Delving into knowledge modeling software supporting collaborative ontology development

Reza Kalbasi[1], Saeid Amanzadeh[2], Ali Alesheikh[3]

Abstract – *The proficiency of ontologies as the formal explication of knowledge in different domains has motivated numerous software engineers to develop miscellaneous ontology editors. The recent propensity of ontology engineering community, collaborative ontology development, necessitates exclusive ontology platforms supporting collaborative aspects. Although a few research have strived to look over the collaborative ontology editors, they have not determined a complete list of requirements and benchmarks for their investigations. Therefore, collaborative ontology development requirements have been proposed in this research comprehending collaborative methodologies and platforms. Regarding to these touchstones, two existing categories of collaborative ontology editors have been explored here. The most discussed and applied instances of these platforms, wiki-based and non-wiki-based, have been enumerated in this paper too. The itemization, description, and evaluation of platforms can be fruitful for the ontology-based applications in future. The results of this research (in the context of an ongoing EU project) demonstrate that workflow support, reliability of software, and integration/representation/storing of changes are the prominent gaps of current collaborative ontology development platforms.*

Keywords: *software engineering, collaborative ontology development, wiki-based ontology editors, non-wiki-based ontology editors*

I. Introduction

Ontologies as the explicit specification of a conceptualization have been applied to resolve semantic heterogeneity [1]. The major advantages of ontologies are to reduce concept definition mistakes, supporting common understanding among communities, facilitating communication, and providing formal and detailed description of resources [2, 3, 4]. The process of extracting knowledge and constructing its structure, properties, and interrelationships formally is called ontology engineering which many paradigms have been proposed for this task. This development can be done in a centralized manner where mostly the ontology engineers make decisions about the ontology components. The main resources to extract knowledge are domain experts as well as other repositories such as books, encyclopedias, databases etc.

In contrast to centralized methodology, collaborative ontology development has been proposed to engage more domain experts and end-users of ontology-based application in this process. The ontology engineer's skills are familiarity with ontology editors, methodologies, reasoning and inference engines, and modeling [5]. The centralized methodology's deficiency is revealed when the ontology engineer cannot grasp the domain expert's knowledge well. It can have many reasons such as engineer's inaptitude or expert's inability. However, there is a need for consensus among the engineer and domain expert in this situation about the domain ontology components. This consensus indirectly is also dependent on the end-user of ontology-based application if the ontology-based application does not fulfill the end-user's need. One may ask that how the end-users and domain experts can find some faults among the ontology components while they don't have ontological experience and background. The answer for this question is that the conflicts can appear in the usage of the systems which ontology attempts to enhance their applicability. For example, an ontology-based knowledge-base can result in wrong conclusion or reasoning which is an unsatisfactory answer for a query.

Totally, the centralized ontology engineering has some considerable disadvantages which have been enumerated in previous research [6], [7], [8]: firstly, ontology evolution is not possible as the domain experts and end-users do not have authority to edit and evolve the components of ontology (engineering-oriented ontology building). Secondly, there is compulsion on the other roles to follow the engineer's perspectives (engineering-oriented ontology building). Thirdly, since there is no

Manuscript received January 2007, revised January 2007

communication among the engineers and domain experts/end-users, the latter groups cannot easily grasp the intention of the ontology and its components. And finally, in some domains the ontologies are not static as their corresponding knowledge is evolving and changing such as geosciences where the conceptual models of physical phenomena are altering temporally.

Most of the time, in the process of ontology engineering, ontology engineer consults with the experts who comprehend the whole territory of a domain as the best resource for knowledge acquisition. It would be the best case if the domain experts are ontology engineers themselves as there may be misinterpretation while consulting. Although being both of domain expert and ontology engineer rarely occurs in a person, it is possible to engage the domain experts in the process of ontology engineering somehow in a collaborative infrastructure.

The development of ontologies collaboratively has been emerged recently. Traditionally the collaboration was done by using email and other communication services which is not effective since it is not possible to track the discussion threads [9]. As a result, there is a need to develop the proper facilities for effective collaborative development.

The software which helps for the creation of ontologies is called ontology editor (platform). This type of software supports the ontology developers with user-friendly Graphical User Interface (GUI) for each type of ontology component. Although ontologies can be built by the ontological languages (text files) which are intended for computer processing, the ontology editors have evolved their environments to support more user-friendly graphical interface for human interaction. It is similar to the evolution of recent integrated development environments (IDE) [10] like Microsoft .Net Visual Studio to provide pre-developed graphical interfaces as the code components. Also they can support other aspects of ontology engineering which increase their capabilities in this context. These aspects can be services for ontology mapping/alignment, ontology merging, ontology visualization, reasoning engines, natural language processing, ontology validation, specific methodology support, and conversion techniques (from DB Schema, XML Schema, UML, and other modeling languages).

In the next section, the related works and previous research related to this paper will be discussed. The third section introduces the wiki-based collaborative ontology editors. In section four, the other type of platforms, non-wiki-based platforms, will be introduced. Section five, discussion, evaluates the capabilities of these platforms to guarantee collaborative ontology engineering. The basis for this evaluation is upon a number of requirements for collaborative ontology engineering. And the last section pertains to the made conclusion of this research besides some recommendations.

