The Knowledge Engineering Review, Vol. 29:1, 101–131. © Cambridge University Press, 2013
doi:10.1017/S0269888913000192
First published online 3 May 2013

Collaborative ontology engineering: a survey

ELENA SIMPERL[1] and MARKUS LUCZAK-RÖSCH[2]

[1]*Web and Internet Science, University of Southampton, UK, Highfield Campus, Building 32, SO17 1BJ, Southampton, UK; e-mail: e.simperl@soton.ac.uk;*
[2]*Networked Information Systems Workgroup (AG NBI), Free University of Berlin, Königin-Luise-Straße 24-26, 14195 Berlin, Germany;*
e-mail: markus.luczak-roesch@fu-berlin.de

Abstract

Building ontologies in a collaborative and increasingly community-driven fashion has become a central paradigm of modern ontology engineering. This understanding of ontologies and ontology engineering processes is the result of intensive theoretical and empirical research within the Semantic Web community, supported by technology developments such as Web 2.0. Over 6 years after the publication of the first methodology for collaborative ontology engineering, it is generally acknowledged that, in order to be useful, but also economically feasible, ontologies should be developed and maintained in a community-driven manner, with the help of fully-fledged environments providing dedicated support for collaboration and user participation. Wikis, and similar communication and collaboration platforms enabling ontology stakeholders to exchange ideas and discuss modeling decisions are probably the most important technological components of such environments. In addition, process-driven methodologies assist the ontology engineering team throughout the ontology life cycle, and provide empirically grounded best practices and guidelines for optimizing ontology development results in real-world projects. The goal of this article is to analyze the state of the art in the field of collaborative ontology engineering. We will survey several of the most outstanding methodologies, methods and techniques that have emerged in the last years, and present the most popular development environments, which can be utilized to carry out, or facilitate specific activities within the methodologies. A discussion of the open issues identified concludes the survey and provides a roadmap for future research and development in this lively and promising field.

1 Introduction

Several decades ago, ontologies were introduced to information and communication technologies (ICT) as a novel means to represent the kinds of things that can be talked about in a system in a formal and explicit manner. Used in information retrieval, information extraction, as well as data and process integration (Fensel, 2001), ontologies provide reusable pieces of declarative knowledge, which can be—together with problem-solving methods and reasoning functionality assembled into high-quality technology and application systems in an economical fashion (Neches *et al.*, 1991; Guarino, 1998). The Semantic Web is one of the most important application areas of ontologies. Initially introduced by Tim Berners Lee (Berners-Lee *et al.*, 2001), the originator of the World Wide Web, the idea of extending the current Web into a computer-processable knowledge infrastructure in addition to its actual, semi-structured and human-understandable content foresees the usage of knowledge components, which can be easily integrated into and exchanged by ICT systems in an operationalized manner. In this context, the knowledge components, that is, the ontologies, are formalized using Web-suitable, semantically unambiguous representation languages such as Resource Description Framework (RDF) Schema

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

(Brickley & Guha, 2004) and Web Ontology Language (OWL; Patel-Schneider *et al.*, 2004), and are pervasively accessible and shared across the Web.

Notably, the popularity of ontologies on the Semantic Web has led during the past years to the reinforced study of ontology engineering as a consensus-building process, in which a—potentially open and geographically distributed—group of stakeholders, or a community of practice, agrees upon a common view upon a domain of interest, and upon the way their shared knowledge can be structured in terms of concepts, attributes, relationships and constraints. This understanding of ontologies and ontology engineering processes is the result of intensive theoretical and empirical research in the Semantic Web community, supported by technology developments such as Web 2.0. Over 6 years after the publication of the first methodology for collaborative ontology engineering (Holsapple & Joshi, 2002a), it is generally acknowledged that, in order to be useful, but also economically feasible, ontologies should be developed and maintained in a community-driven manner, with the help of fully fledged environments providing dedicated support for collaboration and user participation (Vrandecic *et al.*, 2005; Tempich, 2006; Hepp, 2007). Wikis, and other similar communication and collaboration platforms enabling ontology stakeholders to exchange ideas and discuss modeling decisions, are probably the most important technological components of such environments. In addition, process-driven methodologies assist the ontology engineering team throughout the ontology life cycle, and provide empirically grounded best practices and guidelines for optimizing ontology development results in real-world projects. The goal of this article is to analyze the state of the art in the field of collaborative ontology engineering. We will survey several of the most outstanding methodologies, methods and techniques that have emerged in the last years, and present the most popular development environments, which can be utilized to carry out, or facilitate specific activities within the methodologies surveyed. A discussion of the open issues identified concludes the survey and provides a roadmap for future research and development. The article targets collaborative ontology engineering researchers, but also technology providers and potential adopters, interested in getting a comparative overview of the state of the art in the field, pointers to ongoing projects, and some ideas about directions in which this field could evolve.

This article is structured as follows. Section 2 reviews the key aspects of collaborative ontology engineering. In Section 3, we describe several established methodologies, which address these aspects from a process-oriented perspective, while in Section 4 we present software environments typically used when developing and maintaining an ontology by a group of stakeholders. We discuss our findings and based upon them recommend future directions of research and development for the ontology engineering community in Section 5.

2 Collaborative ontology engineering

Ontology engineering refers to the study of the 'activities that concern the ontology development process, the ontology life cycle, and the methodologies, tools and languages for building ontologies' (Gomez-Perez *et al.*, 2004). In a *collaborative ontology engineering* scenario process, methods and tools are explicitly designed to support a decentralized group of stakeholders or community of interest—in the sense of geographical dispersion, varying levels of skills, experience and responsibilities, as well as potentially divergent agendas—to reach a consensus in an incremental and asynchronous fashion[1].

A collaborative ontology engineering process typically starts with an analysis of the domain to be captured by the ontology, and of the requirements imposed by the ontology-based application—as it is common in any other ontology engineering process. However, in this special case, the stakeholders

[1] While collaboration is not characteristic to decentralized scenarios, we argue that methodological and tool support becomes a critical issue particularly when the ontology engineering team is diverse in terms of location, skills, experience, responsibilities and interests. Collaborative ontology engineering emerged as an independent research topic in the ontology engineering community in response to the needs of process and technology-level support in such decentralized scenarios.

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

agree on these requirements and their priorities, and propose and discuss various alternatives to create a conceptual model complying with these requirements and reflecting both their individual interests and the shared goals of the community of interest. The stakeholders need to investigate the different ways to model specific domain knowledge as concepts, attributes, relationships or constraints; to decide upon the adequate level of granularity of the model, upon conventions for labeling and documenting ontological entities, and upon the overall engineering process and associated (decision-making) procedures to be followed. The conceptual model is implemented in a formal knowledge representation language such as the Semantic Web ontology languages RDFS and OWL. In response to changes in the target domain, evolving application requirements, and discussions on whether and how to capture specific domain aspects in ontological terms, the community continuously revises and extends the ontology, and releases new versions of it. In parallel, different stakeholders may continue to maintain and use different versions of the ontology, while some of the changes may be integrated into a new shared release. In order to facilitate the systematic evolution of the joint ontology and to operationalize consensus-building, the community needs to undertake each activity in a controlled manner, and to be able to monitor the engineering process and the results achieved so far. Instruments for resolving conflicts, which might arise if several parties hold irreconcilable views at some point in the process, may become crucial to ensure progress. New stakeholders joining the community need to understand the rationales behind particular engineering-related decisions, and to follow the history of various releases of the ontology, in order for them to effectively participate in the process. Carefully documenting every step facilitates the inclusion of newcomers into the process.

A notable aspect of traditional ontology engineering, preserved by collaborative practices, is the foundation of an iterative approach to developing and maintaining ontologies, based on combinations of the activities discussed above. There are, however, several aspects in which the two differ extensively, which lead to the creation of specific methodological and technical tools assisting the ontology editors and contributors in handling these aspects. Collaborative ontology engineering projects can host potentially very large, open communities with diverse backgrounds in domain, technical or organizational terms. In such a scenario, it is thus essential to define appropriate roles and policies for developing and modifying the shared ontology, and for managing different versions thereof, and to provide widely accessible software support to document the entire process and its results and to facilitate interaction between participants. The core ontology engineering life cycle known from the literature (Gomez-Perez *et al.*, 2004) is adapted to reflect the collaborative nature of the process. For example, Braun *et al.* (2007) argue for the need of a pre-conceptualization step in the collaborative ontology engineering process. In this step, the participants share knowledge informally—for example, through tagging. The tag collection built in this way undergoes continuous evolution with gradual formalization through discussions and decisions within the community. Another example is the use of customizable workflows, as explicitly stated guidelines for running collaborative ontology engineering projects in specific environments. These workflows are specified by the community itself, thus reflecting their (organizational) requirements toward the ontology engineering process. In the following, we detail the main features of collaborative ontology engineering practices.

2.1 Key roles

The team developing the shared ontology consists of stakeholders with different, and perhaps divergent, interests and complementary competencies. The number of participant parties greatly varies across application scenarios, from dozens to thousands of organizations, and their roles in the project need to be precisely defined in order to allow for a smooth operation of the process. Classical ontology engineering distinguishes between three roles: knowledge engineers, ontology engineers and domain experts (Gomez-Perez *et al.*, 2004). Domain experts are knowledgeable in the domain that is captured by the ontology; they have intricate knowledge about domain-relevant concepts and their attributes, as well as their interdependencies and relationships. The role

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

of knowledge engineers is to elicit these insights from the domain experts—for instance, via interviews—to create a conceptual model of the domain. This conceptual model is then represented in a suitable knowledge representation language by ontology engineers. In collaborative ontology engineering, each member of the community can play several roles, depending on the types of contributions the respective individual is allowed to perform on the shared ontology, but also on the level of technology support in place and on the type of ontology that the project targets. Recent approaches to collaborative ontology engineering have in fact investigated the trade-offs between the level of expressivity of the ontology and the level of expertise predicated by the underlying formalization task, arguing for lightweight ontologies that are possibly less powerful with respect to the knowledge they can cover and the associated reasoning functionality, but whose development and maintenance can be undertaken by laymen (Siorpaes & Hepp, 2007).

Two commonly occurring roles in collaborative ontology engineering are ontology editors and ontology contributors, where the former have the authority to perform changes in the ontology. Contributors may give feedback on the ontology and propose changes based on new and evolving requirements of their particular setting. This role may overlap with the one of ontology users, based on the obvious fact that an important class of ontology change requests are identified during the usage of the ontology. There is no commonly agreed view on the distinction between these two roles in the collaborative ontology engineering literature. One could argue that only a designated subset of the users—those, who, for instance, possess some ontology engineering expertise—document usage-driven changes requests, and discuss them with the editors. In addition, changes are not always a consequence of user feedback, but can be equally motivated by changes in the domain of the ontology and in the application scenario. The (strategic) decision to, for instance, expand the domain of an ontology to cover areas that have not yet been considered, is not necessarily directly related to usage feedback. Some methodologies introduce additional roles in response to the importance of discussion and decision-making activities in a collaborative ontology engineering scenario, as outlined in Section 3.

Many recent collaborative ontology engineering enterprises come from the biomedical sector. We survey some of the most prominent ones, which illustrate the diverse collaboration models in which community-accepted ontologies are developed and maintained. We start with the FMA project[2], in which a small team of editors jointly created a foundational model of anatomy containing over 75 000 classes. In contrast, the Gene Ontology, a project which aims to provide a widely accepted terminology for the description of gene products, exhibits a different form of collaboration[3]. The underlying ontology is developed by a large base of scientists, who report on issues occurred during the usage of the ontology, whereas changes at the level of the shared ontology are undertaken by a small dedicated editing team (Tudorache *et al.*, 2008). Another interesting example is the NCI Thesaurus[4], a medical ontology that captures knowledge about cancer biology and oncology. Changes in the NCI Thesaurus are implemented by a team of around 20 editors, under the supervision of a lead editor, who coordinates the process and approves the changes. The user community can accept or reject these changes based on their individual needs. Collaborative ontology engineering is popular beyond the biomedical sector. eClass[5] (or its OWL variant eClassOWL)[6] is a standard classification for products and services to support interoperability in electronic commerce by allowing stakeholders to refer to a common vocabulary. The standard is maintained according to formally defined procedures by the eClass Foundation. Changes can be proposed by the user community via fax; in regular meetings, these proposals are discussed and, if consensus is achieved, a new release of the standard is issued.

