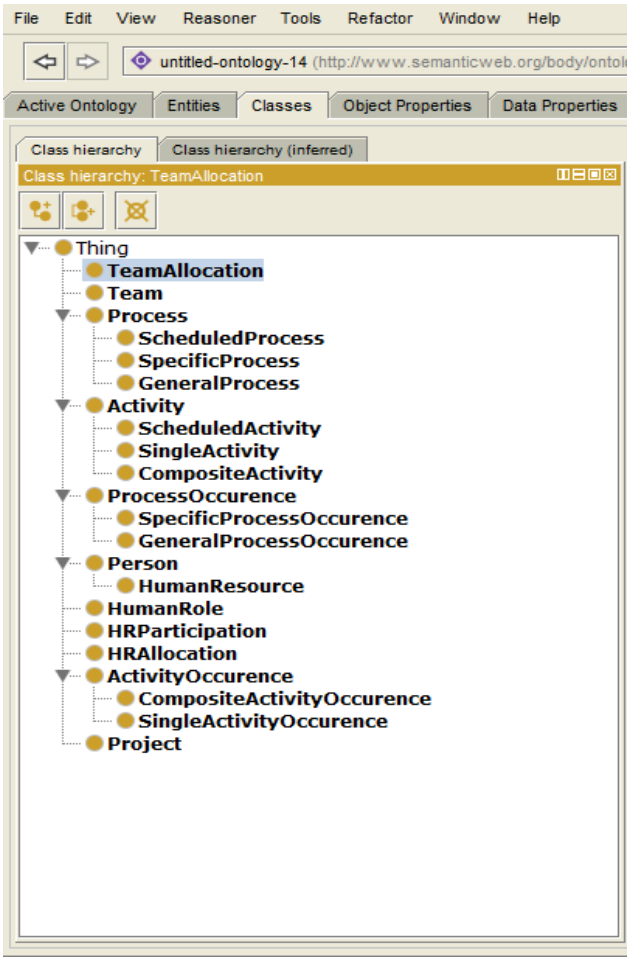

Figure. 4 SPTM ontology in protégé

5. Discussion and assessment

To evaluate AMOD, a comparative assessment against existing methodologies was conducted. The compliance of each methodology with the IEEE standard was defined [40]. As shown in Table 5, The IEEE standard includes three kinds of process as follows:

- *Project management processes*: include the creation of a framework for the ontology lifecycle.
- *Ontology development processes*: include three types of processes: pre-development, development and post-development.
- *Integral processes*: involve the processes required to successfully complete the project activities.

Table 6 analyses the compliance of each methodology with the different processes of the IEEE standard. Each process was rated based on the ratings in Table 7. Table 8 summarizes the results of this compliance analysis. For a certain rating r (r=1…n) and a methodology d, (d=1…m), we have

a number of processes $X_{r,d}$. Therefore, the present for ratings was calculated using the following formula:

$$P_r = \frac{X_{r,d}}{\sum_{r=1}^{n} X_{r,d}} \times 100 \quad (1)$$

Table 9 explains this analysis through depicting the coverage percentage per methodology. Fig. 5 presented data from Table 9 in graphical from. Based on the analysis, only Methontology propose performing scheduling, control and quality assurance. However, it does not propose how to carry out the project initiation. Installation, operation, support and retirement processes are missing in most methodologies. Some methodologies (such as UPON and Methontology) refer the need of carrying out some integral processes. However, these processes were not covered in detail.

In conclusion it can be said that none of the existing methodologies are fully mature if they are compared with the IEEE standard. Most of these methodologies focus on the development activities and they do not consider the aspects related to project management. AMOD supports not only the core activities of building ontologies, but also the project management and integral processes. The results show that AMOD has resulted in 56% satisfaction and achieved 17% partial satisfaction for the processes of the IEEE standard. This resembles 22% enhancement in the satisfaction against the other methodologies.

The experience form developing the SPTM ontology showed that adopting the agile principles and methods in the ontology development produced the following benefits: reducing the complexity of ontology development activities such as conceptualization, improving communication between ontology engineers and domain experts, the continuous assessment of the project status, keeping domain experts involved during the ontology development, focusing on the most important requirements and the ability to respond to changing knowledge rapidly. It also indicated that AMOD was easily followed during the ontology development.

