Expert Systems With Applications 224 (2023) 119968

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Loki – the semantic wiki for collaborative knowledge engineering

Krzysztof Kutt *, Grzegorz J. Nalepa

Jagiellonian University, Faculty of Physics, Astronomy and Applied Computer Science, Institute of Applied Computer Science, and Jagiellonian Human-Centered AI Lab, and Mark Kac Center for Complex Systems Research, S. Lojasiewicza 11, Kraków, 30-348, Poland

ARTICLE INFO

Dataset link: https://loki.re/

Keywords:
Knowledge engineering
Semantic wiki
Software engineering
Unit tests
Prolog

ABSTRACT

We present Loki, a semantic wiki designed to support the collaborative knowledge engineering process with the use of software engineering methods. Designed as a set of DokuWiki plug-ins, it provides a variety of knowledge representation methods, including semantic annotations, Prolog clauses, and business processes and rules oriented to specific tasks. Knowledge stored in Loki can be retrieved via SPARQL queries, in-line Semantic MediaWiki-like queries, or Prolog goals. Loki includes a number of useful features for a group of experts and knowledge engineers developing the wiki, such as knowledge visualization, ontology storage, or code hint and completion mechanism. Reasoning unit tests are also introduced to validate knowledge quality. The paper is complemented by the formulation of the collaborative knowledge engineering process and the description of experiments performed during Loki development to evaluate its functionality. Loki is available as free software at https://loki.re.

1. Introduction and motivation

In recent years, a shift from traditional knowledge engineering (KE) to group-based KE has been observed. This change corresponds to the transformation from a process in which the main role was played by knowledge engineers, to a group effort, in which domain experts are the main contributors (McGuinness, 2017). This is in line with the emerging movement of knowledge socialism, which promotes the development of technologies that facilitate the exchange of knowledge between different stakeholders for the public good (Peters, 2021). The research literature provides several terms for such a group-based KE process: collaborative KE, distributed KE, collective KE, and co-operative KE. There are no well-established definitions of what they mean (Jackson, 2016), and experts can often understand them in different ways (Noy et al., 2008), or use some of them interchangeably, for example, distributed KE and collaborative KE in Baumeister et al. (2016), or collaborative KE and collective KE in Nalepa (2010). However, they should be distinguished on the basis of the characteristic features indicated by experts and supported by dictionary definitions. Making this distinction transparent allows one to differentiate various situations where multiple users are involved in the creation of knowledge. It will also support us in the specification of the context of this work:

- *Distributed KE* is a general term that describes the KE process divided into a number of users who work together on a knowledge base using different terminals (Baumeister & Nalepa, 2009; Distributed system, 2015). This definition is similar to the notion of "distributed systems", which covers a wide range of different systems in which communication is provided by sending messages over the network (Coulouris et al., 2011).
- *Cooperative KE* is a process in which the problem is divided into separate fragments, as in the "divide and conquer" approach. Each participant is responsible for one part (Roschelle & Teasley, 1995). They pursue their own goals, but are willing to help others (Villines, 2014). They ultimately formulate knowledge that benefits each of them (Cooperative, 2015). An example of such a process can be found in HermesWiki (Reutelshoefer et al., 2010), where tasks are divided between students who can work on them independently.
- *Collaborative KE* is the joint participation of the project participants for a common purpose, although it may result from different motivations. In this scenario, users operate on the same part of the problem, associated with the occurrence of conflicts and the dynamics of opinions (Adrian et al., 2013; Richards, 2009). Collaborative KE examples are ACKTUS-dementia (Lindgren & Winnberg, 2010) and Dispedia (Elze et al., 2011) where patients and physicians share knowledge about the same diseases, as well as the Turkic thesaurus (Gatiatullin & Kubedinova, 2020) developed by a group of engineers and domain experts.
- *Collective KE* is done by a whole group of people who aim towards a common goal, who have similar motivations and socioeconomic

* Corresponding author.
E-mail addresses: krzysztof.kutt@uj.edu.pl (K. Kutt), gjn@gjn.re (G.J. Nalepa).

https://doi.org/10.1016/j.eswa.2023.119968
Received 21 July 2022; Received in revised form 17 March 2023; Accepted 21 March 2023
Available online 24 March 2023

interests. The members of the group are equal, which means a similar level of skills (Collective, 2015; Villines, 2014). Examples of this group are the German Tourism Knowledge Graph (Simsek et al., 2023), International Classification of Diseases (ICD-11) (Tudorache et al., 2013) and the OpenResearch.org semantic wiki (Vahdati et al., 2019). Projects are developed by a group of similar participants (tourism industry professionals, physicians, and researchers) with a shared goal, using their own resources.

Our work is established within *collaborative knowledge engineering* (referred to as CKE). It is the most common variant of the KE performed by groups. However, it presents many challenges that still await a proper solution and requires appropriate support tools (Noy et al., 2008). It is also worth noting that collaborative KE and collective KE differ primarily on levels of organization and communication, so probably collective KE problems may be similar to the ones in collaborative KE, and proposed solutions will also apply to collective KE.

The difference between KE and CKE, i.e., the shift of the main role in the knowledge engineering process to domain experts, is not only the highlighting of a different group of stakeholders (McGuinness, 2017; Simsek et al., 2023). In a traditional KE, an engineer trained in computer science learns the domain and prepares the solution for the assumed application using tools that allow them full control over the knowledge base. This often leads to a complex and expressive model that allows for making rich inferences. In the CKE, the knowledge base is mainly created by domain experts and the community, who usually do not have detailed knowledge of computer science, but want to rely on knowledge engineering methods to make the knowledge base they develop useful (Thanachawengsakul & Wannapiroon, 2021). Such an approach usually implies a simple but flexible model, focused on creating multiple instances (Simsek et al., 2023). This change therefore entails the need to redesign many of the established knowledge engineering tools and techniques and adapt them to this new situation. Although there are successful deployments based on a set of spreadsheets (Debruyne et al., 2022), CKE usually requires the creation of dedicated groupware tools that support it (Kutt, 2018; Torres, 2018).

In our work, we argue that one of the most suitable tools for CKE is *semantic wikis*. They are built on classic wiki systems that have gained immense popularity, due to their usability, enhanced by easy installation and maintenance without long introduction training (Prasarnphanich & Wagner, 2009). Wikis are pretty simple and robust systems for collaboration, as indicated by the relatively high quality of the Wikipedia project (Junjie et al., 2010). They were further extended by using Semantic Web technology. The term "semantic" is here understood as a meaning of the information (Allemang & Hendler, 2011), and as a machine-processability of the information, which will also allow information exchange between agents (Hitzler, 2021). Incorporation of selected technologies, such as URIs for entity identification (Allemang & Hendler, 2011), RDF for knowledge representation (Tomaszuk & Hyland-Wood, 2020), OWL for more sophisticated relations in the form of an ontology (OWL Working Group, 2009), and SPARQL for querying the knowledge base (Harris & Seaborne, 2013) extends the capabilities of wiki systems for automatic processing of knowledge contained in the wiki and for conducting inference. Such enriched wikis, called semantic wikis, became useful tools for CKE: On the one hand, they are easy to use and, on the other, they offer the ability to process knowledge at a certain level of formalization (Baumeister et al., 2011a). However, in order for them to be fully useful in the CKE area, they need to be enhanced to support all the requirements for CKE, identified in Section 2.

Furthermore, we observe that today CKE can take advantage of decades of software engineering research (Sommerville, 2004). We believe that this is due to the fact that there are many software engineering methods that address challenges similar to those of CKE. This includes the gradual and dynamic group-based development process. This is why in our work we propose a specific CKE process that organizes the main KE phases and extends them with a number of software engineering methods. Furthermore, at the level of the groupware tool supporting the CKE process, we use selected techniques established in practical software engineering.

In the context mentioned above, the objective of this paper is twofold. First, we introduce a specific and detailed workflow or process for CKE. In this specification, we combine the methods of classic KE with the above-mentioned software engineering techniques. Second, we present the *Loki* groupware tool, which is a semantic wiki that we developed to support our CKE workflow. It is targeted at a group consisting of knowledge engineers(s) and domain experts with short training in specific knowledge representation techniques used in a project. The choice of specific solutions for this system was based on our assumption that specific solutions developed recently in software engineering can be useful in addressing current KE issues. Among such practical solutions, there are, e.g., agile methodology and unit tests. Finally, to make our considerations more specific, in this paper, the KE process is narrowed to the creation of various types of conceptual models, such as structured vocabularies.

The rest of the paper is organized as follows. In Section 2 we discuss the requirements for tools that support CKE. Then, we provide the specification of our CKE process in Section 3. The main part of the paper, that is, the presentation of the Loki wiki, is placed in Section 4. We discuss the use of reasoning tests to control the quality of the knowledge base in Section 5. The experiments we carried out during Loki development are described in Section 6. Related work and comparison with existing wiki systems in the context of CKE are discussed in Section 7. The summary and future works are given in Section 8.

2. Requirements for CKE

KE experts seem to agree that specific use cases of collaboration require adequate tools. Therefore, it is not possible and there will never be a universally accepted tool for all possible situations. On the one hand, there may be cases that require a specific change acceptance protocol and, on the other hand, the situations in which everyone can make immediate changes (Noy et al., 2008). However, there is the possibility of specifying the list of common requirements that should be addressed to provide the appropriate CKE tools. There were many attempts to define such a list (Alobaid et al., 2019; Baumeister & Reutelshoefer, 2011; John & Melster, 2004; Noy et al., 2008; Nurminen et al., 2003; Paraiso et al., 2016; Richards, 2007, 2009; Santhosh, 2019; Wang & Chen, 2004). Each of them paid attention to other aspects of the problem. We have reviewed them and eventually partitioned them into six fields of CKE support (Kutt, 2018):

- *Agile CKE process* – there is a need for a mature agile methodology for CKE, as it has not yet been developed (Furth & Baumeister, 2014), in particular:
  - R1. Development should be fast and done in a robust agile way,
  - R2. Different roles of users involved in the process should be identified and supported in a proper way,
  - R3. There should be a possibility for users to specify expectations which will have an impact on the direction of the project's development.
- *Supporting tools* – CKE systems must be built on a base that allows collaboration:
  - R4. Provide compatibility with mainstream tools, standards and methods, especially domain specific ones,
  - R5. Manage current KB status and make it possible to download the stable version of KB,
  - R6. Adapt the system to specific project requirements.
- *Knowledge representation and reasoning* – as domain knowledge should be managed by domain experts, not knowledge engineers, there is a need to develop a suitable knowledge representation:

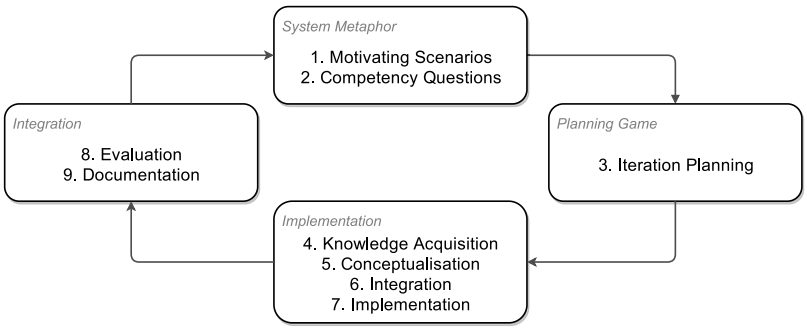

Fig. 1. CKE agile process.

R7. The representation should be adapted specifically to the project and group that will use the CKE tool and should be easy to use for experts and powerful in terms of inference capabilities.