II. Related works

Although many collaborative ontology editors have been developed recently, only a few works have evaluated their capabilities. Noy and Chugh et al. have introduced and evaluated some platforms in the context of Collaborative Knowledge Construction (CKC) Challenges [11]. They have determined 12 features as the main parameters to evaluate these tools. The other work is by Lu [12] which is an M.Sc. research as a roadmap for developers of collaborative ontology editors. In his work, five different platforms were evaluated.

In 2002, Michael Denny [13] surveyed more than 50 ontology editors which some of them support collaborative ontology development. Currently, most of these tools are not active and have been expired. He has determined the main characteristics of ontology editors such as: versioning, release date, modeling features/limitations (the representational and logical qualities that can be expressed in the developed ontology), supporting the native or primary language used to encode the ontology, web support & use (support for web-compliant ontologies e.g. URIs and use of the software over the web such as browser client), importing/exporting formats, supporting graph view which is the extent to which the built ontology can be created, debugged, edited and/or compared directly in graphic form, consistency checking, referential and/or logical correctness of the ontology, multi-user support (features that allow and facilitate concurrent development of the developed ontology), merging and supporting for easily comparing and merging independent built ontologies, lexical support (capabilities for lexical referencing of ontology elements such as synonyms and processing lexical content like searching/filtering ontology terms), and information extraction.

Also in the other work by McGuinness and Patel-Schneider had investigated the usability of six knowledge representation tools [14]. Their usability parameters for knowledge representation tools have been determined such as learning time, error handling, system efficiency, and adequacy of programming interfaces.

Garcia Barriocanal et al [15] in their recent work have evaluated the usability of ontology editors regarding to the interactivity of naive users. In their research, the target users are the motivated people to learn ontology editor's functions while they had no any familiarity with these tools.

In our research, we have found some debates on these results. Firstly, they have claimed that the languages in this type of software should be oriented towards the non-specialized user community. This is not logical as the ontology language is related to the software processes and interactions, not to the user. Secondly, they recommended that richer navigation and filtering mechanisms should be developed according to the user

task model. For instance the user should be capable to navigate from a class to its instances. In our experience of ontology engineering, it is in contrast with some platforms like Protégé which had supported this service before. Hence we decided to explore the previous versions (published from 2003 to 2005) of Protégé [16] such as Protégé -2000, Version 1.8 with the release date of April 9, 2003. This showed that the previous versions of Protégé support this service too. Besides these outputs, the collaborative aspects of ontology engineering have been ignored in this test. The other related work is a continuous endeavor by a portal for publishing information on research and development related to the topics Semantic Web and Wikis [17]. In this portal, a brief description of each ontology editor and its developers has been explained. This can be a good resource to become aware of latest released versions of both collaborative and centralized ontology editors. In some of the previous works, only a brief description of ontology editors has been given. These works are mainly related to the development of new ontology editors such as [8], [18], [19], [20], [21], [22], [23], [24].

All in all two main categories of platforms support collaborative ontology development that they have been called wiki-based and non-wiki-based categories. In the next session the reason for this kind of denomination and their specifications will be mentioned. Also, the most discussed and applied platforms of these categories will be described.

III. Wiki-based platforms

Being one of the most discussed technologies of Web 2.0, wikis are the web portals where the users can edit and modify the contents of them. This complies with the roadmap of web 2.0 which engage the users to share ideas and talk about the contents of web pages. As a metaphor, wiki is a graph which its each node is a page about a concept with unique identifier (URL[1]) and each edge is a hyperlink between two pages. The most famous wiki that becomes the biggest knowledge base of the world is Wikipedia[2] with more three million pages (concepts). Schaffert [25] has listed a number of common characteristics for the wikis such as being user-friendly editable interfaces via browsers, holding simplified wiki syntax (the contents of wikis are simplified HTML), supporting rollback mechanism as the previous versions can include important information related to the current version, supporting collaborative editing by the contribution of multiple clients to do a task, holding strong linking through the hyperlinks, supporting search function throughout the wiki pages, and uploading of other contents like multimedia and codes.

Recently the user-friendly interface of wikis has motivated many domains to apply them for their tasks. One of these tasks is ontology engineering where the wikis have become as the ontology editors. It is interesting that wikis have applied semantic web technologies to enhance their applicability too. So the union of semantic web and wikis has resulted in two different state-of-the-art platforms. In the first type, aka semantic web for wikis, wikis have evolved their accomplishments such as query on wiki contents by using semantic queries which can return more related results [20]. In the second type, aka wikis for ontologies, wikis have been used as the ontology editors which have more user-friendly interface and support collaborative development. Only the latter one is related to this research (ontology engineering). However, in the remained part of this section, the instances of both types will be described.

