



AGH University of Science and Technology in Kraków

Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering

PHD DISSERTATION

COLLABORATIVE KNOWLEDGE ENGINEERING. METHODS AND TOOLS FOR SYSTEM DESIGN

AUTHOR:

Krzysztof Kutt

SUPERVISOR:

Grzegorz J. Nalepa, Ph.D.

Kraków 2018



Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

ROZPRAWA DOKTORSKA

METODY PROJEKTOWANIA SYSTEMÓW Z UŻYCIEM KOLABORATYWNEJ INŻYNIERII WIEDZY

AUTOR:

Krzysztof Kutt

PROMOTOR:

dr hab. inż. Grzegorz J. Nalepa,
prof. AGH

Kraków 2018

Then the Dean repeated the mantra that has had such a marked effect on the progress of knowledge throughout the ages. 'Why don't we just mix up absolutely everything and see what happens?' he said. And Ridcully responded with the traditional response. 'It's got to be worth a try,' he said.

(Terry Pratchett, "Hogfather")

Abstract

This dissertation concerns methods for supporting the knowledge engineering (KE) process. In recent years, a shift from the traditional KE to collaborative KE has been visible – one that represents a transformation from process, in which the main role was played by knowledge engineers, to a collaborative effort, in which domain experts are main contributors. This change involves new challenges related to the massiveness of the process and the need to adapt it to stakeholders who do not have much technical knowledge.

The main research goal of this dissertation is twofold: to capture and formulate a collaborative knowledge engineering process that provides a general framework for defining roles which should be identified in a group, and to prepare steps that should be taken in this process. Likewise, its aim is to propose methods and tools that support the defined CKE process, leading to the creation of good quality knowledge base in reasonable time, through means convenient for target users.

To address this research goal, the dissertation makes the following original contributions. First of all, critical analysis of requirements for systems that support collaborative KE process was conducted. Based on that, the formulation of a general process for collaborative KE was proposed, which was then followed by the proposal of set of methods and tools for three important fields of collaborative KE support: quality management, change management and user involvement. Most important methods include: 1) a method of summarizing characteristics of changes into one metric, 2) a graph-based semantic changelog, i.e. a meta-layer that describes all changes being made in knowledge base that can be further queried and processed, and 3) a set of gamification techniques adapted to the CKE process. The theoretical part of the dissertation is supported by the description of a prototypical toolkit that supports collaborative KE process within wikis, which was developed during the research.

The results of the dissertation were evaluated in three different ways. Firstly, each of proposed modules was tested in a sample project to check whether they work properly. Secondly, five experiments were conducted during the course of the thesis to examine the whole collaborative KE process and the developed toolkit in KE practice with knowledge engineers. Finally, the usability study with users was conducted.

Streszczenie

Rozprawa dotyczy metod wspierania procesu inżynierii wiedzy. W ostatnich latach widoczne jest przejście od tradycyjnego procesu, w której główną rolę odgrywali inżynierowie wiedzy, do kolaboratywnego procesu, w której głównymi uczestnikami są eksperci dziedzinowi. Zmiana ta pociąga za sobą nowe wyzwania związane m.in. z masowością procesu czy z potrzebą dostosowania go do uczestników, którzy nie mają dużej wiedzy technicznej.

Głównym celem badawczym niniejszej rozprawy jest stworzenie opisu iteracyjnego procesu kolaboratywnej inżynierii wiedzy, który wyznaczałby jego ogólną strukturę poprzez zdefiniowanie ról, które powinny zostać zidentyfikowane wewnątrz grupy i kroków, które powinny być realizowane przez tę grupę. Opis ten jest uzupełniony zbiorem metod i narzędzi wspierających tak sformułowany proces, w celu stworzenia dobrej jakości bazy wiedzy w rozsądnym czasie i przy użyciu środków, które będą wygodne dla jego uczestników.

Realizując założenia wyznaczone przez cel badawczy, autor rozprawy prezentuje w niej następujące osiągnięcia stanowiące oryginalny wkład w dziedzinę. W pierwszej kolejności przeprowadzono analizę wymagań stawianych przed systemami, których celem jest wsparcie omawianego procesu. Opierając się na wnioskach z tej analizy, zaproponowano definicję iteracyjnego procesu kolaboratywnej inżynierii wiedzy. W dalszej kolejności zaproponowano zbiór metod i narzędzi dla trzech obszarów wsparcia, tj. dla zarządzania jakością, zarządzania zmianami i zaangażowania użytkowników. Pośród nich najbardziej istotnymi metodami są: 1) sposób podsumowania właściwości zmian w jedną metrykę, 2) graf zmian, tj. możliwa do dalszego przetwarzania i odpytywania metawarstwa opisująca wszystkie zmiany dokonywane w bazie wiedzy, oraz 3) zbiór technik grywalizacji dostosowany do procesu. Teoretyczną część pracy uzupełnia opis prototypowego zbioru narzędzi wspierającego proces kolaboratywnej inżynierii wiedzy w systemie wiki, stworzony w czasie prowadzonych badań.

Ewaluacja wyników rozprawy została przeprowadzona na trzy sposoby. Po pierwsze, każdy z zaproponowanych modułów został przetestowany w przykładowym projekcie dla zweryfikowania poprawności ich działania. Po drugie, w czasie prac przeprowadzono pięć eksperymentów mających na celu sprawdzenie zaproponowanego procesu kolaboratywnej inżynierii wiedzy i prototypowego zbioru narzędzi w praktyce. Po trzecie, przeprowadzono z użytkownikami badanie użyteczności stworzonego zbioru narzędzi.

Contents

| | |
|---|-------------|
| Abstract | v |
| Streszczenie | vii |
| List of Figures | xi |
| List of Tables | xiii |
| 1 Introduction | 1 |
| 1.1 Motivation and Scope | 1 |
| 1.2 Goal and Plan of the Work | 4 |
| 1.3 Original Contribution | 6 |
| 1.4 Exclusions | 7 |
| 2 Collaborative Knowledge Engineering | 9 |
| 2.1 Examples of Distributed Knowledge Engineering | 9 |
| 2.1.1 Use Cases | 9 |
| 2.1.2 Tools | 11 |
| 2.2 Taxonomy of Distributed Knowledge Engineering | 14 |
| 2.3 Issues and Challenges | 16 |
| 2.3.1 Knowledge Bases Quality | 17 |
| 2.3.2 Version Control | 20 |
| 2.3.3 Agile Development | 22 |
| 2.3.4 User Involvement | 26 |
| 2.4 Summary | 28 |
| 3 Semantic Wikis as CKE Tools | 29 |
| 3.1 Basic Wiki Systems | 29 |
| 3.2 Semantic Web | 31 |
| 3.3 Semantic Wikis | 35 |

| | | |
|----------|--|------------|
| 3.3.1 | Semantics in Wikis | 35 |
| 3.3.2 | Overview | 36 |
| 3.3.3 | Semantic MediaWiki | 36 |
| 3.3.4 | KnowWE | 40 |
| 3.3.5 | OntoWiki | 40 |
| 3.4 | Loki | 42 |
| 3.5 | Summary | 45 |
| 4 | Approach Proposal | 47 |
| 4.1 | Fields of CKE Support | 47 |
| 4.2 | Area of Interest | 48 |
| 4.3 | Approach Overview | 49 |
| 4.4 | CKE Agile Process | 52 |
| 4.5 | European Wiki Case Study | 54 |
| 4.6 | Summary | 56 |
| 5 | Quality Management | 57 |
| 5.1 | Reasoning Unit Tests | 57 |
| 5.2 | Metrics of Changes | 69 |
| 5.3 | Opinions and Discussion | 76 |
| 5.4 | Summary | 79 |
| 6 | Change Management | 81 |
| 6.1 | Change Ontologies | 81 |
| 6.2 | Semantic Changelog | 86 |
| 6.3 | Summary | 97 |
| 7 | User Involvement | 99 |
| 7.1 | Gamification | 99 |
| 7.2 | Usability | 104 |
| 7.3 | Summary | 110 |
| 8 | Experiments and Evaluation | 113 |
| 8.1 | Conducted Experiments | 113 |
| 8.1.1 | First Experiment: Pokemons, Simpsons, et al. | 114 |
| 8.1.2 | Second Experiment: CSP Library | 115 |
| 8.1.3 | Third Experiment: Pubs in Cracow | 118 |

| | | |
|----------|--|------------|
| 8.1.4 | Fourth Experiment: Artificial Intelligence Class | 120 |
| 8.2 | Final Evaluation: Cookbook and Movies KB | 122 |
| 8.3 | Summary | 125 |
| 9 | Concluding Remarks | 127 |
| 9.1 | Conclusions | 127 |
| 9.2 | Future Work | 130 |
| A | Code for Metrics Calculation | 133 |
| B | Semantic Changelog Files | 135 |
| C | Report Scheme Used in Experiments | 141 |
| D | Survey Used in the Fourth Experiment | 143 |
| | Bibliography | 145 |

List of Figures

| | | |
|-----|--|----|
| 1.1 | Gartner’s Hype Cycle for Advanced Analytics and Data Science as of July 2015 | 2 |
| 1.2 | Gartner’s Hype Cycle for Emerging Technologies as of July 2016 | 3 |
| 2.1 | Group KE taxonomy | 16 |
| 2.2 | CodeCity Metrics example for ArgoUML java package [171] | 18 |
| 2.3 | PROV model overview [58] | 21 |
| 2.4 | The agile process model for knowledge-based systems [12] | 25 |
| 2.5 | Usability Stack for knowledge-based systems [52] | 27 |
| 3.1 | The Semantic Web Stack of technologies [23] | 31 |
| 3.2 | Sample RDF graph that represents selected properties of London available in the DBpedia . | 32 |
| 3.3 | Semantic MediaWiki wiki page [86] | 41 |
| 3.4 | KnowWE wiki page [15] | 41 |
| 3.5 | OntoWiki wiki page about professor Ernst Heinrich Weber | 42 |
| 3.6 | A set of plugins for various knowledge formalizations: Loki, BPwiki and SBVRwiki | 43 |
| 3.7 | XTT2 model visualization inside Loki wiki page | 44 |
| 3.8 | Loki wiki page text with recommendations generated by HearT rule engine [108] | 45 |
| 4.1 | The BiFröST framework architecture | 51 |
| 4.2 | <i>CKE agile process</i> compared to the agile process model proposed by Baumeister [12] | 53 |
| 5.1 | Sample test structure for <i>European Wiki</i> project | 59 |
| 5.2 | <i>Reasoning unit tests</i> module for Loki. Classes diagram | 63 |
| 5.3 | The tests summary at the very beginning of the project | 63 |
| 5.4 | The results of the <code>unittest:citiestest test</code> | 66 |
| 5.5 | The results of the <code>unittest:eu:contradictionstest test</code> | 66 |
| 5.6 | The results of the <code>unittest:eu:largesttest test</code> | 66 |
| 5.7 | The results of the <code>unittest:names:emptytest test</code> | 68 |
| 5.8 | The results of the <code>unittest:names:detailed:manynametest test</code> | 68 |

| | | |
|------|---|-----|
| 5.9 | The summary of all tests used for validation | 68 |
| 5.10 | Discussion button in page menu | 78 |
| 5.11 | Discussion among users about <code>city:paris</code> page | 79 |
| 5.12 | Evaluation form for two subsequent changes made to <code>city:paris</code> page | 80 |
| 6.1 | Full version of general CKE change ontology proposed in this dissertation | 85 |
| 6.2 | Default and extended DokuWiki/Loki edition form | 86 |
| 6.3 | <i>Semantic changelog</i> provides a meta-layer for subsequent states of KB | 88 |
| 6.4 | <i>Semantic changelog</i> skeleton that describes the change presented on Figure 6.3 | 88 |
| 6.5 | Metadata representation within the <i>semantic changelog</i> | 89 |
| 6.6 | Semantic statistics saved in the <i>semantic changelog</i> | 89 |
| 6.7 | Number of reasoning tests passed and metrics in the <i>semantic changelog</i> | 89 |
| 6.8 | Vote for specific concept's revision within the <i>semantic changelog</i> | 90 |
| 6.9 | Change type and goal assignment to the specific change in the <i>semantic changelog</i> | 90 |
| 7.1 | TOP 10 list in a sidebar of a wiki. It is consistent with values presented in Table 7.3 | 103 |
| 7.2 | SPARQL Endpoint form for Loki | 106 |
| 7.3 | Ontology edition form for Loki. Presented ontology is saved in the XML file on Listing 7.2 | 109 |
| 7.4 | Ontology visualisation for Loki. Presented ontology is saved in the XML file on Listing 7.2 | 110 |
| 7.5 | Code hint and highlight mechanism for Loki | 110 |
| 8.1 | Summary of factual change descriptions selected from the first ontology version | 117 |
| 8.2 | Summary of goal descriptions selected from the first ontology version | 117 |
| 8.3 | Summary of factual change descriptions selected from the second ontology version | 119 |
| 8.4 | Summary of goal descriptions selected from the second ontology version | 119 |
| 8.5 | Summary of SUMI scales | 124 |

List of Tables

| | | |
|-----|--|-----|
| 1.1 | Transitions in ontology development (adapted from [99]) | 3 |
| 2.1 | Distributed KE use cases discussed in the text | 10 |
| 2.2 | Distributed KE tools discussed in the text | 12 |
| 2.3 | Ontology metrics that may be used to evaluate KB quality | 19 |
| 2.4 | Main differences between classical approach and the agile way (based on [136]) | 24 |
| 3.1 | Semantic wikis comparison with regard to the CKE Requirements (see Section 2.3) | 37 |
| 5.1 | Set of assertions for SELECT queries | 58 |
| 5.2 | Set of assertions for ASK queries | 58 |
| 5.3 | Set of assertions for SELECT queries within Loki | 61 |
| 5.4 | Set of assertions for ASK queries within Loki | 61 |
| 5.5 | Set of assertions for DESCRIBE queries within Loki | 62 |
| 5.6 | Rules for normalization of differences in metric values to 1-5 scale | 72 |
| 5.7 | Metrics values for subsequent states and changes in a sample KB | 75 |
| 6.1 | Sample commit messages from analysed projects | 83 |
| 6.2 | Factual change and goal annotations selected by users in <i>European Wiki</i> project | 87 |
| 7.1 | Points and badges awarded for changes within <i>European Wiki</i> project | 104 |
| 7.2 | Users' credibility calculation after the first iteration of the <i>European Wiki</i> project | 104 |
| 7.3 | Summary of status of users after the first iteration of the <i>European Wiki</i> project | 105 |
| 8.1 | Modules (and plugins' versions) coverage in conducted experiments | 114 |
| 8.2 | Summary of the course of the first experiment | 114 |
| 8.3 | Statistics of CKE processes within the first experiment | 116 |
| 8.4 | Statistics of CKE processes within the fourth experiment | 121 |
| 9.1 | BiFröST framework compared to other semantic wikis with regard to the CKE requirements | 128 |

List of Abbreviations

| | | |
|---------|---|---|
| BiFröST | – | BiFröST Framework für Semantical Tracking |
| BPwiki | – | Business Processes Wiki |
| CKE | – | Collaborative KE |
| HalVA | – | HeKatE Verification and Analysis |
| HaDEs | – | HeKatE Design Environment |
| HeaRT | – | HeKatE RunTime |
| HeKatE | – | Hybrid Knowledge Engineering Project |
| HMR | – | HeKatE Meta Representation |
| HQEd | – | HeKatE Qt Editor |
| KB | – | Knowledge Base |
| KBS | – | Knowledge-Based System |
| KE | – | Knowledge Engineering |
| KnowWE | – | Knowledge Wiki Environment |
| Loki | – | Logic in Wiki |
| OWL | – | Web Ontology Language |
| PIWiki | – | Wiki based on Prolog |
| RDF | – | Resource Description Framework |
| RDFS | – | RDF Schema |
| SBVR | – | Semantic Business Vocabulary and Business Rules |
| SE | – | Software Engineering |
| SMW | – | Semantic MediaWiki |
| SPARQL | – | SPARQL Protocol And RDF Query Language |
| UML | – | Unified Modeling Language |
| URI | – | Uniform Resource Identifier |
| VCS | – | Version Control Software |
| XTT2 | – | eXtended Tabular Trees, Version 2 |
| W3C | – | World Wide Web Consortium |

Chapter 1

Introduction

This chapter introduces the reader to the contents of this dissertation. It consists of four sections: Section 1.1 presents the background of the research, defines the scope of the thesis and provides arguments why it is an important issue in the field of computer science and artificial intelligence. Section 1.2 specifies the goal of the research and describes the steps that were made in order to achieve it. Section 1.3 emphasizes the original contribution of the thesis. Finally, Section 1.4 discusses the issues that were deliberately not addressed in this thesis, and left for further research.

1.1 Motivation and Scope

Knowledge Engineering (KE), a subdomain of Computer Science and Artificial Intelligence, is *an engineering discipline that involves integrating knowledge into computer systems in order to solve complex problems normally requiring a high level of human expertise*, as stated by classical definition by E. Feigenbaum and P. McCorduck [47]. The long tradition of KE is manifested, among others, in several text-books that were written in this field [43, 59, 63, 93, 169]. While it deals with various methods of knowledge representations, this dissertation is focused on ontologies, i.e. *formal, explicit specifications of a shared conceptualizations* [150, 62]. This is a well-defined and well-established field with ongoing research in many areas: ontologies are currently developed as a base for various Knowledge-Based Systems (KBS), ranging from academic and medical, through enterprise, to government use cases.

Technological development, in particular the transition to Web 2.0 and the emergence of the Semantic Web, has a significant impact on KE. The former describes the transition from static web pages to the web created by users, e.g. via blogs or Twitter messages, and the Semantic Web¹ is *a collection of technologies and standards that allow machines to understand the meaning (semantics) of information on the Web* [7].

¹Semantic Web is sometimes called Web 3.0 [23], as a next step in the Web development or Linked Data² [67], as it provides a way to connect data from various sources into one big knowledge base.

²One should not confuse Linked Data with the Linked Open Data project that collects information about Linked Data data sets available under an open licence.