Table 5. IEEE standard processes

Project management processes	P1.1	Project initiation
	P1.2	Monitoring and control
	P1.3	Quality management
Ontology development-oriented processes	P2.1	Environment study
	P2.2	Feasibility study
	P2.3	Requirements
	P2.4	Design
	P2.5	Implementation
	P2.6	Installation
	P2.7	Operation
	P2.8	Support
	P2.9	Maintenance
	P2.10	Retirement
Integral processes	P3.1	Knowledge acquisition
	P3.2	Evaluation
	P3.3	Configuration management
	P3.4	Documentation
	P3.5	Training

Table 7. Index of processes rating

Rating	**criteria**
Covered (C)	Means that IEEE process is fully covered by the methodology.
Partially Covered (P)	Means that some parts of IEE process is fully covered by the methodology.
Uncovered (U)	Means that IEEE process is not covered by the methodology.

Table 6. Compliance of each methodology with IEEE standard [18]

	P1.1	P1.2	P1.3	P2.1	P2.2	P2.3	P2.4	P2.5	P2.6	P2.7	P2.8	P2.9	P2.10	P3.1	P3.2	P3.3	P3.4	P3.5
Uschold and King [13]	U	U	U	U	U	P	U	C	U	U	U	U	U	C	C	U	C	U
Grüninger and Fox [14]	P	P	U	U	U	C	C	C	U	U	U	U	U	P	U	U	U	U
Methontology [15]	U	P	P	U	U	P	P	P	U	U	U	P	U	C	C	C	P	U
On-To-Knowledge [16]	C	C	U	U	C	C	P	C	U	U	U	P	U	U	C	U	U	U
UPON [18]	U	U	U	P	U	C	C	C	U	U	U	P	U	C	C	U	C	P
AMOD	C	C	P	P	P	C	C	C	U	U	U	C	U	C	C	C	C	U

Table 8. Summary of compliance analysis results

	Uschold and King [13]	Grüninger and Fox [14]	Methontology [15]	On-To-Knowledge [16]	UPON [18]	AMOD
Covered	4	3	3	6	6	10
Partially Covered	1	3	7	2	3	3
Uncovered	13	12	8	10	9	5

Table 9. Coverage percentage per methodology

	Uschold and King [13]	Grüninger and Fox [14]	Methontology [15]	On-To-Knowledge [16]	UPON [18]	AMOD
Covered	22%	17%	17%	33%	33%	56%
Partially Covered	6%	17%	39%	11%	17%	17%
Uncovered	72%	67%	44%	56%	50%	28%

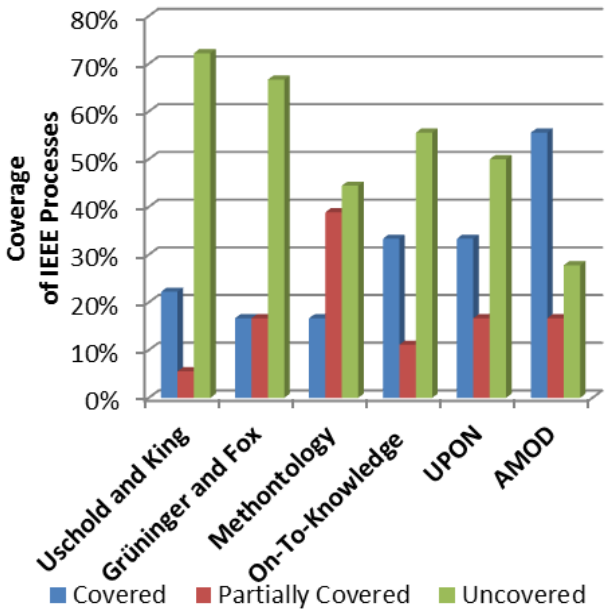

Figure. 5 Coverage percentage of each methodology

6. Conclusion and future work

In this paper, an ontology development methodology, AMOD, based on the agile software engineering methods was presented. The development of ontology is different from developing software, but the fundamental principles and activities are the same. Therefore, building ontologies should follow the standards propped of software that should be tailored to the special characteristics of ontologies. A comparative evaluation with existing ontology engineering methodologies based on the IEEE standard was conducted. Results showed that AMOD is more compliance with the IEEE standard than the other methodologies. AMOD resulted in achieving a better stratification with additional 23% coverage that is 41% development.