- *Quality management* – there is a need to ensure the proper level of knowledge base quality:

R8. Provide automatic methods for knowledge verification, e.g., knowledge consistency check or execution of automatic tests,
R9. Determine the credibility of user expertise and sources of knowledge, especially in open environments where everyone can make changes,
R10. Keep in mind that users are domain experts: give them a possibility to approve, disagree, or discuss with others, but also be prepared to resolve user conflicts.

- *Change management* – management of factual changes in the collaborative setting is more difficult than in the classical KE:

R11. There is a need for a robust versioning mechanism that takes care of different types of change and covers various characteristics of the process.

- *User involvement* – since participation in the CKE process is not spontaneous, various incentives should be considered to motivate people:

R12. Take care of usability, that is, a powerful and easy-to-use interface, compatibility with established practices, proper visualization, and automation capabilities.
R13. Consider the use of gamification techniques. However, it should be noted that some users prefer "hard work", and do not find the game elements motivating at all.

This list has set the scope of the work undertaken to adjust an existing semantic wiki system to the needs of the CKE process. However, to achieve this goal, we next provide a complete specification of a CKE process.

3. Formulation of the CKE complete process

3.1. CKE process description

The proposed CKE process is the result of the analysis of existing KE methodologies (Fernández-López et al., 1997; Gruninger & Fox, 1994; Holsapple & Joshi, 2002; Noy & McGuinness, 2001; Uschold & King, 1995). The intermediate results of these analyses were presented earlier (Nalepa, Slazynski, et al., 2015). They were then combined with inspirations from agile methodology in software engineering, particularly Scrum (Schwaber & Sutherland, 2020). An important role was also played by the general outline of the agile CKE process proposed in Baumeister (2004), which is refined and clarified here. Finally, the CKE process consists of 9 steps grouped into 4 stages (cf. Fig. 1) performed in multiple iterations until the expected state of the system is reached:

1. *System metaphor*: each iteration begins with the *motivating scenarios* step to regularly discuss the use cases and scenarios and to match them to the current needs. The *competency questions* are also specified here. They can be written down formally, e.g., in a form of reasoning unit tests.
2. *Planning game*: the *iteration planning* step is explicitly introduced into the proposed methodology to reflect an important step in the agile process. During this step, specific tasks are created, estimated and selected for the current iteration.
3. *Implementation*: the most important stage of the iteration is actual integration of knowledge into the system. It covers the whole process that starts with *knowledge acquisition* from experts, domain texts and other resources. The gathered knowledge is then *conceptualized* and the *integration* with the existing well-established vocabulary is considered. Finally, the knowledge is *implemented* using selected formalized system, e.g., in a semantic wiki.
4. *Integration*: each iteration is concluded by a proper *evaluation* of the knowledge base. In addition to the verification of knowledge at the formal level, it should also be checked at the conceptual level to check how reliably it represents the domain, e.g., using the reasoning unit tests specified previously. The entire process is supplemented by documentation in a form of e.g., competency questions and reasoning unit tests.

During the initial iterations of the presented process, a common language is established (Holsapple & Joshi, 2002; Silva et al., 2009), and later, in the next, knowledge is created from less formal to more formal. This is achieved by a team in which specific roles can be distinguished: *the product owner*, who represents the client and has the whole system vision, *the CKE process master* (named as scrum master in Scrum methodology), who oversees the project course, is responsible for the proper implementation of the process, ensures that all obstacles preventing the team from completing the project are removed, *the team of domain experts* and 1–2 *knowledge engineers*, who support the team. One can also distinguish some roles within the team: *the adders*, who create a lot of material, but not always well semantically marked, *the synthesizers*, who take care of semantics and concepts interrelations, *the cops*, who are responsible for imposing standards and schemes and then they are monitoring them (cf. Majchrzak et al., 2006).

3.2. European Wiki case study

The sample *European Wiki* project is introduced here to present the process proposed above in action. It is developed by a group of colleagues that will work online to develop the wiki with a description of all European countries. They will use it as a support system during their travels across the continent. As there is no "client" in this setting,

a group member has been selected as the owner of the product, who will supervise the main purpose of the project. In addition, one of them was selected as the CKE process master who will take care of the whole process. Finally, all of them except the product owner will work as a development team that will build the wiki. To simplify the description, only the first iteration will be presented here,together with the KB made up of three wiki pages prepared by two users (`kkutt` and `yoda`).

1. The process begins with the definition of *motivating scenarios*. During this step, three use cases for the *European Wiki* were proposed: (a) I want to see cities: A, B and C. In what order do I have to visit them to be able to use direct flights? (b) I want to see all European capitals. List all of them. (c) I am in city A. What is interesting here?
2. In the next step more specific *competency questions* were proposed. They were formally specified using the reasoning unit tests.
3. When such system specification is described, the *iteration planning* step takes place. In the first iteration of the *European Wiki* project, three tasks were defined: (a) Describe London, (b) Describe Paris, (c) Describe Cracow.
4. After planning game, the actual implementation stage begins. During this block, two users made six changes to the *European Wiki* project and discussed the difficult points. In the background, the semantic changelog was created and parsed to provide a gamification-based incentives for users. All task-related decisions were marked on an iteration board (a scrum board; a place with a list of all tasks selected for current iteration grouped into three main categories: “to do”, “in progress” and “done”), i.e., user `kkutt` assigned task (a) to himself and moved it to the “in progress” section.
5. Each iteration is concluded by an *evaluation*. During this step, reasoning unit tests results are consulted to check the current knowledge quality and the level of requirements fulfillment.
6. Everything is supplemented by a *documentation* step. Within the first iteration of *European Wiki* project, the most important action was to comment the first design decision made: cities will be described by a `city` category. A note appeared in the proper place in the system.

To make such a process successful, it is necessary to develop the right tool. It should allow such a process to be carried out, facilitate it and, moreover, it should meet the requirements listed in Section 2. We argue that the Loki platform is such a system. It is described in the next section.

4. The Loki platform

4.1. The architecture of the Loki ecosystem

Originally, we developed the preliminary version of Loki (a short for “Logic in wiki” or “Logic-based wiki”)[1] engine in 2009 to show the possibility of combining wiki systems with the representation of knowledge in Prolog. As a base, the DokuWiki engine[2] was selected. Loki was created as a plugin for this wiki system (Nalepa, 2010, 2011). Later, we extended them to subsequent modules, as part of the Ph.D. thesis (Kutt, 2018), leading to the emergence of the ecosystem presented in Fig. 2. It is now made up of five groups of modules. All of them are developed as plugins for DokuWiki that provide additional functionalities for Loki engine.

- *Loki core.* This group represents the minimum needed for the basic functioning of a semantic wiki. The DokuWiki engine manages the basic functions (page editing, user accounts), while the Loki engine handles the processing of knowledge contained in the pages in the form of semantic annotations and in the form of Prolog’s code. It adds the appropriate notation, allows one to query the knowledge base, and allows one to manage quality in the form of reasoning unit tests. Among the core modules, there are also RDFLoki for knowledge visualization on given page and LokiOntology for storing the simple ontology within wiki using selected concepts from the OWL language (OWL Working Group, 2009).
- *Knowledge representation modules.* These modules provide the possibility to use additional knowledge representations than the core ones, i.e., text, images, tables, semantic annotations, Prolog clauses. Currently, we provide modules for three such representations: Business Process Model and Notation (BPMN; business processes)[3] (Nalepa et al., 2012), Semantics of Business Vocabulary and Rules (SBVR; business rules)[4] (Nalepa, Kluza, & Kaczor, 2015) and eXtended Tabular Trees (XTT2; production rules)[5] (Nalepa, 2018).
- *Semantic change graph.* These modules are related to the creation (PROV) and visualization (PROVViz) of the semantic change graph. This is an RDF-based graph that covers various aspects of CKE process in the wiki: statistics of triples added/removed, metrics of KB’s subsequent states, results of reasoning unit tests, rates from wiki users (by RevisionsRater plugin). As an RDF graph, it forms a (meta-)knowledge base that can be further processed, leading to more thoughtful analyses of the CKE process.
- *User-oriented modules.* This group consists of three modules. The first provides gamification elements as incentives for users in an KE process. The second module focuses on the development of a user interface that adapts to the needs of identified user groups, e.g., adders that create a lot of content and synthesizers that are aimed at knowledge consistency. Finally, the ScrumIDE module is an attempt to connect a wiki with an IDE that supports agile software development.
- *DokuWiki plugins.* Due to the fact that Loki uses the DokuWiki system as a base, one can extend Loki features with all available DokuWiki plugins.[6] They give a possibility to e.g., create multilingual wiki, compile TEX commands on wiki pages or create image galleries inside wiki.

The subject of this paper is to present the current state of *Loki core*. The description of the remaining modules is beyond the scope of this paper.

4.2. DokuWiki engine

The DokuWiki system was selected as the basis for Loki, as it is a very compact and fast implementation of the Wiki concept. Controls the creation of pages and saves them in text files, purely using the Unix filesystem with no need for a database engine. Provides a simple markup through which one can create headers, links, lists, format text, or insert drawings (cf. Fig. 3). DokuWiki engine handles all the processing of wiki pages that can be easily extended through the plugin mechanism that modifies *syntax* processing, engine *actions*, and page *rendering*. All Loki-related functionalities presented below are implemented as these plugins. Specifically, this architecture enforces all Loki-related actions to be executed by PHP scripts triggered when

[1] For documentation and downloads see: https://loki.re/.
[2] See: https://www.dokuwiki.org/.
[3] See: http://bpmn.org, https://loki.re/wiki/docs:bpwiki-about.
[4] See: https://www.omg.org/spec/SBVR/ and https://loki.re/wiki/docs:sbvr-about.
[5] See: https://loki.re/wiki/docs:xttviewer.
[6] See: https://www.dokuwiki.org/plugins.

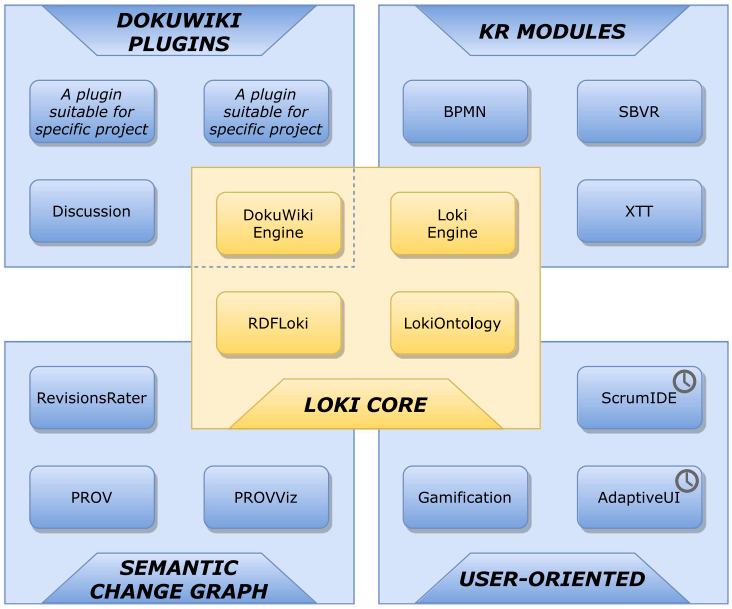

Fig. 2. Architecture of Loki ecosystem.