[2] http://sig.biostr.washington.edu/projects/fm/AboutFM.html

[3] http://www.geneontology.org/

[4] http://nciterms.nci.nih.gov/

[5] http://www.eclass.de

[6] http://www.heppnetz.de/eclassowl/

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

Table 1 Access policies in collaborative ontology engineering pocesses

Permission	Lead editor/project administrator/ super user/ moderator	Editor/ content reviewer	Reporter/ contributor/ content provider (named user)	User/content consumer (unknown)
Implementation level				
Approve and propagate changes	★			
Commit changes	★	★		
Conceptualization level				
Propose changes	★	★	★	
Discuss issues and ideas	★	★	★	
Report issues and ideas	★	★	★	★

In general, and as shown through the previous examples, the model of the roles and associated access policies depend on the characteristics of the ontology engineering project, including the size of the community, the nature of the project and the intended collaboration style. Policies can range from informal agreements between participants to formal documents whose infringement is avoided through the implementation of specific monitoring and enforcement mechanisms, and can be defined upfront (statically) or dynamically—similarly to those in a Web forum where contributors can work their way up to a certain role subject to pre-defined rules. With reference to the previously mentioned projects, we provide a common set of access policies related to the user roles in Table 1. Defining such policies in an optimal way is one of the challenges of collaborative ontology engineering.

2.2 Ontology development and evolution

Publishing new ontology versions in a collaborative ontology engineering project is different to centralized scenarios due to the need to synchronize editing. Ontology editing can be done using desktop and Web-based (dedicated) development environments. To facilitate concurrent access, versioning is managed through software such as SVN (Subversion)[7], and distributed versioning approaches. However, as ontology languages allow to express the same semantics by use of a number of different syntactic variants, general-purpose versioning software relying on textual differences in files is not a perfect fit to the requirements of ontology versioning. To avoid the emergence of ontology versions that are not consensual, an agreement—usually informally documented—is reached between participants prior to carrying out specific editing actions (Tempich, 2006). In addition to challenges associated with mediating among potentially divergent viewpoints and interests, difficulties arise when using versioning software, which has not been designed to support ontology-specific efforts, especially in projects with a large number of contributors, handling a complex ontology, or both. Eventually, interests of different stakeholders are reflected in the emergence of branches of the shared ontology, in which certain parts of the ontology are preferentially edited without taking into account the opinions of the rest of the community. In such situations using general-purpose versioning software that use linear text files is likely to fail when applied for ontologies that are not serialized in a unique way. These issues are tackled by versioning software that builds around the structure of the conceptualization such as Noy and Musen (2007), Luczak-Rösch *et al.* (2010). Alternatively, wiki-based technology has proven to be feasibly applicable for facilitating community participation and feedback. Turning back to the biomedical sector, platforms such as, for instance, LexWiki enable users of the community-developed BiomedGT to browse the ontology, make comments and propose changes in textual form[8]. The wiki stores textual change suggestions as annotations of the ontology, while

[7] http://subversion.tigris.org/

[8] http://biomedgt.nci.nih.gov/wiki/index.php/Main_Page

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

editors can access these suggestions and implement changes directly in the ontology. As will be elaborated in Section 4, wikis offer an intuitively usable community-based forum for discussions. They can also be extended into additional ontology engineering features improving their usability in ontology-related projects, such as class hierarchy browsing and auto-completion; despite their natural appeal, other features are much more difficult to implement: editors must switch back and forth to implement changes, while simple semantic checks on the data are not supported due to the textual nature of the annotations provided. Additionally, each wiki software implements its own collaboration process that cannot be customized during the project.

2.3 Collaboration process

A common characteristic of early ontology engineering methodologies has been the notion of a pre-defined process model that guides the ontology engineering activities and yields a dedicated set of roles, policies and tools to perform it in a consistent fashion—examples, can be found, for instance in Holsapple and Joshi (2002a), Gomez-Perez *et al.* (2004) and Vrandecic *et al.* (2005). With increasing adoption of collaborative ontology engineering principles and practices, the trend moved toward greater flexibility for the engineering team in defining their own models to optimize the results of the project, and providing the tools to support this flexibility (Braun *et al.*, 2007). Tudorache *et al.* (2008) provide several examples of such projects, which have been publishing and refining their workflows for years, including the Gene Ontology project discussed above. In other cases, knowledge and ontology engineers have been actively working on how to formalize the process models they follow. Given proper tool support, the availability of such formal models promise to further enhance the flexibility of the underlying approach by allowing changes to be made during the (now adaptable) ontology development life cycle. An instance thereof is the NeOn project[9] developing an ontology for the United Nations Food and Agriculture Organization (UN FAO).

Factors to be taken into account in collaborative engineering scenarios are the organizational structure underlying the project, the size and the openness of the community of contributors, the required level of rigor in quality control and the complexity of the representation (Tudorache *et al.*, 2008). At the same time, a proper balance must be struck between the formal representation of the process model and the added value of this representation as per automatic use in the project. VoCamps[10] denote informal bar-camp-style events organized by the Semantic Web community, in which a group of stakeholders and enthusiasts (usually around 20) meet at a physical location to develop lightweight vocabularies capturing domains of interests that are proposed democratically by the participants. From a methodological point of view, the VoCamp approach is the most unstructured from all examples presented in this survey. It does not introduce any process model, neither for the ontology development activities nor for the deployment, maintenance and use of the resulting ontology. Moreover, there is no explicit model of roles and policies. Nevertheless, the appeal of the approach lies precisely in its participatory and informal nature. It builds upon the real needs of the community interested in developing ontologies in specific domains.

With respect to reaching consensus, generic techniques that found applicability in the ontology engineering field include the Nominal Group (Dunnette *et al.*, 1963) and the Delphi (Linstone & Turoff, 1975) approaches, supported by elementary communication channels such as discussion forums and chats (Holsapple & Joshi, 2002a; Tudorache *et al.*, 2008). Further examples of collaboration are provided in the following sections.

2.4 Collaboration tools

Collaborative ontology engineering environments can be characterized by a significant amount of deliberation between contributors regarding both the ontology to be developed and the engineering

[9] http://www.neon-project.org

[10] http://vocamp.org

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

process itself. Discussions between participants usually take place via email, instant messaging and discussion forums. While these channels can provide general-purpose communication and archiving support, a direct linking between the threads of discussions and the content of the ontology the discussions refer to is largely missing. As such, gaining an understanding of the status of the discussion, and of the rationale behind a certain decision require extensive effort, especially for any part of the community that is not at the core of the editing team, or that joins the project at a later stage. These limitations have been recently addressed within the Protégé initiative, which released a collaborative version of their popular ontology engineering environment (Tudorache *et al.*, 2008). Going beyond such links, taking an informed decision on any ontology-related issue, be that the resolution of a conflicting situation, the detection of inconsistencies, or the assessment of the necessity to introduce specific changes in the ontology, requires dedicated tool support. Such support is not offered per default through the channels mentioned above, or even by more specialized tools such as (wiki-based) ontology editors, if these are not customized to the task and the domain at hand. One exception might be the topic of argumentation, which is at the core of several methodologies and collaborative engineering environments (Vrandecic *et al.*, 2005; Dellschaft *et al.*, 2008). Furthermore, while face-to-face meetings commonly do occur in the majority of collaborative ontology engineering projects, recording minutes, decisions and actions as structured information linked to the ontology is not fully supported by the technology available.

The remaining sections will present and perform a comparative analysis of the most important methodologies and associated software environments created in the context of collaborative ontology engineering over the past decade. Each methodology will be presented in terms of their main activities and tasks, as well as tool support and real-world application. The second part of the survey will concentrate on the most promising tools in this area and on how they support particularly challenging themes such as evolution and collaboration.

3 Ontology engineering methodologies

In order to facilitate the operationalization of the ontology engineering process in terms of results, labor and duration, significant efforts have been spent in the Semantic Web community to understand the life cycle of semantic content and to design methodologies providing descriptions of the process through which user needs are translated into semantic artifacts. In general a methodology can be defined as 'a comprehensive, integrated series of techniques or methods creating a general systems theory of how a class of thought-intensive work ought be performed' (IEEE Computer Society, 1990). In particular, a methodology includes a description of the process to be performed and of the roles involved in the process, assigns responsibilities to activities and people, and gives recommendations in form of best practices and guidelines. It can be related to a specific process model, which provides additional details on the order and relationships between activities foreseen by the corresponding life cycle[11].

Depending on the setting in which they can be applied in, methodologies can be divided into two main categories:

Methodologies for centralized ontology engineering: The ontology engineering team is concentrated in one location and communication between team members occurs, among others, in regular face-to-face meetings. This setting is particularly relevant for the development of ontologies for a specific purpose within an organization.

Methodologies for decentralized ontology engineering: This type of approach applies to the Semantic Web or any other open, large-scale environment where the ontology engineering team and IT systems and infrastructure are distributed. The ontology engineering team is composed of different stakeholders dispersed over several geographical locations, applying

[11] See the NeOn project for a recent analysis of ontology life cycles at `http://http://www.neon-project.org`

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

the shared ontology in different settings. The ontology provides a lingua-franca within the contributing community or ensures interoperability between machines, humans or both.

Early ontology engineering methodologies such as Uschold and King (1995), Swartout *et al.* (1996), and Fox and Gruninger (1998) (see Fernández-López & Gómez-Pérez, 2002 for an overview) focussed on core ontology development activities: requirements analysis, conceptualization, implementation, evaluation and maintenance. They assume that the formal specification of the domain knowledge to be used in an application system precedes the actual development of the system. A second generation of methodologies shifted this focus towards a more iterative engineering process in which application-specific requirements are seen as an integral part of the requirements analysis activity. Furthermore, several versions of the ontology are released incrementally in order to ensure that requirements are optimally met, and to respond to changing requirements. Common to all these approaches is the division between domain experts, knowledge engineers, ontology engineers and users with respect to their development and post-development responsibilities. The ontology engineering process is driven by engineers, who gather requirements from domain experts and users, implement these requirements, test the resulting ontology and steer its evolution. Representative for this second generation of methodologies are Methonotology (Fernandez-Lopez *et al*., 1997) and OnToKnowledge (Sure, 2002; see (Gomez-Perez *et al*., 2004) for an overview) The third and current generation of ontology engineering methodologies follows a participatory approach. The emphasis is on making ontology engineering a truly collaborative effort carried out by a potentially large group of contributors with diverse backgrounds and skills, and on providing the technological support that makes it easier for non-experts to become involved in ontology-related activities beyond requirements elicitation. In the following, we describe several of the most prominent methodologies in the field of collaborative ontology engineering in the last of the three aforementioned categories in chronological order of their publication.

3.1 The methodology of Holsapple and Joshi

Holsapple and Joshi (2002a) proposed the first comprehensive methodology to collaborative ontology design based on a Delphi-like (Linstone & Turoff, 1975) approach to structure the consensus-building process. First, an initial ontology is developed by merging or integrating existing ontologies. This ontology provides a starting point for the design process, which is performed collaboratively by revising the ontology based on the feedback received from the various parties involved. The engineering process is divided into four phases (Figure 1, Holsapple & Joshi, 2002a).

Preparation: Defines design criteria, and determines boundary conditions and standards that can then be used for evaluation.