Both of them led to a situation in which virtually anyone can take part in knowledge engineering process. The potential behind this observation was emphasized in Gartner's Hype Cycle. In 2015, Linked Data was identified as a technology that enters the *Slope of Enlightenment*, i.e. the real potential of the technology is being understood and companies are starting to invest in this technology (see Figure 1.1). The next year, Enterprise Taxonomy and Ontology Management was recognized as one of important emerging technologies (see Figure 1.2). This transition from traditional KE to distributed KE, undertaken by a lot of people, changes the research perspective and creates new fields of study, as there are many challenges that appear in this setting [156]. Due to the fact that distributed KE covers many diverse processes, more specific terms appeared in the literature: cooperative KE, collaborative KE and collective KE. As they are sometimes used to define different cases, and other times used interchangeably, thereby causing confusion, *there is a need for clear definitions that describe this research area.*

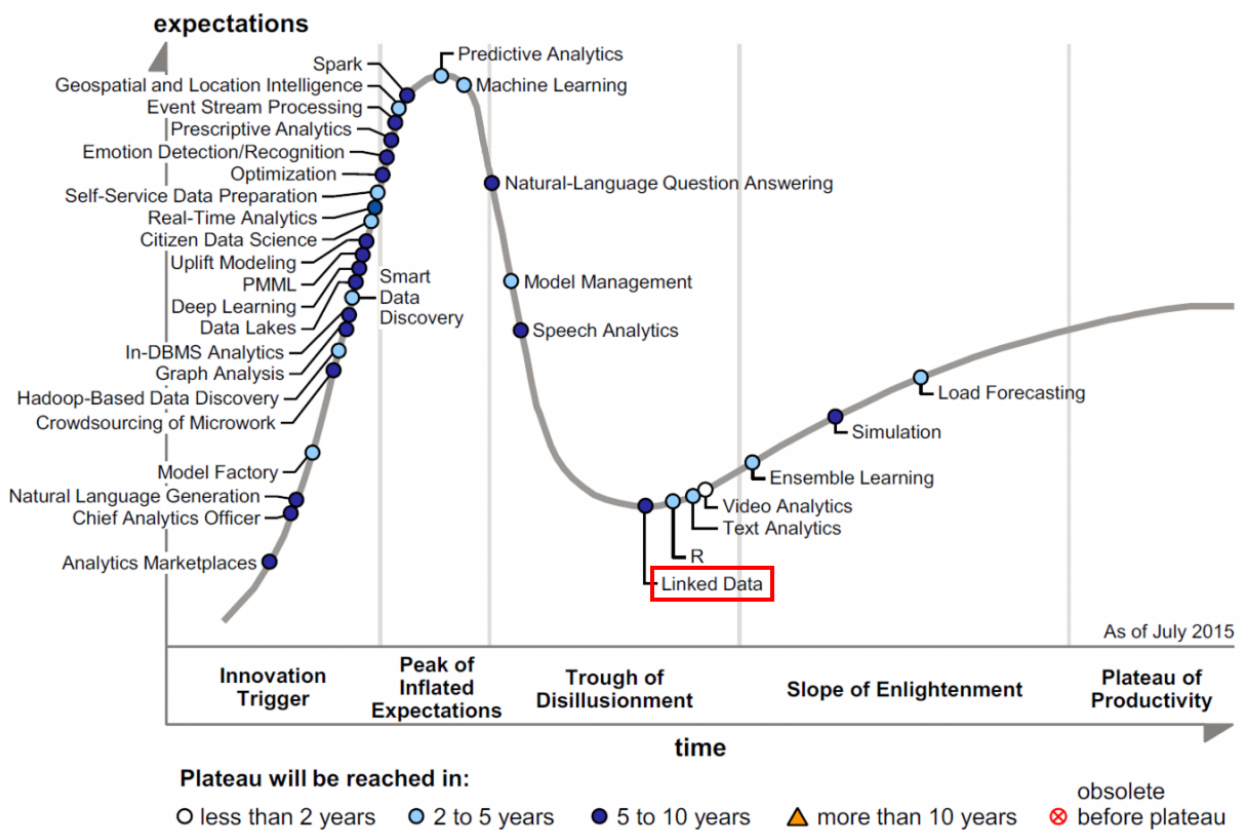


Figure 1.1: Gartner's Hype Cycle for Advanced Analytics and Data Science as of July 2015³

The shift of paradigm has led to the change of main contributors in KE process. Modern age ontologies are no longer prepared and maintained only by a small group of specialists trained in KE. Nowadays, emphasis is put to allow domain experts to be the main driving force in this process (see Table 1.1). Due to the fact that people from various environments will be now involved in ontology development, there is

³Source: <https://www.gartner.com/>.

⁴Source: <https://www.gartner.com/>.

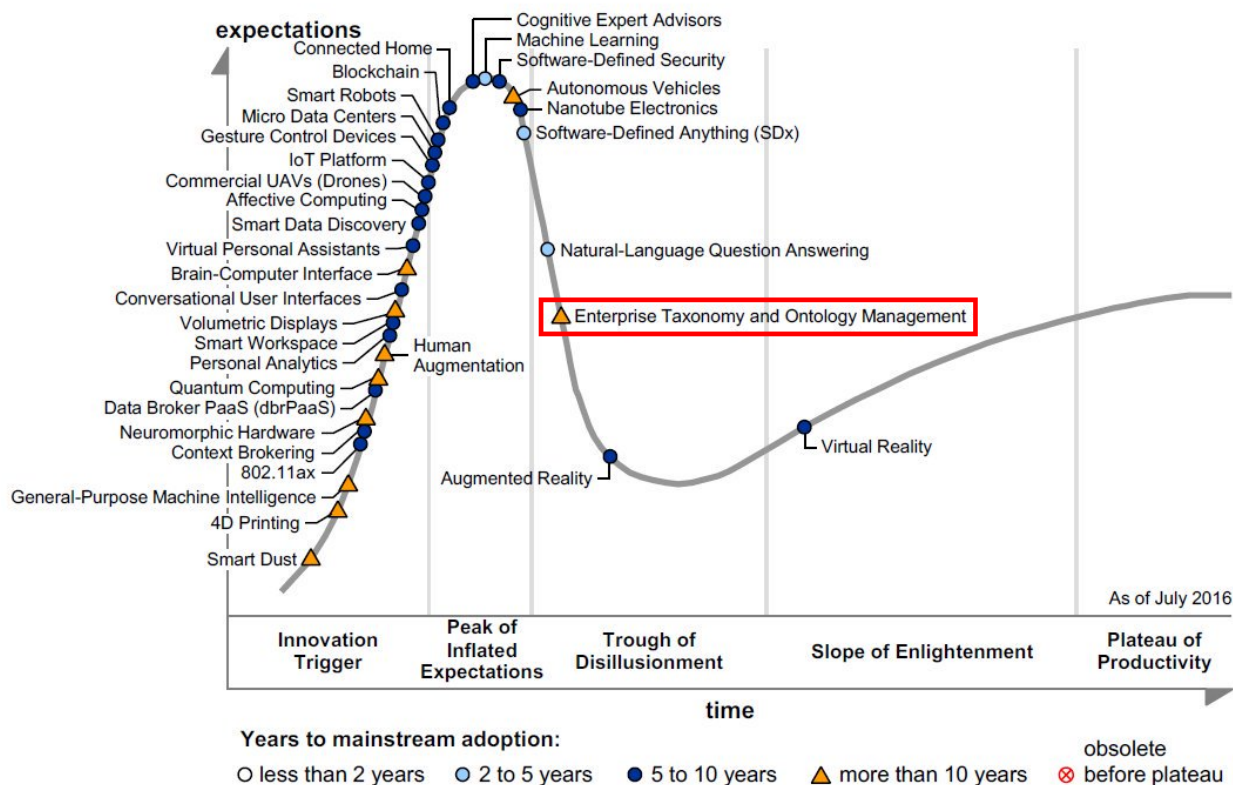


Figure 1.2: Gartner’s Hype Cycle for Emerging Technologies as of July 2016⁴

a need to clearly state the responsibilities and activities of each participant. It is also urgent to keep in mind that knowledge, i.e. *theoretical or practical understanding of a subject or a domain* [113], is something that depends on time and context, and therefore can change often [32]. *This gives a motivation to develop a distributed KE process based primarily on domain users, one that is done in an iterative manner in order to react to knowledge changes.*

Table 1.1: Transitions in ontology development (adapted from [99])

| | Past | Present |
|-------------------|-------------------------------------|---|
| Design | KE expert with domain expert access | KE expert paired with domain expert(s) |
| Population | KE expert learns domain | KE and domain experts determine the vocabulary |
| Evolution | KE expert heavily involved | KE expert involved in customizing tools that domain experts use |
| Tool Users | Trained in Computer Science | Trained in Domain Sciences |
| Application Users | Well understood group | Diverse and evolving group |
| Reuse | Well thought out | Expect the unexpected |

The fact that contemporary ontologies are developed mainly by domain experts entails important consequences. First of all, they are often no technology specialists and they know nothing about KE techniques. For this reason, there is a need to hide some technical and more advanced details of selected system and knowledge representation, so as to make it easier to understand by users. On the other hand, proper level of

reasoning capabilities is demanded to ensure that the prepared system will work flawlessly. Secondly, due to the fact that new technologies allow for a massive increase in the knowledge base [166], there is a need to provide some methods that ensure decent quality level of created KBS. The collaborativeness aspect makes it increasingly important to emphasize the fact that knowledge is something that has some origin [32]. It is especially critical nowadays, since introduction of Semantic Web technologies allows that *Anyone can say Anything about Any topic* (so called AAA slogan) [7]. Conflicts of information are commonplace in such a scenario. The natural consequence is the need to prepare methods to evaluate credibility and trustworthiness of knowledge and users [40], as well as methods to track the provenance of information. This will allow the system to choose the best piece of knowledge and make a decision. Finally, it is also essential to address the social aspect of the whole process: many discussions and conflicts will appear, there will be different levels of motivation, etc. *This gives an inclination for designing a set of methods and tools adjusted to capabilities and needs of domain users that complement the distributed KE process.*

Distributed KE has been present as a research subject for more than a decade. In 2001, the Semantic Web ideas were presented by Tim Berners-Lee in Scientific American [21], and in 2005-2006 “semantic wiki explosion” was observed. Since then, many tools have been created, but they were often developed without any consideration regarding actual users needs [117]. There have also been several attempts to define a distributed KE process. Nevertheless, there is still a lack of mature methodology [56]. *This confirms that distributed KE, with many unsolved issues, is a field worth researching.*

1.2 Goal and Plan of the Work

On the presented background, the research objectives of the thesis were stated. *The main goals were to describe a Collaborative Knowledge Engineering process that provides a general framework for defining roles which should be identified in a group and steps that should be taken in this process, as well as to propose methods and tools that support the defined CKE process, leading to the creation of good quality KB in reasonable time, through the means that will be convenient for target users.*

The work plan consists of the achieving following *subgoals*:

1. Definition of the Collaborative KE (CKE) and its relation to other terms: Distributed KE, Cooperative KE and Collective KE, to give a clear understanding of what CKE is and what CKE is not.
2. Description of an CKE process that defines the roles, responsibilities, tasks and a general timeline. Developed process should provide a robust framework that adapts to changing requirements and takes care of knowledge variability, i.e. is prepared in an agile way.
3. Identification of a set of requirements for tools that support CKE, which should be grouped in order to define areas of CKE support.
4. Development of methods and tools for selected CKE support areas that fulfill the identified require-

ments. Prepared solutions should take care of final users, i.e. domain experts who are not KE specialists.

A good source of inspiration for solutions may lie in Software Engineering. In this area, the need for transition from traditional models to iterative ones, that quickly respond to changing requirements and user needs, was observed at the beginning of the 21th century, when the “Manifesto for Agile Software Development”⁵ was prepared. For almost 20 years, Software Engineering has faced many problems, some of which are relevant to the CKE challenges. It is suggested to look at the solutions that were worked out, e.g. the definition of a software development process, and to analyze the possibilities of their use in the CKE area.

The set of methods and tools described within this dissertation is supported by a prototypical implementation of proposed modules. As a base for them, the semantic wikis were selected, which are based on wiki systems. Their popularity may come from their simplicity [91]: on the user’s side they require only a web browser, they use a simple syntax that can be quickly learned, and allow instant publishing of content. Semantic wikis are wikis extended with Semantic Web technologies, what makes them very promising tools for CKE – especially in massive CKE setting, where many users manage big knowledge base [166]. Specifically, Loki [4, 103, 105, 106] system developed at AGH University of Science and Technology was used as a base platform for prepared set of methods and tools.

The discussion on the aforementioned research objectives in this dissertation was partitioned into five chapters, and organised as follows. The comprehensive analysis of the state-of-the-art in the Collaborative Knowledge Engineering was given in Chapter 2. It begins with the description of sample existing use cases and tools in Section 2.1 to outline the area of interest. Then, in Section 2.2, Distributed KE taxonomy is proposed to differentiate concepts used in literature for the description of the research. Within the defined area of CKE, list of issues and challenges that should be addressed is identified in Section 2.3. Chapter 3 gives an overview of semantic wikis technology. First of all, the classical wikis idea is presented in Section 3.1. It is followed by the Semantic Web stack of technologies description in Section 3.2. The combination of these two technologies led to the emergence of semantic wikis. Section 3.3.1 provides their types classification, capabilities description and comparison of systems currently under development. Finally, Loki, selected as the base for prototypical implementation, is the subject of Section 3.4.

In Chapter 4 our original outline of the approach is presented. This chapter groups the requirements for CKE tools into six fields of interest in Section 4.1 and then in Section 4.2 identifies *CKE agile process*, *quality management*, *change management* and *user involvement* as the ones that are addressed in this dissertation. General idea of the whole proposed framework is provided in Section 4.3. It is followed by a definition of the *CKE agile process* in Section 4.4. Finally, Section 4.5 introduces a sample *European Wiki* project that is used across the dissertation to present the proposed approach in action. Detailed description

⁵See: <http://agilemanifesto.org/>.

of all modules is given in Chapters 5-7. It starts with the specification of modules for *quality management* and presentation of their prototypical implementation in Chapter 5. Then in Chapters 6 and 7 methods and prototypes for *change management* and *user involvement* are presented respectively.

Description of proposed framework is complemented by evaluation given in Chapter 8. It consists of two main parts. The first one, in Section 8.1, describes experiments that were conducted during the course of the work to evaluate sub-results and to gather data about the CKE process. The second one, in Section 8.2, presents the experiment conducted on the final version of the framework to evaluate the whole proposed approach.

The research presented in this dissertation is summarized in Chapter 9.

1.3 Original Contribution

The following results are considered to be the most important *original contributions* of this thesis:

1. *Formulation of distributed KE taxonomy* – analysis of concepts used in literature to describe the area of interest and formulation of their clear definitions was given in Section 2.2.
2. *Analysis of issues and challenges for systems that support CKE process* – a set of requirements that should be addressed in order to provide proper solutions for CKE was formulated in Section 2.3. It was then grouped into fields of CKE support in Chapter 4.
3. *Definition of CKE agile process* – the description of a general process for CKE, alongside with the specification roles, their responsibilities, the series of steps and some good practices were given in Section 4.4.
4. *Conceptualization of change ontology* – a formal representation of factual changes made in knowledge base during the CKE process along with their goals was given in Section 6.1.
5. *Proposal of graph-based semantic changelog* – a meta-layer that describes all changes being made in knowledge base, and is de facto a meta-base that can be queried, and processed was described in Section 6.2.
6. *Formulation of involvement metrics* – preparation of gamification module led to definition of a set of involvement metrics, useful for e.g. scrum owner in CKE agile process.
7. *Implementation of prototypical toolkit that support CKE process within wikis* – a set of modules in a form of plugins for Loki that support CKE process in the quality management, change management and user involvement fields was described in Chapters 5-7.

This thesis also proposed *improvements* of existing methods and tools. The following ones are considered to be the most important:

1. *Definition of hierarchy for reasoning unit tests* – an idea of adopting unit tests from SE into the KE was already described in literature. In Section 5.1 the hierarchical structure is proposed as an extension

to provide a possibility to group the tests based on their specificity and to execute only the subset of them.

2. *Proposal of method of summarizing characteristics of changes into synthetic metric* – a weighted average metric that provides one simple value, calculated based on many metrics, and also captures various characteristics of changes being made in knowledge base was proposed in Section 5.2.
3. *Design of gamification module for CKE process* – a selection of gamification techniques and their adaptation to CKE needs was given in Section 7.1.

1.4 Exclusions

In the literature there are many areas related to the Collaborative Knowledge Engineering. Some of them are used interchangeably, but in reality they define fields that are not equivalent to CKE:

Collaborative Knowledge Management is a management domain related to establishing what kind of information we have, what data we need to solve the problem, and applying it in new situations [32]. It covers analytical tasks which define problems that are then solved in CKE by a group of knowledge engineers and domain experts.

Collaborative Knowledge Creation is a term very often used interchangeably with CKE (e.g. [117]). In fact, in both of them a group of users is involved into knowledge base preparation. Both of them may even use the same tools, e.g. wiki systems described in Chapter 3). Nevertheless, they differ in the area of application. CKE term is used to describe issues related to knowledge-based systems development and to acquisition of knowledge from experts, while CKC defines processes that appear in a self-learning group, often a group of students [22, 75, 90, 124].

Collaborative Knowledge Building mainly regards process of social discourse that leads to creation of knowledge possessed by a specific (social) group. It is an area strongly related to social and cognitive psychology [64, 70, 148, 149].

Collaborative Knowledge Acquisition is sometimes referred to as a standalone process, but in fact is simply one of stages of CKE process, when various techniques to gather knowledge from experts are considered [137].

Collaborative Knowledge Curation is a process of searching information by experts in specialized sources, choosing valuable knowledge, structuring it properly and placing it in the database. This is analogous to curators job in museums, who are responsible for the search for art and exhibition management [183, 184].

This dissertation is focused solely on CKE, i.e. on aspects related to acquiring expert knowledge into information systems and building proper tools for this task.

Also, even if ontology evaluation and ontology change management methods are used in this work, they define a separate research area that is far beyond the scope of this dissertation. Here, attention is focused on methods that provide some interaction with domain users. This was the motivation for selecting unit tests as a method that enables users to define their expectations in a simple formal way, and for quality metrics that give an instant feedback about user credibility and knowledge quality. For comprehensive review of ontology evaluation methods one can read [163]. Overview of ontology change fields of interest and methods is provided by [50].

Finally, there are many Collaborative Knowledge Engineering use cases. The first which may come to mind is Wikipedia⁶. With more than 5,330,000 articles in English⁷, it is the biggest distributed database in the world. Within the Wikipedia there are also smaller communities that group specialists in specific field e.g. the WikiProject Medicine⁸, which groups 500 participants who have created over 31,000 articles [184]. Likewise, intelligence services have knowledge bases created in a distributed way. Intellipedia, used by 16 intelligence agencies in the United States, is a wiki system created as a response to 9/11 attacks, that allows analysts to share information, analyses, and premonitions with others in a secure network. The system also has the ability to search users who are experts on a particular topic or country [32]. The system was launched in 2006. After three years, there were 900,000 pages created by 100,000 users, who made over 15,000 edits a day [19]. However, neither of these wikis are interesting from this dissertation's point of view, because they are not related to knowledge engineering defined as ontology engineering, as they are simple systems of interlinked wiki systems, without underlying semantic model.

⁶See: <https://www.wikipedia.org/>.

⁷As of 07.02.2017.

⁸See: https://en.wikipedia.org/wiki/Wikipedia:WikiProject_Medicine.

Chapter 2

Collaborative Knowledge Engineering

This chapter outlines the Collaborative Knowledge Engineering area, in which this PhD dissertation is set. In particular, Section 2.1 presents examples of projects and tools that can be considered as cases of Distributed Knowledge Engineering. Then, in Section 2.2 naming issues are discussed, to differentiate Distributed, Cooperative, Collaborative and Collective Knowledge Engineering. The main part of this chapter is the presentation of issues and challenges associated with building methods and tools that support Collaborative KE. Section 2.3 begins with the list of requirements that should be fulfilled for this purpose. Then, current state-of-the-art in four areas that are important from the dissertation's point of view is presented (Sections 2.3.1–2.3.4).

2.1 Examples of Distributed Knowledge Engineering

Chapter starts with the presentation of Distributed KE examples to outline the dissertation's area of interest. It is provided in two parts. First, sample use cases description is provided in Section 2.1.1. It is followed by overview of related tools in Section 2.1.2.

2.1.1 Use Cases

The goal of this section is to provide a general idea of the wide range of fields which deal with Distributed KE, the size of conducted projects, as well as the interesting solutions used in these projects. The list of presented use cases is located in Table 2.1.

The first use case is International Classification of Diseases (ICD-11)¹. This catalogue is used by doctors, nurses, psychologists and others as guidelines to classify diseases and other health problems, as well as for the preparation of health and mortality statistics by WHO Member States. Eleventh revision of ICD catalogue, which will be published in 2018², is being prepared as a Distributed KE. 270 doctors have created

¹See: <http://www.who.int/classifications/icd/en/>.

²Information from <http://www.who.int/classifications/icd/revision/en/>, 7.02.2017.