III.1. Semantic Web For Wikis platforms

MAKNA [8], [20], [21], [22], [23], [24], [25], [26]: An AJAX-based[3] system which extends the JSPWiki[4] with generic and easy to use ontology-based components for collaborative authoring, querying by using JENA reasoning engine, and browsing the semantic web information. The main objective to develop Makna is to extend the wikis (RDF-based) while preserving their conventional characteristics. It should have an easy-to-use procedure for concept creation and semantic data management, and versatile use of semantic technologies (the flexible semantic wikis which use ontologies on the web seamlessly). The evaluation of MAKNA[5] by Krotzsch et al [22] demonstrates that although it supports more expressive ontologies, it is only good for the closed domains and the cases which only administrators have rights to change the ontology. As the designers claim that: ‘‘our system is not targeted at collaborative ontology engineering yet’’, this is considered in the semantic web for wikis category.

Platypus [8], [19], [20], [21], [22], [23], [24], [25], [26], [27]: As a rapid and useful personal knowledge management system (designed in JAVA) supports a simple user interface to create the wiki pages whose metadata is structured in RDF, RDFS or OWL.

Semantic MediaWiki [28]: as a free extension of MediaWiki, semantic MediaWiki engine enriches wikis with semantic annotations. By using semantic MediaWiki we can search, organize, tag, browse, evaluate, and share the wiki’s content.

III.2. Wikis For Ontologies

[1] Uniform Resource Locator
[2] www.wikipedia.org

[3] Asynchronous JavaScript and XML
[4] 6A feature-rich and extensible WikiWiki engine built around the standard J2EE components (Java, servlets, and JSP); http://www.apps.ag-nbi.de/makna/
[5] Standing for ‘‘knowledge’’ in Indonesian

MyOntology [8], [29], [30]: in this project, the developers have proposed a wiki-based system to facilitate users to create and edit ontologies without much expertise in ontology engineering. In the following their reasons to apply collaborative ontology engineering are indicated: creating more up-to-date ontologies, existing more commitment about created ontology because of user participation, no misconception for the users (because of community grounding), less expressive ontology (to have a bigger user community), not able to add more axioms by the users (to have a bigger user community), and more inconsistencies that can be tracked by the users themselves.

OntoWiki [18], [21], [31]: as an AJAX-based semantic wiki, the main specifications of OntoWiki are: intuitive display and editing for instance data both generally and specific, semantic views (different views and aggregations of the knowledge base), versioning and evolution by keeping the track of changes, semantic search and filtering, community support via discussion and vote/rating, online statistics by measuring the popularity of contents and user activities, and Semantic syndication by distributing information and its integration into desktop applications. Four main parts can be considered in this platform such as knowledge bases, classes (to browse the knowledge base classes), instances of classes, and properties. These parts facilitate the accessibility of collaborators to ontology elements.

IkeWiki [22], [32], [33]: As a semantic wiki for collaborative knowledge management and ontology engineering (among non-technical domain experts and knowledge engineers), IkeWiki[6] uses Java-based semantic web framework JENA to become one of the popular editors in this category. The main characteristics of this platform are: easy to use, interactive interface applying technologies such as AJAX (as domain expert are non-technical people), compatibility with Wikipedia by importing existing contents, compatibility with semantic web standards, immediate exploitation of annotations, supporting different levels of experience, supporting different levels of formalization, and supporting reasoning (unlike the other semantic wikis, the default installation uses OWL-RDFS reasoning). The main components of IkeWiki are: Postgres DB, Jena RDF framework to represent the knowledge base and using SPARQL engine to make semantic queries, page store to store and retrieve data from database as page contents and supporting versioning of data, rendering pipeline to combine page contents and semantic annotations to generateWIF[7] [25] document enriched by relevant semantic annotations, the WIKlets helping for representation of added information only for some users, supporting transformation for XSLT (eXensible Style Language Translation), IkeWiki Servlet to serve the outputs of transformation layer for the end-user browser, and supporting editing by using AJAX technology.

IV. Non-wiki-based platforms

These platforms are the common type of ontology editors which have been applied from the first days of knowledge representation emergence. The building and the maintenance of knowledge structures (e.g. ontologies) are supported by ontology editors through the GUIs [15]. The evolution of the ontology editors has to happen on several axes such as single-user applications to multi-user, thick client to thin clients (web-based), and single-user control of ontology content to multiple user content control including the simultaneous access problems, discussions, annotations, conflict resolution and so on [34]. In below, some of the most discussed and applied non-wiki-based collaborative ontology editors will be described.

Tadzebao system [35]: As the developers may be dispersed in different time zones, asynchronous discussion as well as synchronous discussion is supported in this system. Also, it integrates platforms for the text editing and drawing the ontology. For the sake of collaboration and dialogue between users, a notepad interface supports both texts and images to be exchanged among the users.

WebOnto system [35], [36]: This Java-based system is composed of a central server and clients designed to overcome the problems and drawbacks of HTML user interface. In fact, it is the complementary of Tadzebao for discussions while the goal of this system is to support collaborative browsing, creation, and editing of ontologies through an easy way and supporting large ontologies.

OntoEdit [23]: by considering the ontology development process, inference mechanisms, and consensual achievements, it supports collaborative ontology engineering through three phases such as requirements specification, refinement, and evaluation. So this software is based on a methodology for collaborative development of ontology. The phases of this methodology are described here to show the rationale behind this software's design.

The first phase, requirement specification, has been originated from the software development community. This includes two detailed tasks:

1. From scratch, the ontology engineers and domain experts interact together to describe the domain of discourse and ontology goals (via the competency question interface).