Anchoring: Produces a first ontology that helps for orientation of the participants.

Iterative improvement: Adjusts and extends the anchor ontology developed in the previous step. This is achieved with the help of an expert panel, which is interviewed through questionnaires to collect their feedback on the ontology. The consolidated results are handed to the experts with the aim to achieve a consensus on all design issues. The ontology is edited by ontology developers, who implement the changes consensually agreed among the participants.

Application: Is the actual usage of the ontology in a specific context.

3.1.1 Collaborative aspects

In this approach, collaboration is facilitated through the application of a Delphi-like approach to incrementally improve the shared ontology.

3.1.2 Roles, policies and tasks

The methodology differentiates between an expert panel, which takes the role of the content reviewers and contributors, and a team of lead editors.

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

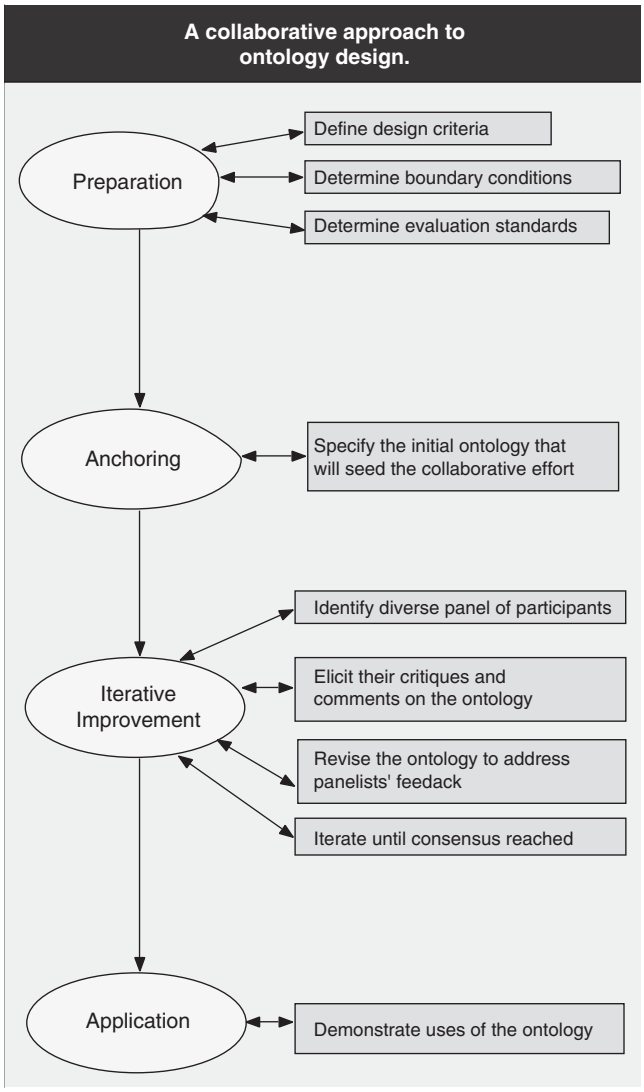


Figure 1 Overview of the Methodology of Holsapple and Joshi

3.1.3 Application areas

The methodology was applied in a knowledge management project in order to describe the conduct of knowledge management in organizations (Holsapple & Joshi, 2002b). The findings of this case study confirm the usefulness of the Delphi method to guide decentralized ontology engineering; they also raise several issues related to the resource-intensive nature of collaboration, the need for additional support tool to ensure a consistent execution of the Delphi process (e.g. by facilitating the access to the information gathered from the panelists in each round), and to potential biases induced by the anchoring ontology.

3.2 Dogma-Mess

Dogma-Mess (De Moor *et al.*, 2006; Spyns *et al.*, 2007) is an extension of the Dogma methodology (Jarrar & Meersman, 2009) toward inter-organizational support. In Dogma, an ontology consists of a base of lexons, holding conceptualizations of a domain[12] and a layer of ontological commitments.

[12] Lexons can be considered as a combination of RDF/OWL triples and their inverses, defining taxonomical or domain-specific relationships.

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

Dogma-Mess was explicitly conceived for those ontology engineering settings that involve multiple stakeholders. It distinguishes between five major phases:

Formulate vision statement: The stakeholders develop a shared specification of scope and aim of the ontology.

Conduct feasibility study: The vision statement is refined and evaluated in terms of costs, benefits and technology.

Project management: Project management activities (time management, planning, controlling) are initiated.

Preparation nd scoping: This phase is carried out as a sequence of five tasks: (i) definition of user requirements; (ii) definition of purpose; (iii) identification of domain experts; (iv) compilation of knowledge resources; and (v) scoping of knowledge resources.

Domain conceptualization: This is the core of the ontology engineering methodology. It involves the analysis of the domain and leads to a Dogma-style ontology. It involves the following activities:

Knowledge discovery: This is performed semi-automatically within the following tasks:

- collect, select and pre-process an appropriate corpus;
- discover sets of equivalent words and expressions;
- validate the sets with the help of a domain expert;
- discover sets of semantic relations and extend the sets of equivalent words and expressions;
- validate the relations and extended concept definitions with the help of a domain expert;
- create a formal representation.

Knowledge elicitation: This activity allows domain experts produce a conceptualization based on their domain expertise. The activity encompasses brainstorming, abstraction and the compilation of the baseline taxonomy.

Knowledge negotiation: This activity concerns a conversational gathering of feedback from domain experts with respect to the meaning of concepts based on efficiently handling context dependencies, in particular specialization dependencies.

Knowledge breakdown: Here the aim is to generate a hierarchical structure using linguistic techniques. For this purpose, the methodology recommends (i) verbalizing elementary sentences, which involves extracting elementary facts; and (ii) engineering lexons, which aims at the creation of verbalized facts in natural language.

Application specification: This final phase includes structuring the applications domain, tailoring the domain conceptualization according to application-specific constraints and preparing the validation of the ontology.

3.2.1 Collaborative aspects

In Dogma-Mess, collaboration is considered in the context of what the authors call 'inter-organizational' ontology engineering. The authors propose a pragmatic approach to handle adaptations of shared ontologies in local environments by looking into ways to use formal techniques to context management in ontology engineering projects, while ensuring the efficiency of these projects. The methodology does not give any details on how to reach consensus on the shared ontology, in fact the core activities rely exclusively on Dogma, which did not target collaborative settings.

3.2.2 Roles, policies and tasks

Dogma-Mess involves domain experts covering the role of content reviewers and providers and core domain experts covering the role of the lead editors. To support the core domain experts, who are typically not ontology-engineering experts, knowledge engineers may take the role of lead editors, too.

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

3.2.3 Application areas

In De Moor *et al*. (2006), the authors introduce a Web-based system that applies Dogma-Mess to engineer ontologies within and across organizations. Preliminary evaluation results in a project in the Dutch bakery sector are mentioned in the same publication, however, they remain very limited. A second application sector was Human Resources (De Leenheer *et al*., 2009).

3.3 DILIGENT

DILIGENT (Vrandecic *et al*., 2005) proposes a methodology for collaborative ontology engineering based on the IBIS argumentation model (Kunz & Rittel, 1970). The process model is divided into several phases to be carried out in multiple iterations:

Build: A core team of domain experts, users, knowledge engineers and ontology engineers build an ontology that is not required to be complete as with respect to the requirements to be fulfilled.

Local adaptation: The ontology is made available and users adapt it to their own needs in their local environments. However, the 'original' ontology remains unchanged while local adaptions are logged for future analysis.

Analysis: Local branches of the shared ontology are analyzed with respect to their mutual differences and an ontology engineering board selects changes to be carried over into the next version of the shared ontology.

Revision: The changes agreed in the previous phase are implemented in the shared ontology and a new version thereof is released.

Local updates: Users may decide to align their local ontologies with the new version of the ontology released by the board in order to ensure compliance to commonly agreed standards, as well as communication and interoperability benefits arising from the usage of an ontology that is shared within the community.

An overview of the methodology is presented in Figure 2 (see Tempich *et al*., 2007).

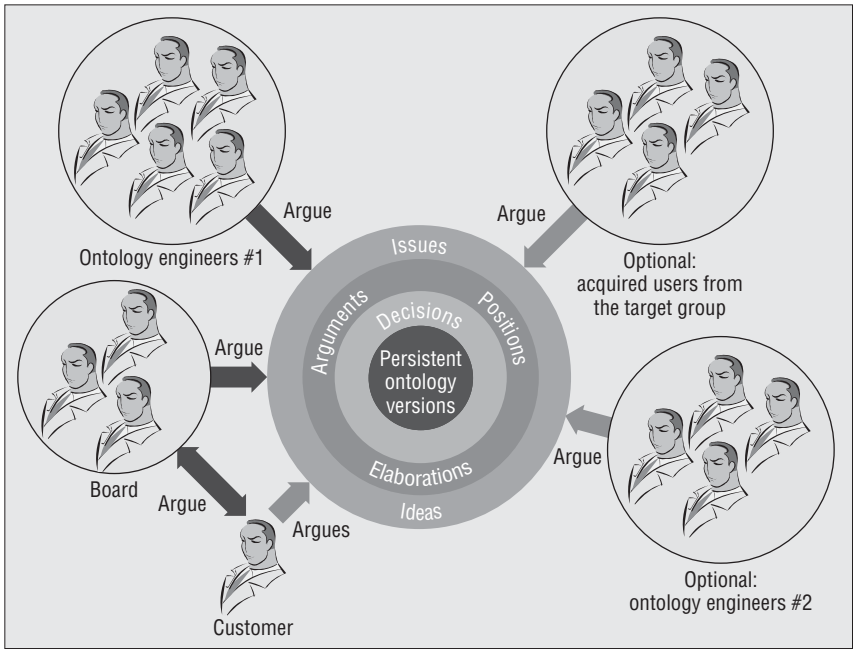


Figure 2 Overview of DILIGENT

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

3.3.1 *Collaborative aspects*

Collaboration is performed throughout all phases of the engineering cycle as all the different stakeholders argue for and against the implemented ontology primitives. It is based on an argumentation process consisting of several phases. First, the participants in an ontology engineering discussion choose a moderator. The basic rules for moderation also apply in this case: the moderator does not contribute to the discussion, but structures it; he does not take part in decision, but organizes the decision process. Any participant may take the role of the moderator and the moderator role may move from one participant to the next. In the second step, the participants agree on a mechanism to reach agreements during the discussions. They decide upon a voting procedure such as majority voting, and on the conditions triggering a new voting round. For instance, they can vote within fixed time intervals or if no new arguments have been brought forward for a certain period of time. Then ontology engineering discussions are initiated by specifying issues that arose during the process, corresponding to domain or application requirements for the ontology to be built. Once the discussion evolves, issues can be grouped according to their priority for the target setting and treated accordingly. Discussions around 'issues' are structured with the help of 'ideas'. Provided a generally agreed relevant issue, participants bring forward ideas to formalize it. Other participants may express their agreement or disagreement with arguments and alternative ideas in order to strengthen or weaken them. This step is of particular importance for the ontology design as the effectiveness and efficiency of the entire process depends on the decisions based on the provided arguments. DILIGENT proposes the use of the Rhetorical Structure Theory (RST; Mann & Thompson, 1987) to define the types of arguments that can be used during the discussions while reaching a balance between the manageability of the overall process and the ease-of-use of the approach by a large community.

3.3.2 *Roles, policies and tasks*

In DILIGENT, users take part in the engineering process by proposing issues and ideas and arguing on them. The users, as well as knowledge engineers, domain experts and, potentially, a customer take the role of the contributors. Ontology engineers act as the editors implementing changes to the ontology and a dedicated board of ontology engineers is allowed to decide on the deployment of changes to the consensual ontology model as a group of lead editors.