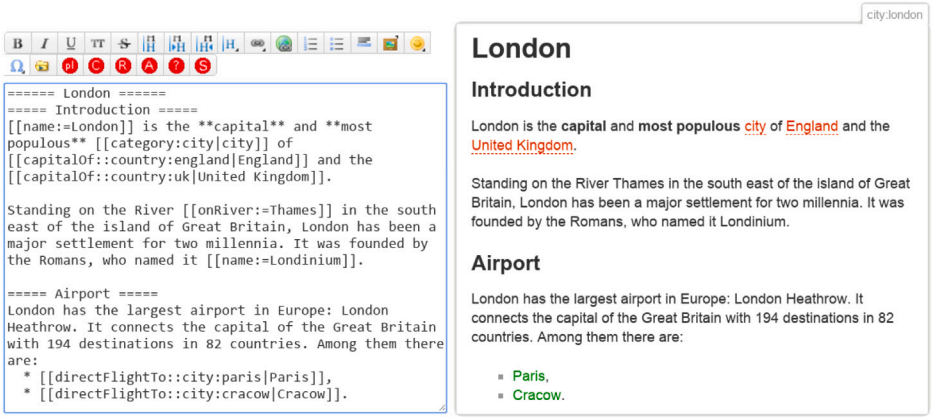

Fig. 3. Sample Loki page: Wiki markup (on the left) and rendered page (on the right).

the page is loaded (e.g., executing SPARQL queries, cf. Section 4.3), when the editor is opened (e.g., loading the code hint mechanism, cf. Section 4.6), or when the page is saved (e.g., running unit tests, cf. Section 5).

4.3. Knowledge engineering with Loki engine

For the implementation of Loki, we selected SWI-Prolog,[7] as it is a mature, popular, and well supported open implementation of Prolog. The basic knowledge representation in semantic wikis is semantic annotations. To provide compatibility with existing solutions, it was decided to use annotation markup similar to the one available in Semantic MediaWiki (SMW), one of popular semantic wikis (see Section 7 for description and comparison with Loki). When the wiki page is saved, it is parsed and the annotations are translated into Prolog code for further processing. There are three types of annotation available in Loki. All of them will be depicted with examples from the `city:london` page of *European Wiki* (see Fig. 3 for the full wiki page).

- *Category* specifies the type of concept described on a given page.
  Example: London page describes a city.
  Annotation: `[[category:city]]`
  RDF triple: `:city:london rdf:type :city .`
  Prolog statement: `wiki_category('city:london','city').`
- *Relation* describes the connection between two concepts (wiki pages). It is equivalent to an object property in OWL.
  Example: London is the capital of England.
  Annotation: `[[capitalOf::country:england]]`
  RDF triple: `:city:london :capitalOf :country:england .`
  Prolog statement:
  `wiki_relation('city:london','capitalOf','country:england').`
- *Attribute* connects a concept (wiki page) with a literal value. It is equivalent to a data property in OWL.
  Example: The name of the city is "London".
  Annotation: `[[name:= London]]`
  RDF triple: `:city:london :name 'London' .`
  Prolog statement: `wiki_attribute('city:london','name','London')` .

Annotations can be supplemented by the Prolog code placed directly in the wiki text with the `<pl>` tags. The result is a homogeneous knowledge base that can be queried in three different ways (cf. Fig. 4):

[7] See: http://www.swi-prolog.org/.

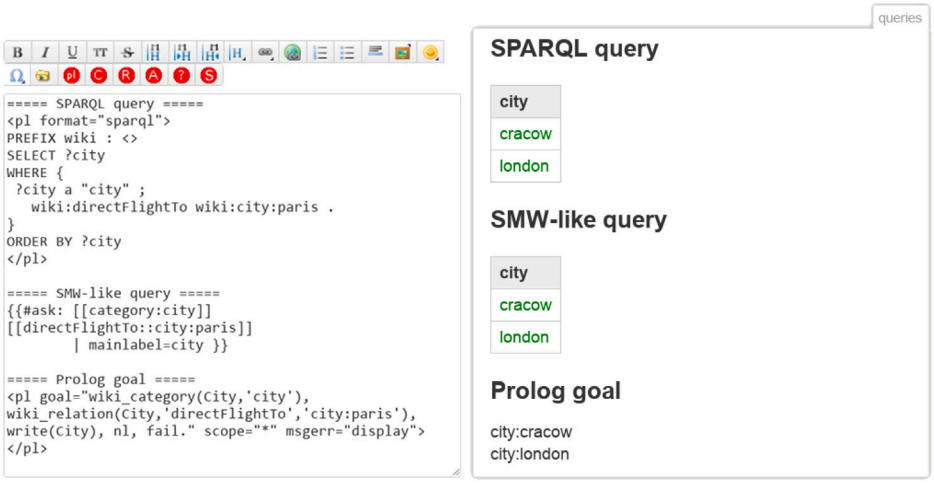

Fig. 4. Three types of queries (on the left) with sample results (on the right).

- *SPARQL query language* (Harris & Seaborne, 2013). A subset of the SELECT, ASK, and DESCRIBE constructs is supported.[8] Such queries may be placed in the wiki text (see Fig. 4) or may be asked using the built-in SPARQL Endpoint. It allows users to specify queries to a wiki via a simple web interface without the need to create a separate wiki page with the query. Queries can also be remotely sent via HTTP GET Requests. Results may be returned as HTML, JSON or RDF/XML. This functionality is currently a standard that allows data to be retrieved from various RDF-based systems.
- *SMW-like queries*. Loki adopted the query language used in Semantic MediaWiki (Krötzsch & Vrandecic, 2011). It is easier to understand than SPARQL and Prolog because it offers a syntax similar to semantic annotations used in the wiki (see Fig. 4), but at the cost of simpler querying capabilities.
- *Prolog goals*. One can also specify arbitrary Prolog goals within the wiki. They can be used to retrieve knowledge written in both forms: annotations and Prolog code, as they form one knowledge base.

Besides storing and querying the knowledge in the wiki, Loki also has the functionality to export each page as RDF/XML files. After pressing the "Export to RDF" button, all semantic annotations saved on a given page are translated into a fully compatible RDF/XML document and available for download. A more detailed technical specification of the Loki engine is available in Nalepa (2009) and on the Loki website.[9]

4.4. Visualization of knowledge

Knowledge written in the wiki is not visible to the user at a glance. One simply sees the text and links on the page, but does not know which fragments are formally written in the form of annotations. It is possible only when the edition form is opened and the wiki text is visible. To facilitate a quick look at the knowledge available on a given page, a visualization module has been created. It is based on the RDF/XML export provided by the Loki engine (described in Section 4.3). Due to the fact that the export function saves the generated RDF/XML file in the wiki file system, it can be easily used later. For visualization purposes, this file is parsed by the jQuery script, which immediately generates a graph in SVG format using the `<svg>` tags supported by HTML5. The sample graph generated for the London page (see Fig. 3) is presented in Fig. 5.

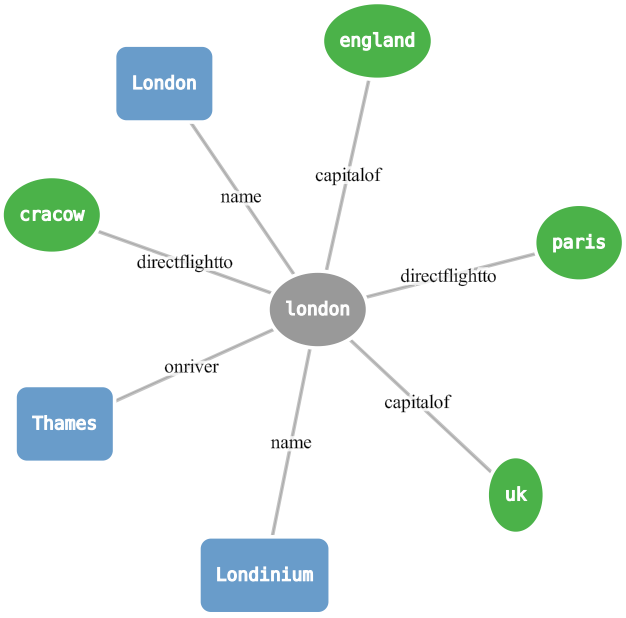

Fig. 5. RDF graph visualization for London page (cf. Fig. 3).

Generated graphs use different types of nodes to differentiate between different types of elements: the current page is represented as a gray node, other wiki pages, i.e., objects of object properties, are shown as green nodes, while literals, i.e., objects of data properties, are blue nodes. All properties are represented by arcs. This representation is consistent with the convention used in the W3C RDF Recommendation (Gandon & Schreiber, 2014), where the RDF resources are represented as ovals and literals as rectangles. All oval nodes are links to proper wiki pages, providing a simple and intuitive way to explore the whole knowledge graph by going through the small visualizations on each page.

4.5. Ontology storage

A separate module for ontology storage was developed for Loki, allowing an explicit definition of the ontology and its structure. First, to provide an easy way to store the general idea of ontology used in one place, and secondly to help users with stating new annotations on

[8] See: https://loki.re/wiki/docs:userman#sparql_queries.
[9] See: https://loki.re/.

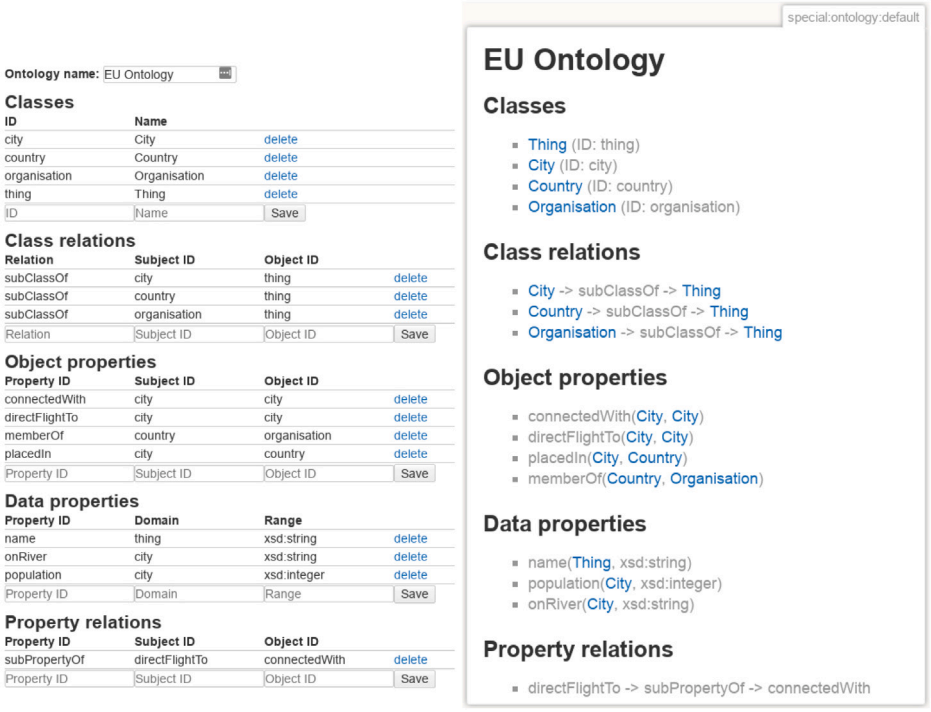

Fig. 6. Ontology editor (on the left) and visualization (on the right).

wiki pages – with an ontology in one place, it is easy to look at the available relations and classes and copy them to new wiki page.

Given the architecture of Loki, we decided to store ontologies in `special: ontology:{name}`[10] pages, where *{name}* represents the name of the ontology stored on a specific page: `default` for base ontology, `foaf` for Friend-of-a-Friend,[11] etc. Ontologies are stored on these pages using XML syntax. When combined with XSLT stylesheets,[12] it offers great visualization capabilities. Two stylesheets were prepared: one with simple edition form, that gives a possibility to enter a new statement or delete an existing one, and the second to view the ontology (see Fig. 6).