Table 2.1: Distributed KE use cases discussed in the text

| Domain | Use Cases |
|------------|--|
| Medical | International Classification of Diseases (ICD-11), ACKTUS-dementia, Dispedia |
| Academic | Catalogus Professorum Lipsiensis, HermesWiki, Caucasian Spiders |
| Enterprise | Business Innovation in Virtual Enterprise Environments (BIVEE), SlideWiki |
| Government | KnowSEC |

45,000 classes, have made 260,000 changes and have provided 17,000 external links [159]. It is probably the biggest group ontology engineering project. Behind ICD-11 there are two different ontologies [184]: core ICD domain ontology that describes the diseases, and the Change and Annotation Ontology (ChAO) [118] that describes changes made in the ontology. Another two medical use cases are ACKTUS-dementia and Dispedia. Both are currently in prototype phase. ACKTUS³ (shortcut for *Activity-Centred Modelling of Knowledge and Interaction Tailored to Users*) is a decision support system for dementia treatment. Experts can expand it, among others, by filling clinical guidelines, informal rules of thumb and context of individual interpretation. Based on them, the system helps to detect possible alternative interpretations or ambiguities [94]. Dispedia⁴ is a knowledge base about rare diseases. Prototype describes only Amyotrophic Lateral Sclerosis (ALS). In this project, base ontology was created jointly with the experts. The project will be tested in a real environment, where it will be broadly accessed [44].

Second important Distributed KE area considers historical databases. Among them there is Catalogus Professorum Lipsiensis project⁵, where the life and work of professors of Universitat Leipzig in 1409-2009 is described. 10 historians and a group of volunteers created a description of 1,300 professors, 10,000 life periods, 400 institutions and many more. The database gives a possibility to conduct original historical investigations, e.g. social network analysis [134]. Another historical base describes ancient Greece history. About 700 pages in German, grouped within HermesWiki project, were created by a research team in cooperation with selected students. Participants were divided into two roles: domain experts (professor and assistant professor), who determined the tasks and later accepted or refused them, and ordinary users, who created knowledge. The system also had a “Hermes Quiz” – a test in which questions were generated automatically, based on the selected ontology relations (e.g. “Begin of the Persian Empire?”) [129]. Academic examples are not limited to historical ones. Caucasian Spiders Database⁶ is developed by the community of scientists, who search for spiders in the Caucasus region, take photos and describe them using the mobile application [45]. The knowledge base consists of 13321 records about 1107 spider species [123].

³See: <http://acktus.cs.umu.se/>.

⁴See: <http://dispedia.de/>.

⁵See: <http://catalogus-professorum.org/>.

⁶See: <http://caucasus-spiders.info/>.

Distributed KE is also used in various enterprises. Business Innovation in Virtual Enterprise Environments (BIVEE)⁷ is an EU Project realized for years 2011-2014. Within this project, base ontology for innovations were developed, as well as a methodology for building a specific innovations' ontology within a company [96]. SlideWiki⁸ is a tool for distributed authoring of presentations. Although it does not have a real ontology, it has a set of tags that provide semantic knowledge. They are used in searching for slides. In the future, possibility of online presentation translation into different languages will be provided, which will be useful for multinational companies [9, 82].

The last party taken into the account in this overview of Distributed KE is government. KnowSEC system⁹ (full name: *Managing Knowledge of Substances of Ecological Concern*) is a knowledge-based decision support system that helps with chemical substances assessment for German Environment Agency. More than 6,000,000 statements in the database describe 13,400 substances, which are regulated by European Regulation REACH (*Registration, Evaluation, Authorization, and Restriction of Chemicals*). This number is continuously increasing. Authors estimate that in 2018 there will be 30,000 substances characterized in the system [18].

2.1.2 Tools

All systems presented in the previous section were created using the appropriate tools that support ontology engineering. This field has a long history, with a lot of different tools that may or may not enable simultaneous group work. See [1, 6, 48, 141, 142] for more information. In this section, short revised overview of group ontology creation systems will be given. Only tools created and/or updated recently are taken into the account. Not only formal ontology-based OWL tools are considered, but also "lighter" ones, which allow eg. only for semantic annotation of the documents. List of tools discussed in the section is located in the Table 2.2.

The first group comprises tools that support the creation of OWL ontologies. One of the popular tools serving this purpose is WebProtégé¹⁰ and its previous version named Collaborative Protégé¹¹. The old version [158] was an extension to Protégé Desktop, that provides two types of group work: (a) standalone mode, where ontology was saved on some shared drive and only one user at a time could change the project, and (b) client-server mode, where users could edit the ontology simultaneously. Web interface was then provided and project evolved into WebProtégé. Both versions need some experience or training with OWL or other formal language as a prerequisite [156]. Currently, WebProtégé does not support many features of Protégé Desktop, e.g. it does not support inference engines, but there is a possibility to export an ontology

⁷See: <http://bivee.eu/>.

⁸See: <http://slidewiki.org/>.

⁹For demo see: <https://knowsec-demo.denkbares.com/>.

¹⁰See: <http://webprotege.stanford.edu/>.

¹¹See: http://protegewiki.stanford.edu/wiki/Collaborative_Protege.

Table 2.2: Distributed KE tools discussed in the text

| Group of Tools | Tools |
|-------------------------------------|--|
| Full OWL support | WebProtégé, NeOn Toolkit, TopBraid Composer, OntoStudio |
| Documents with semantic annotations | Semantic Wikis, KnowCat (Knowledge Catalyser), Noctua, Visual Analyser |
| Overlays for other data sources | SemiT++, RDFauthor, HyperTwitter, owl2vcs |
| Mobile solutions | IRENE Project, OntoWiki Mobile |

from WebProtégé to the Desktop version and perform more sophisticated tasks from there. Customized version of WebProtégé, named iCAT, is used in the ICD-11 project, mentioned before, as well as in two other WHO classifications: the International Classification of Traditional Medicine (ICTM) and the International Classification of Patient Safety (ICPS) [159].

NeOn Toolkit, one of the results of EU Project for years 2006-2010¹², is a set of Eclipse plug-ins¹³ for collaborative ontology authoring accordingly to OWL 2 specification. Additional plug-ins allow for the creation of ontologies repository or adding the discussion to individual elements of the ontology. Basing on the popular IDE allows developers to create their own plug-ins and customize the tool to suit their needs [156, 177]. Another Eclipse-based tool is TopBraid Composer¹⁴. It is built on the Jena API¹⁵ to support OWL language, and provides a possibility to edit and refactor multiple models at the same time with use of the graphical or form-based editor. What is more, it has the ability to perform inference with using of many popular engines and to visualize knowledge in the form of tree, graph or UML-like view [177]. The last tool in the OWL group is OntoStudio¹⁶ (formerly known as OntoEdit), a commercial product for developing OWL ontologies. Within this tool, group begins with creation of semi-formal system specification, and then they iteratively build more formal ontology. At the last stage, it is compared with the specification. Similarly to the previous two, it is based on Eclipse IDE [151].

Among tools that allow for building a collection of documents with semantic annotations, the most important group are Semantic Wikis systems. They will be further described in Section 3.3. Another tool in this category is KnowCat (Knowledge Catalyser) that allows building web pages (within the tool) grouped into topics with structured knowledge. Built-in mechanism called “Knowledge Crystallisation” analyses users’ interaction with the system and indicates the best written documents [26]. KnowCat also evaluates if users are well-organized or not. The former opens page once for all and reads the pages for a longer time while the latter “jumps” between pages [41]. Noctua¹⁷, is a tool that provides a mechanism called “Virtual Catalyst”.

¹²For EU Project description see: <http://www.neon-project.org>, for Toolkit overview see: <http://www.neon-toolkit.org>.

¹³See: <https://eclipse.org/>.

¹⁴See: <http://www.topquadrant.com/tools/IDE-topbraid-composer-maestro-edition/>.

¹⁵See: <https://jena.apache.org/>.

¹⁶See: <http://www.semafora-systems.com/en/products/ontostudio/>.

¹⁷See: <http://projetos.dia.tecpar.br/noctua/index.php?clicked=info>.

Such a mechanism interacts with users by asking questions (e.g. “Could you write something about X?”) to confirm the existing knowledge and point out conflicts. It is based on automatically created user profiles that describe their fields of knowledge and interests [22, 124]. Last tool within this category is a Visual Analyser (also known as Timeline-Based Analyser), created during KP Lab (Knowledge Practices Laboratory)¹⁸ EU Project for years 2006-2011. The project concerned technology-enhanced learning, but the tool has abilities that may prove useful in KE. It graphically draws users’ interaction with system: each user is represented by a horizontal line, each event by a specific icon (different images for creation, opening, etc.), and events related to the same page are connected with a curve. It also enables to search the graph with specified patterns, e.g. to find all situations where (1) a user uploads a document, (2) another user opens this document, (3) another user deletes it [133].

The next category groups tools that are some kind of overlays for other data sources or tools. The first one is SemT++ project, aimed at collaborative management of digital resources. It provides a possibility to add two types of knowledge: public (visible to all users) and private (visible only for the creator). It is a way to resolve conflicts: the most controversial pieces of information can be in a private zone to help the user and to not disturb others [60]. RDFauthor¹⁹ is a kind of overlay for RDFa[2] annotations enriched websites. It extracts them into the automatically-generated form-based interface, and allows for viewing and changing them. Changes are saved in the database, if appropriate mechanisms are available, e.g. SPARQL Endpoint. The tool works well e.g. with the knowledge bases created with the use of OntoWiki (see Section 3.3.5) [157]. On the other hand, there is a HyperTwitter²⁰. It is a protocol and a tool for placing simple statements in Twitter messages (using an RDF standard), which then can be further processed. Protocol allows for defining relations between tags, users and external URIs. E.g. it can be stated that two Twitter tags mean the same thing [69]. The last tool in “overlays” category is Owl2vcs²¹, previously known as OntoCVS. The tool itself does not support the creation of a knowledge base, but it can facilitate cooperation between those who use some other tools for developing OWL ontologies. The tool is a Java-based plug-in for Git and Mercurial versioning systems that extends classical diff feature to compare ontologies in OWL (in RDF/XML, Turtle and other syntaxes). It compares both logical and technical levels of the ontology [178].

In the last category there are two mobile solutions for Distributed KE. The first one is a work-in-progress framework for sharing the knowledge among group of mobile application users created within IRENE EU Project²². It incorporates the reputation calculation mechanisms for both content and users based on e.g. number of editions, references, votes and reputation of raters [29]. Second solution is a lightweight HTML5-based interface named OntoWiki Mobile. Reliance on common Web technologies allows for independence

¹⁸See: <http://kplab.evtek.fi:8080/wiki/>.

¹⁹See: <http://aksw.org/Projects/RDFauthor.html>.

²⁰See: <https://semantictwitter.appspot.com/>.

²¹See: <https://github.com/utapyngo/owl2vcs>.

²²See: <http://www.planetmedia.es/proyecto-irene/>.

from hardware platform. Application can be used in scenarios, where users do not have a device or power supply that supports full heavy-weighted tool, e.g. scientific expeditions, like Caucasian Spiders mentioned above [45].

As it was presented in this section, Distributed Knowledge Engineering is under interest of a number of different groups. Applications range from strictly academic, through medical and enterprise, to the government projects. The need for the implementation of these multiple goals led to the creation of diverse group of tools, examples of which were discussed here. They include tools for creating formal ontologies, “lighter” tools relying on groups of documents with semantic annotations, as well as a number of auxiliary tools, such as mobile applications and additions to version control systems. Based on this short survey, in the next section an attempt to systematize the definitions of the Distributed KE research area will be provided.

2.2 Taxonomy of Distributed Knowledge Engineering

The most popular term for KE developed by a group of people is Collaborative KE (Google Scholar results: 304²³). This concept describes many different cases, which, in general, boil down to a number of users working on one knowledge base [117, 135]. Another concepts that describe group KE are: Distributed KE (Google Scholar results: 160), Cooperative KE (15) and Collective KE (56). There are no well-established definitions of what they mean [74]. Experts can often understand them in a dramatically different ways [117], or use some of them interchangeably (e.g. Distributed KE and Collaborative KE in [8, 18], or Collaborative KE and Collective KE in [105]). However, they should be distinguished based on the characteristic features indicated by experts and supported by dictionary definitions. This will bring the power to differentiate various situations where multiple users are involved in the creation of knowledge. It will also clarify the context of this dissertation.

Distributed KE is a general term that describes KE process partitioned into a number of users working together on one knowledge base by using different terminals [13, 38]. This process can be spatially distributed [55], but it is not important at all, because users communicate with each other and with the system through a network. 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 [31]. All examples mentioned in Section 2.1 can be considered as a Distributed KE.

Cooperative KE is a process, in which the problem is divided into separate fragments. Each user is responsible for one part [135]. After dividing the task, communication between participants is not required.

They carry out their own goals, but are willing to help others within the possibilities and needs [162].

²³Number of search results in <https://scholar.google.com/>. It is the sum of both “Distributed KE” and “Distributed Knowledge Engineering” search terms. As of 20.02.2017.

Despite the fact that they do not share the common motivation, they ultimately form knowledge, which brings benefits to each of the participants [30]. Cooperative KE can be compared to “divide and conquer” approach. An example of such resolution is HermesWiki (see Section 2.1.1), in which tasks are divided between students and then they can work on them independently.

Collaborative KE is a joint involvement of participants in the project for a common purpose, although it may result from different motivations. In contrast to Cooperative KE, here a few users are operating on the same piece of the problem, which is associated with the need of communication inside the group. There is no necessity for equal involvement in the process, but each of participants often helps others [27, 135, 162]. Collaborative KE is associated with the occurrence of conflicts and dynamics of opinions. This is due to the fact that each of participants has her own experience and beliefs that may differ from others’ [5, 132, 138]. Conflicts can escalate to “trolling” or “edit wars”, but it is not under the scope of Collaborative KE, because of assumption of a common goal which is good quality of knowledge. Collaborative KE examples are ACKTUS-dementia and Dispedia (see Section 2.1.1) where both patients and doctors are sharing knowledge about the same diseases.

Collective KE is done by a whole group of people, who aim towards a common goal, have similar motivations and socio-economic interests. Members of the group are equal, which means the similar level of skills and access to shared resources that are not owned by someone else [162, 28]. In accordance with the results obtained in (Computational) Collective Intelligence, when the group operates in a collective way, it can achieve more (quantitatively and qualitatively) than the sum of achievements of individual agents (both human and artificial systems) [154, 10]. Examples of this group are Caucasian Spiders database and International Classification of Diseases (ICD-11) (see Section 2.1.1). The database is developed by a group of equal participants (biologists and doctors) with shared goal, using their own resources.

It can be easily seen that KE approaches may be treated as a continuum from scattered and chaotic to structured group work (see Figure 2.1). This dissertation is set within the Collaborative Knowledge Engineering (further referred to as CKE). It is the most popular group KE, but, despite this, it generates a lot of challenges, that still await for proper solution and appropriate tools [117]. They will be presented in the following section. 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 solutions proposed within this dissertation will also apply to Collective KE.

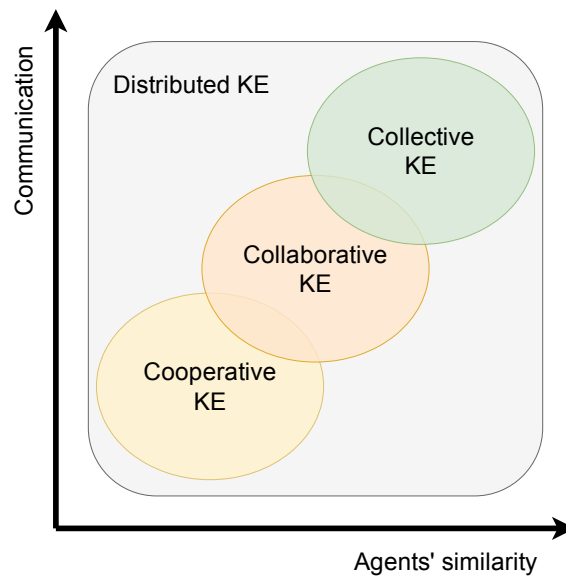


Figure 2.1: Group KE taxonomy

2.3 Issues and Challenges

Experts seem to agree that different use cases require different tools and there never will be a good general tool for all 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 [117]. There is, however, possibility to specify the list of issues and challenges that should be addressed in order to provide proper CKE Methods and Tools.

The first attempt was made in [76], where authors drew attention to the fact that it is necessary to:

- R1. Use a shared repository that provides opportunities for sharing and reuse,
- R2. Guarantee the possibility to create the content in users' own way,
- R3. Prepare an ontology that will provide a big picture of the project,
- R4. Allow the users to prepare knowledge in bottom-up, top-down or spontaneous manner.

It was further extended by seven more requirements [131, 132, 168]:

- R5 Support different expertise levels and access rights,
- R6 Give a possibility to approve, disagree or discuss with others (see also [117, 124]),
- R7 Keep track of the knowledge consistency and warn about conflicts,
- R8 Be compatible with existing tools, standards and methods,
- R9 Conduct a simple knowledge maintenance cycle,
- R10 Keep in mind that the knowledge should be owned and managed by the domain experts,
- R11 Provide a possibility to configure the system to specific enterprise ontologies and processes.

Authors [117] noted that the first prototype CKE tools were created without real investigation about needs and expectations of actual users of such systems. Based on the interviews following requirements that

should be taken into account were highlighted [117, 124]:

- R12. Provide a visualization that helps in understanding the knowledge structure,
- R13. Determine the user's and knowledge's credibility, especially in open environments where everyone can make changes,
- R14. Consider the usability, especially web interface, which is powerful and easy to use, because it does not require installation of many tools,
- R15. Automatically identify conflicts and provide mechanisms to resolve them.

It is worth to pay attention to selected hypotheses that determine which KBs will survive for a long time [119]:

- R16. Provide a high degree of automation, but keep in mind that users prefer usability over automation,
- R17. Experts should be involved quickly in the process of creating the system and they should quickly receive the first benefits,
- R18. Development should be fast and done in an agile way,
- R19. Use mainstream software and hardware, and domain specific tools if it is possible.

CKE process should be considered as an agile process (see Requirement R18). To ensure that, some requirements should be addressed, especially set of practices known as continuous integration. To support it, CKE tool should meet the following requirements [14]:

- R20. Give the possibility to define and execute automatic tests,
- R21. Have a versioning control system to ensure backtracking of bad changes,
- R22. Provide a visualisation of current and previous knowledge base state,
- R23. Make it possible to download a stable version of knowledge base at any time.

The rest of this chapter will provide a description of the works associated with presented requirements that are related to the goal of the dissertation – that is, Ontology Engineering within CKE setting with the use of Semantic Wikis technology.

2.3.1 Knowledge Bases Quality

Knowledge bases quality is probably the most important issue in the KE process. The quality of knowledge influences future possibilities of processing it, as well as utility of the entire system. Transition from traditional KE to CKE raises new issues in this area (see Requirements R6, R7, R13, R15, R20 and R23) that are connected, among others, with the following problems [13]: (1) Heterogeneity: the same concept can be understood differently by different users, e.g. temperature as a number and as a description (cold/warm), which leads to the lack of automatical use of both fragments without explicit translation. (2) Oscillating knowledge: database that can be often changed can lead to its uselessness, because the system is not reliable. It can also lead to stress, as it may be a result of conflicts between users. More issues, especially

connected with Linked Data, are discussed in [180].

One of the solutions for such a problems is an idea of adapting unit tests that are available in SE to CKE needs [164]. Such tests can perform queries to the knowledge base and check whether the acquired results are consistent with desired ones. They can be performed after each change in the knowledge base as traditional unit tests in SE, as well as according to a schedule (e.g. larger tests that burden the system, performed during the night) or on demand (e.g. more complex tests that are triggered by an expert when necessary) [15]. Tests result can be then showed to the users (see Requirement R22). Such tests can be also adapted to KB's debugging in search of inconsistent statements written in the past, as in an ontology debugger [56], based on the Delta Debugging idea from SE [181].