3.3.3 *Application areas*

The methodology was evaluated at various stages of its development through case studies in domains as diverse as tourism, law (Casanovas *et al.*, 2007) and academic research. While the findings of the case studies are positive with respect to the applicability of the overall approach to collaboratively engineer an ontology, they also emphasize the importance of software support in many phases of the engineering process. This issue has been taken up in several research projects, which produced a suite of (Web-based) ontology editors facilitating RST-like discussions and translating the results of these discussions in changes at the level of the ontology. Examples of such tools are coefficientMakna (Tempich *et al.*, 2007) and Cicero as part of the NeOn Toolkit (Dellschaft *et al.*, 2008).

3.4 *Human-Centered Ontology Engineering Methodology*

The Human-Centered Ontology Engineering Methodology (HCOME; Kotis & Vouros, 2005) is, like DILIGENT, a methodology that explicitly focuses on the distributed creation of ontologies in knowledge-intensive organizations. It differentiates between several information spaces in which conceptualization efforts are carried out: a personal information space, reflecting the view of an individual party on the domain of interest and a shared information space based on which different parties synergistically develop a commonly agreed conceptualization, by aligning individual viewpoints and putting joint results into the context of their own experiences. Orthogonally, the methodology is organized in three major phases, each with several tasks and goals, which may refer to these spaces.

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

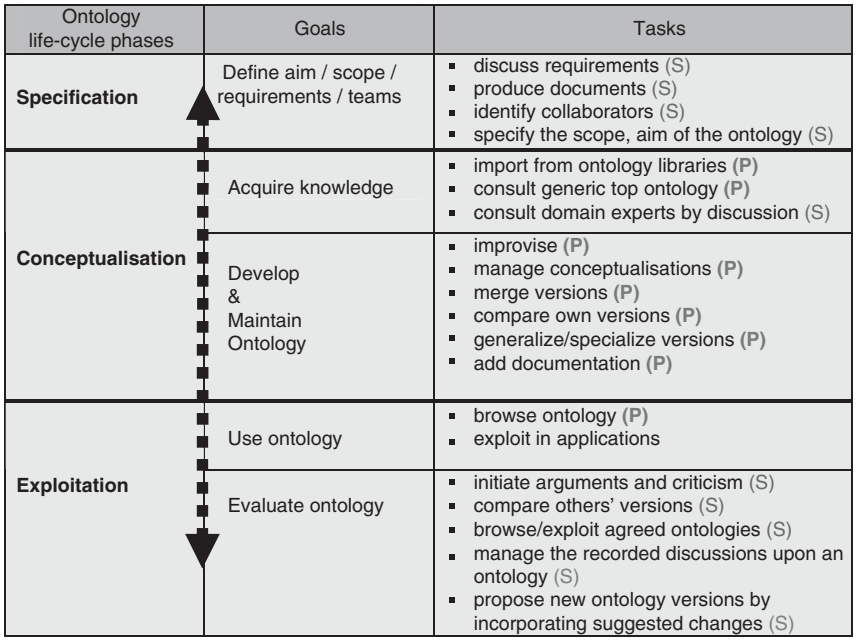


Figure 3 Overview of HCOME, (S) denotes shared space, and (P) denotes private spaces

Specification: This phase establishes the engineering teams that collaborate toward defining a joint aim and scope of the ontology and analyzes the requirements for developing a shared ontology, which are recorded in a requirements specification document.

Conceptualization: Conceptualization first takes place locally within the various teams. It covers the following tasks: (i) importing existing ontologies from ontology libraries; (ii) consulting generic top ontologies for better understanding; (iii) improvising ontologies, that is, from-scratch-development, based on the input of domain experts; (iv) managing, mapping and merging of ontology versions; and (v) ontology evolution.

Exploitation: In this phase, the ontology is used and against alternatives developed by other stakeholders. Structured conversation and critical dialogue facilitate achievement of a common understanding with respect to the directions in which the shared ontology should be adjusted or extended. The tasks performed in this phase include: (i) inspection of ontologies by collaborators; (ii) comparisons of versions of one ontology in order to spot differences; and (iii) publication of comments and feedback.

An overview of the methodology is provided in Figure 3 (from Kotis & Vouros, 2005).

3.4.1 Collaborative aspects

HCOME also identifies a number of principles, which should be fulfilled by collaborative ontology engineering environments. Besides the need for an eclectic approach to the development of ontologies, the authors argue in favor of providing techniques and tools that leverage various information sources as input for the ontological conceptualization, and for a conversational collaboration style by which knowledge workers can seamlessly deploy and evaluate the shared ontology and become active in its evolution. Finally, it is seen as critical that ontology engineering environments allow knowledge workers to interact with ontologies in a natural and consistent way, which means not only different levels of details, but also interfaces that abstract from the particulars of knowledge representation languages, and features for consistency checking and ontology matching.

To support the usage of the methodology in real-world projects, HCOME is accompanied by the HCONE tool suite, which includes, among other things, features for the development and

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

management of shared ontologies. Collaboration is supported through a version of the IBIS argumentation model and a notification system, which updates the participants on the status of the discussions. Users of the tools can browse the shared ontology, examine how it differs from its personal counterparts and consult the rationale behind specific design issues. Feedback can be collected by posting to a moderated discussion thread.

3.4.2 Roles, policies and tasks

HCOME introduces no dedicated model for roles and policies. Instead, it differentiates between the personal space, the shared space and the agreed space. Each member of the community can deploy her ontologies to the personal space and the shared space for personal and collaborative revision, respectively. Finally, a consensual ontology is deployed to the agreed space as the result of an argumentation process between all community members.

3.4.3 Application areas

There is scant evidence of the application of HCOME/HCONE in real-world projects. As a continuation of the work, the authors proposed several years later the HCOME-3O framework, which allows for an improved management of the collaboration process using meta-ontologies (capturing administrative information, changes to various ontology versions, and the rationale therefor) and semantic wiki technology (Kotis, 2008).

3.5 RapidOWL

RapidOWL applies the agile engineering paradigm to ontology engineering (Auer & Herre, 2006). The approach does not commit to a specific process model or an ontology life cycle, but aims at providing a number of guidelines to be taken into account by the engineering team. The general goal is to allow domain experts to become an integral part of the ontology development process, by identifying simple, tool-supported strategies and techniques, which they can apply without extensive intervention of knowledge representation experts. The main components of RapidOWL are, as illustrated in Figure 4 (Auer & Herre, 2006):

Values: RapidOWL subscribes to the philosophy of eXtreme Programming: Communication, as crucial enabler of a collaborative approach to any engineering endeavor; Feedback, to steer the evolution of the shared ontology following the needs and requirements of the stakeholders; Simplicity, to facilitate the maintainability of the ontology and the underlying data; and Courage, to foster progress despite potential modeling dead-ends. Communication and Feedback are merged into a new category, termed Community, which covers both the social aspects involved in collaborative design and the evolution of the ontology by gathering the feedback of the participants. In addition to these category, RapidOWL introduced a new one, Transparency, by which the full record of changes to the ontologies are made visible to the entire community, thus acknowledging the contributions of the corresponding parties and facilitating an effective monitoring of the activities of the participants.

Principles: Complementarily to these four tenets of XP, which are understood as long-term goals of every agile ontology engineering project, RapidOWL recommends that every agile ontology engineering process be guided in the mid-term by several principles. They are partially adopted from the design goals for wiki systems defined by Ward Cunningham[13]. Among the most important ones we highlight: uniform authoring methods for both schema and instance representation and modeling, observable development and rapid feedback (to ensure that the ontology reflects the views of a high number of community members and to ease the maintainability).

[13] `http://c2.com/cgi/wiki?WikiDesignPrinciples`

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

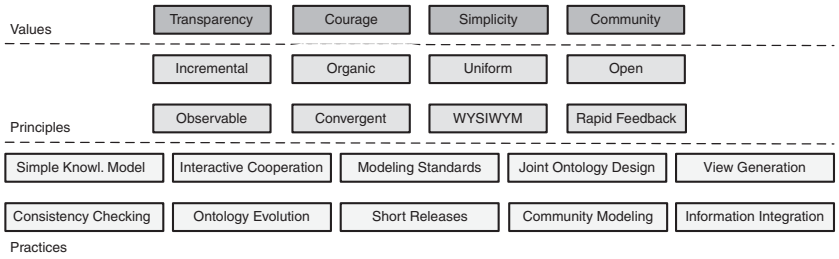


Figure 4 Overview of RapidOWL

Practices: A third component of the RapidOWL approach are engineering best practices, which are inspired both by general-purpose eXtreme Programming and by other works on collaborative knowledge base design such as Knublauch (2002). Important aspects are the Joint Ontology Design (between knowledge engineers, domain experts and users), View Generation (to reflect the needs and requirements of individual stakeholders) and Modeling Standards (to ensure reusability and interoperability).

3.5.1 Collaborative aspects

Collaboration is addressed in RapidOWL at the level of general guidelines, be that long-term values to be taken into account, mid-term principles according to which the engineering process should be carried out, or concrete practices, which aid knowledge engineers, domain experts and users during this process. There are no concrete methods, techniques and tools proposed as integral part of the methodology; as such, the approach could be used as an assessment framework for other methodological attempts in the ontological engineering field, and less as a 'cookbook', which describes how the engineering process should be performed in terms of phases, activities, tasks, roles and best practices.

3.5.2 Roles, policies and tasks

The set of general guidelines to which RapidOWL subscribes distinguishes between knowledge engineers, who act as lead editors, and domain experts and users, who act as editors in the ontology engineering process.

3.5.3 Application areas

RapidOWL guidelines have been applied to build the ontology underlying the 'Vernetzte Kirche' (in English: Networked Church) project, which runs a series of Web-based portals on behalf of the Lutheran Church in Bavaria (Auer & Pieterse, 2005). The methodology is supported by the POWL[14] software. Recently, the University of Leipzig realized its catalogue of professors using ontologies that were build following RapidOWL[15].

3.6 Ontology maturing

Ontology maturing is a community-driven approach to ontology engineering (Braun *et al.*, 2007) that puts emphasis on the role of the user community in steering a sustainable, long-term evolution of an ontology. The maturing process consists of four phases, which result in ontologies with an increasing degree of formality and expressivity (Figure 5, from Braun *et al.*, 2007):

Emergence of ideas: In the first phase, new concept ideas are collected in an *ad hoc* fashion. This is achieved through the assignment of simple tags.

[14] `http://sourceforge.net/projects/powl/` and `OntoWikihttp://code.google.com/p/ontowiki/`

[15] `http://catalogus-professorum.org/cpm/`

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

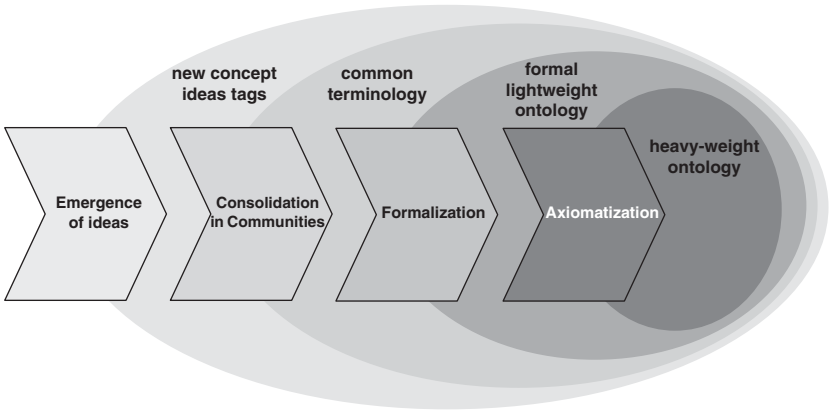


Figure 5 Overview of Ontology Maturing

Consolidation in communities: The tags generated in the first phase are reused and adapted by the user community. The aim is to extract concepts from the available tags leading to a common terminology.