4.6. Code hint and completion mechanism

The code hint and completion mechanism was selected as the most useful extension for Loki, as a solution for a problem that appears in all performed experiments (see Section 6). That is, user-made typos or errors in the semantic annotations led to the emergence of knowledge bases that were not fully useful and required time-consuming debugging. As annotations in Loki are case-sensitive, even mistake `[[category:name]]` with `[[category:Name]]` leads to two very different statements.

To overcome the discussed problems, as a part of the lokiontology plugin used also for ontology storage within Loki, we implemented a code hint mechanism that follows the rules using JavaScript (JS).[13] The mechanism aims at suggesting matching annotations, classes, and the available object and data properties. Its operation is based on a continuous scan for annotations in the input entered by the user. It is necessary to define an ontology in the wiki (cf. Section 4.5) for the whole mechanism to work. First, the user must specify the category. All categories defined in the ontology are suggested as a hint. Proposing/verifying properties is based on a custom JS function that checks which properties are defined for a given category in the ontology. Object suggestion/validation is a two-phase process. First, the object type defined for a given category and property in the ontology is determined with a JS function. Then, a SPARQL query is executed to generate a list of all instances of this type.

For example, consider the ontology presented in Fig. 6. On the current page, the user has already put the `[[category:city]]` statement. As a result, `directFlightTo`, `onRiver` and other properties are proposed (cf. Fig. 7). If the user selects the former, all `city` class instances are proposed as objects (cf. Fig. 7). If the second is selected, the user has to write the string value, so nothing is proposed.

5. Controlling knowledge base quality with reasoning unit tests

CKE is related not only to ways of gathering and integrating knowledge or its subsequent processing. The means to ensure the required level of quality of this knowledge are also very important (see List R8 on page 6). In software engineering, there are various types of tests to address this issue. Among them, there are unit tests that are used on a daily basis to provide a quick quality check with immediate results that will signal potential problems with the code (Khorikov, 2020).

This idea was formally adopted into KE (Vrandecic & Gangemi, 2006) and implemented in the KnowWE semantic wiki (Baumeister et al., 2012). It was also incorporated into Loki with further improvements. First of all, it is important to note that these tests can and should be created by system users (i.e., domain experts) to reflect the real system expectations. This allows unit tests to check not only the quality of "abstract" knowledge, but also how well the system actually performs its task according to the user's requirements. Second, within the unit tests module implemented in KnowWE, all tests are thrown into a single "big bag", making it difficult to use with a larger number of tests. In Loki, the hierarchy is proposed, so similar tests can be grouped in the nested structure, with more detailed tests placed deeper in the hierarchy. The advantage of this solution lies in the ability to not run tests when they are not needed. This means, for example, that when any of the parent space tests fails, it is assumed that currently checked part

[10] `special:` namespace is used in Loki to store special pages, e.g., the list of concepts, the URI resolution system, etc.

[11] See: http://www.foaf-project.org/.

[12] Needs xslt plugin: https://www.dokuwiki.org/plugin:xslt.

[13] See: https://loki.re/wiki/docs:lokiontology.

Fig. 7. Code hint mechanism in Loki: List of properties for `city` (on the left), list of possible objects for `directFlightTo` (in the center) and errors highlighting (on the right).

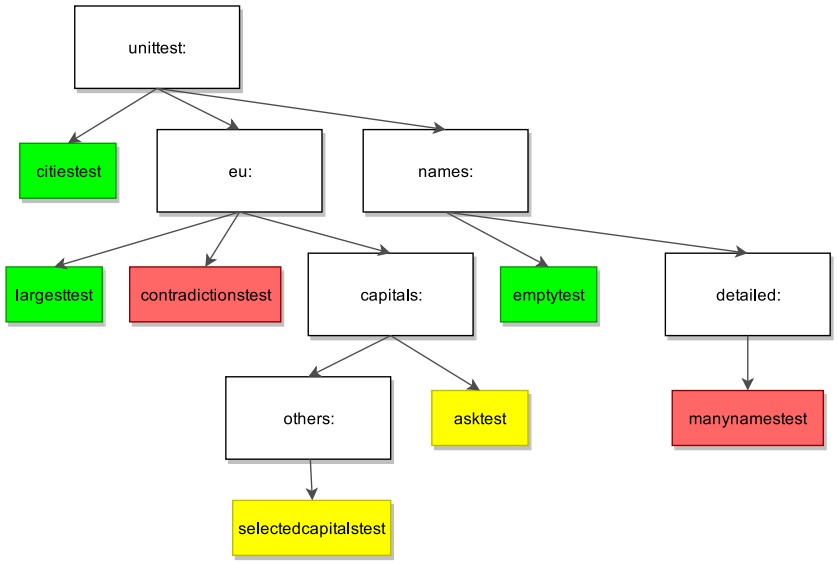

Fig. 8. Sample test structure for *European Wiki* project.

of the KB has errors, so nested tests are not executed, because they are very likely to fail. A sample tests structure, with indication of whether the test was passed (green), failed (red), or not executed (yellow), is shown in Fig. 8. The white nodes represent levels in the hierarchy.

The test structure within Loki is implemented as a tree of nested namespaces with the `unittest` namespace as the root. Each test within this structure is a separate page that consists of exactly one SPARQL query with a set of assertions that specify restrictions on the expected query result. The sample test structure in Fig. 8 can be represented as follows:

```
unittest:citiestest
unittest:eu:largesttest
unittest:eu:contradictionstest
unittest:eu:capitals:asktest
unittest:eu:capitals:others:selectedcapitalstest
unittest:names:emptytest
unittest:names:detailed:manynamestest
```

The whole test structure is executed after each change in the wiki, i.e., after every page save event. The results of the most recent test execution are presented on a corresponding page in a `unittestresults` namespace, e.g., for a test saved in `unittest:eu:largesttest`, the results are available on the `unittestresults:eu:largesttest` page. The test summary is generated as a table with their status in the test. `unittestsoverview` page (Fig. 9).

As mentioned above, each test page is made up of an SPARQL query and a set of assertions. Any Loki-valid SELECT, ASK, or DESCRIBE query is acceptable as a test. A set of possible assertions is available for each of the query types (see Tables 1–3). All of them are added to the test by putting a single line on the test page, according to the scheme:

Reasoning unit tests results:

Test	Status
citiestest	PASSED
eu:capitals:asktest	NOT EXECUTED
eu:capitals:others:selectedcapitalstest	NOT EXECUTED
eu:contradictionstest	FAILED
eu:largesttest	PASSED
names:detailed:manynamestest	FAILED
names:emptytest	PASSED

Fig. 9. Summary of tests. It represents the structure in Fig. 8.

```
[[unittest_assert_{type}:?{parameter}:{value}|{comment}]]
```

where: `type` is an assertion type (see Tables 1–3[14]), `parameter` is a name of query column used for comparison or special value, `value` is a value used for comparison in the assertion. There is also a place for optional `comment` to document the assertion. The sample query with two assertions taken from the *European Wiki* project is presented in Listing 1.

[14] As Loki treats all attributes' values as strings, there are only assertions for equality available. Other comparisons are not as useful for strings.

Table 1
Set of assertions for SELECT queries within Loki.

type	parameter	Description
anyequal allequal noneequal	Attribute name	Checks whether any row/all rows/no rows contain the given value of specified attribute
rowcount	equal, notequal, lessequal, less, greater, greaterequal	Checks the number of rows returned

```
===== Capitals test =====
<pl format="sparql">
PREFIX wiki: <>
SELECT ?page ?name
WHERE {
   ?page a "city" .
   ?page wiki:name ?name .
   ?page wiki:capitalOf ?country .
}
</pl>

[[unittest_assert_anyequal:?name:London|Is London one
      of the capitals?]]\\
[[unittest_assert_noneequal:?name:Cracow|Cracow is
     not one of the capitals!]]
```

Listing 1: A sample test from the *European Wiki* project.

Table 2
Set of assertions for ASK queries within Loki.

type	parameter	Description
asserttrue	Empty – one can omit the parameter: [[unittest_assert_asserttrue:?|]]	Checks whether the query returns true value
assertfalse		Checks whether the query returns false value

DESCRIBE queries require more attention, as they are not well documented by the W3C (Harris & Seaborne, 2013). Within Loki, they take a form similar to the SELECT queries , but instead of listing the selected pages , they return all attributes that characterize the selected pages; i.e., they return all triples where a given page is a subject. This characteristic gives the possibility to provide an additional type of assertion (`attributecount`) for checking the number of occurrences of a given attribute, e.g., to check if Canada has exactly two `official_lang` values (`english`, `french`) and Poland has only one value (`polish`). These cannot be validated by the `allequal` values, as they test only whether the requested value is available.

SPARQL Recommendation (Harris & Seaborne, 2013) also defines the CONSTRUCT query form which returns an RDF graph. They were not presented here due to the lack of implementation in Loki. However, it is worth noting that from the testing point of view, they work in the same way as SELECT queries. The main difference is that the result is not a table, but a graph. Therefore, for these queries, one could use the same set of assertions as for SELECT queries.

6. Experimental evaluation of Loki

We evaluated selected concepts of CKE, as well as the corresponding features of Loki during six experiments we conducted to evaluate specific hypotheses during modules' and CKE process' development, Each of these experiments was aimed at creating a knowledge base on the wiki by a group of computer science students. They were participants in two courses organized at AGH-UST: "Semantic Web Technologies" and "Artificial Intelligence Foundations". None of the participants had any previous experience with knowledge engineering methods. Before launching the experiments, each person attended a 1.5-hour training session on how to use the wiki.[15] All instances of the wiki were hosted on a university server and available on the Internet to all participants. General information and statistics of all experiments are given in Table 4. In the following subsections, we briefly characterize the most important results.

6.1. First experiment: Pokemons, Simpsons, et al.

In the first experiment, *CKE process* was carried out by a team of experts and a team of knowledge engineers who worked alternately and independently. Specifically, there were four independent processes led by four two-person expert teams.

The first phase was aimed at creating a wiki about the topic chosen by the team (Pokemons, Simpsons, Dragon Ball Z, and Drinks). The participants took on the role of domain experts. Based on their knowledge and on publicly available materials, they prepared "classic" wikis containing text, links, and media files. Then, the teams exchanged wiki systems, simulating the appearance of new actors: knowledge engineers. Within this phase, their task was to identify the right vocabulary and to use it to annotate the content, to build a "semantic" wiki. Finally, during the third phase, their task was to evaluate the content of the wiki, both text and semantic annotations.

The results of this experiment finally led to two conclusions. Firstly, during *CKE process*, domain experts and knowledge engineers must collaborate at every stage. Second, experiments with students should be conducted under more strictly controlled conditions, not performed occasionally in the students' free time.

6.2. Second experiment: CSP library

In the second experiment, students participated in the project aimed at transferring the knowledge available in the Constraint Satisfaction Problem Library (CSPLib)[16] to the wiki system and preparing relevant semantic annotations. In the experiment, a simple *CKE process* based on Ontology 101 (Noy & McGuinness, 2001) was implemented. It was done in an iterative manner. Each of the 5 iterations lasted 2 h. During the first iteration, the first set of competency questions for the CSP Library was developed. After that, ontology prototypes were created.

Subjective teacher observations and analysis of the reports prepared by the participants led to the following conclusions. First, users identified the lack of a proper code hint and completion mechanism as a major disadvantage. Then it was implemented as described in Section 4.6. Second, the formalization of the competency questions to SPARQL queries at the end of the second iteration is too late. In the final version of *CKE process* (Section 3), this issue is addressed by incorporation of the *reasoning unit tests* definition (in the form of SPARQL queries), as part of the *competency questions* step. It was also noted that tasks assignment may be done in a more controlled way. The final version of *CKE process* incorporates an iteration board to address this issue. Finally, three roles were observed during the process and implemented later on in the wiki: a leader, a scheme designer, and a developer.