Another possibility to take into account regarding KB quality is to provide metrics that allow the automatic system or expert to assess whether knowledge base improves or deteriorates, e.g. with the use of sophisticated visualisation as in CodeCity Metrics in SE [171], see Figure 2.2 (color = nesting level, width = number of attributes, height = number of methods). Simple metrics for calculating KBs' various characteristics may be the foundation as well. Review based on metrics described in [42, 46, 120, 155, 176, 182] is presented in Table 2.3. For review of metrics related to Linked Data see also [180]. On the top of simple metrics, as well as other system parameters (e.g. number of revisions or ratings from other users), complex metrics such as content's reputation and users' reputation may be calculated [29]. Such an evaluation system may be adapted to current needs: some modules can be disconnected, weights on metrics and other parameters can be adjusted [29].

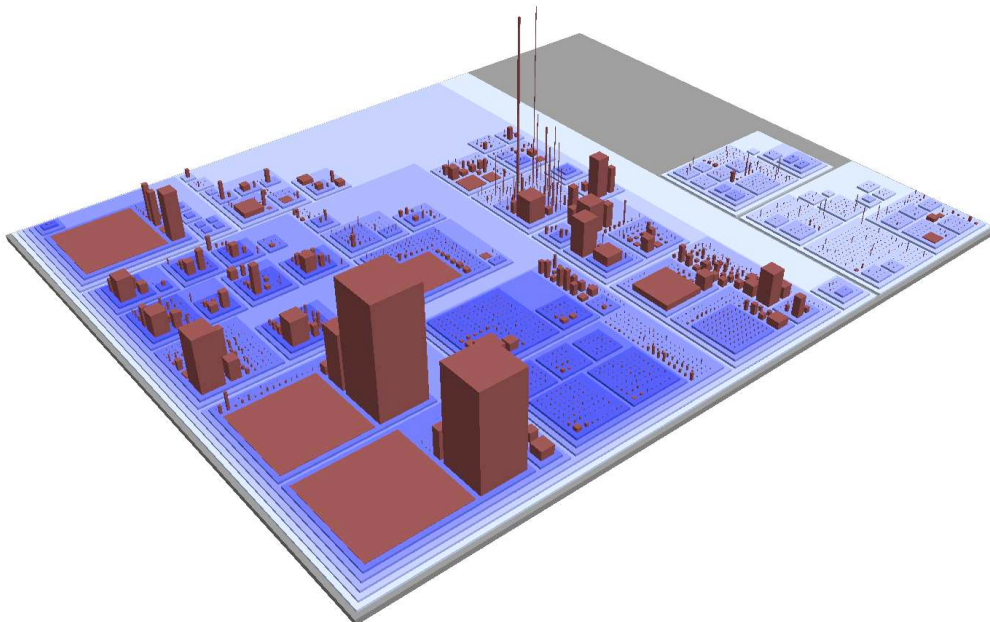


Figure 2.2: CodeCity Metrics example for ArgoUML java package [171]

Table 2.3: Ontology metrics that may be used to evaluate KB quality

| Name | Description | References |
|---|--|----------------|
| Size of vocabulary | Number of defined vocabulary (named classes, instances and properties) | [182] |
| Knowledge enriched | A measure of reasoning ability of ontology: ratio of the amount of all isolated axioms to the amount of all the axioms of a domain | [46] |
| Lack of cohesion in methods | Independence of components | [42] |
| Domain modularity | Number of subdomains in ontology (ability to group the knowledge) | [46] |
| Edge node ratio | Density of connections between nodes | [182] |
| Coupling between objects | Average number of connected classes | [42, 155] |
| Tree impurity | Deviation from a tree structure | [182] |
| Entropy of graph | Variety of graph structure | [182] |
| Schema tree balance | Level of tree balance | [155] |
| Depth of inheritance | (Average) length of the path from leaves to the root | [42, 176, 182] |
| Specific classes count | Number of root / leaf classes, classes with only one subclass, classes with more than 25 subclasses, classes with no definition | [120, 176] |
| Schema inheritance richness | Variety of inheritance | [155] |
| Schema deepness | Average number of subclasses | [155] |
| Number of children / ancestor classes | Average number of direct sub- and superclasses | [42] |
| Tangledness | Average number of classes with more than one parent | [42] |
| Schema relationship / attributes richness | Relationships / attributes variety | [42, 155] |
| Properties richness | Number of properties divided by sum of properties and relationships | [42] |
| Characteristics relevancy | Ratio of essential attributes amount to all attributes of class X | [46] |
| Class relationship richness | Level of relationships usage within class X | [42, 155] |
| Class in-degree | Degree of use of class X by other classes | [182] |

Continued on next page

Table 2.3 – Continued from previous page

| Name | Description | Reference |
|--|---|-----------|
| Class out-degree | Degree of dependence of classes on class X | [182] |
| Class importance | Class X importance based on instances' distribution | [155] |
| Fullness | Ratio of amount of instances belonging to the subtree of the root, which is a class X | [155] |
| Class richness | (Average) Distribution of instances among classes | [42, 155] |
| Knowledgebase instance coverage (cohesion) | Instances coverage of classes | [155] |
| Readability | Degree of readability for human, as the sum of available comments and descriptions | [155] |
| Annotation richness | Average number of annotations | [42] |

As stated by Requirement R6, CKE supporting tool should provide some possibility to report disagreement. Clear vote up/down method is not very useful, therefore there should also be a place for discussion, where user can justify one assessment [117]. Such a discussion may be structured in a form of a specific protocol that describes the rules of how a consensus should be reached (for overview see [124]).

2.3.2 Version Control

KB's building in CKE setting have to be based on tools that support collaboration. Especially, they should provide a robust Version Control Software (VCS; see Requirement R21). In SE there are many VCS systems. One can divide them into three groups: (a) local, where all files are stored on one machine, (b) centralised, based on client-server architecture, like CVS²⁴ or SVN²⁵, (c) distributed, based on peer-to-peer architecture, like Git²⁶. Each of them describes the change in a similar way, as a linear series of changes (with possible branches), where every change is accompanied by information about its author, timestamp and a short commentary that describes the change. Similar linear VCSs are available in the currently existing CKE Tools, especially in wiki systems, which easily allow the possibility to make changes by many users [55].

Another form of change tracking system may be a change ontology. Such an ontology may be low-level, as Collaboration Ontology proposed in [156] that describes changes such as ConceptCreated, ConceptPropertyRemoved and PropertyTypeChange, or like Change and Annotation Ontology (ChAO) [118] that incorporates concepts of KB_Change, Class_Change and connects them with authors and appropriate descriptions provided in annotations. Change ontology may also describe high-level types of contribution [97], e.g. adding content to existing pages, editing others' grammar or spelling, or rewriting whole paragraphs.

²⁴See: <http://cvs.nongnu.org/>.

²⁵See: <https://subversion.apache.org/>.

²⁶See: <https://git-scm.com/>.

Relations between types of changes are not the only characteristic that can be described in a form of a graph. Even entire changelog may be written in that form with the use of PROV standard provided by W3C [58]. It describes data provenance in the form of a graph (see Figure 2.3) that represents relations between entities (physical, digital and other kinds of things), activities (actions and other system dynamics) and agents (people, software, etc). Such a graph can be serialized using RDF notation (see Section 3.2 for more information) and then parsed and analyzed with the use of general RDF libraries or specialized PROV tools:

- Prov Viewer²⁷ [85] – simple Java visualisation tool for small graphs.
- PROV-O-Viz²⁸ [71] and ProvStore²⁹ [73] – powerful online tools for visualisation in a form of a Sankey diagram, as well as a graph similar to the one presented in the Figure 2.3.
- Komadu³⁰ [152] – tool for collecting, analysing and visualising PROV files.
- Prov Python³¹ – a Python library for analysing PROV files, used e.g. in ProvStore.

PROV graph can be used, among others, for data quality assessment, as it provides a way to analyze content trustworthiness and authors' level of expertise [101]. Another kind of graph was developed in KP-Lab Project. It is based on three main concepts, analogous to the ones in the PROV model: Thing, Activity, Actor. However, here attention is directed towards analysis of the interaction between users, not to the analysis of the origin documents as in W3C PROV [39, 133].

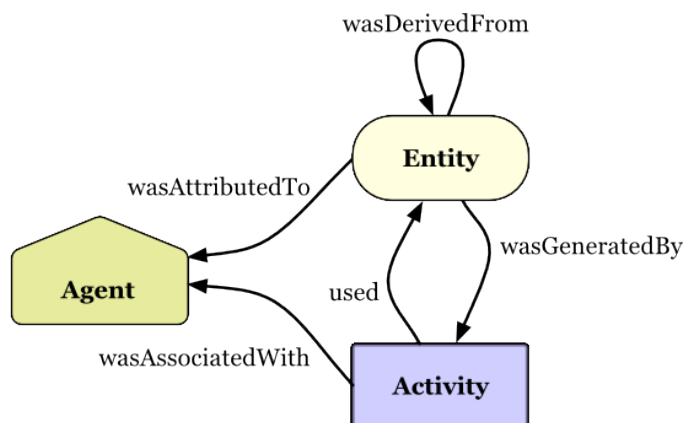


Figure 2.3: PROV model overview [58]

It is also worth mentioning that there are attempts to automatically generate some kind of semantic changelog graph based on existing historical data, e.g. [114, 115] try to parse unknown changelog text file, to discover the types of columns and the relationships between them, and on this basis to generate a RDF graph.

²⁷See: <https://github.com/gems-uff/prov-viewer>.

²⁸See: <http://provoviz.org/>.

²⁹See: <https://provenance.ecs.soton.ac.uk/store/>.

³⁰See: <https://pti.iu.edu/impact/data-sets/komadu.html>.

³¹See: <http://prov.readthedocs.io/en/latest/index.html>.

2.3.3 Agile Development

Regardless of the tools' issues described above, associated with the guarantee of proper KB's quality and traceability of the changes, there is also need for mature collaboration methods: best practices, development conventions, patterns about organization of collaborative groups and knowledge engineering cycle (see [175, 18], as well as Requirements R4, R9, R17, R18). The base should be constituted by a general KE methodology. It consists of six main phases [93, 169, 43, 107], similar to waterfall process in SE [147]:

1. Identification of the problem,
2. Knowledge acquisition,
3. Knowledge modeling within the KB,
4. Application of inference mechanisms,
5. Evaluation of the system,
6. Maintenance.

In Ontology Engineering this process was detailed by several specific methodologies, but there is no standard approach in this domain (for overview see [77, 20, 141]). Among existing methods, there are seven most significant approaches (that are well-described and were used in big projects): TOVE (*Toronto Virtual Enterprise*), Enterprise Model Approach, METHONTOLOGY, Ontology 101, Holsapple and Joshi approach, Dogma-Mess and DILIGENT.

TOVE [61] process consists of the following steps:

1. stating motivating scenarios,
2. asking informal competency questions,
3. specifying terminology (in formal logic),
4. formalising competency questions,
5. specifying the whole ontology,
6. evaluating the ontology.

Enterprise Model Approach [160] starts with identification of purpose and required level of formalization. Then, the scope is identified and ontology is formalised according to the specification. Finally the ontology is evaluated.

METHONTOLOGY [49, 57] process starts with a formal specification. After that, knowledge is acquired and conceptualized. Next, integration with existing ontologies is considered. Ontology is then finally implemented and evaluated. The whole process is complemented with the documentation.

Ontology 101 [116, 177] defines an iterative process that consists of several steps carried out in a cycle:

1. Determination of the domain and the scope of the ontology,
2. Consideration of reusing existing ontologies,
3. Enumeration of important terms,

4. Definition of the general taxonomy,
5. Definition of the properties and their constraints,
6. Creation of instances.

Holsapple and Joshi approach [72] starts with preparation of boundary conditions and selection of evaluation techniques. Then, first version of ontology is prepared to anchor the users. After that, the ontology is iteratively extended by the knowledge achieved from experts' interviews until the consensus is reached.

Dogma-Mess [33] methodology starts with scope and aim specification, as well as evaluation in terms of costs and benefits. Next, after time management process is started, preparation step is undertaken, when requirements and knowledge resources are defined. After that, ontology is finally written down by brainstorming and negotiation with domain experts. Process is concluded by tailoring the ontology to application-specific context.

DILIGENT [165] methodology consists of the following steps done in a cycle:

1. New version of ontology is built.
2. Users adapt it to their own needs.
3. Local users' changes are analysed and a set of them is selected for implementation in the base ontology.
4. Ontology is released and users can update their local copies to the new version.

Transition from KE to CKE requires modernization of methodologies – from those developed 20 years ago, to more robust agile methods that respond to frequently changed requirements or non stable domains [159, 130]. Analogical transition was made in SE, from traditional waterfall model to agile [153]. This is associated with a big change that resulted from many differences between the two approaches (see Table 2.4). In particular, implementation of agile methodology is connected with steps differing from traditional model. For example, Scrum methodology consists of the following phases [153]:

1. Select a Product Owner, a person with the overall vision of the project, a team of 3 to 9 people, and a scrum master, a person who will lead the team through the framework.
2. Create a Product Backlog, a single list of everything that should be done in the project.
3. Estimate the Product Backlog. Estimation should be done by people who will execute the tasks and should be done in some abstract measure, not in hours (e.g. small, medium or large).
4. Plan a Sprint. The Product Owner, the Team and the Scrum Master should evaluate how much of the Product Backlog they can do in the Sprint (one-two weeks of work). Furthermore, a Sprint Goal should be stated by the Team. It cannot be changed during the Sprint.
5. Make work visible. Prepare a Scrum Board that groups tasks into three categories: To Do, Doing, and Done, or prepare a Burndown Chart to present how much work has been done in the Sprint.
6. Arrange daily Stand-up meetings. For no more than fifteen minutes, the Team answers three questions:

"What did you do yesterday?", "What will you do today to help the team finish the Sprint?" and "Is there any obstacle blocking you?".

7. Remember that the Team is autonomous. Each member will assign tasks themselves. There's no assigning of tasks from above.
8. Review the Sprint. At the end, the Team shows what they have done at the open meeting, where also management board and customers can attend. Only completed features are presented. After that, the Team reflects on what went right and what can be done better in the next Sprint.
9. Finally, the next Sprint cycle starts with the Sprint Planning session.

Table 2.4: Main differences between classical approach and the agile way (based on [136])

| | Classical Approach | Agile Way |
|--|--|---|
| Process structure | Phase-based and sequential | Iterative and incremental |
| Change/emergence | Change should be avoided | Embrace change in an economically sensible way |
| Uncertainty management | Eliminates uncertainty at the start | Reduces uncertainties simultaneously |
| Decision making | In the proper phase | Keeps options open |
| Achieving right results the first time | Assumes one can create the proper requirements and plans | One cannot achieve right results up front |
| Predictive versus adaptive | Highly predictive | Mix of predictive plans with adaptive just-in-time work |
| Feedback | Late (near the end of the process) | Fast (at the end of every iteration) |
| Cost of delay | Rarely considered | Always considered |
| Conformance to plan | Conformance is important to achieve a success | Adapt and replan rather than conform |
| Centricity | Follow the process (process-centric) | Deliver the value (value-centric) |
| High quality | At the end, after an extensive test-and-fix phase | Built from the beginning |

Such an agile methodology for CKE should be generalized based on experience from many projects. Nevertheless, in the literature there is probably only one detailed analysis that tries to specify phases within the real project post factum. Logs analysis within the Catalogus Professorum Lipsiensis (described in Section 2.1.1) gives the possibility to enumerate six different phases within this project [134]:

1. Analysis of information within existing non-ontological database and transformation to preliminary ontology.
2. Initialization of the ontological system with the preliminary data.
3. Knowledge acquisition and ontology refinement as a result of close cooperation between KE experts and historians.
4. Publication of the catalogue to the public on the web. Knowledge acquisition and engineering activities were intensified due to the feedback of the web community.

5. Interlinking with other datasets, especially with DBpedia project and the German National Library.
6. Alignment to other prosopographical ontologies.

An agile CKE methodology can be also build in a top-down manner based on SE experiences. Such a general high-level adaptation for Knowledge-Based Systems (KBS) development has been done in [12] (see Figure 2.4). It starts with the System Metaphor phase when an overall plan is developed. Then, the Planning Game is undertaken to decide about the scope and priority of development in the near future (e.g. Sprint perspective in the Scrum methodology). After that, selected tasks are implemented and tested. Finally, new functionalities are integrated into the KBS and integration tests are done to ensure the proper work of the extended system. Such a framework fits into the CKE Requirements, but it is only a general idea. A mature agile methodology for CKE still has not been developed [56].

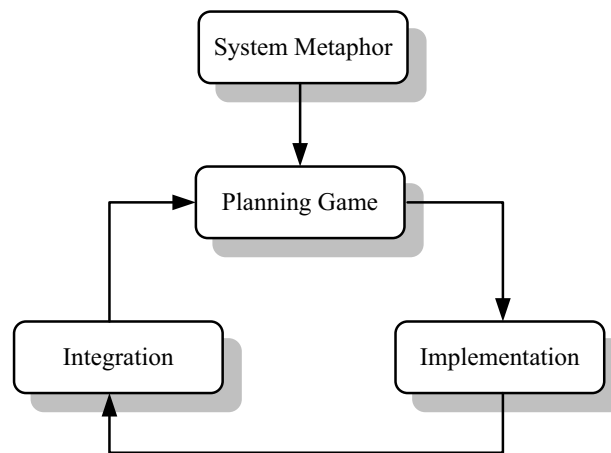


Figure 2.4: The agile process model for knowledge-based systems [12]

With CKE agile methodology, relation between knowledge engineers and domain experts is inextricably intertwined. Traditionally, knowledge engineers use a variety of methods, including structured interviews, to obtain all the knowledge from domain experts and fit it into the system (for overview of knowledge capturing methods see [32]). However, CKE requires that domain users should be the owners and managers of the domain knowledge, not knowledge engineers (see Requirement R10). One solution for this requirement is to separate both groups. Engineers may prepare the knowledge schemas that domain experts will fill [81], but this does not satisfy the Requirement R2, which states that people should be allowed to create the content in their own way. Another solution is to use less formalised knowledge representation than OWL, such as semantic annotations, which are still powerful, but also easy to understand by domain experts.

Finally, CKE agile process should be supported by some metrics that ease process's evaluation. Such metrics could be used to describe the self-organization of the user [41], based on the time she was viewing the documents and how often she returns to the document (well-organized user opens the document once for a longer time). In turn, [80] built machine learning models to evaluate collaboration quality. Analyses indicated that for such an assessment (within the tool that allows for communication with other users)

the following metrics are important: (a) communication-related metrics: the number of chat messages, the alternation of chat messages and the mean response time in chat messages, (b) workspace-based metrics: symmetry in main actions and overall workspace actions. Their findings also indicate that too much activity can be related to redundant actions, which are result of a bad coordination within the team.

2.3.4 User Involvement

Last but not least, users' motivation should be discussed. The participation in CKE process is not spontaneous [124]. It should be rather considered through the prism of Prisoner's Dilemma (PD) [89], a classical problem in the game theory, or even as "Tragedy of the commons" (me versus they) [65]. In brief, within the CKE process, a user wants to get as much as possible, while giving as little as possible [126]. This is especially apparent in larger group, because the contribution of a single person is difficult to see (for discussion see [124]). Such a PD motivation model is rooted in three dimensions [126]: individual vs collaborative, influence on others vs not, short vs long term. The last dimension is particularly interesting, because it differentiates the classical PD (short term perspective), where non-cooperation is the best choice, and the repeated PD (long term perspective), where collaboration is the best strategy, as it gives larger income during longer game.

Motivation is exceptionally important issue nowadays – when people modify their work and projects frequently, or when they leave the project, the knowledge they have leaves the project with them. This causes the so-called "corporate amnesia" [32]. In general, researchers [32, 124] state that to enable participation, a sense of community and a climate of trust, where knowledge sharing is encouraged, are necessary. In such a setting, a mix of three main motives plays the role [126]: (a) traditional cost-benefit rationality (if monetary compensation is available), (b) self-interest, (c) community-interest and altruism. Authors stress especially the role of the second category, e.g. recognition by peers, improving future work opportunities and gaining better software. Individual motives can be supported by different positive and negative incentives, both for encouraging collaboration and for discouraging negative behaviour, as in classical instrumental conditioning proposed by B.F. Skinner [146]. These incentives can be grouped into three categories [25, 32]: (a) Material rewards: especially money, (b) Moral incentives: sense of self-esteem and approval for doing the right thing, as well as sense of guilt and condemnation for doing something wrong, (c) Coercive incentives: punishments, imprisonment or even the physical force as a result of failure or doing the wrong thing.