2. Then they should reach to a semi formal ontology

[6] In KIWI project (http://www.kiwi-project.eu/); The KiWi - "Knowledge in a Wiki" project proposes a new approach to knowledge management that combines the wiki philosophy with the intelligence and methods of the Semantic Web.

[7] ''Basic WIF merely describes the page content and structure, but allows to add custom, application specific information in separate namespaces''

description as the output of this part (graph of named nodes and (un-)named, (un-)directed edges). Two plugins have been applied for the sake of these issues such as OntoKick and Mind2Onto. OntoKick helps for describing the ontology meta aspects (the mentioned requirement specifications outputs) and supports the creation of a semi formal ontology description. Mind2Onto supports brainstorming and discussion about ontology structures via mind maps[8]. Although mind maps do not consider the semantics of knowledge completely, they can represent the association of different concepts and their structures properly.

The second phase is the refinement where the semi formal description of the ontology (first phase's output) should become a mature ontology in several iterations and formalizations. One of the capabilities of OntoEdit is the transaction management (the least concurrency and more consistency) in this phase which is done by a locking and transaction protocol and implementing a distributed event model upon the Java RMI[9].

And the last phase is the evaluation of formal ontology corresponding to the requirements specification in the first phase. This is done by the analysis of typical queries by an instance editor and an axiom editor, error avoidance and location (the localization of error), usage of competency questions as they provide a checklist which ontology must answer all of them through a collaborative evaluation interface.

COE [4]: as a peer-to-peer collaborative ontology editor, its developers believe that collaborative ontology building through the client-server (C/S) architecture takes the advantage of central data repositories, establishing user ID and authentication, maintaining standard ontologies, and keeping the overall state consistent. In contrast to these advantages, they also believe in their cons such as difficulties in synchronous communications (as data is stored only in server), interface problems, and taking too much time for the responses. In contrast, in P2P architecture the knowledge is divided into different knowledge bases and distributed repositories (instead of a centralized server) which resemble the human communities better than the C/S architecture and has greater tolerance to failures. It seems that the P2P architecture is more suitable for the cases that each of ontology collaborators is responsible for a specific and exact portion of knowledge corresponding to its rules and criteria (e.g. different organizations separated geographically).

Related to its specific characteristics, firstly it does not support multiple-inheritance[10], because the developers believe that the graph with multiple-inheritance is difficult to perceive for the users. Secondly discussions in COE is done by chat and file exchange after user registration on the group of peers.

Ontolingua Server [37], [38]: As a famous platform developed by Stanford Knowledge Systems Lab, it has been implemented in client-server architecture to develop ontologies from its repository collaboratively. The main specifications of Ontolingua Server are: a semi formal representation language, possibility for browsing and retrieval of ontologies from the repositories, possibility for assembly/customization/extension of ontologies from the repositories, facilities for translating ontologies from repositories into typical application environments, facilities for programmatic access to ontologies so that remote applications have reliable access to the up-to-date term definitions, support for distributed and collaborative development of consensus ontologies, and finally concurrent ontology development via locking mechanism (and sending alert while changing in ontology).

Ontoverse [39], [40]: In general the main characteristics of this system are: browser-based, enough graphical interfaces for ontology engineering and maintenance, copyright protection of ontology elements, keeping large amounts of data, concurrent access, supporting community awareness features, conflict resolution, user management, special visualization, and independent modules for more flexibility and adaptability of system to the current requirements. Its advanced visualization technique is called SmartTree to help the inexperienced users (especially for the large ontologies). E.g. one can zoom in the interested and related concepts for the user concentration.

WebODE [6], [41], [42]: The WebODE developers (from polytechnic university of Madrid) motivations to create this platform are some deficiencies in the other ontology editors such as no correspondence between ontology environments and methodologies, not considering all the activities of the ontology life cycle, and no interoperability among ontology development platforms. To cope with these problems, WebODE workbench has covered three main components:

1. Ontology development activities like knowledge acquisition, edition, browsing, integration, merging, reengineering, evaluation, and implementation and management activities like configuration management and ontology evolution.

2. Ontology middleware services through the well-defined service-oriented Application Programming Interfaces (APIs) which help for accessing ontologies, integration with databases, ontology upgrading, and query services.

[8] The tree of knowledge integrated with ontologies for better interaction between ontologists and domain experts.

[9] The Java Remote Method Invocation (RMI) system allows an object running in one Java virtual machine to invoke methods on an object running in another Java virtual machine. RMI provides for remote communication between programs written in the Java programming language (Sun Microsystems; http://java.sun.com/docs/books/tutorial/rmi/)

[10] A feature of some object-oriented programming languages that allows a new class to be derived from several existing classes. Multiple inheritance both extends and combines existing types. Microsoft dictionary

3. Ontology-based application development suites as the last step towards a real integration of ontologies into the enterprise information systems.

And finally, its collaborative capabilities are provided by both form-based and graphical user interfaces. The main facilities for the collaborative ontology engineering in WebODE are inconsistency checker, an inference engine, an axiom builder, the documentation service, visualization of ontology corresponding to the user customization, and synchronization.