The strength of AMOD lies in its ability to be customized to fit a number of factors including ontology complexity, domain of interest, ontology size. The phases and activities of AMOD were applied to develop Software Project Time Management (SPTM) ontology. The experience from building SPTM ontology indicated its applicability and high degree of acceptance by

ontology engineers. It also showed that the use of agile methods simplifies the implementation of the ontology development activities. It is recommended as a future work to adopt AMOD in new applications to obtain additional validation cases.

References

[1] T. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", International *Journal of Human-Computer Studies*, Vol. 43, pp. 907-928, 1995.

[2] A. Abdelghany, N. Darwish, and H. Hefny, "Towards a Hybrid Approach for Software Project Management using Ontology Alignment", *International Journal of Computer Applications*, Vol. 168, No. 6, 2017.

[3] A. Khattak, R. Batool, and Z.Pervez, "Ontology Evolution and Challenges", *Journal of Information Science and Engineering*, Vol. 29, pp. 851-871, 2013.

[4] M. Suárez-Figueroa, R. García-Castro, B. Villazón-Terrazas, and A. Gómez-Pérez, "Essentials In Ontology Engineering: Methodologies, Languages, And Tools", In: *Proc. of the 2nd Workshop on eeBuildings Data Models*, pp. 6-28, 2011.

[5] M. Gawich, "A Methodology for Ontology Building", *International Journal of Computer Applications*, Vol. 56, No. 2, 2012.

[6] I. Al-Baltah and A. Abdul Ghani, "A Comparative Study on Ontology Development Methodologies towards Building Semantic Conflicts Detection Ontology for Heterogeneous Web Services", *Research Journal of Applied Sciences, Engineering and Technology*, Vol. 7, pp. 2674-2679, 2014.

[7] R. Iqbal, M. Murad, A. Mustapha, and N.Sharef, "An Analysis of Ontology Engineering Methodologies: A Literature Review", *Research Journal of Applied Sciences, Engineering and Technology*, Vol. 6, pp. 2993-3000, 2013.

[8] M. Copeland, A. Brown, H. Parkinson, R. Stevens, and J. Malone, "The SWO Project: A Case Study of Applying Agile Ontology Engineering Methods in Community Driven Ontologies", In: *Proc. of the 3rd International Conf. on Biomedical Ontology*, 2012.

[9] D. Giudice, "The 2015 State Of Agile Development", *Forrester Research*, 2105

[10] Agile Alliance, "Manifesto for Agile Software Development", Internet: http://agilemanifesto.org, [Feb. 1 2018].

[11] A. Farid, A. Abdelghany, and Y. Helmy, "Implementing Project Management Category Process Areas of CMMI Version 1.3 Using Scrum Practices, and Assets", *International Journal of Advanced Computer Science and Applications*, Vol. 7, No. 2, 2016.

[12] A. Mohamed, N. Darwish, and H. Hefny, "Towards a Machine Learning Model for Predicting Failure of Agile Software Projects", *International Journal of Computer Applications*, Vol. 168, pp. 20-26, 2017.

[13] M. Uschold and M. King, "Towards a Methodology for Building Ontologies", In: *IJCAI'95 Workshop Basic Ontological Issues in Knowledge Sharing*, pp. 6.1-6.10, 1995.

[14] M. Grüninger and M. Fox, "Methodology for the design and evaluation of ontologies", In: *IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing*, pp. 6.1-6.10, 1995.

[15] M. F. Lopez, A. Gomez-Perez, J. Sierra, and A. Sierra, "Building a Chemical Ontology Using Methontology and the Ontology Design Environment", *IEEE Intelligent Systems*, Vol. 4, pp. 37-46, 1999.

[16] S. Staab, R. Studer, H. Schnurr, and Y. Sure, "Knowledge Processes and Ontologies", *IEEE Intelligent Systems*, Vol. 16, pp. 26–34, 2001.

[17] P. Spyns, Y. Tang, and R. Meersman, "An ontology engineering methodology for DOGMA", *Applied Ontology*, Vol. 2, No. 3, pp. 13–39, 2008.

[18] A. Nicola, M. Missikoff, and R. Navigli, "A Software Engineering Approach to Ontology Building", *Information Systems*, Vol. 34, pp. 258-257, 2009.

[19] M. Suárez-Figueroa, A. Gómez-Pérez, and M. Fernández-López, "The NeOn Methodology for Ontology Engineering", *Ontology Engineering in a Networked World*, pp. 9-34, 2012.

[20] M. Annamalai and M. Rosli, "Software Engineering: A Pathway to Enterprise-Strength Ontology Engineering", *International Journal of Digital Content Technology and its Applications*, Vol. 6, No. 22, pp. 98-107, 2012.