[15] The training is available at https://loki.re/wiki/docs:tour and is updated with each major change in Loki.
[16] See: http://www.csplib.org/.

Table 3
Set of assertions for DESCRIBE queries within Loki.

type	parameter	Description
`anyequal` `allequal` `noneequal`	Attribute name	Checks whether any/all/no pages contain the given value of specified attribute
`attribute-countXX`		Checks whether all pages contain the given attribute in specified amount. `XX` should be replaced with one of the types of assertions: `lessequal`, `less`, `greater`, `greaterequal`, `equal`, `notequal`
`pagecount`	`equal`, `notequal`, `lessequal`, `less`, `greater`, `greaterequal`	Checks the number of pages returned

Table 4
Summary of the CKE experiments.

Exp.	Objectives	No. Participants	Changes	Wiki pages	Triples
1st	Evaluation of the basic CKE process in wiki	8	3000	345	1891
2nd	Extension of the CKE process based on Ontology 101 (Noy & McGuinness, 2001)	13	1186	265	821
3rd	Evaluation of Loki features after the 2nd experiment	15	665	202	1031
4th	Practical comparison of Loki to SMW	105	5046	285	2535
5th	Evaluation of the modules implementing the agile process	16	579	275	890
6th	Evaluation of the final version of the process and Loki	105	2807	436	431

6.3. Third experiment: Pubs in Cracow

The objective of the experiment was to evaluate the new Loki modules in action and to collect feedback on the improved *CKE process*. The topic of the project was selected by the participants. It was decided to create a wiki on pubs located in the center of Cracow. A new observation was that one person in each team had to be responsible for quality assurance, but in fact such a task may be automated. This was later addressed by the introduction of a code hint and completion mechanism that also checks the annotations and gives instant feedback of all errors found. Furthermore, the lack of a task assignment mechanism was problematic. To address it in a final version of *CKE process*, a *iteration planning* step was introduced that uses an iteration board.

6.4. Fourth experiment: Artificial intelligence class

In the fourth experiment, students created a wiki on topics of artificial intelligence covered during the "Artificial Intelligence Foundations" course. Among the main goals of the experiment there were: (1) to verify whether there are differences in difficulty levels of wiki usage between Loki and the popular Semantic MediaWiki (SMW), (2) to test the *CKE process* conducted by a larger group than in previous experiments. Due to the specific setting, that is, the fact that the project was done as a part of a course, the process was adjusted wrt to the original one (Reutelshoefer et al., 2010).

To address the first goal, a two-sided Mann–Whitney U test was used to check whether there were differences between the difficulty levels for Loki and SMW reported in the survey. This statistical test was selected as it is designed to determine whether two samples represent different distributions and does not require the assumption of normal distributions. The test results[17] do not allow the null hypothesis of no difference between Loki and SMW to be rejected for each of four measures of wiki aspects: overall wiki usage ($U = 463.0$, $p = .26$), a wiki text editor ($U = 633.0$, $p = .20$), semantic annotations ($U = 597.5$, $p = .46$), and queries ($U = 576.0$, $p = .66$).

Analysis of the process and feedback gathered from users led to the following main observations: (1) a 1-hour introductory training at the beginning was not sufficient for the purposes of the project, (2) groups of 50 people are definitely too big for self-management. The need for knowledge schemes, proper naming conventions, or even the language used for annotations was quickly noticed, (3) some users were not convinced of wiki technology. The most frequently pointed problems were related to the interface and were addressed in the next version of Loki. It is interesting that, even with these problems in mind, the vast majority (54 out of 66 votes) decided that the wiki project should be repeated during the next course edition. In conclusion, there are no significant differences between the difficulty reported for Loki and SMW.

6.5. Fifth experiment: Cookbook and movies KB

Once the complete set of Loki modules for *CKE agile process* was developed, the fifth experiment was conducted. It was done in two parallel ways: (1) students took part in a *CKE agile process* done in a way presented in Section 3 aimed at cookbook development within Loki, (2) the online evaluation call was announced on Loki web page[18] and sent to KE researchers at AGH-UST, as well as graduates of "Engineering of Intelligent Systems".

Among the objectives of this study, there were to (a) check whether all modules are compatible with each other, (b) conduct a usability study, and (c) verify whether the proposed framework improves *CKE*

[17] Results consists of two values. U is a Mann–Whitney test statistic, while p represents the probability that the null hypothesis of no difference is true. If the p score is less than 0.05, one can reject a null hypothesis and state that there is a difference.

[18] See: https://loki.re/wiki/evaluation2017.

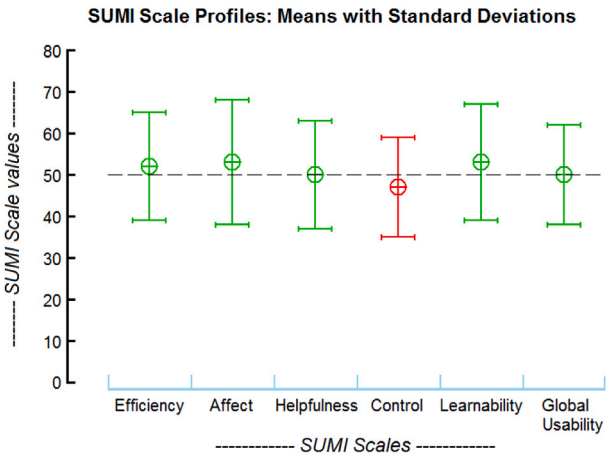

Fig. 10. Summary of SUMI scales.

process. At the end of the experiment, as well as at the end of the tasks related to the evaluation call, all participants completed the Software Usability Measurement Inventory (SUMI) (Kirakowski & Corbett, 1993). The questionnaire allows for the evaluation of global usability, as well as its specific aspects covered by five subscales. The results are compared with a standardization database that contains profiles for various software products. Each scale has a mean of 50 and a standard deviation of 10. As a result of the evaluation, the following observations can be made:

1. All modules smoothly interact with each other. No major flaws were noticed (the 1st goal).
2. 18 users filled out the SUMI inventory (see Fig. 10). The results indicates that users generally felt satisfied with the use of the Loki (the 2nd goal).
3. Proposed framework improved the *CKE process* (the 3rd goal): (a) in contrast to previous experiments, no one reported that was "the cop" responsible only for monitoring the quality. This task was supported by the *reasoning unit tests* module, as well as the hint and completion system. (b) thanks to the simple iteration board, everyone knew what tasks are currently being carried out in the wiki. Therefore, there was no redundancy in the work. It was proposed that more advanced iteration boards, like Trello[19] or JIRA,[20] may be used in the future, e.g., to group tasks.

6.6. Sixth experiment: Artificial intelligence class

The final experiment was a repetition of the fourth experiment (Section 6.4). There were again two groups, with two independent wiki instances: one of them was a pure Loki engine, here called "wiki A" (as described in previous papers (Nalepa, 2010, 2011) and summarized in Section 4.3) and the second was the new Loki core, "wiki B", (as presented in Fig. 2 and described in this paper). The goal of this experiment was to investigate how the new Loki core improves the CKE process on the wiki. Trello was used as a professional iteration board.

Wiki A was used by 58 participants, while wiki B by 47 students. The number of changes made in both wikis was comparable (1540 in wiki A and 1267[21] in wiki B) but in wiki B there were more wiki pages (257 vs. 179) and more triples created (274 vs. 157). It is noticeable that a small number of triples is created compared to the number from the fourth experiment (cf. Table 4). The construction of the experiment itself has not changed, but the way of evaluating participation was different. In the fourth experiment, the creation of relationships with other pages was more restrictively assessed, as a result of which participants were "forced" to create multiple triples if they wanted to complete the course with high scores. In the last experiment, the rules have been loosened, and it was enough to do one triple on a wiki page to "pass".

All improvements of Loki found in wiki B were found useful. From the new remarks that emerged after this experiment: The use of Trello as a professional task management tool was very well received. The transparent CKE Agile process made it clear what steps were being taken. Gamification mechanisms were found to be a useful addition to the knowledge base development process.

7. Related works and comparative evaluation

The first semantic wiki (Platypus) was proposed in 2004. Then, in 2005–2006 "the semantic wiki explosion" was observed when many semantic wiki implementations appeared. In subsequent years, more systems were created, but not as many as in the beginning. Bry et al. provided a comprehensive description of these systems and their history in 2012 (Bry et al., 2012).[22] Only systems that have been updated over the last 6 years (i.e., the latest release or commit was in the 2016–2022 period) are considered here: Semantic MediaWiki (Section 7.1) (SMW), KnowWE (Section 7.2), and OntoWiki (Section 7.3). The summary of their features with respect to CKE requirements (as defined in Section 2) is presented in Table 5. This summary also provides a comparison of these state-of-the-art wikis with Loki. Loki is described separately in two columns: the Loki engine (as described in Sections 4.2–4.3 and published previously in Nalepa (2010, 2011)) is presented in the "Loki engine (2011)" column, and a fully updated Loki core (as described in Section 4.2–5) is presented in the "Loki core (2022)" column. The WikiMeningitis system (Thiombiano et al., 2021) is not included, as there are no details on the source code or the ability to install the system. As the system is inspired by SMW, it probably has similar capabilities.

7.1. Semantic MediaWiki

Semantic MediaWiki (Krötzsch & Vrandecic, 2011; Krötzsch et al., 2007) (see Fig. 11) is an extension of the MediaWiki system.[23] The wiki extends MediaWiki markup to provide a way to describe categories, relations with other wiki pages (object properties), and relations with literal values (data properties):

```
Category: [[Category:City]]
Object property: [[Is capital of::United Kingdom]]
Data property: [[Has population::7,421,329]]
```

The wiki also provides the possibility to: define the ranges of the relations (using categories or data types), convert state values (e.g., from miles to kilometers), and specify the hierarchy of categories and relations (Krötzsch & Vrandecic, 2011). Notable was the attempt to create a distributed SMW system that stored knowledge on multiple servers to become independent of a single point of failure (Skaf-Molli et al., 2010).

The wiki knowledge base may be queried using SPARQL, as well as with a specific SMW's query language. To list all cities located in the United Kingdom with their population size, one can simply state:

[19] See: https://trello.com/.
[20] See: https://www.atlassian.com/software/jira.
[21] 1563 when the value is rescaled to the same number of participants.
[22] For list of current and "historical" semantic wikis see also: https://www.semanticweb.org/wiki/Semantic_Wiki_State_Of_The_Art.html and http://www.mkbergman.com/sweet-tools/.
[23] See: https://www.mediawiki.org/.

Table 5
Loki core compared to other semantic wikis with regard to the CKE requirements.