Protégé [24], [34], [43]: Protégé is one of the most popular ontology editors among the ontologists. It is an open source platform which helps for the creation of knowledge bases and ontologies. Its specific characteristic is the capability to be extended and customized via accessible Java-based API. Protégé supports the development of both frame-based (upon Open Knowledge Base Connectivity protocol) and OWL (W3C's Web Ontology Language) ontologies. Its plugin library supports many services for ontology engineering such as collaborative ontology engineering. In below the related plugins to this research will be explained.

Client Server Protégé/ The client-server, aka multi-user, Protégé is a mode in this software which ontology (hosted on the server) can be edited by multiple clients simultaneously (via RMI API). This is in contrast with the stand-alone (consecutive) mode where the simultaneous access to the ontology is impossible. The advantage of multiuser mode is the possibility to define project policies and server administration. An implemented example of these policies can be reviewed in [5].

Collaborative Protégé/ This extension of Protégé is developed based on two modes: client-server and stand-alone. Corresponding to the efforts and ideas of its developers in [24], [34], [43], it supports: annotation of the ontology elements, annotation of the ontology changes such as class creation, deletion, and renaming, extending the annotation types individually, change proposals and voting of proposals, filtering of existing annotations, annotation searching based on simple or complex criteria, discussion threads as the users can reply and make comments about the other person's discussions, rating and voting for proposals, to reuse the ontology components by the other developers, synchronous and asynchronous discussions, and capturing information about the performed action and storing it in the knowledge base (to notify other users about every change). The heart of collaborative Protégé is Changes and Annotations Ontology (ChAO) knowledge base. The instances of changes and annotations (corresponding to each ontology entity) are stored as ChAO ontology individuals.

All in all, the collaborative ontology engineering is supported by four different Java APIs such as Changes API, Annotations API, Ontology Components API, and Ontology API. The provided methods by these APIs are illustrated in Figure 1:

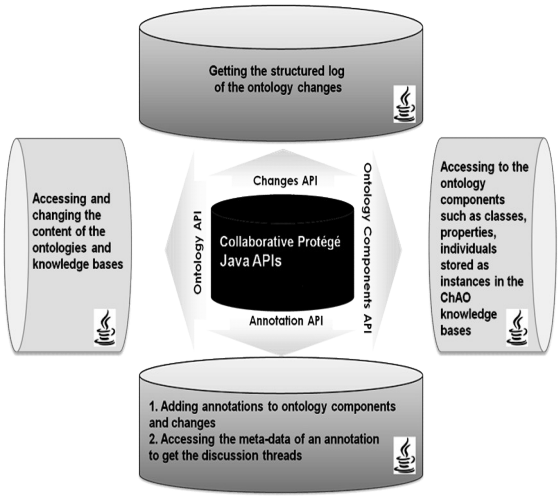

Fig. 1. Provided methods by Java APIs in Protégé [5]

V. Discussion

In the previous section multiple ontology editors (wiki-based and non-wiki-based) were introduced. These platforms have been evaluated by different criteria in previous works. For instance in [11], the parameters to compare these category of software are hierarchy of concepts, properties, instances of concepts in the hierarchy (includes tags), comments on ontology components, asynchronous editing of ontology modules, discussion, ratings, personal space, web browser interface, creation of content through an integrated browser button, and instant Chat. Nevertheless the rationale behind the selection of these features was not determined in this work. These features have been determined only because of their existence in some of the ontology editors. For example, only SOBOLEO and Collaborative Protégé support the facilities for chat and discussion threads [11].

One of the main targets of our previous works such as [5] is to determine the touchstones for both collaborative ontology engineering methodologies and platforms. These touchstones (aka collaborative ontology development requirements) are: 1) integration, representation, and storing of discussions and annotations and changes in ontology development, 2) ontology expressivity and visualization, 3) continuous editing, 4) periodic archiving of different versions (versioning), 5) determination of jury, 6) support for evaluation such as semantic/syntactic inconsistency checking, 7) provenance of information, 8) scalability (both in the size of ontology and the number of collaborators), 9) reliability, 10) robustness, 11) the methods to reach to the consensus when different ideas exist, 12) access control, 13) workflow support, 14)

supporting synchronous collaboration on ontologies, 15) and supporting asynchronous collaboration on ontologies. Among these requirements, some of them have not been considered for the platforms (being exclusive for methodology) such as the determination of jury and the consensus methods. Overlooking these parameters, 13 parameters will be remained. For the sake of this research some of the others such as continuous/periodic editing and robustness are also overlooked for testing. This is because of their less importance corresponding to the other parameters of collaborative ontology development. For instance, the developers can store different versions manually if there is no versioning support in the editor.