Incentives can be quite tricky, because different individuals can perceive the same thing as a punishment or as a reward, so it is advised that incentives should fit particular person or group [32]. For example, users can be divided accordingly to the expertise-level. It can be done manually, by assigning users to specific groups within system, as in HermesWiki project [129] (see Section 2.1.1), where there were experts and regular users, but also automatically, e.g. based on semantic similarity methods used to calculate expertise

centroids within the system [184]. Such different expertise groups may have different access rights within the system [129] or different system interfaces, designed to relieve pressure on the more advanced users, so they can concentrate on providing high-quality product [29]. Users can also be divided accordingly to the types of changes. Clustering analysis lead to the specification of two main groups [97]: Adding (content, pages and comments, small corrections) and Synthesizing (integration, reorganization and rewriting the whole paragraphs). Synthesizers can be then motivated by the impact they can have: on the organization, task, or other users of the system. On the other hand, adders are not concerned about their reputation and impact, but rather about easy work process and fulfilling their formal roles.

It is also worth mentioning that proper user interface (UI) is important and may be decisive for success or failure of the whole project [126] (also Requirements R4, R5, R7, R11, R12, R15, R16 and R19). Determination of proper UI is a matter of research in usability domain. In particular, the usability stack for knowledge-based systems was developed [52] (see Picture 2.5). Each layer of the stack depends on the previous one, and all of them indicate the proper way of building usable KBs: (a) It begins with the definition of system model that takes into account the real-world context (System Image). (b) Then, immediate feedback is guaranteed to provide Self-Descriptiveness of the system in both successes and failures, while (c) user control is developed (e.g. undo/redo actions and constraints on inputs). (d) Based on them, minimalistic aesthetical Interface Design is built to clearly present information to user. All layers are supported by errors handling and easily accessible documentation. Building an UI accordingly to the Usability Stack may be integrated with previously presented Agile Process Model (see Figure 2.4). On different process stages, different UI prototypes can be delivered [54]. UI prototyping can be also partially automated, based on the previously defined templates [53].

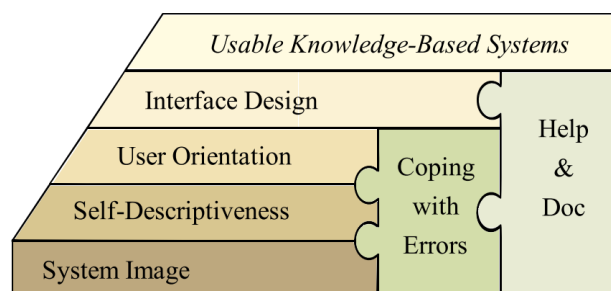


Figure 2.5: Usability Stack for knowledge-based systems [52]

Developed UI may be then tested to ensure that it copes with users' needs. The most popular techniques are [95]:

1. *usability testing*: observation of actual future users using the system under controlled conditions;
2. *heuristic evaluation*: experts evaluate whether the system fulfills certain usability requirements;
3. *cognitive walkthroughs*: a group of people tries to perform specific tasks on the pages to determine whether the typical tasks run without any problem, e.g. number of clicks needed to start the most

important functions is checked [156];

4. *focus group*: experts conduct a group interview with users (requires at least 6 users);
5. *thinking aloud*: users are thinking aloud while performing tasks to provide better feedback;
6. *questionnaire*: the cheapest and fastest way of collecting data, at the cost of not being able to ask more sophisticated questions that would be able to capture very detailed nuances of the system. Within this group there is e.g. Software Usability Measurement Inventory (SUMI)³² [84], that consists of 50 statements with possible answers: Agree, Don't Know, or Disagree. The questionnaire provides reliable usability assessment with minimal sample size of 10-12 respondents [83].

For more comprehensive review of usability evaluation techniques see [51].

Users' motivation may be also promoted by the usage of gaming techniques. Firstly, game mechanics elements may be implemented into the working process (so-called gamification). Among them there are: achievements ("Best employee of the month" reward), combos related to completing a combination of actions (certificate given after finishing X hours of theoretical training and Y hours of practical one), quests (after completing a series of certification, user is given a higher status) and points (granted for completing tasks, may be further exchanged for e.g. day out or meeting with the boss). For more comprehensive overview see [170, 112]. Gamification in CKE was proposed in Semantic Knowledge Management System as a primary source of motivation for annotating documents [100] and was successfully used in ICD-11 project (see Section 2.1.1), where Top 10 boards were implemented. Further reports stated that users really cared about them [159]. On the other hand, ontology-related concepts may be hidden behind the multiplayer online game that has a real problem as a goal. Example of the so-called serious game is the WhoKnows project, presented as a quiz, but with the goal of cleaning DBpedia database [167]. Challenges associated with creating Serious Games in CKE environment are discussed in [143, 144].

2.4 Summary

In this chapter Collaborative Knowledge Engineering domain was presented. Firstly, some cases were described to provide an intuition about the scope of the area. Then, definitions were clarified to distinguish CKE from other close areas. Finally, list of issues associated with supporting CKE was presented and related work were discussed. To narrow the scope of the dissertation, following chapter will cover Semantic Wikis as an example of simple yet powerful CKE tool. After that, in Chapter 4, approach that extends Semantic Wikis technology to address requirements presented in this section will be proposed.

³²See: <http://sumi.uxp.ie/>.

Chapter 3

Semantic Wikis as CKE Tools

Research suggests that Wiki systems are proper tools for collaborative work because of their usability, brought by easy installation and maintenance without a long introduction training [126]. The development of the Semantic Web technology has allowed for extending the capabilities of the Wiki systems for the 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, easy to use, and on the other, offering the ability to process knowledge at a certain level of formalization. This chapter covers all these topics. It begins with description of Wiki systems in Section 3.1. Then, in Section 3.2 Semantic Wikis technologies overview is provided. General Semantic Wikis capabilities description as well as comparison of the current wikis is placed in Section 3.3. Chapter is concluded with description of Loki, the Semantic Wiki system used as a base for prototypical implementation of methods and tools proposed in this thesis.

3.1 Basic Wiki Systems

The first wiki system was c2 wiki¹, developed in 1995 by Ward Cunningham to provide a place for software patterns repository [166]. These days, there are a lot of wiki implementations with various features, written in almost every programming language². All of them are connected by the fact that they allow users to create knowledge “the wiki way”³. They are containers for text pages (so called “wiki pages”) and media files (images and others) and provide an environment that is [91, 172]:

- Simple – easy to use,
- Open – any reader can edit an incomplete page,
- Incremental – a page can cite other pages that are not prepared yet,
- Organic – structure and content easily evolve,

¹See: <http://wiki.c2.com/>.

²For more information see: <http://wiki.c2.com/?WikiEngines> and <http://www.wikimatrix.org/>.

³Term “wiki way” was proposed by Cunningham in [91]. “Wiki” comes from Hawaiian “wiki wiki” what means “fast”.

- Mundane – simple markup for plain text pages,
- Universal – writing and editing use the same mechanisms,
- Overt – based on the output, one can easily predict the markup of the page,
- Precise – page titles are specific enough to avoid name clashes,
- Tolerant – pages are rendered even if they contain errors,
- Observable – activity can be reviewed by all visitors,
- Convergent – duplicated content can be removed by finding and citing related one.

Wiki's popularity is revealed in the fact that they are used in many projects, developed in cathedral-style (centralized effort of a closed group of developers), bazaar-style⁴ (everyone can contribute) and mixed-style [128, 166]. Surveys [122, 125, 138] group the usages into three categories:

- *single-contributor wikis* for personal knowledge management,
- *group collaboration tools*, e.g. for software development documentation and coordination, project knowledge management (e.g. ideas and notes) and collaborative writing (short story, novel, etc.),
- *encyclopedia systems*.

There are several other collaborative technologies, e.g.. video conferences, online chats and instant messaging systems, discussion forums, or weblogs [32, 126]. Still, among those, wikis have become the most popular tools for massive collaboration. It is only thanks to such systems that we are able to keep up with recording our knowledge into the KB [166]. Their design principles force specific social norms that promote collaboration over individual work. The resulting knowledge creation process is characterized by [126]:

- *Collective authorship*: there is no single author, everything is done as a group, which is good for community and task-oriented people.
- *Instant publication*: all changes are visible just after page is saved and others can use them immediately.
- *Versioning*: there is always a possibility to revert changes, so it is difficult to spoil a wiki.
- *Simplicity of authorship*: low cost of entry to the new technology encourages even these ones that will provide only small changes.
- *Division of labor* [97]: contributors can focus on different aspects, ie. content creation or integration.

Wikis are pretty simple and robust systems for collaboration, as indicated by the relatively high quality of the Wikipedia project [79]. However, there is also much to do to provide a powerful CKE system. Especially, most of the ordinary wikis are written using natural language [121]. Even if they have a lot of data and a lot of interlinks between pages making a big knowledge graph, they cannot be easily queried in an automatic way [55, 79]. Also, the knowledge quality in such a wiki cannot be measured easily [166]. These problems can be assessed by an incorporation of semantic web technology, which is the matter of the next section.

⁴Cathedral- and bazaar-style terms were coined by E. Raymond to describe two approaches to free software development [128].

3.2 Semantic Web

There is a lot of data on the web presented in various forms: text, tables, images, audio files, etc. They are easily accessible to humans, but on the other hand they are also “dumb”, because they are not synchronized and are difficult to process by automated tools. One possible solution for these problems is an attempt to improve the consistency, availability and inter-connectivity of the data on the Web. Such data could be then processed by appropriate tools, that will prepare more “smart” results [7]. This is the idea that led to emergence of the Semantic Web: to add semantics to existing web infrastructure, which will enable information exchange between the agents [7, 21]. Semantic is here understood as a meaning of information in the web [7], and as a machine-processability of the information [140].

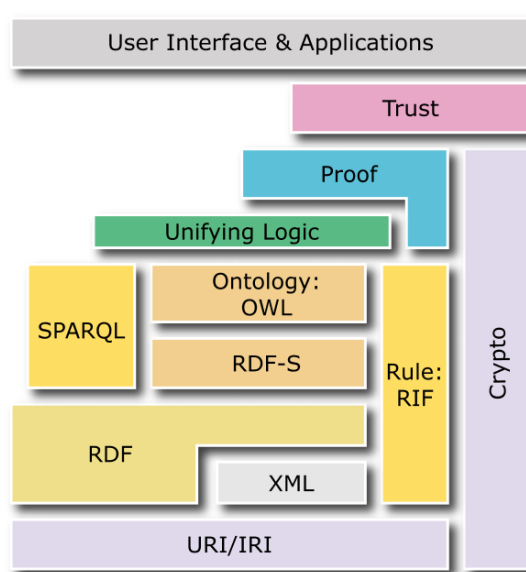


Figure 3.1: The Semantic Web Stack of technologies [23]

The Semantic Web is a set of standards that can be depicted in a form of a stack, where each layer is based on the ones below (see Figure 3.1). At the bottom of the Stack, there are Uniform Resource Identifiers (URIs). They are character strings that provide unique names for physical (e.g. people, books) and conceptual (e.g. scientific theories) resources [140]. It is recommended to use the same global URI when two or more agents are referring to the same resource [7].

Semantic knowledge is described with the use of the Resource Description Framework (RDF), a basic representation language of the Semantic Web [7]. RDF statements are simple triples that always consist of three elements: subject, predicate and object. Subject and object are resources, the relationship between which is described by the predicate, e.g. <Krzysztof Kutt> <is a> <PhD candidate>. RDF triples can be visualized as a directed graph with nodes representing subjects and objects, as well as arcs representing predicates [139]. Subject and predicate are identified by URIs. Object is identified by URI or by a literal (string, number, date, ...). RDF graphs can be serialized by the use of one of four available

syntaxes [139, 174]):

- RDF/XML – The original scheme that uses XML notation (see Listing 3.1),
- Turtle – A simple, human- and machine-readable format (see Listing 3.2),
- RDFa – A standard to put the RDF triples directly in HTML (see Listing 3.3),
- JSON-LD – The newest scheme aimed especially at web services and web developers (see Listing 3.4).

More sophisticated relations between concepts can be written down in a form of an ontology in the Web Ontology Language (OWL), an extension for RDF. OWL allows users to define Classes, Properties and their instances, as well as to describe their characteristics, e.g. to define that one property is an inversion of another one [174].

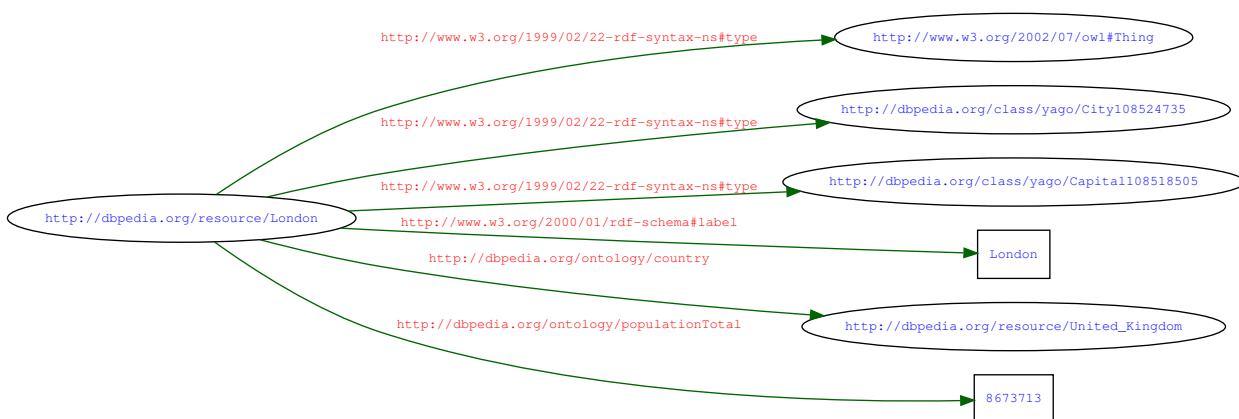


Figure 3.2: Sample RDF graph that represents selected properties of London available in the DBpedia

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <rdf:RDF
3   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
5   xmlns:dbo="http://dbpedia.org/ontology/">
6
7 <rdf:Description rdf:about="http://dbpedia.org/resource/London">
8   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
9   <rdf:type rdf:resource="http://dbpedia.org/class/yago/City108524735"/>
10  <rdf:type rdf:resource="http://dbpedia.org/class/yago/Capital108518505"/>
11  <rdfs:label xml:lang="en">London</rdfs:label>
12  <dbo:country rdf:resource="http://dbpedia.org/resource/United_Kingdom"/>
13  <dbo:populationTotal rdf:datatype="http://www.w3.org/2001/XMLSchema#
14  nonNegativeInteger">8673713</dbo:populationTotal>
15 </rdf:Description>
16 </rdf:RDF>

```

Listing 3.1: Sample RDF graph (see Figure 3.2) serialized using RDF/XML syntax

As was mentioned before, it is recommended to use the same URI when referring to the same thing. So it is recommended to use one of existing dictionaries where possible [174]. Among the most established vocabularies there are:

```

1 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2 PREFIX owl: <http://www.w3.org/2002/07/owl#>
3 PREFIX dbc: <http://dbpedia.org/class/yago/>
4 PREFIX dbo: <http://dbpedia.org/ontology/>
5 PREFIX dbr: <http://dbpedia.org/resource/>
6 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
7
8 dbr:London a owl:Thing, dbc:City108524735, dbc:Capital108518505 ;
9     rdfs:label "London"@en ;
10     dbo:country dbr:United_Kingdom ;
11     dbo:populationTotal "8673713"^^xsd:nonNegativeInteger .

```

Listing 3.2: Sample RDF graph (see Figure 3.2) serialized using Turtle syntax

```

1 <div prefix="rdfs: http://www.w3.org/2000/01/rdf-schema#
2     owl: http://www.w3.org/2002/07/owl#
3     dbc: http://dbpedia.org/class/yago/
4     dbo: http://dbpedia.org/ontology/
5     dbr: http://dbpedia.org/resource/
6     xsd: http://www.w3.org/2001/XMLSchema#">
7 <p resource="dbr:London" typeof="owl:Thing
8     dbc:City108524735
9     dbc:Capital108518505">
10 <span property="rdfs:label" lang="en">London</span>
11 is the capital city of
12 <a property="dbo:country" resource="dbr:United_Kingdom" href="http://dbpedia.org/
13 page/United_Kingdom">United Kindom</a>.
14 The total population of London is estimated
15 <span property="dbo:populationTotal" datatype="xsd:nonNegativeInteger">8673713</
16 span>.
</p>
</div>

```

Listing 3.3: Sample RDF graph (see Figure 3.2) embedded into HTML using RDFa syntax

- The Dublin Core Metadata Initiative (DCMI)⁵ – a set of metadata for media objects such as audio files, books or images, e.g. creator, title, language, SizeOrDuration.
- The Friend-of-a-Friend (FOAF)⁶ – a dictionary for describing people and relations between them, e.g. familyName, knows, member, currentProject.
- Simple Knowledge Organization System (SKOS)⁷ – a model for representing the basics of taxonomies, e.g. Concept, broader, definition, example.
- The Description of a Project (DOAP)⁸ – a vocabulary to describe software projects, e.g. Project, Repository, developer, programming-language.

Prepared RDF graph can be accessed and processed by the use of SPARQL Query Language [66]. Queries can be executed against and RDF file loaded into the local database (so-called triple-store), or remotely against the SPARQL Endpoint – a web service connected with specific dataset. SPARQL is a graph pattern matching language: a query is like a set of RDF triples with the possibility to use variables in place of any element of the triple. Such a pattern is then compared to the RDF database, and subgraphs that are

⁵See: <http://dublincore.org/documents/dcmi-terms/>.

⁶See: <http://xmlns.com/foaf/spec/>.

⁷See: <https://www.w3.org/TR/skos-primer/>.

⁸See: <https://github.com/ewilderj/doap/wiki>.