	SMW	KnowWE	OntoWiki	Loki engine (2011)	Loki core (2022)
Publications	Krötzsch and Vrandecic (2011), Krötzsch et al. (2007)	Baumeister et al. (2012, 2011b)	Frischmuth et al. (2015), Heino et al. (2009)	Nalepa (2010, 2011)	*Presented in this paper*
Web page	https://semantic-mediawiki.org/	https://www.d3web.de/	http://ontowiki.net/	http://loki.re/	http://loki.re/
Last update[a]	24.03.2022 (v. 4.0.1)	11.02.2022 (v. 12.7)	04.10.2016 (v. 1.0.0)	23.03.2022 (v. 2022-03-23)	23.03.2022 (v. 2022-03-23)
Technology	Extension to MediaWiki (PHP). Storage in MediaWiki relational database or any Triple Store[b]	Built one the JSPWiki (Java EE). Storage in relational database	PHP (Zend Framework). Storage in relational database or Triple store (via Erfurt API[c])	Extension to DokuWiki (PHP). Uses SWI-Prolog. Storage in plain text files	Extension to DokuWiki (PHP). Uses Loki engine as a base
Agile CKE process					
R1: Agile development	*Agile development is related to the appropriate CKE methodology, not specific tools, so all wikis may be used with CKE process*				
R2: User's roles	No, access rights only	No, access rights only	No, access rights only	No, access rights only	Yes, access rights & automatic identification of roles
R3: Expectations	Yes, Unit tests	Yes, Ontology & unit tests	Yes, Ontology	No	Yes, Ontology & unit tests
Supporting tools					
R4: Compatibility	Yes, RDF & SPARQL support	Yes, RDF & SPARQL support. Compatible with domain-specific standards	Yes, RDF & SPARQL support. Uses well-established vocabularies (SKOS, FOAF, ...)	Yes, export to RDF; SPARQL, BPMN & SBVR support; SMW-style queries	Yes, as in Loki engine + ontologies in OWL subset, changelog in PROV
R5: Current KB state	*In all Wikis except KnowWE and Loki core there are no automatic tests to validate current state, so the stable and unstable versions are not differentiated. Only KnowWE gives a possibility to download the last stable version of the KB*				
R6: Adaptation to project	*All Wikis have plugin systems that enable the possibility to connect with domain specific tools and adapt the system to the specific project needs*				
Knowledge representation and reasoning					
R7: Adjustable KRR methods	Yes: media, plain text, annotations, business processes, UML class diagrams	Yes: media, plain text, annotations, production rules, decision trees, decision tables, flowcharts, set-covering models	No, only forms with different types of fields' values	Yes: media, plain text, annotations, Prolog facts and clauses, XTT2 rules, business processes and rules[d]	Yes, as in Loki engine
Quality management					
R8: Auto quality checks	No	Yes, unit tests	No	No, only consistency checks limited to XTT2 rules	Yes, as in Loki engine + unit tests
R9: Credibility determination	No	No	No	No	Yes, for users & partially for sources used
R10: Tools for experts' conflicts	Yes, special page for discussion for each wiki page	No	Yes, discussion about small information chunks	Yes, special page for discussion for each wiki page	Yes, as in Loki engine + approval/disagreement mechanism
Change management					
R11: Versioning control	Yes, simple & linear	Yes, simple & linear	Yes, simple & linear	Yes, simple & linear	Yes, robust & graph-based
User involvement					
R12: Usability	Tested in experimental setting, giving good overall results (Zander et al., 2014)	Not tested	Not tested	Not tested	Tested in experimental setting, giving good overall results (Kutt, 2018)
R13: Gamification	Yes, very simple	No	No	No	Yes
Requirements fulfilled	**9**	**8**	**6**	**6**	**12**

[a] *Checked on 25.03.2022.*
[b] *See: https://www.semantic-mediawiki.org/wiki/Help:Using_SPARQL_and_RDF_stores.*
[c] *See: http://aksw.org/Projects/Erfurt.html.*
[d] *Processes and rules may be edited and visualized, but they cannot be queried.*

```
{{#ask:
  [[Category:City]]
  [[Located in::United Kingdom]]
  |?Population
}}
```

SMW queries can be used to define concepts; for example, concept `UK City` may be a result of `[[Category:City]] [[Located in::United Kingdom]]` query. Such concepts may be used in the same way as categories in future queries, allowing for grouping pages in a more sophisticated way (Krötzsch & Vrandecic, 2011).

Importantly, in addition to the core SMW team,[24] there is also a large community that not only utilizes the system (e.g., Pilkington & Pretorius, 2020; Willmes et al., 2018), but also creates their own modules to extend its functionality (e.g., RDFIO (Lampa et al., 2017)).

[24] See: https://www.semantic-mediawiki.org/wiki/Help:SMW_Project.

Fig. 11. Semantic MediaWiki wiki page (Krötzsch & Vrandecic, 2011).

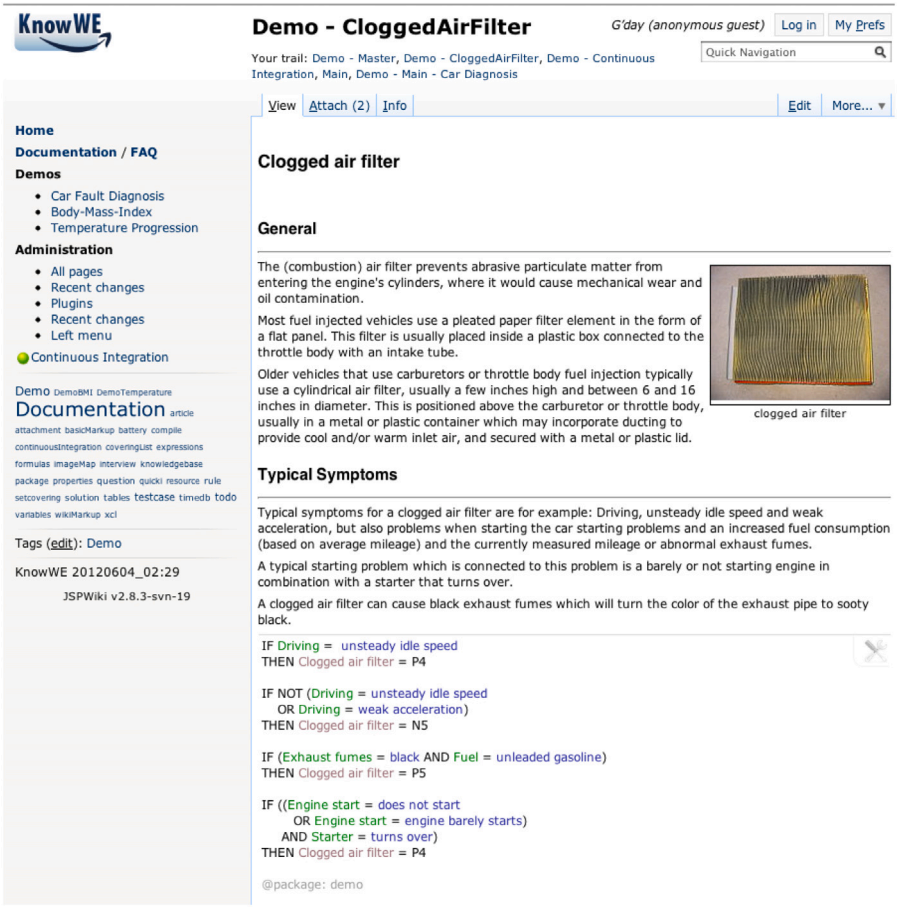

Fig. 12. KnowWE wiki page (Baumeister et al., 2012).

7.2. KnowWE

KnowWE (Knowledge Wiki Environment) (Baumeister et al., 2012, 2011b), based on JSPWiki,[25] is not a simple wiki with semantic annotations. It was developed as a decision support system that mixes knowledge at different formalization levels, ranging from pictures to strong problem solving knowledge, for example, production rules (Baumeister et al., 2012, 2011a) (Fig. 12). KnowWE is constantly evolving, including the introduction of new methods for quality assessment (Baumeister, 2020) and the automatic creation of technical knowledge from documents (Furth & Baumeister, 2017). The wiki was used to define clinical guidelines and HCI devices configuration (Baumeister et al., 2012), chemical substances assessment (Baumeister et al., 2016), ancient Greece description (Reutelshoefer et al., 2010), and more. KnowWE does not have a generic annotation mechanism. It uses markup and editors created for specific use cases to reduce complexity for domain experts (Baumeister et al., 2012; Krötzsch & Vrandecic, 2011).

[25] See: http://jspwiki.apache.org/.

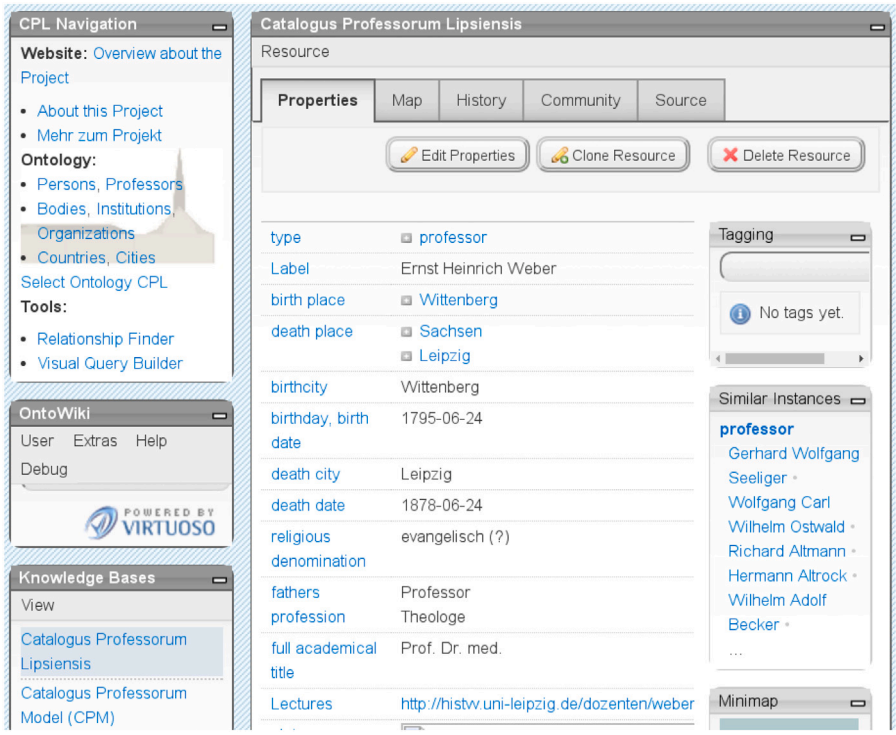

Fig. 13. OntoWiki wiki page about professor E. H. Weber.

7.3. OntoWiki

OntoWiki (Frischmuth et al., 2015; Heino et al., 2009) is not an extension of the conventional text-based Wiki. It is based on the pOWL editor, which is used for collaborative ontology engineering.[26] As a result, OntoWiki does not provide text pages with specific markup, but special forms (see Fig. 13) that structure knowledge into the form of an ontology. More precisely, OntoWiki is an overlay for an arbitrary RDF/OWL dataset, which allows its edition without a deep knowledge of ontology engineering (Frischmuth et al., 2015; Heino et al., 2009). OntoWiki knowledge can be queried locally using SPARQL queries, but it also allows external queries via the SPARQL endpoint and the LinkedData endpoint (Frischmuth et al., 2015).

It was used successfully in many situations, ranging from historical data in Catalogus Professorum Lipsiensis (Riechert et al., 2010), through medical diagnosis in Dispedia (Elze et al., 2011), to animal classification in the Caucasian Spiders project (Ermilov et al., 2011). The latter was done in the wilds of the Caucasus area, where the desktop client is not useful, so there is also a lightweight HTML5-based version of OntoWiki that can be browsed on mobile phones (Ermilov et al., 2011).

8. Conclusions

In this paper, the field of collaborative knowledge engineering is outlined. To clearly define this concept and differentiate it from related terms, a group-based knowledge engineering taxonomy was proposed. This allowed for further analysis of the CKE and its challenges, which led to the identification of 13 requirements for tools supporting the CKE. Among them, the need for a mature agile methodology for CKE appeared, as it has not yet been developed. To address this gap, we have prepared a detailed description of the collaborative knowledge engineering process, which derives both from knowledge engineering methods and from practices developed in software engineering.