TABLE I
EVALUATION OF ONTOLOGY EDITORS

Platform	Inconsistency checking	Workflow Support	Expressivity and Visualization	Synchronous/ Asynchronous	Access control	Reliability	Robustness	Provenance	Integration and Representation and storing of changes, annotations, etc
MyOntology	No (manually by human)	No	No	Yes	Yes	No	Yes	Yes	Not changes
OntoWiki	No	No	No	Yes	Yes	Yes	Yes	Yes	Not changes
IkeWiki	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Not changes
Tadzebao	Yes (just for OCML code)	No	Yes	Yes	Yes	No	Yes	Yes	Not changes
WebOnto	Yes (just for OCML code)	No	Yes	Yes	Yes	No	Yes	Yes	Not changes
OntoEdit	Yes	Ye s	Yes	Yes	Yes	No	Yes	Yes	Not changes
COE	No	No	No	Yes	Yes	Yes	Yes	Yes	Not changes
Ontolingua Server	Yes	No	Yes	Yes	Yes	Yes but difficult to access	Yes	Yes	Not changes
Ontoverse	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
WebODE	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Not changes
Protégé	Yes	No	Yes	Yes and also Chat facility	Yes	Yes	Yes	Yes	Yes

Also the scalability can only be tested for the big ontologies that it was not possible for the case study of this research. This case study was ForeStClim[11] as an EU funded project. To select an ontology editor for this research, the enumerated platforms in Table 1 were evaluated. Regarding to the results of this evaluation, some of the parameters are essential to be considered by these platforms such as workflow-support (except OntoEdit), reliability (as a big challenge for software engineering), and integration/representation/storing of changes. However each ontology-based application can apply the evaluated tools corresponding to its needs like this research's case study which applied Protégé.

In below, the specific characteristics of some of these platforms will be discussed.

MyOntology- Corresponding to the developers of this system, the current version cannot work reliably to preserve data.

OntoWiki- Due to its weakness in reasoning, there is a need to examine the possibilities to integrate the Description Logic reasoning services into OntoWiki.

Tadzebao- The deficiency of this system is its HTML interface design which poses to the following problems: centralization of all data in the server and not having direct data manipulation that interface designers have to use repeated control widgets, one-shot connections and having the problem of redisplay time and the user remembering of the last page for more than one object.

COE- Its developers believe that they would have used Protégé as their ontology editor if it were supported by the P2P architecture. This shows the popularity of Protégé among ontology engineers community.

Ontoverse- The SmartTree of Ontoverse is its special feature helping collaborative ontology engineering.

WebODE- The developers of WebODE believe that the usage of plugins in Protégé is the main instigation for WebODE to apply built-in services which enhance the capabilities of these systems. However, this system is not

[11] ForeStClim is an EU-funded environmental project addressing forests and climate change. The short name stands for ''Transnational Forestry Management Strategies in Response to Regional Climate Change Impacts'';http://www.forestclim.eu/

up-to-date as the last update of this system is 03-Dec-2003.

Protégé- Firstly, the access control has some problems in this software. The users should not have capability to change the other participants' comments. This is a bug in its current version 3.4.1. Secondly, the possibility to define a workflow in this software should be supported while not being a predefined workflow. Thirdly, for the visualization of ontology two plugins have been discussed more such as OWLViz and Jambalaya. Both plugins are not mature enough to represent ontology elements. OWLViz only shows the structure of ontology (superclass and subclass relationships). And the deficiency of Jambalaya is its messy related edges of graphs when it represents the big and complicated ontologies. In contrast to this characteristic, Nested View of this plugin is more effective for the user perception of ontology.

VI. Conclusion and Recommendation

For the sake collaborative ontology development, the ontology editor should be compatible with the collaborative aspects. Although numerous software support the collaborative ontology development, each one considers some specifications such as discussion tab etc. These criteria and parameters are miscellaneous among these platforms. In our paper a complete list of parameters to support collaborative ontology development is proposed (for both methodologies and platforms). These parameters (aka collaborative ontology development requirements) helped the evaluation of ontology editors. It was determined that workflow support, reliability of software, and integration/ representation/ storing of changes are the gaps in most of the current collaborative ontology editors. This evaluation was performed throughout a geospatial ontology-based application in the context of an EU project. The participation of domain experts and ontology engineers demonstrated that discussions/annotations/changes, ontology visualization, inconsistency checking by reasoning engines, and access control aspects are motivating for their participation. So the user-friendly GUIs of these features have to be taken into account when software engineers develop this type of software.

The existing collaborative ontology editors are in two main categories such as wiki-based and non-wiki-based platforms. The evaluation and specification of wiki-based platforms (related to their developers' idea like [22]) demonstrated their capability and usage mostly for the lightweight ontologies. Also some of these platforms have missed the simplicity of wikis (their exclusive characteristic upon web 2.0) by integrating some of ontology elements with wikis such as object properties and data type properties among the pages of wiki. Regarding to the other category, non-wiki-based, they can support the development of heavyweight ontologies (like OWL DL). This characteristic can result in their support for automatic inference by reasoning engines.

However the browser-based characteristic of wikis is an advantage for them to support collaborative ontology engineering throughout the web environment. Although this characteristic is not considered as the collaborative ontology development requirements, recent non-wiki-based platforms have been inclined to their web-based versions such as Web Protégé [44]. For the future works, it is recommended to investigate other collaborative ontology development requirements such as web support. Also these platforms can be evaluated in detail only upon one of the mentioned requirements. For example, visualization of ontology as numerous techniques exists in this context like 3D graphs and concept clouds. The mentioned gaps of current platforms can be useful topics for the enhancement of upcoming software versions.