Formalization: This phase adds taxonomic and *ad hoc* relations to the common terminology yielding lightweight but formal ontologies.

Axiomatization: The ontology is further refined with logical axioms.

3.6.1 Collaborative aspects

How collaboration is carried on in the Ontology Maturing methodology is underspecified. However, the core principle is a focus on the consolidation of knowledge in communities that is core phase of the overall process of ontology maturing.

3.6.2 Roles, policies and tasks

Just as for the collaborative aspects, the concrete roles policies and tasks in this methodology are underspecified.

3.6.3 Application areas

While the collaboration-related aspects of the methodology, including roles, policies and tasks are underspecified, Ontology Maturing excels in providing a number of detailed case studies in which the methodology was successfully applied (Braun *et al*., 2008; Braun *et al*., 2010)[16]. The case studies are supported by the tool SOBOLEO (Braun & Zacharias, 2010), which facilitates the collaborative editing of the ontology in a user-friendly manner and its subsequent usage in tasks such as semantic search or semantic annotation. While these case studies show the feasibility of the approach, their practical findings remain so far very generic to have significant implications for the future research and development in the ontology engineering community.

3.7 Comparative analysis

In order to analyze the collaborative ontology engineering, methodologies introduced above we adapted the framework proposed in Fernández-López and Gómez-Pérez (2002), which is used in the ontology engineering literature to compare methodologies in terms of the following nine criteria:

C1. Inheritance from knowledge engineering: This criterion assesses the influence of traditional knowledge engineering on the methodology. Building upon the insights and results of a

[16] http://mature-ip.eu/demonstrators

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

research field as renowned as knowledge engineering ensures a solid grounding of the new approaches. In the same time, compatibility with established practices has a positive effect on adoption and impact.

C2. Detail of the methodology: This criterion assesses the level of detail of the methodology, as the availability of elaborated process descriptions accompanied by empirically driven best practices and guidelines is crucial for adoption.

C3. Recommendations for knowledge formalization: This criterion considers the knowledge formalism(s) the methodology is tailored to. Strong dependencies toward a particular formalism or knowledge representation language may influence the applicability of the methodology in settings that have constraints in this respect.

C4. Strategy for building ontologies: This criterion discusses whether the strategy underlying the methodology is application-dependent, application-semidependent or application-independent. The underlying assumption is that there is a trade-off between the level of assistance provided by a fine-grained methodology in a specific environment and its applicability to other environments, which might not exhibit the same characteristics.

C5. Strategy for identifying concepts: This criterion researches whether the strategy for identifying concepts is top-down, bottom-up, or middle-out (Uschold & Grueninger, 1996). The choice of an appropriate strategy depends on the type of ontology engineering scenario (centralized vs. decentralized), and on the availability of domain-related documentation and requirements, which describe the knowledge to be encoded in the ontology.

C6. Recommended life cycle: This criterion investigates whether the methodology proposes a life cycle; the life cycle needs to match the requirements of the scenario for which the ontology is being developed.

C7. Differences between the methodology and IEEE 1074-1995: This criterion summarizes the differences between the methodology and the IEEE 1074-1995 standard. This standard provides a schema to generate software development and maintenance (in short, software life cycle processes) processes. Similarly, criterion C1 gives an assessment of the potential impact of the methodology, as standards-compliant approaches are expected to yield a greater potential for adoption.

C8. Recommended techniques: This criterion investigates whether the methodology proposes the use of specific techniques for carrying out activities. The availability of such techniques can have a positive effect on the usability of the methodology, as these techniques provide support the ontology engineering team.

C9. Usage of the methodology: This criterion outlines ontologies that have been developed following the methodology, giving an account on the degree to which that methodology has been validated in real-world projects and on the types of scenarios to which it is well or less suited.

We extended and adapted this catalogue of criteria to reflect the characteristics of decentralized ontology engineering projects. The result contains ten criteria, which are explained below. Table 2 summarizes the results of our analysis following these 10 criteria.

C-I. Detail of the methodology: Same as above.

C-II. Recommendations for knowledge formalization: Extending the scope of the original definition, this criterion analyzes for which level of expressivity of the ontology the methodology is designed (e.g. controlled vocabulary or fully axiomatized ontology). It accounts for the trend observed in decentralized ontology engineering methodology to put more emphasis on lightweight ontologies, which can be feasibly developed and maintained by a community of non-experts.

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

Table 2 Overview of collaborative ontology engineering methodologies

Criterion	Holsapple and Joshi	Dogma-Mess	DILIGENT	HCOME	RapidOWL	Ontology maturing
Detail of the methodology	Detailed	Detailed	Very detailed	Very detailed	Not detailed	Not detailed
Rec. for knowledge formalization	Fully axiomatized ontologies	Controlled vocabularies for inter-organizational communication formalized via lexons	Fully axiomatized ontologies	Fully axiomatized ontologies formalized in NeoClassic	Degree of formalization not prescribed, all knowledge representation based on RDF-triples	Lightweight ontologies
Strategy for building ontologies	Application-semi-independent	Application-dependent	Application-independent	Application-semi-independent	Application-independent	Application-semi-independent
Strategy for identifying concepts	Delphi method, top-down	Brainstorm, abstract and compile baseline taxonomy; generate a hierarchical structure using linguistic techniques	Middle-out, proposals of abstract design issues and collection of ideas of concrete implementations of the issues	Bottom-up, ontology reuse, improvising of ontologies, alignment of multiple ontologies	Open community modeling and information integration, middle-out	Emergence of ideas, consolidation within the community and formalization, bottom-up
Rec. life cycle	Iterative	Iterative	Iterative	Iterative	Rapid prototyping	No life cycle model proposed
Rec. techniques	Delphi method for feedback collection	Decomposition of generic ontology base and application-dependent commitment layer, template and context dependencies management, knowledge discovery based on linguistic resources	IBIS decision support	IBIS decision support, Latent Semantic Indexing, alignment to linguistic resources like WordNet, support for views and conversational feedback gathering, ontology alignment	View-based editing for different roles, providing concrete techniques for performing different practices is stressed but not explicitly defined	Tagging, wikis

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

Table 2 *(Continued)*

Roles	Knowledge workers	Ontology engineers, domain experts	Domain experts, knowledge and ontology engineers, ontology users	Knowledge workers	Domain experts, knowledge engineers, users	Domain experts
Evolution	Iterative improvement based on feedback collected by use of questionnaires	Complex selection process to identify relevant concepts from the individually evolved organizational ontologies for adoption in the new version of the inter-organizational ontology	Iterative process of local adaption and change of the shared ontology and then analysis, revision, and implementation of relevant changes from the local branches back to the shared ontology	Iterative process of local use and evaluation of the shared ontology and then propagation, discussion and implementation of agreed changes	Frequent evolution based on the feedback of the community	Continuous process that results ontologies with an increasing level of formality
Collaboration	Delphi method	Knowledge negotiation	IBIS-based argumentation framework and argumentation ontology	Shared HCONE discussion model based on IBIS, later semantic wikis	–	Wiki-based collaboration
Usage of the methodology	Describing the conduct of knowledge management in organizations	Experimental inter-organizational ontology engineering in the Dutch bakery and HRM sectors	Used within Iuriservice, a Web-based intelligent FAQ for judicial use (Casanovas *et al.*, 2007)	Graduate project at the University of the Aegean	Catalogue of professors at University of Leipzig, semantic application for the Lutheran Church in Bavaria	Image-based navigation and bookmarking of digital cultural and scientific resources

C-III. Strategy for building ontologies: Same as above.

C-IV. Strategy for identifying concepts: Same as above.

C-V. Recommended life cycle: Same as above.

C-VI. Recommended techniques: Same as above.

C-VII. Roles: Collaborative ontology engineering is based on a potentially complex roles model of editors and contributors, which ensures that the engineering process is performed in an efficient manner despite the size of the engineering team and the decentralized nature of the setting.

C-VIII. Evolution: This criterion reflects on how the methodology supports the publication and management of ontology versions. The evolution of the ontology to which many stakeholders contribute might cause inconsistencies in conceptual modeling and discussions about how to optimally implement the needs and preferences of these stakeholders.

C-IX. Collaboration: This criterion investigates the procedures, techniques and support tools facilitating consensus finding within the ontology community.

C-X. Usage of the methodology: Same as above.

Most of the surveyed methodologies provide a detailed description of the process model to be followed within a decentralized ontology engineering scenario, describing the phases, activities and tasks to be performed by each category of contributors. RapidOWL relies on existing principles from related engineering disciplines, leaving out any details on how these principles should be translated into ontology engineering terms. DILIGENT, in contrast, gives a full account of the approach to be applied, describing all activities in terms of the roles, tasks, inputs, outputs and tool assistance required. As such, it gives an excellent example of how such a methodology should be documented for feasible external use. Despite this promising baseline, all methodologies would benefit from a more elaborated and precise description of the criteria to be taken into account when taking certain decisions along the engineering process and from the availability of comprehensive case study findings and domain-specific guidelines. In this respect, the case studies around the Ontology Maturing approach provide a useful baseline, besides being an interesting reading of the types of systems and projects collaborative ontology engineering is currently being applied.

With respect to the strategy for building ontologies, it is important to highlight that several methodologies have been classified as 'application-independent', though no substantial empirical evidence could be found of their usability across application scenarios. This assessment is based on the declared aim of the methodologies. Other methodologies have a clear application-semi-independent character (Fernández-López & Gómez-Pérez, 2002), as the authors identify a series of core application scenarios for which the methodology is implicitly or explicitly designed for. Only Dogma-Mess relies on application-specific linguistic resources for the development of the ontology; naturally, the resulting ontology is application-dependent, though the two domains in which it was applied speak in favor of the generality of its principles.

Regarding the recommended life cycle model, the only two methodologies that do not explicitly propose an iterative model are RapidOWL and Ontology Maturing. The methodologies are not described at the same level of detail as the others surveyed, and do not explicitly propose an integrative phase model for the full set of engineering activities. Especially, RapidOWL is rather a set of principles and guidelines than an engineering methodology (in the IEEE sense). Most methodologies recommend various techniques that can be applied as part of specific phases of the engineering process. The two most prominent examples therefor are DILIGENT and HCOME. As part of the HCONE environment, the latter offers an impressive portfolio of features that are expected to crucially ease the development of the ontology, especially when it comes to leveraging

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

existing resources such as thesauri and others. As for DILIGENT, the methodology includes, in addition to a description of the phases, activities, tasks and associated roles, recommendations for techniques and tools that could be used to assist the engineering team at various stages of the process. The work is also useful for the detailed analysis of the requirements for automation support for collaborative ontology engineering, and as such, as inspiration for development roadmaps for ontology development environments (ODEs).

With respect to the roles defined within the process, we observe an average split of three approaches, which address less technically skilled target groups, and other three that address both technically savvy ontology engineers and laymen. Interestingly, in case of RapidOWL and Ontology Maturing, this correlates with the complexity level of the technologies recommended to be used, such as wikis and other Web 2.0 tools, which are rather lightweight. Both inspired by the IBIS model, DILIGENT and HCOME propose an argumentation/discussion model to facilitate consensus finding. Holsapple and Joshi's methodology goes along with this and recommends a Delphi-inspired approach. RapidOWL and Ontology Maturing assume that Web 2.0 communication platforms will enable collaboration and exchange of ideas among the participants, whereas Dogma-Mess recommends a knowledge negotiation phase in which domain experts discuss the meaning of concepts in particular application contexts.

To summarize, there are several methodologies that provide a detailed description of the collaborative engineering process according to which ontologies are developed and maintained in decentralized scenarios. DILIGENT and HCOME are the most prominent examples therefor, however, they are limited in terms of the available concrete case study descriptions, and, meanwhile, also with respect to the associated technological support. In this respect, an approach such as Ontology Maturing can be seen as complementary. RapidOWL offers an interesting list of guidelines, which could be used to design collaborative ontology engineering methodologies, and to better align existing ones to the more general principles of agile engineering and rapid prototyping.