The central part of the paper presents a description of the semantic wiki Loki, a groupware tool that implements the proposed CKE process. Designed as a set of DokuWiki plug-ins, it provides a variety of knowledge representations, including semantic annotations, Prolog clauses, business processes, and rules (in BPMN and SBVR notations, respectively). It can be easily extended to other ones using the plug-in mechanism. Knowledge stored in Loki can be retrieved via SPARQL queries, inline Semantic MediaWiki-like queries, or Prolog goals. There is also a possibility to export knowledge in RDF/XML files or to query the wiki remotely using the SPARQL endpoint. Loki also includes a set of features that facilitate the work of domain experts and knowledge engineers, including knowledge visualization, ontology storage, and a code hint and completion mechanism. The introduction of reasoning unit tests, apart from the method of verifying the quality of knowledge, also facilitates the definition of users' expectations regarding the final state of the knowledge base. Finally, a description of six experiments conducted to evaluate specific hypotheses during Loki's and CKE process' development was provided.

As experiments with students have their limitations, e.g., concerning the motivation of participants, we plan to establish cooperation with external partners and use Loki for real use case related to digital humanities or software engineering project. The latter will be possible thanks to the ability to store knowledge in the form of business processes and rules, and the plug-in system that facilitates the expansion with new knowledge representations. We consider this to be a solid basis for storing software engineering project specifications in the wiki. Unlike simple wiki systems hosted in GitLab or Azure DevOps, Loki will allow better processing and querying of stored knowledge, as well as verification of its consistency. A connector between the programming IDE and the wiki is in development, allowing users to link specific pieces of code and fragments of specification stored in the wiki. The current development also aims to improve the human-wiki interaction through the gamification and adaptive user interface modules (see Fig. 2).

Finally, detailed studies of the scalability of Loki are planned. In the experiments conducted, no performance problems were encountered, however, they were carried out on small knowledge bases of less

[26] See: http://aksw.org/Projects/Powl.html.

than 3,000 triples. Further experiments on larger knowledge bases are planned. Considering the overall architecture, the narrowest bottleneck will be the execution of queries against the knowledge base performed by the SWI-Prolog engine. Its high performance is reported in the literature (Lampa, 2010; Wielemaker et al., 2003), however, we do not know what the upper acceptable limit of the knowledge base size is. To overcome the potential limitations of SWI-Prolog, it is planned to introduce the possibility of storing knowledge in a native triplestore, which will increase the efficiency of the system for larger knowledge bases.

CRediT authorship contribution statement

Krzysztof Kutt: Conceptualization, Methodology, Software, Investigation, Visualization, Writing – original draft, Writing – review & editing. **Grzegorz J. Nalepa:** Conceptualization, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Loki is an open-source solution. To download it, simply go to the project webpage: https://loki.re/.

Acknowledgments

This publication was funded by a flagship project "European Heritage in the Jagiellonian Library – Digital Authoring of the Berlin Collections, Jagiellonian University, Poland" under the Strategic Programme Excellence Initiative at Jagiellonian University.

This paper was funded by the National Science Centre, Poland under CHIST-ERA programme, the CHIST-ERA 2017 BDSI PACMEL Project, NCN 2018/27/Z/ST6/03392.

The research for this publication has been supported by a grant from the Priority Research Area DigiWorld, Jagiellonian University, Poland under the Strategic Programme Excellence Initiative at Jagiellonian University.

References

Adrian, W. T., Nalepa, G. J., & Ligęza, A. (2013). On potential usefulness of inconsistency in collaborative knowledge engineering. In *Proceedings of the 8th international conference on knowledge, information and creativity support systems*.

Allemang, D., & Hendler, J. A. (2011). *Semantic web for the working ontologist - Effective modeling in RDFS and OWL* (2nd ed.). Morgan Kaufmann.

Alobaid, A., Garijo, D., Poveda-Villalón, M., Santana-Pérez, I., Fernández-Izquierdo, A., & Corcho, Ó. (2019). Automating ontology engineering support activities with ontoology. *Journal of Web Semantics*, *57*, http://dx.doi.org/10.1016/j.websem.2018.09.003.

Baumeister, J. (2004). *Agile development of diagnostic knowledge systems* (Ph.D. thesis), Germany: Julius Maximilians University Würzburg.

Baumeister, J. (2020). Experience-based quality assessment of distributed knowledge graphs. In P. Heisig, R. Orth, J. M. Schönborn, & S. Thalmann (Eds.), *WM 2019 - Wissensmanagement in digitalen Arbeitswelten: Aktuelle Ansätze und Perspektiven - Knowledge management in digital workplace environments: State of the art and outlook* (pp. 123–138). Bonn: Gesellschaft für Informatik e.V.

Baumeister, J., & Nalepa, G. J. (2009). Verification of distributed knowledge in semantic knowledge wikis. In H. C. Lane, & H. W. Guesgen (Eds.), *FLAIRS-22: Proceedings of the twenty-second international Florida Artificial Intelligence Research Society conference* (pp. 384–389). FLAIRS Menlo Park, California: AAAI Press.

Baumeister, J., & Reutelshoefer, J. (2011). Developing knowledge systems with continuous integration. In *I-KNOW 2011, 11th international conference on knowledge management and knowledge technologies* (p. 33). http://dx.doi.org/10.1145/2024288.2024328.

Baumeister, J., Reutelshoefer, J., Belli, V., Striffler, A., Hatko, R., & Friedrich, M. (2012). KnowWE–a wiki for knowledge base development. In *Knowledge engineering and software engineering*.

Baumeister, J., Reutelshoefer, J., & Puppe, F. (2011a). Engineering intelligent systems on the knowledge formalization continuum. *International Journal of Applied Mathematics and Computer Science (AMCS)*, *21*, http://ki.informatik.uni-wuerzburg.de/papers/baumeister/2011/2011-Baumeister-KFC-AMCS.pdf.

Baumeister, J., Reutelshoefer, J., & Puppe, F. (2011b). Knowwe: A semantic wiki for knowledge engineering. *Applied Intelligence*, 1–22. http://dx.doi.org/10.1007/s10489-010-0224-5.

Baumeister, J., Striffler, A., Brandt, M., & Neumann, M. (2016). Collaborative decision support and documentation in chemical safety with knowsec. *Journal of Cheminformatics*, *8*, 21.

Bry, F., Schaffert, S., Vrandecic, D., & Weiand, K. A. (2012). Semantic wikis: Approaches, applications, and perspectives. In *Reasoning web. Semantic technologies for advanced query answering - 8th international summer school 2012, Vienna, Austria, September 3-8, 2012. proceedings* (pp. 329–369). http://dx.doi.org/10.1007/978-3-642-33158-9_9.

Collective (2015). In M. Deuter, J. Bradbery, & J. Turnbull (Eds.), *Oxford advanced learner's dictionary* (9th ed.). Oxford University Press, http://www.oxfordlearnersdictionaries.com/definition/english/collective_1.

Cooperative (2015). In M. Deuter, J. Bradbery, & J. Turnbull (Eds.), *Oxford advanced learner's dictionary* (9th ed.). Oxford University Press, http://www.oxfordlearnersdictionaries.com/definition/english/cooperative_1.

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2011). *Distributed systems: Concepts and design* (5th ed.). Pearson.

Debruyne, C., Munnelly, G., Kilgallon, L., O'Sullivan, D., & Crooks, P. (2022). Creating a knowledge graph for Ireland's lost history: Knowledge engineering and curation in the beyond 2022 project. *ACM Journal on Computing and Cultural Heritage*, *15*, 25:1–25. http://dx.doi.org/10.1145/3474829.

Distributed system (2015). In M. Deuter, J. Bradbery, & J. Turnbull (Eds.), *Oxford advanced learner's dictionary* (9th ed.). Oxford University Press, http://www.oxfordlearnersdictionaries.com/definition/english/distributed-system.

Elze, R., Hesse, T.-M., & Martin, M. (2011). Dispedia.de – a linked information system for rare diseases. In A. Holzinger, & K.-M. Simonic (Eds.), *Information quality in e-Health: 7th conference of the workgroup human-computer interaction and usability engineering of the Austrian Computer Society* (pp. 691–701). Berlin, Heidelberg: Springer Berlin Heidelberg, http://dx.doi.org/10.1007/978-3-642-25364-5_50.

Ermilov, T., Heino, N., Tramp, S., & Auer, S. (2011). Ontowiki mobile – knowledge management in your pocket. In G. Antoniou, M. Grobelnik, E. Simperl, B. Parsia, D. Plexousakis, P. De Leenheer, & J. Pan (Eds.), *The semantic web: Research and applications: 8th extended semantic web conference, ESWC 2011, Heraklion, Crete, Greece, May 29-June 2, 2011, proceedings, part I* (pp. 185–199). Berlin, Heidelberg: Springer Berlin Heidelberg, http://dx.doi.org/10.1007/978-3-642-21034-1_13.

Fernández-López, M., Gómez-Pérez, A., & Juristo, N. (1997). Methontology: from ontological art towards ontological engineering. In *Proceedings of the ontological engineering AAAI-97 spring symposium series*. American Association for Artificial Intelligence.

Frischmuth, P., Martin, M., Tramp, S., Riechert, T., & Auer, S. (2015). Ontowiki–an authoring, publication and visualization interface for the data web. *Semantic Web*, *6*, 215–240.

Furth, S., & Baumeister, J. (2014). An ontology debugger for the semantic wiki KnowWE (tool presentation). In G. J. Nalepa, & J. Baumeister (Eds.), *Proceedings of 10th workshop on knowledge engineering and software engineering (KESE10) co-located with 21st European conference on artificial intelligence*. http://ceur-ws.org/Vol-1289/.

Furth, S., & Baumeister, J. (2017). Constructing technical knowledge organizations from document structures. In *Lecture notes in computer science: vol. 10260, NLDB* (pp. 210–213). Springer.

Gandon, F., & Schreiber, G. (2014). RDF 1.1 XML syntax. W3C recommendation W3C. https://www.w3.org/TR/rdf-syntax-grammar/.

Gatiatullin, A., & Kubedinova, L. (2020). Multilingual thesaurus for turkic languages. In *2020 5th international conference on computer science and engineering* (pp. 393–398). http://dx.doi.org/10.1109/UBMK50275.2020.9219378.

Gruninger, M., & Fox, M. S. (1994). The design and evaluation of ontologies for enterprise engineering. In *Workshop on implemented ontologies, European workshop on artificial intelligence*.

Harris, S., & Seaborne, A. (2013). SPARQL 1.1 query language. W3C recommendation W3C. https://www.w3.org/TR/sparql11-query/.

Heino, N., Dietzold, S., Martin, M., & Auer, S. (2009). Developing semantic web applications with the ontowiki framework. In T. Pellegrini, S. Auer, K. Tochtermann, & S. Schaffert (Eds.), *Networked knowledge - Networked media* (pp. 61–77). Berlin, Heidelberg: Springer Berlin Heidelberg, http://dx.doi.org/10.1007/978-3-642-02184-8_5.

Hitzler, P. (2021). A review of the semantic web field. *Communication of the ACM*, *64*, 76–83. http://dx.doi.org/10.1145/3397512.

Holsapple, C. W., & Joshi, K. D. (2002). A collaborative approach to ontology design. *Communication of the ACM*, *45*, 42–47. http://dx.doi.org/10.1145/503124.503147.

Jackson, M. H. (2016). Collaboration and cooperation. In K. B. Jensen, R. T. Craig, J. D. Pooley, & E. W. Rothenbuhler (Eds.), *The international encyclopedia of communication theory and philosophy*. John Wiley & Sons, Inc., http://dx.doi.org/10.1002/9781118766804.wbiect203.

John, M., & Melster, R. (2004). Knowledge networks – managing collaborative knowledge spaces. In G. Melnik, & H. Holz (Eds.), *Lecture notes in computer science*: *vol. 3096*, *Advances in learning software organizations* (pp. 165–171). Springer Berlin Heidelberg.