```

1 {
2   "@context": {
3     "owl": "http://www.w3.org/2002/07/owl#",
4     "dbc": "http://dbpedia.org/class/yago/",
5     "dbr": "http://dbpedia.org/resource/",
6     "xsd": "http://www.w3.org/2001/XMLSchema#",
7     "label": {
8       "@id": "http://www.w3.org/2000/01/rdf-schema#label",
9       "@language": "en"
10    },
11    "country": {
12      "@id": "http://dbpedia.org/ontology/country",
13      "@type": "@id"
14    },
15    "populationTotal": {
16      "@id": "http://dbpedia.org/ontology/populationTotal",
17      "@type": "xsd:nonNegativeInteger"
18    }
19  },
20  "@id": "dbr:London",
21  "@type": [
22    "owl:Thing",
23    "dbc:City108524735",
24    "dbc:Capital108518505"
25  ],
26  "label": "London",
27  "country": "dbr:United_Kingdom",
28  "populationTotal": "8673713"
29 }

```

Listing 3.4: Sample RDF graph (see Figure 3.2) serialized using JSON-LD syntax

equivalent to the query are selected [66]. There are five types of SPARQL queries [66]:

- SELECT queries return the specified data directly as plain text, XML or JSON document.
- CONSTRUCT queries combines the results into a single RDF graph.
- ASK queries are used to test whether the given query pattern has a solution. They return simple true/false values.
- DESCRIBE queries are executed to receive a single RDF graph that describes all specified resources. They are only informally indicated in the SPARQL specification, so their behaviour may depend on the specific implementation.
- UPDATE queries are performed to modify the graph. It includes the methods to update, create, and remove RDF triples.

At the top of all standards in the Semantic Web Stack, there is applications layer. Here, the most popular group are the so-called semantic mashups – the web applications that combine data from various semantic sources [21, 98]. There are also other tools, like HypperTwitter – an extension for Twitter service⁹, that adds a possibility to embed an RDF triple into Tweets to provide some semantics for them [69]. Another popular group of Semantic Web applications are Semantic Wikis that will be described in the next section. For more comprehensive description of Semantic Web technologies see available textbooks [7, 67, 140, 174, 177].

⁹See: <https://twitter.com>.

3.3 Semantic Wikis

Wiki systems enriched with Semantic Web technologies are simply called Semantic Wikis, which are the subject of this section. It begins with a discussion of different ways in which semantics can be added to a wiki in Section 3.3.1. Then, in Section 3.3.2, overview of systems currently under maintenance is provided. Finally, one specific Semantic Wiki called Loki, maintained at AGH-UST, is described in Section 3.4.

3.3.1 Semantics in Wikis

Wikis allow users to write the knowledge using a natural language. Semantic Wikis extend this by adding a possibility to also use a semi-formal language to explicitly model the relationships between pages. As a result, better information retrieval, wiki navigation and knowledge reuse and reasoning is achieved [55, 79, 121]. Particularly within Semantic Wikis one can query the knowledge directly (“list all cities”), navigate the content using the knowledge (“list other cities in UK”) or specify background knowledge (“every city has a mayor and is placed in a country”) [79]. To do this properly, Semantic Wikis should address the following questions [121]: 1) how to annotate content?, 2) how to formally represent content?, 3) how to navigate content? Based on how they realize this, they could be grouped in the following categories¹⁰:

- *Simple Semantic Wikis*. In them, wiki markup is simply extended to add a possibility to specify RDF-like triples. Mostly, the first element of the triple (subject) is a page on which the annotation is specified. Second element (predicate) is then the annotation name and third one (object) is a value of an annotation. As wiki page is a subject of the triple, it is most often used to describe a single concept or instance [24]. The main advantage of classical wiki systems is their simplicity, resulting in high usability. During the transition to Semantic Wikis, one has to keep in mind, when introducing new semantics, to not complicate the whole system [55].
- *Semantic Data Wikis* [55]. These systems are not only simply extensions to traditional wikis, but also they are built from scratch to handle semantic data, e.g. instead of simple text annotations within plain text, knowledge is edited in properly prepared forms. Such a structure avoids redundancy, often found in Simple Semantic Wikis, where knowledge is stored both in the text annotations of the page and in the external database, which allows for real-time responses to queries [55]. On the other hand, the imposition of the structure in forms restricts the possibility of arbitrary page layout.
- *Semantic Knowledge Wikis* [11]. Both Simple Semantic Wikis and Semantic Data Wikis can handle semantic data and provide basic search and navigation functionalities, but not necessarily anything more. It is important, therefore, to specify the third category (not disjoint with the others), Semantic Knowledge Wikis. These are systems that use specific knowledge representation (e.g. XTT2 in

¹⁰For another classification see [24]. Author listed simple semantic wikis, ontology-based systems and unconventional ones.

Loki [3, 108]) and reasoning capabilities (e.g. HeaRT inference engine in Loki) to solve some problems. Knowledge in such wikis can then be verified similarly to verifying knowledge in knowledge systems, e.g. one can verify the completeness of the knowledge base and the occurrence of contradictions. For discussion about using such methods in Semantic Knowledge Wikis see [13]. If a system supports different knowledge representations, wiki could be iteratively developed starting with less formal knowledge that will be then manually or semi-automatically precised. In particular. one can start with natural language and images, then extend it with semantic annotations and finally generate the set of “if ... then ...” rules in KnowWE [16].

3.3.2 Overview

The first semantic wiki (Platypus) was proposed in 2004. Then, in 2005-2006 “semantic wiki explosion” was observed when many semantic wikis implementations appeared. In subsequent years, more systems were created, but not as many as in the beginning. A comprehensive description of these systems and their history were provided by Bry *et al.* in 2012 [24]¹¹. Authors also discuss a wide range of semantic wikis knowledge engineering use cases, like Encyclopedias or eGovernment. It is worth mentioning that such systems are also successfully used in software engineering projects, see e.g. [34, 37, 92, 109, 110]. Here, only systems that are currently under maintenance are listed, that is the systems that have the latest releases or commits in the last 1.5 years (15.11.2016-15.05.2017):

1. Semantic MediaWiki (SMW, Section 3.3.3),
2. KnowWE (Section 3.3.4),
3. OntoWiki (Section 3.3.5),
4. Loki (Section 3.4),

Their summary with regard to CKE Requirements (see Section 2.3) is presented in Table 3.1.

3.3.3 Semantic MediaWiki

The most popular¹² semantic wiki implementation is Semantic MediaWiki (SMW) [35, 36, 86, 87, 88], an extension to the MediaWiki system¹³. The wiki extends markup to give the possibility to describe: category of the page: `[[Category:City]]`, relations with other wiki pages (object properties): `[[Is capital of::United Kingdom]]` and relations with literal values (data properties): `[[Has population::7,421,329]]` (see Figure 3.3). Object properties may be specified by the definition of object type, e.g. relation `Is capital of` may have a `Country` object. Wiki also provides the possibility to: state

¹¹For list of current and “historical” semantic wikis see also: http://semanticweb.org/wiki/Semantic_Wiki_State_Of_The_Art and <http://www.mkbergman.com/sweet-tools/>.

¹²See list of selected wiki instances at: https://www.semantic-mediawiki.org/wiki/Category:Wiki_of_the_Month.

¹³See: <https://www.mediawiki.org/>.

Table 3.1: Semantic wikis comparison with regard to the CKE Requirements (see Section 2.3)

| | SMW | KnowWE | OntoWiki | Loki |
|-----------------------------------|---|--|--|--|
| Publications | [35, 36, 86, 87, 88] | [15, 17] | [8, 55, 68] | [105, 106] |
| Web page | https://www.semantic-mediawiki.org/ | https://www.d3web.de/ | http://ontowiki.net/ | http://loki.ia.agh.edu.pl/ |
| Last update ¹⁴ | 17.05.2017 (v. 2.5.2) | 09.08.2016 (v. 11.0) | 04.10.2016 (v. 1.0.0) | 26.10.2016 (v. 2016-10-26) |
| Technology | Extension to MediaWiki (PHP). Storage in MediaWiki relational database or any Triple Store ¹⁵ | Built on the JSPWiki (Java EE). Storage in relational database | PHP (Zend Framework). Storage in relational database or Triple store (via Erfurt API ¹⁶) | Extension to DokuWiki (PHP). Uses SWI-Prolog. Storage in plain text files |
| R1: Shared repository | All Wikis (by definition) keep the knowledge in a shared repository (database or files on wiki server) accessible by all users | | | |
| R2: Create in the own way | Media, plain text, annotations, business processes, UML class diagrams | Media, plain text, annotations, production rules, decision trees, decision tables, flowcharts, set-covering models | Forms with different types of fields' values | Media, plain text, annotations, Prolog facts and clauses, XTT2 rules, business processes and rules ¹⁷ |
| R3: Ontology as a big picture | Yes, OWL-DL | Yes, OWL-DL | OntoWiki is only an interface for arbitrary RDF datasets, so it depends on the data | Only list of existing annotations |
| R4: Bottom-up and top-down manner | All Wikis (by definition) give the possibility to refer to concepts and instances not described yet, giving the possibility to start from general concepts, specific instances and any others | | | |
| R5: Access rights | All Wikis support specification of access rights. None of them support different expertise levels | | | |

*Continued on next page*¹⁴Checked on 21.05.2017.¹⁵See: https://www.semantic-mediawiki.org/wiki/Help:Using_SPARQL_and_RDF_stores.¹⁶See: <http://aksw.org/Projects/Erfurt.html>.¹⁷Processes and rules may be edited and visualized, but they cannot be queried.

Table 3.1 – Continued from previous page

| | SMW | KnowWE | OntoWiki | Loki |
|---|---|---|--|---|
| R6: Approval, disagreement and discussion | Special page for discussion for each wiki page | No | Discussion about small information chunks | No |
| R7: Consistency tracking | No | No | No | Yes, limited to XTT2 rules |
| R8: Compatibility | RDF and SPARQL support | RDF and SPARQL support. Compatible with domain-specific standards | RDF and SPARQL support. Uses well-established vocabularies (SKOS, FOAF, ...) | Export to RDF; SPARQL and BPMN support; SMW-style queries |
| R9: Simple maintenance cycle | All Wikis are quite simple tools that do not have many options, so maintenance is also quite simple | | | |
| R10: Experts as owners | All discussed Wikis do not require a lot of specific KE knowledge, so they can be simply used by domain experts without a long training | | | |
| R11: Adaptation to use case | All Wikis have plugin systems that enable the possibility to adapt to specific use case. KnowWE developers also specifies a markup for each use case for better adaptation | | | |
| R12: Visualization | Yes, limited to the specific page | Yes, limited to the specific page | Yes, limited to the specific page | Yes, limited to the specific page |
| R13: Credibility determination | No | No | No | No |
| R14: Usability | All Wikis provide simple syntax and intuitive browser-based interface. Only Semantic MediaWiki usability was tested in experimental setting, giving good overall results [179] | | | |
| R15: Conflicts identification | No | No | No | Yes, limited to XTT2 rules |
| R16: Automation | No | No | No | No |
| R17: Experts in the process | All discussed Wikis, thanks to simple components, allow the experts to enter at the very beginning of the process: first, while creating the underlying ontology in conjunction with the Knowledge Engineers, and later, when they can independently develop the knowledge | | | |
| R18: Agile development | Agile development is related, inter alia, to the appropriate methodology, which at this moment has not yet been developed for KBs [56]. Agile has also some technical requirements, e.g. set of practices known as Continuous integration (see Requirements R20-R23). These requirements are fulfilled only by KnowWE | | | |
| R19: Mainstream and domain specific tools | All Wikis have plugin systems that enable the possibility to connect with domain specific tools | | | |

Continued on next page

Table 3.1 – Continued from previous page

| | SMW | KnowWE | OntoWiki | Loki |
|----------------------------------|---|-----------|-----------|-----------|
| R20: Automatic tests | No | Yes | No | No |
| R21: Versioning control | All Wikis (by definition) keep track of previous versions to give the possibility to rollback bad changes. Changes are kept as lists of changes for each page | | | |
| R22: Current state visualization | No | Yes | No | No |
| R23: Stable version at any time | In all Wikis except KnowWE there are no automatic tests to validate current state, so the stable and unstable versions are not differentiated | | | |
| Requirements fulfilled | 17 | 17 | 14 | 14 |

values conversion (e.g. from miles to kilometres), and specify categories and relations hierarchy [86]. There was a notable attempt to create the Distributed SMW system that kept knowledge on multiple servers to be independent of single point of failure [145].

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

```

1 { {#ask:
2   [[Category:City]]
3   [[Located in::United Kingdom]]
4   |?Population
5 }}

```

SMW's queries can be used to define concepts, e.g. concept UK City may be a result of `[[Category:City]] [[Located in::United Kingdom]]` query. Such concepts may be used the same way as categories in further queries, allowing for grouping pages in a more sophisticated way [86].

3.3.4 KnowWE

KnowWE (Knowledge Wiki Environment) [15, 17], system based on JSPWiki¹⁸, is not a simple wiki with semantic annotations. It was developed as a decision support system that mixes knowledge on different formalization levels, ranging from pictures to strong problem-solving knowledge, e.g. production rules [15, 16] (see Figure 3.4). The wiki was used to define clinical guidelines and HCI devices configuration [15], chemical substances assessment [18], ancient Greece description [129] (see Section 2.1.1), and many more. KnowWE does not have a generic annotation mechanism. It uses markup and editors created for specific use case to reduce the complexity for domain experts [15, 86].

3.3.5 OntoWiki

OntoWiki [8, 68, 55] is an example of Semantic Data Wiki that was established as a response to problems with Simple Semantic Wikis [8] (see Section 3.3.1). The wiki was not developed as an extension to conventional text-based Wiki like Semantic MediaWiki (see Section 3.3.3). As such, it is based on the pOWL editor, which is used for collaborative ontology engineering¹⁹. As a result, OntoWiki does not provide text pages with specific markup, but special forms (see Figure 3.5) that structure the knowledge into the form of an ontology. Strictly speaking, OntoWiki is an overlay for an arbitrary RDF/OWL dataset, which allows for its edition without the deep ontology engineering knowledge [68, 55]. OntoWiki knowledge can be queried

¹⁸See: <http://jspwiki.apache.org/>.

¹⁹See: <http://aksw.org/Projects/Powl.html>.

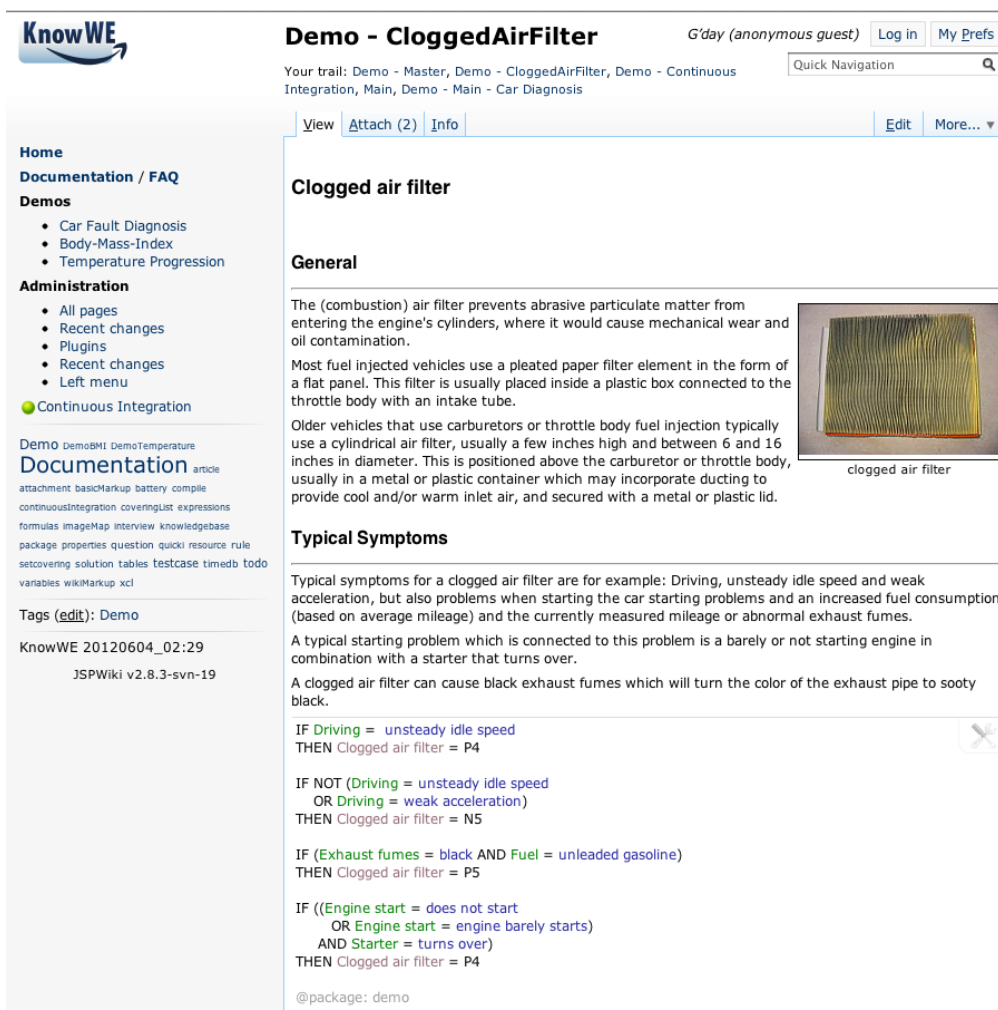


The screenshot shows the Semantic MediaWiki (SMW) interface for the page 'London'. At the top right, there is a 'Log in / create account' link. Below it, navigation tabs include 'Page', 'Discussion', 'Read', 'Edit', and 'View history'. A search box with 'Go' and 'Search' buttons is also present. The main content area features the title 'London' and a paragraph: 'London is the capital city of [England](#) and of the [United Kingdom](#). As of 2005, the population of London was estimated 7,421,328. Greater London covers an area of 609 square miles.' Below this is a 'Category: City' box. A 'Facts about London' section includes a table with the following data:

| Area | 1,577,302,759.2 m ² (1,577.303 km ² , 157,730.276 ha, 609 miles ²) + |
|------------|--|
| Capital of | England +, and United Kingdom + |
| Population | 7,421,328 + |

On the left side, there is a navigation menu with links: 'Main Page', 'SMW homepage', 'Help', 'Browse wiki', 'RDF Feeds', and 'Recent changes'.

Figure 3.3: Semantic MediaWiki wiki page [86]



The screenshot shows the KnowWE wiki page for 'CloggedAirFilter'. At the top right, there is a 'Log in' and 'My Prefs' link. Below it, a 'Quick Navigation' search box is visible. The main content area features the title 'Clogged air filter' and a 'General' section. The text describes the function of an air filter and includes an image of a clogged air filter. Below the text, there is a 'Typical Symptoms' section with a list of symptoms and a diagnostic rule. The diagnostic rule is as follows:

```

IF Driving = unsteady idle speed
THEN Clogged air filter = P4

IF NOT (Driving = unsteady idle speed
OR Driving = weak acceleration)
THEN Clogged air filter = N5

IF (Exhaust fumes = black AND Fuel = unleaded gasoline)
THEN Clogged air filter = P5

IF ((Engine start = does not start
OR Engine start = engine barely starts)
AND Starter = turns over)
THEN Clogged air filter = P4

```

On the left side, there is a navigation menu with links: 'Home', 'Documentation / FAQ', 'Demos', 'Administration', and 'Continuous Integration'. The 'Documentation' section lists various topics like 'attachment', 'basicMarkup', 'battery', 'compile', etc.

Figure 3.4: KnowWE wiki page [15]

Figure 3.5: OntoWiki wiki page about professor Ernst Heinrich Weber²⁰

locally by SPARQL queries, but it also allows external queries via the SPARQL Endpoint and LinkedData Endpoint [55].

The Wiki was successfully used in many situations, ranging from historical data in Catalogus Professorum Lipsiensis [134], through medical diagnosis in Dispedia [44], to animals classification in Caucasian Spiders project [123] (see Section 2.1.1). The latter was done in the wild of the Caucasus area, where full laptop client is not useful, so there is also a lightweight HTML5-based version of OntoWiki that can be browsed on the mobile phones [45].

3.4 Loki

Loki (Logic in wiki) [105, 106] is another Semantic Wiki, developed as a plugin for DokuWiki system²¹ that stores knowledge using Prolog-based representation to provide strong reasoning capabilities. To be more precise, Loki is a part of a set of independent plugins developed in PHP for different knowledge formalizations (see Figure 3.6): Loki, main semantic wiki plugin, BPwiki [109] for storing and visualizing

²⁰The page is available at http://catalogus-professorum.org/view/r/Weber_1030.

²¹See: <https://www.dokuwiki.org/>.

Business Processes in BPMN notation within wiki (business requirements), and SBVRwiki [110] for processing business rules accordingly to the SBVR specification (software engineering). It makes Loki flexible and easily adaptable to current needs by selecting only the plugins that are useful in the domain of interest.

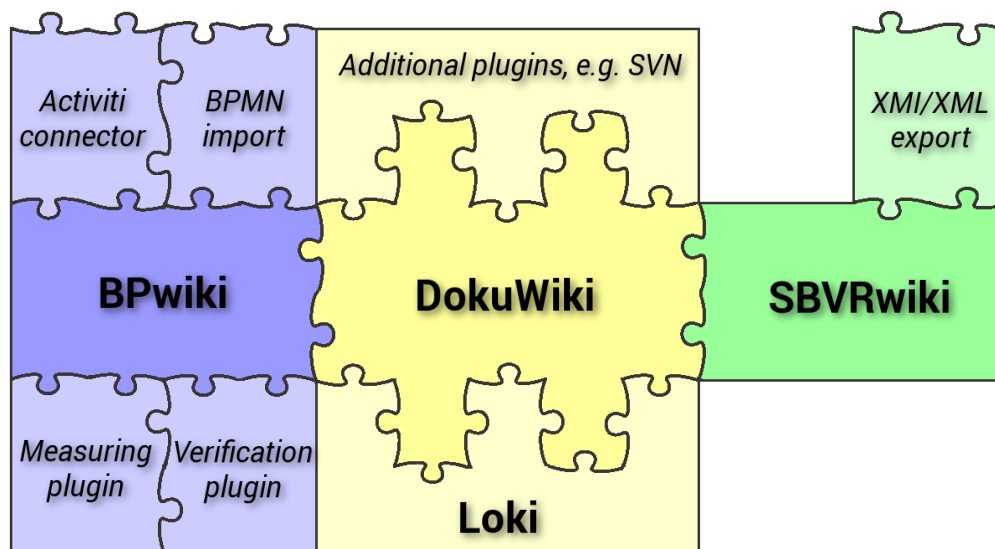


Figure 3.6: A set of plugins for various knowledge formalizations: Loki, BPwiki and SBVRwiki

Loki uses annotations system inspired by the one present in Semantic MediaWiki. In particular, one can specify at london wiki page: a category of the page: `[[category:City]]`, relations with other wiki pages (object properties): `[[is_capital_of::united_kingdom]]` and relations with literal values (data properties): `[[has_population:=7421329]]`. These annotations are then translated to the following Prolog predicates:

```

1 wiki_category('london','City').
2 wiki_relation('london','is_capital_of','united_kingdom').
3 wiki_attribute('london','has_population','7421329').

```

The created database can be queried by SPARQL and SMW-like query language. E.g. to list all cities that are capitals of United Kingdom one can specify the following queries:

```

1 === SMW-like query ===
2
3 {{#ask: [[category:city]] [[is_capital_of::united_kingdom]]}}
4
5 === SPARQL query ===
6
7 <pl format="sparql">
8 PREFIX wiki: <>
9 SELECT ?page
10 WHERE {
11     ?page a "city".

```

```

12     ?page wiki:is_capital_of wiki:united_kingdom.
13 }
14 </p1>

```

Queries are translated into Prolog, executed on the current database and, finally, the results are translated back to the wiki interface.

Loki also gives the possibility to specify more formal knowledge in the form of Prolog statements that can be mixed with classical annotations presented above. E.g. to list all UK cities one can create the following goal:

```

1 <p1 cache="true">
2   uk_cities(X) :-
3     wiki_relation(X, 'is_placed_in', 'united_kingdom') .
4 </p1>

```

and then print results on a selected wiki page:

```

1 <p1 goal="uk_cities(X), write(X), nl, fail" scope="*"></p1>

```

Knowledge can be also expressed as decision rules' tables in the XTT2 (eXtended Tabular Trees) notation [111]. Loki provides a visualization of XTT2 rules for better user experience, after putting them inside `<hmr></hmr>` tags (see Figure 3.7). These decision tables can be then used for inference in HearT Rule Engine [104], e.g. to generate recommendations for the user [3, 108] (see Figure 3.8).

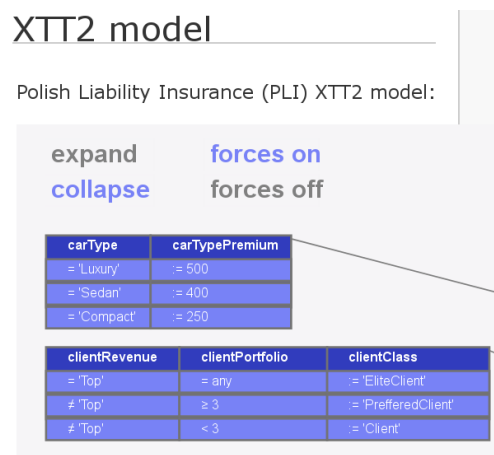
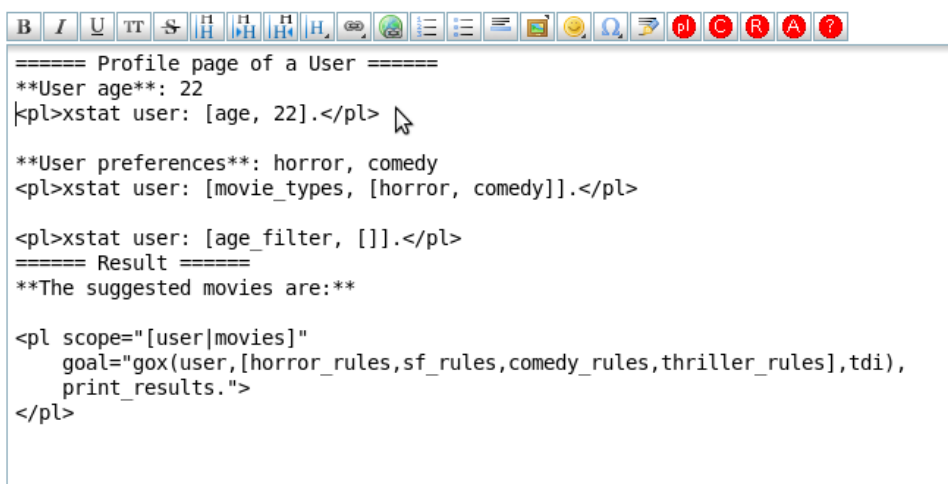


Figure 3.7: XTT2 model visualization inside Loki wiki page

Loki was used in Prosecco (Processes Semantics Collaboration for Companies) project²², which goal was to enhance and simplify the IT system for small and medium enterprises using intelligent technologies such as Semantic Web, Business Process Modeling, Business Intelligence and Machine Learning. Loki was there incorporated as a repository of SBVR rules, using SBVRwiki plugin.

The presented functionalities describe the state of Loki as of June 2014, i.e. at the beginning of research

²²See: <https://geist.re/pub/projects/prosecco:start>.



```

===== Profile page of a User =====
**User age**: 22
<pl>xstat user: [age, 22].</pl>

**User preferences**: horror, comedy
<pl>xstat user: [movie_types, [horror, comedy]].</pl>

<pl>xstat user: [age_filter, []].</pl>
===== Result =====
**The suggested movies are:**