Most methodologies, despite obviously advancing the field of ontology engineering and aligning it with developments such as Web 2.0, have found only modest adoption. Among the factors that one can assume to have led to this somehow disappointing state are the lack of user-friendly development environments, and the general historical limited uptake of semantic technologies. Nevertheless, the key ontology engineering practices promoted by this research have been leveraged by many ongoing projects, not last by those mentioned earlier in this article. This speaks in favor of a mature engineering field, with stable and consolidated principles and process-oriented components. To keep this advantage, collaborative ontology engineering will have to adjust to the latest developments in the Semantic Web area, where we witness a rapid uptake of 'open data' principles for large data sets using structured vocabularies. In the light of these developments, methodologies should investigate how these valuable amounts of publicly available data can be used to bootstrap and speed-up the engineering process. This has to include a revision of the processes and procedures currently in use, putting a stronger focus on data- and reuse-driven ontology development in contrast to development from scratch, but also on improvements of the optimal combination of manual and automatic ontology engineering activities. The availability of large amounts of semi-structured data and ontologies used to describe such data might also be leveraged to define different forms of discussion and argumentation mechanisms, which take into account the statistically grounded impact of an ontology or ontology fragment in relation to a particular design decision.

4 Ontology engineering tools

The availability of automated techniques and tools supporting the application of a methodology is acknowledged to play a crucial role in its adoption. Besides the functionality that is typically exposed by traditional ODEs (e.g. editing, storage, management, alignment), the collaborative scenario raises new challenges in terms of technological assistance: multi-user interfaces, integrated

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

communication channels and concurrency control, to name but a few (Tudorache *et al.*, 2008). Moreover, features related to ontology evolution, maintenance and versioning, which have been investigated also in the context of centralized ontology engineering, become particularly important in a decentralized scenario, with a higher number of participants in the engineering process working on different variants of the same ontology.

In this section, we present examples of collaborative ontology engineering tools developed over the last decade. Our primary focus is on those that support the process of building and maintaining ontologies at the schema level, and integrate any technical mechanism for collaboration and consensus finding. As such, tools such as TopBraid Composer[17], one of the most advanced and mature ODEs, are out of scope of this article, as they do not support collaborative work beyond features such as shared access and version control. An interesting example is also the Hozo ontology editor developed at the Osaka University (Kozaki *et al.*, 2009). Hozo's declared aim is to allow different stakeholders to explore an ontology according to a multitude of viewpoints, with the help of an automatically customized hierarchical structure (for instance, based on relationships such as is-a and part-of) and a conceptual map. In this way, the participants in a joint ontology engineering endeavor are provided with an environment by which a shared ontology is visualized along a variety of perspectives, which, the authors claim, contributes to an enhanced collaboration experience and a better understanding of an emergent knowledge structure. The tool has been applied for developing and using ontologies in the medical domain. Collaborative platforms for the creation and sharing of data expressed as RDF, sometimes as instances of a pre-defined ontology, such as Freebase[18], Semantic MediaWiki[19], DBin[20] and BOWiki (in the area of life sciences)[21] are as well out of the scope of our article, and deserve a dedicated survey.

4.1 Integrated development environments

Features for allowing multiple users to simultaneously access and edit the same ontology have been part of ontology engineering environments since their very beginning. Ontolingua, for instance, was one of the first to provide group access control and multi-user sessions (Farquhar *et al.*, 1997). Users are notified of the changes in the ontologies made by the other users via hyperlinks that describe such changes in terms of basic operations—add, delete and modify. WebOnto supports asynchronous and synchronous discussions on ontologies through the Tadzebao system (Domingue, 1998). As part of such discussions, a user issues an argument or comments on a particular argument through the placement of a poster visible to all other users. In OntoEdit, emphasis is put on facilitating interaction among users during the requirements analysis phase of the engineering process (Sure, 2003). With the OntoKick plug-in, the engineering team collaboratively specifies the domain and goal of the ontology, design guidelines, relevant resources and the means to extract structured knowledge out of these resources. A second OntoEdit plug-in, Mind2Onto, focuses on integration of the extracted structures towards an initial ontology. Collaboration is supported in the form of brainstorming sessions whose results are recorded with the help of the plug-in. It integrates the mind map tool for graphical representations of hierarchical structures. Moreover, it allows users to simultaneously refine the initial version of the ontology into a formal ontology based on the specified requirements. To do so, the system employs a locking and transaction protocol to ensure the model consistency and concurrency. Such features are also covered by the KAON ontology management framework (Maedche *et al.*, 2003), which, among other things, defines a set of interfaces for accessing distributed ontologies, and for consistency checking, change tracking or concurrency conflict detection.

[17] http://www.topquadrant.com/products/TB_Composer.html

[18] http://www.freebase.com/

[19] http://semantic-mediawiki.org/

[20] http://www.dbin.org/

[21] www.bowiki.net/

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

With ontology-based technology, gaining more and more popularity outside research labs industry-strength ontology engineering environments providing fully fledged collaboration support have become available. The NeOn Toolkit and Collaborative Protégé are two of the most prominent freely available examples. The NeOn Toolkit is developed as a set of Eclipse plug-ins for engineering interconnected ontologies[22]. It offers standard features for ontology development, including ontology editing, browsing and import/export (F-logic, OWL), as well as more advanced plug-ins for visual modeling (OntoModel), ontology reuse (Watson, Oyster), ontology learning (Text2Onto), ontology alignment (R2O, FOAM) and collaboration (WikiFactory, Cicero). WikiFactory enables the automatic creation of semantic wiki-based Web sites, their dynamical management at run time, and the synchronization between wiki content and OWL ontologies. In this way, specific aspects of an ontology engineering process, most notably those that take most benefit from an interactive environment, can be carried out in a wiki, and their results transferred into an ontology. Complementary, Cicero allows asynchronous discussions on ontology-engineering-related issues, and supports decision making. It is based on a simplified version of DILIGENT (as discussed in Section 3.3).

Collaborative Protégé is an alternative version of the existing base Protégé system, and probably the most comprehensive tool for collaborative ontology engineering currently available[23]. It implements features for discussion and conflict resolution in order to facilitate interaction between the participants. The system allows multiple users to edit the same ontology in multi-user or standalone mode. In addition to common editing operations, the system enables annotation with pre-defined annotation types for both ontological primitives (i.e. classes, properties, individuals) and ontology changes (i.e. class creation/deletion/renaming). In this way, it allows the user to document the ontology engineering process, in particular the rationale for taking specific decisions, in a systematic way. Collaborative Protégé furthermore offers search and filter functionalities for accessing such user annotations. It also provides discussion threads for users to reply to comments and a chat channel for direct communication. Decisions follow two types of voting mechanisms, a five-star and agree/disagree type of voting, implemented and represented as annotation types in the system.

Wiki-based tools

Wikis have received increasing attention in the Semantic Web community over the last 5 years[24]. This popularity is probably due to the user-friendliness of the core technology—a feature that is not necessarily characteristic to semantic applications—and to their focus on collaborative and community aspects. Existing semantic wikis primarily support the creation of semantic (instance) data expressed in Semantic Web languages such as RDF(S) or OWL; the development of the underlying ontologies is addressed—just as in the case of native ontology editors—mostly at implementation level, while the collaborative nature of wikis is assumed to inherently ease the consensus building within the engineering team. Exceptions from this are, for instance, the argumentation tool Cicero, which was previously mentioned in the context of the NeOn Toolkit, but also the tools that will be presented in the remainder of this section.

IkeWiki (Schaffert, 2006) is a semantic wiki system for collaborative ontology engineering. It aims to formalize informal texts into formal ontologies using interactive user interfaces. Users can annotate pages and links between pages semantically in RDFS and OWL. These annotations are utilized for context-specific presentation of pages, advance querying and consistency checking as well as summarizing conclusions. IkeWiki offers a WYSIWYG editor using AJAX technology to communicate with the server and supports OWL-RDFS reasoning in order to derive implicit information from the facts stored in the knowledge base. Similar features

[22] http://www.neon-toolkit.org

[23] http://protegewiki.stanford.edu/index.php/Collaborative_Protege

[24] See, for instance, the series of workshops on semantic wiki topics at http://www.semwiki.org/

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

are provided by OntoWiki[25], which also includes an alternative visualization for geographical data in the form of Google Maps and calendars automatically generated from the semantic statements stored in the system with the purpose to ease the understanding of semantic information for non-experts (Auer *et al*., 2007).

Similarly to Cicero, the coefficientMakna (Tempich *et al*., 2007) system captures the ontology engineering discussions as instances of the argumentation ontology based on DILIGENT. It allows users to query data for monitoring the status of discussions, progress, and possible conflicts as well as reconstruct the rationale behind certain decisions.

Usability is one of the main concerns of myOntology system (Siorpaes & Hepp, 2007). To allow non-experts to participate in the process, myOntology focuses primarily on engineering light-weight ontologies, and implements several visualization techniques such as tag clouds and topic maps, and automatically builds links to Wikipedia and Flickr[26] for documentation purposes.

4.2 Comparative analysis

In order to perform a comparative analysis of the tools we surveyed, we have defined a list of features that are acknowledged to be particularly challenging from a technical or an organizational point of view in collaborative ontology engineering projects. In general, these features refer either to the development and evolution of the ontology in a multi-user mode, or to the process model according to which the participants interact. Tempich provides a similar features list, which is, however, slightly biased towards the DILIGENT methodology (Tempich, 2006).

Key roles: As mentioned in Section 2, projects in this area adopt various role models specifying the policies according to which stakeholders can contribute to an ontology. But, there is no commonly agreed view on the distinction of a set of standard roles in collaborative ontology engineering and thus it is traceable that the same fact holds for ontology engineering tools. In most cases, the name for a role results from some tool-dependent duties or permissions and not from those in the ontology development process. However, it is possible to draw the line between those users who are allowed to perform changes to the ontology (ontology lead editors), those who use the ontology, give feedback of any form and propose changes (ontology contributors), and those who simply use an ontology (ontology users). Since the distinction between the latter two roles is quite hard to capture, we additionally assume that the ontology contributor performs an explicit feedback in a way that is accessible and usable by the ontology development tool. That yields that the ontology editor may also name reviewers, super users, moderators or administrators. For the ontology contributor, we also find the names of contributors, providers and regular contributors while the ontology users is named content consumer in some cases.

Ontology development and evolution: The main technical challenges in collaborative ontology engineering are related to multi-user access, in particular at the knowledge level, and version management. The shared ontology evolves to accommodate the requirements of the various stakeholders and new versions are released as consensus among these stakeholders is achieved. At the same time, localized variants of the ontology at a particular site will be in use. Consequently, tools have to provide functionality for managing several versions of an ontology in order to allow participants to work with the version that best fits theirs needs and expectations, as well as the usual concurrency control for handling multiple edits on a given ontology. One can differentiate between two collaborative editing modes: synchronous and asynchronous. In a synchronous mode, multiple users simultaneously edit the same ontology with changes taking effect immediately. An asynchronous mode allows users to modify an

[25] http://aksw.org/Projects/OntoWiki

[26] http://www.flickr.com/

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

ontology or a partial ontology, and then submitting their changes to the main version. In order to ensure the consistency of the ontological content during the decentralized development process, it is important that tools implement consistency checking mechanisms, based on a built-in or externally accessible inference engine.

In addition to these specific features, collaborative ontology development benefits from the availability of all kinds of features that facilitate the interaction with ontological content—be that searching, browsing, editing or ontology alignment—in particular for non-expert users. The most important feature in this respect is probably ontology visualization, which provides techniques by which users can easily get a better insight about the content and structure of an ontology. A hierarchy of concepts is the most widely adopted visualization technique. It represents the taxonomical backbone of the ontology through multiple tree views with expanding and contracting levels. A 2D graph visualization may provide more effective presentation in graphical views, illustrating an ontology with many features such as panning and zooming. Another feature is related to the conformance to standard representation formats that enables the development of an ontology in a multitude of technical settings, thus the import and export formats are important.