Acknowledgements

This work was supported by a European Commission project, ForeStClim which is an environmental project addressing forests and climate change. We really appreciate ONF and SERTIT organizations in France who supported us during the implementation of our research.

References

[1] T. R. Gruber, Towards principles for the design of ontologies used for knowledge sharing, *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers, 1993.

[2] G. Xexeo, A. Vivacqua, J. Souza, B. Braga, J. DAlmeida, B. K. Almentero, R. Castilho, B. Miranda. Coe: A collaborative ontology editor based on a peer-to-peer framework. *Advanced Engineering Informatics*, 2005.

[3] L. V. Stoimenov, E. O. Kajan, S. *Djordjevic-Kajan. Ontology-Driven Semantic Interoperability.* International Review on Computers and Software Journal (IRECOS). Vol. 1. n. 2, pp. 132 – 136, 2006.

[4] A. Behaz, M. Djoudi. *Ontological Representation Models for Adaptive Hypermedia Construction.* International Review on Computers and Software Journal (IRECOS). Vol. 6 N. 2, pp. 199-205, 2011.

[5] R. Kalbasi. *Collaborative ontology development for geo-information sharing,* Master's thesis, ITC faculty, University of Twente, 2009.

[6] S. Karapiperis, D. Apostolou. Consensus building in collaborative ontology engineering processes, *Journal of Universal Knowledge Management*, Vol. 1, pp. 199-216, 2006.

[7] F. Reitsma, J. Laxton, S. Ballard, W. Kuhn, A. Abdelmoty, Semantics, ontologies and escience for the geosciences, *Computers and Geosciences*, Vol. 35, pp. 706-709, 2009.

[8] K. Siorpaes, M. Hepp, myontology: *The marriage of ontology engineering and collective intelligence,* In Proceedings of Bridging the Gap between Semantic Web and Web 2.0 (Pages: 127-138, Year of Publication: 2007)

[9] A. M. MacEachren, W. Pike, Ch. Yu, I. Brewer, M. Gahegan, S. D. Weaver, B. Yarnal, Building a geocollaboratory: Supporting human-environment regional observatory (hero) collaborative science activities, *Computers, Environment and Urban Systems*, Vol: 30(2), pp. 201-225, 2006.

[10] Microsoft-Press. Microsoft Computer Dictionary, Fifth Edition. Microsoft Press, 2002.

[11] N. F. Noy, A. Chugh, H. Alani, The ckc challenge: Exploring tools for collaborative knowledge construction, *IEEE Intelligent Systems*, Vol. 23(1), pp.64--68, 2007.

[12] Y. Lu, Roadmap for tool support for collaborative ontology engineering, Master thesis, Department of Computer Science, XiAn Transportation University, 2003.

[13] M. Denny, Ontology building: Ontology Building: A Survey of Editing Tools. Website: http://www.xml.com/lpt/a/1061 ; accessed at April. 2012, 2002.

[14] D. L. McGuinness, P. F. Patel-Schneider, *Usability issues in knowledge representation systems*, In Proceedings of the fifteenth national conference on Artificial intelligence/Innovative applications of artificial intelligence (1998).

[15] E. Garcia-Barriocanal, M. Sicilia, S. Sanchez-Alonso. Usability evaluation of ontology editors. *Knowledge Creation Diffusion Utilization*, Vol. 32, pp. 1-15, 2005.

[16] Website: http://protege.cim3.net/download/old-releases/, accessed at April 2012.

[17] Website: http://semanticweb.org/wiki/Main_Page; accessed at April 2012.

[18] S. Auer, S. Dietzold, T. Riechert, *Ontowiki - a tool for social, semantic collaboration*, In Proceedings of the 5th International Semantic Web Conference- ISWC (Volume: 4273, Pages: 736-749, 2006).

[19] S. E. Campanini, P. Castagna, R. Tazzoli, *Platypus wiki: a semantic wiki wiki web*, In Proceedings of the 1st Italian Semantic Web Workshop Semantic Web Applications and Perspectives (Ancona, Italy, 2004).

[20] K. Dello, E. Simperl, R. Tolksdorf, *Creating and using semantic web information with makna,* In Proceedings of the First Workshop on Semantic Wikis - From Wiki To Semantics (June 2006) .

[21] M. Hepp, D. Bachlechner, K. Siorpaes, *Ontowiki: community-driven ontology engineering and ontology usage based on wikis*, In Proceedings of the 2006 international symposium on Wikis (Page 143-144, New York, USA, 2006).

[22] M. Krötzsch, S. Schaffert, D. Vrandecic, *Reasoning in semantic wikis*, In Proceeding of the Reasoning Web (Volume 4636 of Lecture Notes in Computer Science, Pages 310-329, Springer Berlin/Heidelberg, 2007).

[23] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, D. Wenke, *Ontoedit: Collaborative ontology development for the semantic web,* In Proceedings of the Semantic Web ISWC (Volume: 2342 of Lecture Notes in Computer Science, Pages: 221-235, Springer Berlin/Heidelberg, 2002).

[24] T. Tudorache, N. F. Noy, S. Tu, M. Musen, *Supporting collaborative ontology development in Protégé,* In Proceedings of the Semantic Web ISWC (Volume: 5318 of Lecture Notes in Computer Science, Pages: 17-32, Springer Berlin/Heidelberg, 2008).