[21] R. Falbo, "SABiO: Systematic Approach for Building Ontologies," In: *Proc. of the 1st Joint Workshop Onto.Com/ODISE Ontologies in Conceptual Modeling and Information Systems Engineering*, Vol. 1301, 2014.

[22] M. Rani, S. John, and N. Shah, "Proposal of A Hybrid Methodology for Ontology Development by Extending the Process Models of Software Engineering", *International Journal of Information Technology Convergence and Services*, Vol. 6, No. 1, 2016.

[23] A. De Nicola, and M. Missikoff, “A lightweight methodology for rapid ontology engineering”, *Communications of the ACM*, Vol. 53, No. 3, pp. 79-86, 2016.

[24] S. John, N. Shah, and L. Smalov, “Incremental and Iterative Agile Methodology (IIAM): Hybrid Approach for Ontology Design towards Semantic web Based Educational Systems Development”, *International Journal of Knowledge Engineering*, Vol. 2, No. 1, pp. 13-19, 2016.

[25] S. John, N. Shah, C. Stewart, and L. Samlov, “Software Centric Innovative Methodology for Ontology Development”, In: *Proc. of International Joint Conf. on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pp. 139-146, 2017.

[26] G. Brusa, M. Laura Caliusco, and O. Chiotti, "A Process for Building a Domain Ontology: an Experience in Developing a Government Budgetary Ontology", In *Proc. of the Second Australasian Workshop Advances in Ontologies*, Vol. 72, pp. 7-15, 2006.

[27] N. Rastogi, P. Verma, and P. Kumar, "Analyzing Ontology Editing Tools for Effective Semantic Information Retrieval", *International Journal of Engineering Sciences & Research Technology*, Nol. 6, No. 5, 2017.

[28] T. Slimani, "Ontology Development: A Comparing Study on Tools, Languages and Formalisms", *Indian Journal of Science and Technology*, Vol. 8, No. 24, 2013.

[29] J. Sutherland. (2010, Jul.), *Scrum Handbook*, Scrum Training Institute. [Online]. Available: http://www.knowledgehut.com/images/scrumhandbook.pdf

[30] A. Menolli, H. Pinto, S. Reinehr, and A. Malucelli, "An Incremental and Iterative Process for Ontology Building", In: *Proc. of the 6th Seminar Ontology Research in Brazil*, pp. 215-220, 2013.

[31] K. Schwaber, and J. Sutherland. (2011, Oct.), The Scrum Guide. Scrum.org. [Online], Available:http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf#zoom=100

[32] H. Pinto and J. Martins, "A methodology for Ontology Integration", In: *Proc. of the 1st International Conf. On Knowledge Capture*, pp. 131-138, 2001.

[33] H. Hlomani and D. Stacey, "Approaches, Methods, Metrics, Measures, and Subjectivity in Ontology Evaluation: A Survey", *Semantic Web Journal*, 2014.

[34] V. Jain and S. Prasad, "Evaluation and Validation Of Ontology Using Protégé Tool", *International Journal of Research in Engineering & Technology*, Vol. 4, pp. 21-32, 2016.

[35] H. Zhu, "Quality Model and Metrics of Ontology for Semantic Descriptions of Web Services", *Tsinghua Science and Technology*, Vol. 22, No. 3, 2017.

[36] D. Looser, H. Ma, and K. Schewe, "Using Formal Concept Analysis for Ontology Maintenance in Human Resource Recruitment", In: *Proc. of the Ninth Asia-Pacific Conf. on Conceptual Modelling*, pp. 61-68, 2013.

[37] D. Garijo, "WIDOCO: A Wizard for Documenting Ontologies", *The Semantic Web – ISWC 2017*, Vol. 10588, pp. 94-102, 2017.

[38] M. Blázquez, M. Fernández, J. García-Pinar, and A. Gómez-Pérez, "Building Ontologies at the Knowledge Level using the Ontology Design Environment", In: *Proc. of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems*, Vol. 2, 1998.

[39] A. Haider, "Basic Activities of Software configuration Management", *International Journal of Advancement in Engineering Technology, Management and Applied Science*, Vol. 1, 2014.

[40] M. Lopez and A. Perez, “Overview and Analysis of Methodologies for Building Ontologies”, *Knowledge Engineering Review*, Vol. 17, No. 217, pp. 129-156, 2002.