Collaboration: A basic component of a decentralized ontology engineering approach are the communication channels. Where communication is concerned, integrated communication channels as well as documentation support—for instance, in the form of annotations of ontological primitives—enable effective collaboration within geographically distributed teams. In addition, consensus-building techniques should be put in place in order to structure the discussions and mediate potential conflicts.

Usage: For a proper understanding and a valuable conclusion about the scalable and reliable function of any tool, its usage in well-described real-world scenarios is crucial. This feature does not only provide an insight on how well-developed and -maintained a tool is in general, but potentially also about specific application areas where a tool has been proved and tested exhaustively and could represent a quasi standard.

Tables 3–11 below elaborate on the features offered by each of the aforementioned collaborative ontology engineering tools. Our analysis focuses on those tools that are still under active development and maintenance, and that can be effectively used in ontology engineering projects at present. These tools are representative for the state of the art in this area; they can be divided into two main categories—integrated development environments (IDEs) and wiki-based tools.

In the second category, we observe differences in the degree to which the particulars of formal semantic representations are hidden from the user with the help of abstract conceptual models and visualization techniques. Both aspects are crucial for ensuring a wide usability of the corresponding system, and indirectly, for the adoption of ontology engineering practices across communities of practice. Tools such as **myOntology** and **OntoWiki**, but also **SemanticMediaWiki** that was not considered in this work due to its slightly different focus, offer interesting features in this respect. Another differentiating aspect is related to the expressiveness of the ontologies supported by wiki-based platforms, as most of them trade richer ontology engineering features for ease-of-use and a low barrier of entry. In the choice of the ontology engineering technology optimally matching the needs of a specific project setting one should take into account these issues, and be aware of the fact that the development of an expressive, highly axiomatized ontology will require the use of a traditional (collaborative) ontology editor, possibly on top of a wiki-based system that facilitates knowledge sharing and exchange in an informal manner, but does not support the formalization of complex conceptualizations going beyond classes and relationships between them. **myOntology** and **IkeWiki** provide a feasible mixture of the two; within **myOntology**, for instance, it is possible to develop ontologies in terms of classes, attributes and relationships, to document these entities using various means and to align between different ontologies.

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

Table 3 Shared access and concurrent working

NeOn Toolkit	Local working copy from a distributed version repository and conflict resolution mechanism for concurrently committed changes
Protégé	Multiple clients can edit simultaneously the same ontology hosted on a Protégé server (concurrent mode), changes are immediately visible to other users (concurrent mode), multiple users access the same ontology in succession stored on a shared network drive (consecutive mode)
myOntology	Wiki-based concurrency control at the page level
coefficientMakna	Wiki-based access and editing to each resource individually, wiki-based concurrency control at the page level
Ikewiki	Wiki-based access and editing to each resource individually, wiki-based concurrency control at the page level
OntoWiki	Wiki-based access and editing as well as inline and view-based editing, wiki-based concurrency control at the page level

Table 4 Ontology versioning

Neon Toolkit	Distributed version control system, OWL diff plug-in
Protégé	Repository of versions, check-in/check-out mechanisms, version comparison (diff)
myOntology	Wiki-like version control of all pages, versioning of ontologies calculated from the consensus of an engineering group, direct deployment of all consensual ontologies for use by other people and applications
coefficientMakna	Wiki-like version control of all pages, versioning of ontologies calculated from the consensus of an engineering group, direct deployment of all consensual ontologies for use by other people and applications
Ikewiki	Wiki-like version control of all pages, diff, and rollback, direct deployment of all consensual ontologies for use by other people and applications
OntoWiki	Wiki-like version control of all pages, diff, and rollback, direct deployment of all consensual ontologies for use by other people and applications

Table 5 Visualization

Neon Toolkit	Concept and property trees, alternative visualizations (e.g. graph) via plug-ins
Protégé	Concept and property trees, alternative visualizations (e.g. graph) via plug-ins
myOntology	Tag cloud, icons for specific primitives, info box
coefficientMakna	Wiki pages for ontology resources with specific icons per ontology primitives, infobox with collected information about the semantics
Ikewiki	Infoboxes as an overview over incoming and outgoing RDF edge, context-dependent visualizations
OntoWiki	Context-dependent views (e.g. maps or calendars) for the visualization of ontological data, wiki-like pages

Table 6 Consistency checking

Neon Toolkit	Inconsistency handler plug-in, RaDON plug-in for inconsistency detection within networks of ontologies
Protégé	FaCT++ reasoning engine, Pellet reasoning engine, different reasoning engines as plug-in
myOntology	Manual, community-driven
coefficientMakna	JENA reasoning engine
IkeWiki	JENA reasoning engine
OntoWiki	DL-Learner plug-in for OWL reasoning and consistency checking

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

Table 7 Import/export format

Neon Toolkit	OWL, F-Logic, EMF, UML, XML
Protégé	OWL, RDF, OBO, KRSS2, UML, XML
myOntology	OWL, RDF
coefficientMakna	OWL, RDF
IkeWiki	OWL, RDF, Wiki Interchange Format
OntoWiki	OWL, RDF

Table 8 Roles management

NeOn Toolkit	Contributors (Cicero)
Protégé	Flexible model of roles formalized as an ontology
myOntology	Content consumers, content providers, content reviewers, super users
coefficientMakna	Moderator, contributors
IkeWiki	Regular contributors, administrators
OntoWiki	Regular contributors

Table 9 Collaborative communication

NeOn Toolkit	CICERO wiki plug-in supports discussions conforming the DILIGENT argumentation framework
Protégé	Collaborative annotations of both ontology components and ontology changes
myOntology	Discussion page
coefficientMakna	Discussion pages for the argumentation with reference to design issues and implementation ideas with specific icons per argumentation type
IkeWiki	Threaded discussions for each page
OntoWiki	No explicit communication support

Table 10 Consensus-building techniques

NeOn Toolkit	CICERO wiki plug-in allows the tracking of the design rationale and detection of inconsistencies in argumentations
Protégé	No automatic consensus building but annotations can automatically evaluated
myOntology	No consensus building techniques
coefficientMakna	Discussions conforming the DILIGENT argumentation ontology, decisions incorporate the semantics of the argumentation process automatically
IkeWiki	Threaded discussions without any computational support
OntoWiki	No consensus building techniques

Shared access and version control is usually offered as an integral feature of the underlying wiki software, with tools such as myOntology extending this functionality in order to be able to support different role models and access policies. Consensus-building is covered by the **coefficientMakna** wiki, which realizes structured argumentation flows (conforming the DILIGENT argumentation ontology) related to design issues and ontological primitives. The import and export formats supported by the IDEs are to a large extent common and ranges from RDF(s), OWL, UML, to XML. These tools also implement specific solutions for shared access and version control, which differ in detail. The application of these tools is, in contrast to the wiki-based approaches, intended for distributed, but defined group of stakeholders that pursue an ontology engineering endeavor in

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

Table 11 Usage

NeOn Toolkit	Production of a network of populated fisheries ontologies for a ontology-driven fish stock depletion assessment system, development of ontologies for pharmaceutical case studies, ontology development for a semantic nomenclature, ontology development for pharma case studies invoice management
Protégé	Tool for CommonKADS, ontology modeling in a large number of projects and a variety of domains
myOntology	Collaborative ontology development for Semantic Web-based e-commerce, collaborative ontology creation for knowledge management
coefficientMakna	Students ontology engineering project within a lecture
IkeWiki	Internal knowledge base at a research group, store tutorials of an EU project, conference wiki, prototype for representing mathematical knowledge, prototype of the QVIZ platform (archive platform)
OntoWiki	System for intra-corporate semantic collaboration, base technology for a collaborative portal for tourism related information, ontological collaboration platform, intranet knowledge base

a particular domain of interest—therefore the life sciences domain offers many examples (see also Section 2). Collaborative communication channels and consensus building techniques are, however, still very important. Specific models are sometimes recommended by engineering methodology—for instance, **DILIGENT** emphasizes the importance of an editorial board, while the approach of **Holsapple and Joshi** uses a Delphi-driven expert panel to steer the conceptualization process. **Protégé** implements annotation functionality to enable user comments with reference to classes, properties and instances, as well as to changes performed to the ontology. Annotations can be browsed and filtered, but an automated calculation of the degree of consensus within a team with respect to a particular issue is not possible. This feature is supported by the **NeOn Toolkit** in combination with the Cicero wiki plug-in. In this way, discussions can be structured and recorded according to the DILIGENT argumentation ontology in a similar fashion as it is provided by **coefficientMakna**. Consistency checking is typically supported by IDEs, but also by some of the surveyed wikis, though the links between potential inconsistencies and the argumentation and discussion items could be improved.

5 Conclusions and outlook

In this article, we surveyed some of the most important state-of-the-art methodologies and tools for collaborative ontology engineering. We considered six methodologies that have emerged in the last years in the ontology engineering research community. The methodologies were selected based on their impact in the research community and beyond, as reflected by the adoption of the corresponding approaches by external organizations and initiatives, and by the quality of the related publications. Similar criteria were used for the choice of tools, in addition to their development status. Our assessment was based on a survey of the literature and documentation available, and on our own experience of many years in collaborative ontology engineering projects. All tools have undergone tests of their ground functionality carried out in a hands-on lab setting. To allow for a systematic comparative analysis, we have identified a number of criteria that are acknowledged to be essential in collaborative ontology engineering scenarios. Each approach was briefly described in terms of these criteria and a comparison was performed to identify their strengths and weaknesses.

Our study revealed that collaboration ontology engineering is a maturing, but active research field, in particular in terms of technological development. While there are a series of elaborated methodologies, none of them has found full adoption in real-world ontology-related projects, though there is evidence from case studies that many of their best practices and recommendations have proven useful. Tool support is available, both in forms of ODEs with collaborative features,

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

and as customized wiki platforms that can be used also by non-experts at the cost of reduced knowledge representation expressivity. In order to further advance the state-of-the-art in the field, there is a need for additional case studies; such case studies should offer insights about the types of features that are actually in use and give recommendations about how to achieve a better integration between expert and wiki-based tools. A qualitative, rather than merely incremental improvement could be enabled by leveraging the Web of data as a valuable input for the creation of widely accepted, shared ontologies. The means to achieve this vary from developing methodologies that pay proper account to offering a combination of human and computational intelligence, to tools providing access and using Linked Open Data during design and maintenance.

Collaborative approaches have found wide adoption in particular vertical domains, in which a community of interest, possibly much more limited than originally expected, uses methodologies such as DILIGENT and tools such as Collaborative Protégé to develop and maintain valuable ontologies. In this context wiki-based technology is still far from reaching its full potential; however, if we consider the success this technology has experienced in other scenarios, the focus should be less on using this technology to conceptualize and formalize knowledge in form of ontologies and more in referencing existing ontologies and populating them with real instances.

Numerous initiatives in domains such as life sciences, agriculture or eCommerce demonstrate that ontology engineering practices and development environments are usable and useful, at least when it comes to projects with a clearly defined scope and purpose, operated by a decentralized group of stakeholders. In contrast, the breakthrough toward fully open, community-driven initiatives has still not happened, despite the availability of several wiki-based ontology engineering platforms with interesting functionality and high usability. These platforms enjoy great popularity at the creation of structured, semantically represented content, a scenario that shows many similarities with ontology population. Freebase and Semantic Media Wiki are two of the most prominent examples thereof. We understand this state-of-affairs as an indicator of the real nature of the collaborative ontology engineering, which seem to have been ignored by the wiki-driven research conducted in the field in the last couple of years. A convincing use case for Web-scale ontology engineering is still missing—despite supportive initiatives such as VoCamps.