[25] S. Schaffert, *Ikewiki: A semantic wiki for collaborative knowledge management*, In the Proceedings of the 1st International Workshop on Semantic Technologies in Collaborative Applications (IEEE Computer Society Washington, DC, USA, 2006).

[26] Website: http://makna.ag-nbi.de; accessed at April 2012.

[27] Website: http://platypuswiki.sourceforge.net/; accessed at April 2012.

[28] Website: http://semantic-mediawiki.org/; accessed at April 2012.

[29] Website: http://www.myontology.org/; accessed at April 2012.

[30] K. Siorpaes, *Towards open ontology engineering*, In Proceedings of the Knowledge Web PhD Symposium (KWEPSY)- co-located with the 4th Annual European Semantic Web (Volume: 275, Pages: 77-79, Innsbruck, Austria, 2007).

[31] Website: http://ontowiki.net/Projects/OntoWiki; accessed at April 2012.

[32] S. Schaffert, R. Westenthaler, A. Gruber, *Ikewiki: A userfriendly semantic wiki*, In Proceedings of the 3rd European Semantic Web Conference - ESWC (Budva, Montenegro, 2006).

[33] Website: http://ikewiki.salzburgresearch.at/; accessed at April 2012.

[34] Tania Tudorache and Natasha Noy. Collaborative Protégé . In 16th International World Wide Web Conference (WWW2007), Banff, Alberta, Canada, 2007.

[35] J. Domingue, *Tadzebao and webonto: Discussing, browsing, editing ontologies on the web*, In Proceedings of the 11th Knowledge Acquisition for Knowledge-Based Systems Workshop (1998).

[36] Website: http://projects.kmi.open.ac.uk/webonto/; accessed at April 2012.

[37] A. Farquhar, R. Fikes, J. Rice, The ontolingua server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, Vol. 46, 1997.

[38] Website: http://ksl.stanford.edu/software/ontolingua/; accessed at April 2012.

[39] N. Malzahn, S. Weinbrenner, P. Husken, J. Ziegler, and H.U. Hoppe, *Collaborative ontology development-distributed architecture and visualization*, In Proceedings of German e-Science Conference (Baden-Baden , Germany, 2007).

[40] Website: http://www.ontoverse.org/; accessed at April 2012.

[41] J. C. Arpirez, O. Corcho, M. Fernandez-Lopez, A. Gomez Perez. *Webode: a scalable workbench for ontological engineering*, In Proceedings of the First International Conference on Knowledge Capture (Pages: 21-23, ACM Press, 2001).

[42] Website: http://webode.dia.fi.upm.es/WebODEWeb/index.html; accessed at April 2012.

[43] A. Diaz, G. Baldo, G. Canals, *Coprotege: Collaborative ontology building with divergences*, In Proceedings of the 17th International Conference on Database and Expert Systems Applications (Pages: 156-160, IEEE, Krakow, Poland, 2006).

[44] Website: http://protegewiki.stanford.edu/wiki/WebProtege; accessed at April 2012.

AUTHORS' INFORMATION

The photographs, names, the vitae, the affiliation and the research interests of the authors should be given at the end of the paper.

The photo must be 2.45 cm x 2.45 cm. The text (8 pt)

wrapping style must be around the frame.

[1] Reza Kalbasi, University of Twente, Faculty of Geo-Information Science and Earth Observation (ITC); Email: kalbasi21752@itc.nl
[2] Luc Boerboom, University of Twente, Faculty of Geo-Information Science and Earth Observation (ITC); Email: boerboom@itc.nl
[3] Ali Alesheikh, Khaje Nasir Toosi University of Technology, Department of GIS Engineering; Email: alesheikh@kntu.ac.ir
[4] Saeid Amanzadeh, The University of Tabarestan, Department of Computer Science; Email: saeid.amanzadeh@gmail.com

Reza Kalbasi (Iran/1983) holds an MSc degree in Geo-informatics from University of Twente, Enschede city, the Netherlands. His MSc degree was granted with DISTINCTION honor in 2009. His research interests are semantic web, web 2.0, and geosemantics.
Currently, he is investigating the synergy of semantic web technologies through collaborative infrastructures of web 2.0 research area.

Saeid Amanzadeh (Iran/1985) received his BS degrees in Computer Software from University of Tabarestan, Chalos city, Iran.
His interests are Programming Languages semantic web, web technologies, and Data Mining. He is an active member of Iranian J2EE Society. He has published a number of scientific and research papers in computer science.

Ali Alesheikh received his BS degrees in Surveying Engineering from KNT University of Technology, Iran, his MEng degree in Geomatics Engineering from Uiversity of New Brunswick, Canada, and his PhD in Geospatial Information System (GIS) Engineering from Calgary University, Alberta, Canada in 1997. His interests include GIS, semantic web, and geospatial knowledge engineering.
He has published a numerous scientific and research papers in GIScience. Currently, he investigates on state-of-the-art geospatial technologies and services through spatial cyber-infrastructures.
Dr. Alesheikh has been the initiator and director of national and international level projects in the context of GIS and Iran's National ICT Development Plan.