Junjie, W., Qian, M., Dongxia, M., & Su, L. (2010). Research for collaborative knowledge management based on semantic wiki technology. In *2010 second international workshop on education technology and computer science, vol. 3* (pp. 464–467). http://dx.doi.org/10.1109/ETCS.2010.82.

Khorikov, V. (2020). *Unit testing principles, practices, and patterns*. Manning.

Kirakowski, J., & Corbett, M. (1993). Sumi: The software usability measurement inventory. *British Journal of Educational Technology*, *24*, 210–212.

Krötzsch, M., & Vrandecic, D. (2011). Semantic mediawiki. In *Foundations for the web of information and services - A review of 20 years of semantic web research* (pp. 311–326).

Krötzsch, M., Vrandecic, D., Völkel, M., Haller, H., & Studer, R. (2007). Semantic wikipedia. *Web Semantics*, *5*, 251–261.

Kutt, K. (2018). *Collaborative knowledge engineering. Methods and tools for system design* (Ph.D. thesis), AGH University of Science and Technology, Supervisor: Grzegorz J. Nalepa.

Lampa, S. (2010). *SWI-Prolog as a semantic web tool for semantic querying in bioclipse: Integration and performance benchmarking* (Master's thesis), Uppsala University, Supervisor: Egon Willighagen.

Lampa, S., Willighagen, E. L., Kohonen, P., King, A., Vrandecic, D., Grafstrom, R., & Spjuth, O. (2017). RDFIO: extending semantic mediawiki for interoperable biomedical data management. *Journal of Biomedical Semantics*, *8*, 35:1–13.

Lindgren, H., & Winnberg, P. (2010). Evaluation of a semantic web application for collaborative knowledge building in the dementia domain. In *Electronic healthcare - Third international conference* (pp. 62–69). http://dx.doi.org/10.1007/978-3-642-23635-8_8.

Majchrzak, A., Wagner, C., & Yates, D. (2006). Corporate wiki users: results of a survey. In *Proceedings of the 2006 international symposium on Wikis* (pp. 99–104).

McGuinness, D. L. (2017). Ontologies for the modern age. ISWC2017 Keynote Speech.

Nalepa, G. J. (2009). PlWiki – a generic semantic wiki architecture. In N. T. Nguyen, R. Kowalczyk, & S.-M. Chen (Eds.), *Lecture notes in computer science*: *vol. 5796*, *Computational collective intelligence. Semantic web, social networks and multiagent systems, first international conference, ICCCI 2009, Wroclaw, Poland, October 5-7, 2009. proceedings* (pp. 345–356). Springer.

Nalepa, G. J. (2010). Collective knowledge engineering with semantic wikis. *Journal of Universal Computer Science*, *16*, 1006–1023, http://www.jucs.org/jucs_16_7/collective_knowledge_engineering_with.

Nalepa, G. J. (2011). Loki – semantic wiki with logical knowledge representation. In N. T. Nguyen (Ed.), *Lecture notes in computer science*: *vol. 6560*, *Transactions on computational collective intelligence III* (pp. 96–114). Springer, http://www.springerlink.com/content/y91w134g03344376/.

Nalepa, G. J. (2018). *Intelligent systems reference library*: *vol. 130*, *Modeling with rules using semantic knowledge engineering*. Springer International Publishing.

Nalepa, G. J., Kluza, K., & Ciaputa, U. (2012). Proposal of automation of the collaborative modeling and evaluation of business processes using a semantic wiki. In *Proceedings of the 17th IEEE international conference on emerging technologies and factory automation*.

Nalepa, G. J., Kluza, K., & Kaczor, K. (2015). Sbvrwiki a web-based tool for authoring of business rules. In L. Rutkowski, & et al. (Eds.), *Lecture notes in artificial intelligence*, *Artificial intelligence and soft computing: 14th international conference* (pp. 703–713). Springer.

Nalepa, G., Slazynski, M., Kutt, K., Kucharska, E., & Luszpaj, A. (2015). Unifying business concepts for smes with prosecco ontology. In *Computer science and information systems (FedCSIS), 2015 federated conference on* (pp. 1321–1326).

Noy, N. F., Chugh, A., & Alani, H. (2008). The CKC challenge: Exploring tools for collaborative knowledge construction. *IEEE Intelligent Systems*, *23*, 64–68. http://dx.doi.org/10.1109/MIS.2008.14.

Noy, N. F., & McGuinness, D. L. (2001). *Ontology development 101: A guide to creating your first ontology*. Stanford University.

Nurminen, J. K., Karonen, O., & Hätönen, K. (2003). What makes expert systems survive over 10 years - empirical evaluation of several engineering applications. *Expert System with Applications*, *24*, 199–211. http://dx.doi.org/10.1016/S0957-4174(02)00149-5.

OWL Working Group, W. (2009). OWL 2 web ontology language: Document overview. W3C recommendation W3C.

Paraiso, E. C., Boz, G., Jr., Ramos, M. P., Sato, G. Y., & Tacla, C. (2016). Improving knowledge acquisition in a collaborative knowledge construction tool with a virtual catalyst. *Computing and Informatics*, *35*, 914–940.

Peters, M. A. (2021). Knowledge socialism: the rise of peer production - collegiality, collaboration, and collective intelligence. *Educational Philosophy and Theory*, *53*, 1–9. http://dx.doi.org/10.1080/00131857.2019.1654375.

Pilkington, C., & Pretorius, L. (2020). A connectivist view of a research methodology semantic wiki. In B. Tait, J. Kroeze, & S. Gruner (Eds.), *ICT education* (pp. 131–146). Cham: Springer International Publishing.

Prasarnphanich, P., & Wagner, C. (2009). The role of wiki technology and altruism in collaborative knowledge creation. *Journal of Computer Information Systems*, *49*, 33–41.

Reutelshoefer, J., Lemmerich, F., Baumeister, J., Wintjes, J., & Haas, L. (2010). Taking OWL to athens. In L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, & T. Tudorache (Eds.), *Lecture notes in computer science*: *vol. 6088*, *The semantic web: Research and applications* (pp. 333–347). Springer, http://dx.doi.org/10.1007/978-3-642-13486-9_23.

Richards, D. (2007). Collaborative knowledge engineering: socialising expert systems. In *2007 11th international conference on computer supported cooperative work in design* (pp. 635–640). IEEE.

Richards, D. (2009). A social software/web 2.0 approach to collaborative knowledge engineering. *Information Sciences*, *179*, 2515 – 2523. http://dx.doi.org/10.1016/j.ins.2009.01.031.

Riechert, T., Morgenstern, U., Auer, S., Tramp, S., & Martin, M. (2010). Knowledge engineering for historians on the example of the catalogus professorum lipsiensis. In *International semantic web conference* (pp. 225–240). Springer.

Roschelle, J., & Teasley, S. D. (1995). The construction of shared knowledge in collaborative problem solving. In C. O'Malley (Ed.), *Computer supported collaborative learning* (pp. 69–97). Berlin, Heidelberg: Springer Berlin Heidelberg, http://dx.doi.org/10.1007/978-3-642-85098-1_5.

Santhosh, J. (2019). *Towards a hybrid methodology for domain ontology development* (Ph.D. thesis), Coventry University, Supervisors: Nazaraf Shah and Craig Stewart.

Schwaber, K., & Sutherland, J. (2020). *The scrum guide. The definitive guide to scrum: The rules of the game*: *Technical report*, https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf.

Silva, L. C. L., Jr., Borges, M. R. S., & de Carvalho, P. V. R. (2009). Collaborative ethnography: an approach to the elicitation of cognitive requirements of teams. In *Proceedings of the 13th international conference on computers supported cooperative work in design* (pp. 167–172). http://dx.doi.org/10.1109/CSCWD.2009.4968053.

Simsek, U., Kärle, E., Angele, K., Huaman, E., Opdenplatz, J., Sommer, D., Umbrich, J., & Fensel, D. (2023). A knowledge graph perspective on knowledge engineering. *SN Computer Science*, *4*, 16. http://dx.doi.org/10.1007/s42979-022-01429-x.

Skaf-Molli, H., Canals, G., & Molli, P. (2010). DSMW: distributed semantic mediawiki. In L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, & T. Tudorache (Eds.), *The semantic web: Research and applications, 7th extended semantic web conference* (pp. 426–430). http://dx.doi.org/10.1007/978-3-642-13489-0_37.

Sommerville, I. (2004). Software engineering. In *International computer science* (7th ed.). Pearson Education Limited.

Thanachawengsakul, N., & Wannapiroon, P. (2021). Development of a learning ecosystem using digital knowledge engineering through moocs knowledge repository system. *International Journal of Engineering Pedagogy*, *11*, 35–48. http://dx.doi.org/10.3991/ijep.v11i1.15011.

Thiombiano, J., Traoré, Y., Malo, S., & Sié, O. (2021). Discovery and enrichment of knowledges from a semantic wiki. In J. Mejia, M. Muñoz, Á. Rocha, & Y. Quiñonez (Eds.), *New perspectives in software engineering* (pp. 142–153). Cham: Springer International Publishing.

Tomaszuk, D., & Hyland-Wood, D. (2020). RDF 1.1: Knowledge representation and data integration language for the web. *Symmetry*, *12*(84), http://dx.doi.org/10.3390/sym12010084.

Torres, G. M. (2018). *Collaborative knowledge engineering*. Independently Published.

Tudorache, T., Nyulas, C. I., Noy, N. F., & Musen, M. A. (2013). Using semantic web in ICD-11: three years down the road. In *International semantic web conference* (pp. 195–211). Springer.

Uschold, M., & King, M. (1995). Towards a methodology for building ontologies. In *IJCAI-95 workshop on basic ontological issues in knowledge sharing*.

Vahdati, S., Fathalla, S., Auer, S., Lange, C., & Vidal, M. (2019). Semantic representation of scientific publications. In *Lecture notes in computer science*: *vol. 11799*, *TPDL* (pp. 375–379). Springer.

Villines, S. (2014). Collaborative, collective, cooperative. http://www.sociocracy.info/collaborative-collective-cooperative/.

Vrandecic, D., & Gangemi, A. (2006). Unit tests for ontologies. In *On the move to meaningful internet systems 2006: OTM 2006 workshops* (pp. 1012–1020). http://dx.doi.org/10.1007/11915072_2.

Wang, Y., & Chen, M. (2004). A collaborative knowledge production model for knowledge management in complex engineering domains. In *Proceedings of the IEEE international conference on systems, man & cybernetics* (pp. 5050–5055). http://dx.doi.org/10.1109/ICSMC.2004.1400994.

Wielemaker, J., Schreiber, G., & Wielinga, B. J. (2003). Prolog-based infrastructure for RDF: scalability and performance. In D. Fensel, K. P. Sycara, & J. Mylopoulos (Eds.), *Lecture notes in computer science*: *vol. 2870*, *ISWC* (pp. 644–658). Springer, http://dx.doi.org/10.1007/978-3-540-39718-2_41.

Willmes, C., Viehberg, F., Lopez, S. E., & Bareth, G. (2018). CRC806-KB: A semantic mediawiki based collaborative knowledge base for an interdisciplinary research project. *Data*, *3*(44), http://dx.doi.org/10.3390/data3040044.

Zander, S., Swertz, C., Verdú, E., Pérez, M. J. V., & Henning, P. (2014). A semantic mediawiki-based approach for the collaborative development of pedagogically meaningful learning content annotations. In P. Molli, J. G. Breslin, & M. Vidal (Eds.), *Semantic web collaborative spaces - Second international workshop, SWCS 2013, third international workshop, SWCS 2014* (pp. 73–111). http://dx.doi.org/10.1007/978-3-319-32667-2_5.