A last aspect that deserves particular attention is related to consensus building techniques. Though it seems that the R&D community around ontology engineering shares the same understanding of the features that need to be supported (IBIS model, structured discussions, logging of modeling decisions and rationale thereof, communication channels), there is no evidence how this rich range of features are used in ongoing, successful projects. Detailed case studies and experience reports would be helpful for defining a research roadmap for collaborative ontology engineering for the next years. Several recent technology trends could have an impact on the future directions of research and development in the field; among others, Linked Open Data. As more and more (RDF) data are published online, the need for shared ontologies describing the meaning and structure of this data will become essential for the effective usage of the data. The collaborative ontology engineering community will have to adjust its methods and techniques to the particularities of this new setting, and leverage the huge volume of data on the Web as a valuable input for the machine-supported creation of joint ontologies. Such methods and techniques will have to put emphasis on the smooth integration between human and computational intelligence, using Web-based data as the primary instrument for developing new shared ontologies, be that by extracting structured knowledge from these sources, or by combining available ontologies. This has several consequences. Existing methodologies will have to show commitment to the transition from development from scratch to wide-scale reuse of an arbitrarily high number of ontologies. They will have to shift their focus from a tool-assisted, but primarily manual engineering process to a data-driven approach in which human involvement is optimally leveraged for the resolution of those issues that can not be feasibly automatized. Finally, collaboration and consensus-building procedures, and associated tool support will have to be adapted to reflect these changes.

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

References

Auer, S. & Herre, H. 2006. RapidOWL – An agile knowledge engineering methodology. In *Ershov Memorial Conference*, Virbitskaite, I. & Voronkov A. (eds). Lecture Notes in Computer Science **4378**, 424–430. Springer.

Auer, S. & Pieterse, B. 2005. "Vernetzte Kirche": Building a Semantic Web. In *Proceedings of the 1st International ISWC Workshop on Semantic Web Case Studies and Best Practices for eBusiness SWCASE 2005*, Dublin.

Auer, S., Tramp, S., Lehmann, J. & Riechert, T. 2007. OntoWiki: a tool for social, semantic collaboration. In *Proceedings of the Workshop on Social and Collaborative Construction of Structured Knowledge CKC2007 at the 16th International World Wide Web Conference WWW2007*, Edinburgh.

Berners-Lee, T., Hendler, J. & Lassila, O. 2001. The Semantic Web. *Scientific American* **284**(5), 34–43.

Braun, S., Kunzmann, C. & Schmidt, A. 2010. People tagging & ontology maturing: towards collaborative competence management. In *From CSCW to Web2.0: European Developments in Collaborative Design Selected Papers from COOP08*, *Computer Supported Cooperative Work*, Randall, D. & Salembier, P. (eds). Springer.

Braun, S., Schmidt, A., Walter, A., Nagypal, G. & Zacharias, V. 2007. Ontology maturing: a collaborative Web 2.0 approach to ontology engineering. In *Proceedings of the Workshop on Social and Collaborative Construction of Structured Knowledge at the 16th International World Wide Web Conference (WWW 2007)*, Edinburgh.

Braun, S., Schmidt, A., Walter, A. & Zacharias, V. 2008. Using the ontology maturing process model for searching, managing and retrieving resources with semantic technologies. In *OnTheMove Federated Conferences 2008 (DAO, COOP, GADA, ODBASE)*, Monterrey, Mexico. Lecture Notes in Computer Science, 1568–1578. Springer.

Braun, S. & Zacharias, V. 2010. SOBOLEO – Editor and Repository for Living Ontologies. In *Proceedings of the 1st International Workshop on Ontology Repository and Editors for the Semantic Work ORES2010 at the Extended Semantic Web Conference ESWC2010*, Hersonissos, Greece.

Brickley, D. & Guha, R. V. 2004. 'RDF Vocabulary Description Language 1.0: RDF Schema'. Retrieved 4 April 2013 from `http://www.w3.org/TR/rdf-schema/`

Casanovas, P., Casellas, N., Tempich, C., Vrandecić, D. & Benjamins, R. 2007. 'Opjk and diligent: Ontology modeling in a distributed environment'. *Artificial Intelligence Law* **15**, 171–186.

De Leenheer, P., Christiaens, S. & Meersman, R. 2009. Business semantics management: a case study for competency-centric HRM. *Journal of Computers in Industry: Special Issue about Semantic Web Computing in Industry* **61**(8), 760–775.

De Moor, A., De Leenheer, P. & Meersmann, R. 2006. DOGMA-MESS: a meaning evolution support system for interorganizational ontology engineering. In *Conceptual Structures: Inspiration and Application*, Schärfe, H., Hitzler, P. & Øhrstrøm, P. (eds). Lecture Notes in Computer Science **4068**, 189–202. Springer.

Dellschaft, K., Engelbrecht, H., Barreto, J. M., Rutenbeck, S. & Staab, S. 2008. Cicero: tracking design rationale in collaborative ontology engineering. In *Proceedings of the ESWC 2008 Demo Session*, Teneriffe, Spain.

Domingue, J. 1998. Tadzebao and WebOnto: discussing, browsing, and editing ontologies on the Web. In *Eleventh Workshop on Knowledge Acquisition, Modeling and Management*, 18–23 April 1998, Banff, Alberta, Canada.

Dunnette, M. D., Campbell, J. D. & Jaastad, K. 1963. The effect of group participation on brainstorming effectiveness for two industrial samples. *Journal of Applied Psychology* **47**, 30–37.

Farquhar, A., Fikes, R. & Rice, J. 1997. The ontolingua server: a tool for collaborative ontology construction.

Fensel, D. 2001. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer.

Fernández-López, M. & Gómez-Pérez, A. 2002. Overview and analysis of methodologies for building ontologies. *The Knowledge Engineering Review* **17**(2), 129–156.

Fernandez-Lopez, M., Gomez-Perez, A. & Juristo, N. 1997. Methontology: from ontological art towards ontological engineering. In *AAAI97 Spring Symposium Series on Ontological Engineering*, Stanford, USA.

Fox, M. S. & Gruninger, M. 1998. Enterprise modeling. *AI Magazine* 109–121.

Gomez-Perez, A., Fernandez-Lopez, M. & Corcho, O. 2004. *Ontological Engineering, Advanced Information and Knowledge Processing*. Springer.

Guarino, N. 1998. *Formal Ontology and Information Systems*. IOS Press, 3–15.

Hepp, M. 2007. Possible ontologies: how reality constrains the development of relevant ontologies. *IEEE Internet Computing* **11**(7), 96–102.

Holsapple, C. W. & Joshi, K. D. 2002a. A collaborative approach to ontology design. *Communications of the ACM* **45**(2), 42–47.

Holsapple, C. W. & Joshi, K. D. 2002b. Knowledge manipulation activities: results of a Delphi study. *Information and Management* **39**(6), 477–490.

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press

IEEE Computer Society 1990. IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.121990.

Jarrar, M. & Meersman, R. 2009. Ontology engineering – the DOGMA approach. In *Advances in Web Semantics I*, Dillon, T. S., Chang, E., Meersman, R. & Sycara, K. (eds). Springer-Verlag, 7–34.

Knublauch, H. 2002. *An Agile Development Methodology for Knowledge-Based Systems Including a Java Framework for Knowledge Modeling and Appropriate Tool Support*. PhD thesis, Universitt Ulm. Fakultät für Informatik.

Kotis, K. 2008. On supporting hcome-3o ontology argumentation using semantic wiki technology. In *OTM '08: Proceedings of the OTM Confederated International Workshops and Posters on On the Move to Meaningful Internet Systems*, 193–199, Madeira, Portugal.

Kotis, K. & Vouros, G. 2005. Human-centered ontology engineering: the hcome methodology. *Knowledge and Information Systems* **10**(1), 109–131.

Kozaki, K., Hirota, T., Kou, H., Ohta, M. & Mizoguchi, R. 2009. Viewpoint management for multi-perspective issues of ontologies. *Proceedings of the Poster and Demo Track of the International Semantic Web Conference ISWC 2009*, Washington DC, USA.

Kunz, W. & Rittel, H. W. J. 1970. Issues as Elements of Information Systems. Working Paper 131, Institute of Urban and Regional Development, University of California.

Linstone, H. A. & Turoff, M. 1975. *The Delphi Method: Techniques and Applications*. Addison-Wesley Educational Publishers Inc.

Luczak-Rösch, M., Coskun, G., Paschke, A., Rothe, M. & Tolksdorf, R. 2010. Svont – version control of owl ontologies on the concept level. In *Proceedings of the 5th International Workshop on Applications of Semantic Technologies (AST 2010), part of INFORMATIK 2010*. Lecture Notes in Informatics, Gesellschaft für Informatik **176**, 79–84.

Maedche, A., Motik, B. & Stojanovic, L. 2003. Managing multiple and distributed ontologies on the semantic web. *The VLDB Journal* **12**(4), 286–302.

Mann, W. C. & Thompson, S. A. 1987. Rhetorical structure theory: a theory of text organization. In *The Structure of Discourse*, Polanyi, L. (ed.). Ablex Publishing Corp, 85–96.

Neches, R., Fikes, R. E., Finin, T., Gruber, T. R., Senator, T. & Swartout, W. R. 1991. Enabling technology for knowledge sharing. *AI Magazine* **12**(3), 35–56.

Noy, N. & Musen, M. A. 2007. Ontology versioning in an antology management framework. *IEEE Intelligent System* **19**, 6–13.

Patel-Schneider, P. F., Hayes, P. & Horrocks, I. 2004. Owl web ontology language semantics and abstract syntax. Retrieved 4 April 2013 from `http://www.w3.org/TR/owl-absyn/`

Schaffert, S. 2006. Ikewiki: a semantic wiki for collaborative knowledge management.

Siorpaes, K. & Hepp, M. 2007. myOntology: the marriage of collective intelligence and ontology engineering. In *Proceedings of the Workshop Bridging the Gap between Semantic Web and Web 2.0 at the ESWC 2007*, Innsbruck, Austria.

Spyns, P., Tang, Y. & Meersman, R. 2007. *A Model Theory Inspired Collaborative Ontology Engineering Methodology*. Technical report, VUB StarLab.

Sure, Y. 2002. On-to-knowledge: ontology based knowledge management tools and their application. *Künstliche Intelligenz* **1**, 35–37.

Sure, Y. 2003. *Methodology, Tools and Case Studies for Ontology Based Knowledge Management*. PhD thesis, University of Karlsruhe.

Swartout, B., Patil, R., Knight, K. & Russ, T. 1996. Toward distributed use of large-scale ontologies. In *Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada.

Tempich, C. 2006. *Ontology Engineering and Routing in Distributed Knowledge Management Scenarios*. PhD thesis, University of Karlsruhe.

Tempich, C., Simperl, E., Luczak, M., Studer, R. & Pinto, H. S. 2007. Argumentation-based ontology engineering. *IEEE Intelligent Systems* **22**(6), 52–59.

Tudorache, T., Noy, N. F., Tu, S. & Musen, M. A. 2008. Supporting collaborative ontology development in Protégé. In *ISWC'08: Proceedings of the 7th International Conference on The Semantic Web*, Sheth, A. P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T. & Thirunarayan, K. (eds). Springer, 17–32.

Uschold, M. & Grueninger, M. 1996. Ontologies: Principles, methods, and applications. *Knowledge Engineering Review* **11**(2), 93–155.

Uschold, M. & King, M. 1995. Towards a methodology for building ontologies.

Vrandecic, D., Pinto, S., Tempich, C. & Sure, Y. 2005. The diligent knowledge process. *Journal of Knowledge Management* **9**(5), 85–96.

https://doi.org/10.1017/S0269888913000192 Published online by Cambridge University Press