<pl scope="[user|movies]"
  goal="gox(user,[horror_rules,sf_rules,comedy_rules,thriller_rules],tdi),
  print_results.">
</pl>

```

Figure 3.8: Loki wiki page text with recommendations generated by HearT rule engine [108]

conducted in this dissertation. This base was then extended by a set of modules described in Chapters 4-7, developed in 2015-2017. Loki's development was facilitated thanks to two system properties:

- It is based on DokuWiki, a system that has a robust plugin system. As a result, developed functionalities may be divided into a set of plugins. Such an architecture provides a possibility to adjust the system to current needs by selection of desired plugins.
- Loki core provides a set of methods for managing the stored knowledge base, that can be used in further development, e.g. `list_attributes()` that returns all attributes used in the current wiki instance.

3.5 Summary

In this chapter Semantic Wikis were presented. Firstly, classical Wiki systems were described. Then, introduction to Semantic Web technologies was provided. Based on them, Semantic Wikis' idea and different ways to add semantics into wiki were discussed. Chapter was concluded by the presentation of four currently under-development wiki systems: Semantic MediaWiki, KnowWE, OntoWiki and Loki. The latter, developed at AGH-UST, is a base for prototypical implementation of methods and tools proposed within this dissertation. Their detailed description is provided in Chapters 5-7. Earlier, in the next chapter, approach overview is provided.

Chapter 4

Approach Proposal

This chapter provides an overview of methods and tools proposed within this dissertation. First of all, in Section 4.1 the list of CKE support requirements presented before (see Section 2.3) is extended with the ones identified by the author. All these issues are then grouped into six fields of CKE support. Four of them are selected as an area of interest of this thesis in Section 4.2. Then, in Section 4.3 a general overview of proposed set of methods and tools is given. Detailed description of the solution starts with the definition of the *CKE agile process* in Section 4.4. The chapter is concluded by introduction of sample *European Wiki* project that is used within this dissertation to present the proposed set of methods and tools in action.

4.1 Fields of CKE Support

Within Collaborative Knowledge Engineering setting (see Section 2.2) many issues arised. Except the ones summarized in Section 2.3, some requirements identified by author should also be taken under consideration:

- R24. The sources vary in the knowledge of the topic, in the same way the users do. Is it possible to identify which of them are more useful for knowledge base?
- R25. Conflicts in collaborative setting are natural. People in general, and experts in particular, may have different views on the same subject. How can we identify and resolve such conflicts? (these are different than knowledge conflicts mentioned in Requirement R15)
- R26. There are different kinds of users and different types of changes: some users aim at enhancing the knowledge base, the other ones at bringing coherence, and yet others at improving text quality. Is there a way to identify them and use this information in some way?
- R27. Some users are motivated by well designed “quests” and small, even objectively worthless, rewards, so the use of gamification techniques in the CKE process is considered for them. However, one should keep in mind that some users prefer “hard work”, and they do not find game elements at all.

All presented issues may be grouped into six CKE support fields of interest:

- CKE agile process – From the whole range of issues presented in the dissertation, the most important are the ones related to the process, i.e. the need for defining it in the robust agile way (R9, R18), that takes care of different kinds of users (R26) and their expectations (R3, R4, R17).
- Tools – CKE systems have to be built on the base that allows collaboration. Such tools should have a shared repository (see Requirement R1) with the possibility to control the users' access levels (R5). Compatibility with existing mainstream systems is a great advantage (R8, R19). To provide a robust solution, current KB status should also be properly managed (R22, R23). It is also important to take care of specific project requirements and adapt the tool to them (R11).
- Knowledge representation and reasoning – as domain knowledge should be managed by domain experts, not knowledge engineers (R10, R17), there is a need to develop a proper knowledge representation that will be both easy to use for experts (R2, R17) and powerful in terms of inference capabilities. This representation adjustment should be done specifically to the project, group or enterprise that will use the CKE tool (R11).
- Quality management – the need for assuring the proper level of knowledge quality within the KB. Issues related to review process (R6) and automatic knowledge verification (R20), as well as knowledge consistency (R7, R15) and credibility (R13, R24) are considered. Also, handling of users' conflicts is an important issue (R25).
- Change management – management of factual changes in the database, i.e. the need for a proper versioning mechanism (R21), that takes care of different types of changes (R26). Current state visualization also should be considered (R22).
- User involvement – as participation in CKE process is not spontaneous, various incentives should be considered to motivate people. They may be provided in a form of gamification (R26, R27), i.e. the usage of game elements in a serious tool. Also, usability issues (R14), i.e. taking care of compatibility with established practices (R8, R19), providing proper visualisation and automation capabilities (R12, R16) and taking into account specific users' characteristics (R17, R26) are discussed here.

4.2 Area of Interest

It was decided not to take up the *tools* and *knowledge representation* fields within this dissertation. The former one cover issues related to the preparation of appropriate tools. These are strongly technical tasks connected with various Software Engineering and design challenges, ranging from creation of suitable database, through assertion of adequate security level, to preparation of proper user interface widgets. The tasks considered are also strongly case-specific. As researchers agree, different usage scenarios require different tools and there will never be one common tool for CKE [117]. Knowledge representation and reasoning on the other hand is a well-developed field of research, with many ongoing studies. Within CKE setting, searching

for proper representations for specific tasks is particularly interesting and performed by many researchers. Nevertheless, regarding of the aim of this dissertation, i.e. to provide some methods and tools for general CKE process, existing representations are considered sufficient.

Finally, the scope of this dissertation are four fields of interest: *CKE agile process*, *quality management*, *change management* and *user involvement*. They will be investigated in the CKE setting, where:

- participants are generally motivated to work, which follows the definition presented in the Section 2.2,
- participants form a closed group of people (like in a cathedral style), but the development is done in an agile and decentralized way (like in a bazaar style [128]), which defines many ongoing CKE projects,
- group is working voluntarily, and as a result material (e.g. money) and formal (e.g. risk of being fired) incentives for users are not considered. This is one of the areas of interest of Collaborative Knowledge Management, not CKE (see Section 1.4), so it is beyond the scope of the dissertation.

Within specified setting, methods and tools proposed in the thesis will apply to the CKE process understood as a creation of KB structured using the ontology-like knowledge representation. It is assumed that knowledge is grounded in RDF abstract syntax and may or may not be specified in the OWL ontology language (see Section 3.2).

4.3 Approach Overview

The core of the proposed approach is the *CKE agile process* that provides a description of an iterative procedure executed by a group consisting of product owner, scrum master, domain experts and knowledge engineer(s) (see Section 4.4). It is supported by a set of modules for CKE systems that fulfill selected requirements from the ones presented before. These modules are interrelated, yet generally independent. The decision whether specific group needs all of them or only selected ones is left to this group. The core of the modules' set is a PROV-based graph (see Section 2.3.2). It is a versioning mechanism that extends classical linear changelogs. Using the RDF syntax, it models relations between entities (concepts, wiki pages, articles, ...), agents (users and system bots) and activities (concept annotation, page deletion, ...) (see Section 6.2). It is extended with information from other modules:

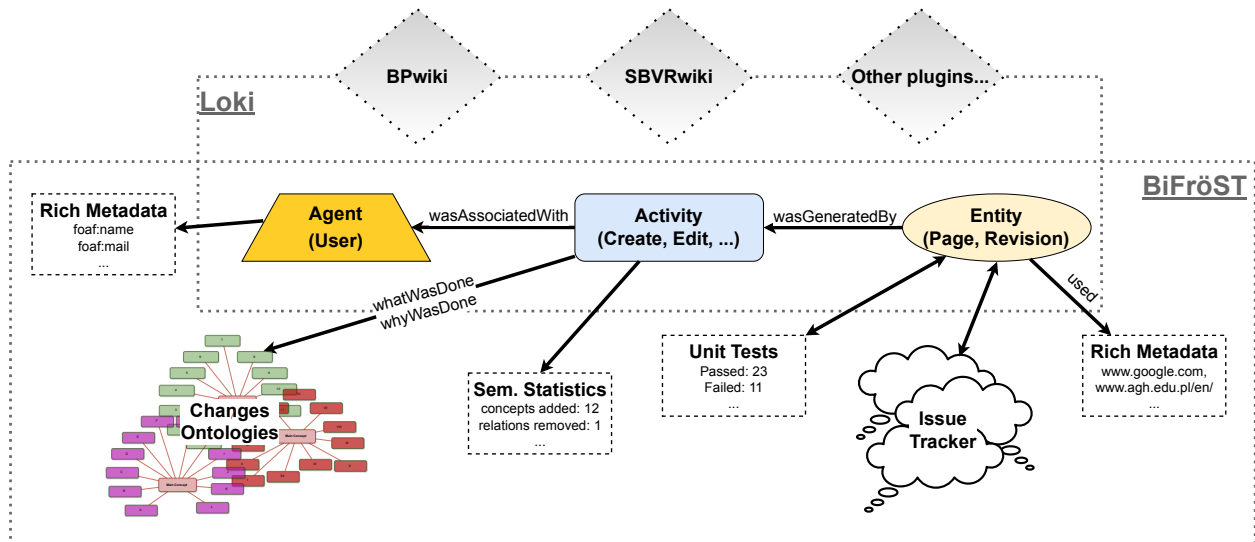
- *Reasoning Unit Tests* is a mechanism of automated unit tests executed after every change in the KB. Tests are defined by the users (domain experts) to reflect their expectations regarding the system and saved in the well-established SPARQL query language, together with the expected results (see Section 5.1 and Requirements R8, R10, R20; page 16). In the PROV-based graph, number of tests are conducted before and after each change is placed.
- *Metrics of Changes* are introduced to cover more sophisticated quality issues than in unit tests. Depending on the knowledge representation used and system capabilities, many metrics can be introduced (see Table 2.3 for overview). As in unit tests, metrics should be calculated before and after each KB

change to reflect the quality of the change made. It is proposed that a set of metrics should be summarized into one “weighted average” metric to provide a simple indicator of current system quality (see Section 5.2 and Requirements R7, R13, R15, R24; page 16). Calculated metrics’ values are finally placed in the versioning graph.

- *Opinions and Discussion* is a module introduced to cover those aspects of changes that cannot be evaluated by automatically computed characteristics (unit tests and metrics of changes). Two mechanisms are proposed: an opinion mechanism that gives users a possibility to evaluate every change made in the system (e.g. on 1-5 scale or using like/dislike buttons) and a place for discussion between experts to provide the possibility to justify the evaluation, as well as to discuss all ambiguities (see Section 5.3 and Requirements R6, R25; page 16). If the graph versioning module is available within the system, all votes are saved in this graph.
- *Change Ontologies* enrich the description of actions made within the system. Core operations, differing in system-level mechanisms (like concept creation, edition, annotation, ...), do not exhaust the issue of describing changes. One should also take into account that change is a complex thing that is stated by the factual change (*What* was done? E.g. *typos or other small bugs fixed* or *new content added*) and the goal (*Why* it was done? E.g. *errors fixing* or *knowledge database expansion*), that form a kind of ontology of changes and their goals. Such an ontology could be prepared for every specific task “type”, e.g. one for preparing conference papers and the other for developing SE system specification (see Section 6.1 and Requirements R21, R26; page 17). Ontological values selected by users are saved in the PROV-based graph for further processing.
- *Rich Metadata and Semantic Statistics* are change properties saved in the versioning graph to extend the inference possibilities. Besides “standard” metadata available in all changelogs (like timestamp, user and small comment), the module also incorporates knowledge sources information, i.e. identifiers (URIs) of books, manuals, reports, web pages, ..., as well as internal sources (other concepts and articles within the KB) used to provide the current change. Additionally, some statistics are provided about the changes on the semantic level of the KB, e.g. how many concepts, instances and relations were added, removed or edited during the current change (see Section 6.2 and Requirements R21, R24; page 17).

They are complemented by discussion of *user involvement* issues, both related to gamification (see Section 7.1) and usability (see Section 7.2).

All proposed modules were prototypically implemented as a part of the thesis. As a base tool, semantic wikis were selected, as – thanks to the plugins systems (see Section 3.3) – they are easy to use by experts and easy to extend by developers. In particular, all prototypes were developed as extensions to Loki system, developed at AGH UST (see Section 3.4). All provided modules form the BiFröST framework (the name is

Figure 4.1: The BiFröST framework architecture¹

an acronym for “BiFröST Framework für Semantical Tracking”). The base for the framework is a PROV-based graph that represents relations between entities (wiki pages and their revisions), activities and agents (users). E.g. the graph states that the `first` revision of a wiki page `dissertation` was created on 31/08/2016 by `Chris` (which consists of three triples that connect specific revision (subject) with the wiki page, creation time and author). New facts (triples in the RDF form) are added to the changelog after one of three possible activities within wiki: page creation, edition and deletion. The graph incorporates also pieces of information conveyed by developed modules described below (see Figure 4.1):

- *Reasoning Unit Tests* are saved in the separate wiki namespace (`tests:`) and are executed after every wiki edition. They are written down as the ASK and SPARQL queries using the same syntax that is used for regular queries within Loki. Expected results are defined by the set of provided types of assertions (e.g. “number of results should be greater than X”, “among results there should be a Y wiki page”).
- *Metrics of Changes* prototypically implemented in Loki are: *Attribute Richness*, *Average Population*, *Size of Vocabulary* and *Edge Node Ration*. These base metrics are then summarized into *Weighted Average* metric that could take one of two available forms: (a) building a base from scratch, which promotes the largest base growth in terms of number of classes, instances and relations, (b) increasing coherence of the existing KB, which promotes new instances addition and the limitation of the number of classes and relationships.
- *Opinions and Discussion* are implemented as a separate `talk:` page for each of the regular wiki pages (e.g. there is a `talk:phd:chapter4` wiki page for discussion about `phd:chapter4` page). These pages consist of lists of all page revisions that can be evaluated on the 1-5 scale and

¹Loki/DokuWiki, and as a result the whole BiFröST framework, may be extended with other plugins to support various more specific knowledge representations, e.g. BPwiki for business processes and SBVRwiki for business rules.

a discussion mechanism that supports the threads to structure the dialogue.

- *Changes Ontologies* are presented to users as two fields in the page edition form, where users have to select “What was done” and “Why it was done” from the available lists or to provide their own description using `Other . . .` option. The presented options are tailored to the general CKE process within Loki.
- *Rich Metadata* within the wiki are essentially pieces of information about sources used to prepare the current revision of the page. These may be specified by the user in the page edition form.
- *Semantic Statistics* include the number of concepts, instances and relations that were added, removed or corrected in the current revision of the wiki page.

The advantage of the proposed solution relies on the power of the PROV graph, that can be (automatically) queried with SPARQL. With this in mind, sample use case scenarios are indicated, as a response for selected requirements from the ones mentioned before: (a) user types identification and support, (b) underdeveloped pages identification, (c) motivation by gamification. More detailed ideas and concrete implementation within the BiFröST framework are provided in Section 6.2.

4.4 CKE Agile Process

In the Chapters 5-7, the set of methods and techniques is presented. They are technical solutions related to the preparation of suitable tools to support a group of users involved in the CKE process. However, to make good use of them, it is important to define the appropriate process of Collaborative Knowledge Engineering first. As has been pointed out in Section 2.3.3, various processes have been developed, which can be applied to the CK, but there is still a lack of mature agile processes.

The author’s work began with an analysis of the existing methodologies. As a result, a process for preparation of KBs was developed² [102]:

1. *Motivating scenarios*: preparation of exemplary use cases that the KBs should address.
2. *Competency questions*: specification of questions which ontology should answer. They can be also written down formally in form of *reasoning unit tests* (see Section 5.1).
3. *Knowledge acquisition*: gathering of the knowledge from experts, domain texts and other resources.
4. *Conceptualisation*: preparation of formal knowledge specification as lists of concepts, objects and relations.
5. *Integration*: prepared idea may be integrated with the vocabulary from existing well-established ontologies, e.g. FOAF to describe people.
6. *Implementation*: ontology is finally written down using selected formal system, e.g. in OWL 2 notation

²This work was done in close collaboration with Mateusz Ślażyński during the Prosecco project. See: <https://geist.re/pub/projects/prosecco:start> for more information about project.

using Protégé editor or in a Semantic Wiki.

7. *Evaluation*: verification of the ontology. It should be done on a formal level to check for completeness, redundancy and contradictions, as well as on a conceptual level to check how reliably it represents the domain, e.g. using the reasoning unit tests specified in step 2.
8. *Documentation*: concepts, relations and all design decisions are commented. A set of competency questions and a set of reasoning unit tests may be also considered KB documentation.

All of these steps are performed in multiple iterations until the expected state of the system is reached.

This CKE process fits into the agile process model proposed by Baumeister [12] as presented on Figure 4.2 (see Figure 2.4 for original model). The author of the model proposed that System Metaphor step should be done once, at the beginning of the project. Here, this step is incorporated into the whole cycle to regularly discuss the use cases and scenarios and to match them to the current needs. Also, planning is explicitly introduced into the proposed methodology to reflect an important step in the agile process. During this step, specific tasks are created, bearing in mind the whole system idea expressed through the motivating scenarios and competency questions. Next, tasks are estimated in a planning game and finally a set of tasks is selected for the current iteration and made available for all at some agile board (for the more comprehensive overview of the agile methodology see Section 2.3.3).

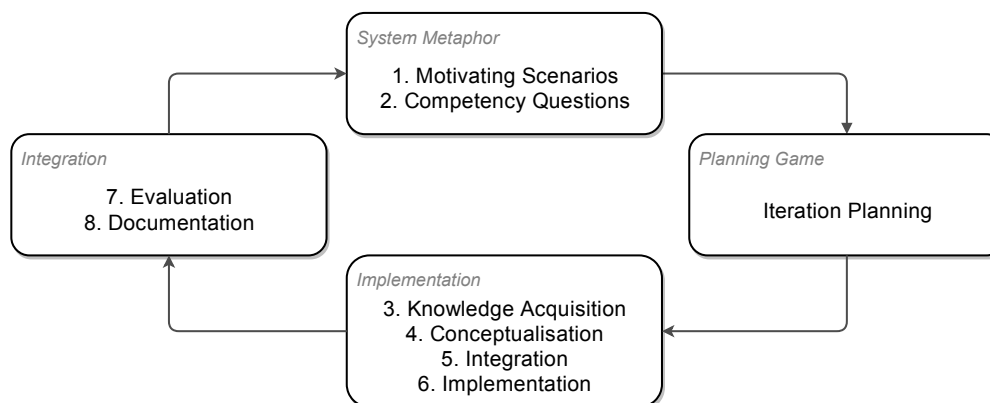


Figure 4.2: CKE agile process compared to the agile process model proposed by Baumeister [12]

Presented iterative process may be conducted in a top-down, bottom-up or middle-out manner [161] (see Requirement R4; page 16), depending on current group needs. It is suggested, however, that during the initial iteration, basic vocabulary should be established as a reference point for further work (see Requirement R3; page 16). It should be treated not only as an engineering task, but also as a cognitive task [78]. For good co-operation within the CKE group, it is required that everyone “speak” the same language. In subsequent iterations, it is proposed to create knowledge from less to more formal. This will allow, according to the agile way, to create rapid prototypes that can be evaluated on an ongoing basis, albeit informally. Such prototypes can then be further refined using more and more formal representation. Progressive formalization should be tailored to the needs and capabilities of the system and its users. To illustrate this, two examples will be

given. In Prosecco project³, the first iteration was made on natural language level, using unstructured texts to represent as much expert knowledge as possible. Then, these notes were conceptualized according to the established structured language during following iterations. Finally, in the third phase, formal ontology in OWL 2 language was developed [102]. Another example is the workflow that can be realized in KnowWE. The first iteration of the project may focus on adding text and images to the wiki. Then, in subsequent iterations, all concepts (wiki pages) are supplemented by more and more complex semantic annotations. Ultimately, knowledge is extended by, partially generated automatically, set-covering models and production rules [16].

Agile methodology is also associated with specific roles of process participants. The following can be distinguished⁴:

- the product owner, who has the whole system vision,
- the CKE process master (scrum master), who oversees the project course,
- the team of 3-9 people.

It should consist of domain experts, preferably with short training in KE techniques, and 1-2 knowledge engineers, who are regular project participants and support the domain experts, who are knowledge owners (see Requirement R10; page 16). The opposite idea of domain experts and knowledge engineers separation was tested in one of experiments (see Section 8.1.1), but it resulted in a rather chaotic process, where knowledge engineers on their own tried to establish the core concepts in domain knowledge. More promising results were achieved in the setting where one knowledge engineer was fully incorporated into the project and available for all experts' requests and problems (see Sections 8.1.2-8.1.3). All experiments conducted by the author (see Section 8.1) suggest that within the team one can also distinguish some roles. Among the most prominent ones, the following three should be mentioned:

- the adders, who create a lot of material, but not always well semantically marked,
- the synthesizers, who take care about semantics and concepts interrelations,
- “the cops”, who are responsible for imposing standards and schemes and then they are monitoring them.

It should be noted that these results are consistent with the analyses presented in [97]. The first two roles discovered in experiments are exactly the same as these presented in the literature.

4.5 European Wiki Case Study

Sample *European Wiki* project is introduced here to present the proposed methods and tools in action. This project is conducted accordingly to the proposed *CKE agile process*. It is developed by a group of seven

³The author was one of the developers of the ontology within this project.

⁴This set of roles is based on scrum methodology, described in Section 2.3.3.

colleagues that will work online to develop the wiki with description of all European countries. This is a non-commercial project that will be used by them as a supporting system during their travels across the continent. As there is no “client” in this setting, one person from the group has been selected as a product owner, who will watch over the main purpose of the project. Also, one of them was selected as a CKE process master that 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. Due to the group’s basic knowledge of the KE, no additional knowledge engineer is needed in this project. The whole *CKE agile process* is based on 9 steps grouped into 4 blocks (see Figure 4.2), done in as many iterations as needed to achieve the final version of the KB. To simplify the description, only the first iteration will be presented here, along with the KB made 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* module (see Section 5.1).
3. When such system specification is described, the *iteration planning* step takes place. During this phase, specific tasks are defined, discussed, estimated and selected for current sprint. 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 block begins. It consists of several steps: *knowledge acquisition*, *conceptualisation*, *integration* and *implementation*, which are seamlessly combined. During this block, two users made six changes to the *European Wiki* project (see Sections 5.2, 6.1) and discussed the difficult points (see Section 5.3). In the background, the semantic changelog was created (see Section 6.2) and parsed to provide a gamification-based incentives for users (see Section 7.1). All task-related decisions were marked on an iteration board (a scrum board), i.e. user *kkutt* assigned task (a) to himself and moved it to the “in progress” section, and after completing the task he moved it to the “ready for evaluation” section, etc.
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 (see Section 5.1).
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 (see Section 5.3 for description of a discussion made by users). A note

appeared on the `special:category:city` page, used by Loki as a description of `city` category.

4.6 Summary

This chapter provided a brief outline of the proposed set of methods and tools prepared as a response to the CKE support requirements (see Section 4.1). The core is the *CKE agile process* described in Section 4.4. It is supplemented by a set of methods for three CKE support fields of interest. Their detailed description will be the matter of next chapters: *quality management* in Chapter 5, *change management* in Chapter 6 and *user involvement* in Chapter 7. Description of every method, technique and tool within these chapters consists of three parts. Firstly, the overall idea and specification are discussed for a general CKE tool, i.e. the one that allows to create the knowledge grounded into RDF syntax (see dissertation assumptions in Section 4.2). Such KB is built up from “information units” that depend on the specific tool to which the methods will be applied: concepts in ontology-oriented tools, wiki pages in semantic wikis, and so on. Then, prototypical implementation for Loki is presented, including semantic wikis and especially Loki-specific aspects. Finally, to show the framework in action, each module is presented from the sample *European Wiki* project point of view.

Chapter 5

Quality Management

Three methods are proposed to allow for a rather comprehensive management of quality. The most coarse quality evaluation may be done by the use of *reasoning unit tests*. They clearly indicate whether the quality requirement is fulfilled or not (see Section 5.1). For a more accurate automatic quality assurance of the knowledge bases, extraction of more sophisticated system characteristics in the form of *quality metrics* may be used (see Section 5.2). Finally, aspects that cannot be checked with automatic systems can be left to the subjective evaluation of the system users (see Section 5.3). On one hand, users are able to capture various problems with the KB. On the other hand, knowledge creation is a social process, based on reconciling a common point of view from different perspectives and experiences, and as such it needs a place for social discourse between users (see collaborative knowledge building in Section 1.4).

5.1 Reasoning Unit Tests

Idea and Specification

Reasoning unit tests is an idea of adapting unit tests from SE into the KE [164]. The goal of unit tests is to check the elementary unit of the system. In the SE, they execute the appropriate function with the given input and check whether it generates the expected output or action. On the other hand, their purpose in KE may be to verify whether the KB contains the data we want, and make sure that the data takes an expected form. These tests can, and should, be created by system users (i.e. domain experts) to reflect the real system expectations. This allows the unit tests to check how well the system actually performs its task. Like unit tests in SE, these tests should be performed after each change as a “first instance” that validates the knowledge base and should be the first signal problems [15]. In the case of larger tests¹, it is possible to postpone their startup for the time when the system is less busy, or at least run them only on demand.

Semantic knowledge bases, as described earlier (see Section 3.2), can be queried using the SPARQL

¹Which, however, are no longer unit tests, as they test more than a small portion of the KB.

query language. It is therefore a natural candidate for unit testing. The general idea of such tests consists of several steps:

1. The user specifies the full SPARQL-compatible query.
2. The user specifies the expected query result using available types of assertions.
3. The test is set as active.
4. After saving changes to the KB, system checks tests' statuses and fires the active ones: executes the SPARQL query and verifies the assertions.
5. Results are then saved for further investigation.

The most crucial is the second step, where the user is provided with a way to determine what are the expected values. The SPARQL language has five types of queries: SELECT, CONSTRUCT, ASK, DESCRIBE and UPDATE (see Section 3.2). The CONSTRUCT query, from the point of view of unit tests, is mainly the same idea as SELECT query, because it also selects data by given constraints. The difference is that it returns RDF triples instead of table results. The UPDATE query modifies the database state, which is not useful for checking the current state. Due to lack of exact specification, the DESCRIBE query may be implemented differently, so the generic test specification will be omitted². In summary, our attention will be focused on SELECT and ASK queries. For these two types of queries, the set of possible assertions is proposed in Tables 5.1-5.2. *Attribute?* column indicates whether the assertion uses the attribute, i.e. any variable from the ones returned by the query.

Table 5.1: Set of assertions for SELECT queries

| Quantifier | Attribute? | Assertion | Description |
|------------|------------|------------------|--|
| Any Row | ✓ | ≤, <, >, ≥, =, ≠ | Checks whether any row / all rows / no rows contain the given value of specified attribute |
| All Rows | ✓ | | |
| None Row | ✓ | | |
| Row Count | – | | Checks the number of rows returned |

Table 5.2: Set of assertions for ASK queries

| Assertion | Description |
|--------------|--|
| Assert True | Checks whether the query returns true value |
| Assert False | Checks whether the query returns false value |

The presented assertions are the main part of the reasoning unit tests module that should meet the following requirements:

- Functional requirements:
 1. Possibility to define, modify and remove test cases.
 2. Ability to check the result of the previously executed test in a readable and easy form.

²Due to the possibility of defining DESCRIBE queries in Loki, the implementation of these tests for this specific platform will be described later in the "Implementation in Loki" part of this section.

3. Hierarchical structure support and selective tests execution (described below).
 4. Presentation of test statistics: last run, number of tests omitted, done correctly and incorrectly.
- Non-functional requirements:
 1. Taking care of the whole system assumptions, e.g. when tests are added to the wiki system, they should meet “the wiki way” principles (see Section 3.1).
 2. Reduction of configuration options to facilitate testing.

The KB unit tests module is already implemented in KnowWE [15]. Unfortunately, all tests are thrown into a one “big bag”, making it difficult to use with greater number of tests. In this dissertation 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 execute the 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 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 on Figure 5.1. White nodes represent levels in the hierarchy.

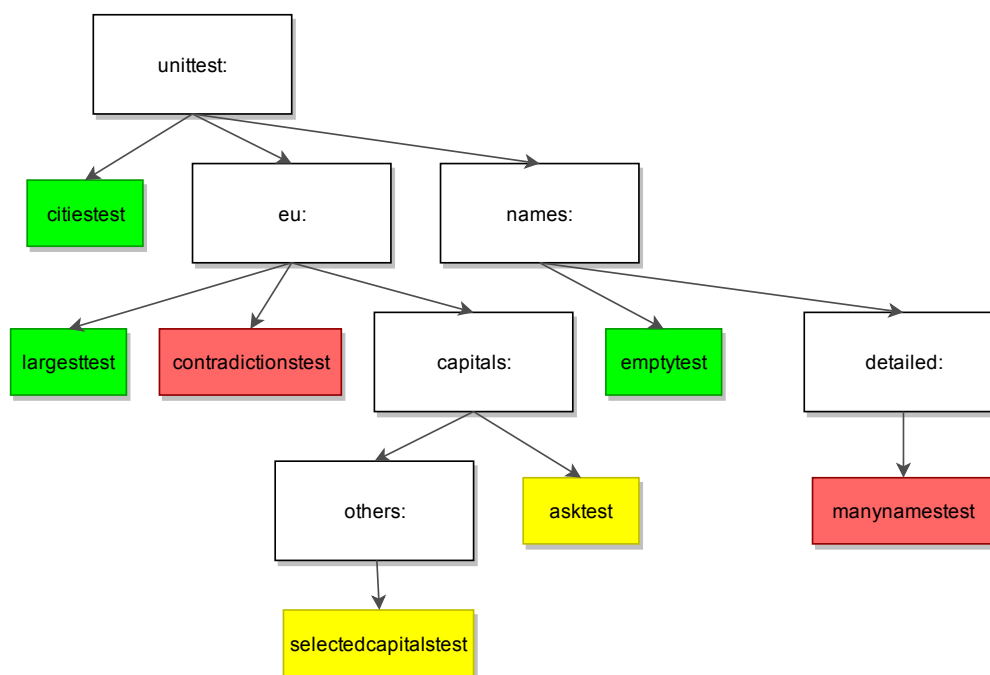


Figure 5.1: Sample test structure for *European Wiki* project

Original Contributions

Idea of unit tests for ontologies was not developed by the author. It was firstly proposed by Denny Vrandečić and Aldo Gangemi [164]. The original contribution of the author is the idea of specification of unit tests in SPARQL to provide a simple way of testing the KB, as users already know this language from regular work. What is more, author proposes to store tests in the hierarchical structure that allows their selective execution.

Implementation of the *reasoning unit tests* module for Loki was also done by the author.

Implementation in Loki

The *reasoning unit tests* module for Loki implements the following procedure³:

1. User specifies the full SPARQL-compatible query on the appropriate page in the wiki.
2. User also specifies the expected query result.
3. The test is set as active.
4. When anyone changes any regular page in the wiki, after saving changes, the wiki system runs all unit tests one by one and checks if the returned result is as expected.
5. Results are saved for further investigation by a user.

The whole module is grounded in the file- and directory-based system of Loki: (a) each page is represented as a text file, (b) each namespace is a directory on a server. Based on this, the architecture was proposed:

- The main node in the test hierarchy (see Figure 5.1) is represented by the `unittest` namespace.
- Each test is represented as a separate wiki page that consists of exactly one SPARQL query and a set of assertions. Test without assertions always returns success.
- Tests inside a nested namespace are executed only if all parent space tests have succeeded.
- Test results are presented on a corresponding page in a `unittestresults` namespace, e.g. for a test saved on a `unittest:examples:test1` page, the results are available on a `unittestresults:examples:test1` page.
- Tests summary is generated as a table with a list of tests and their current status on a `unittestsoverview` page.

A sample tests structure from the Figure 5.1 may be represented by the following set of test pages:

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

As it was mentioned, each test page is composed of a SPARQL query and a set of assertions. Any SELECT, ASK or DESCRIBE query valid in Loki is acceptable as a test. Sample query taken from *European Wiki* project is presented on Listing 5.1. A set of possible assertions is available for each of the query types

³Module is a part of Loki plugin available at <http://loki.ia.agh.edu.pl/>

```

1 <pl format="sparql">
2 PREFIX wiki: <>
3 SELECT ?page ?name
4 WHERE {
5   ?page a "city" .
6   ?page wiki:name ?name .
7   ?page wiki:largestSettlementOf wiki:organisation:eu .
8 }
9 </pl>

```

Listing 5.1: Sample SPARQL query that lists all cities marked as the largest settlement of EU

(see Tables 5.3-5.5). All of them are added to the test by putting a single line on the test page, accordingly to the scheme:

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

where: `type` is an assertion type (see Tables 5.3-5.5), `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. Two sample assertions for query presented on Listing 5.1 may look like these ones:

```

1 [[unittest_assert_anequal:?name:Paris|Is Paris one of the largests?]]
2 [[unittest_assert_noneequal:?name:Cracow|Cracow is not one of the largests!]]

```

Table 5.3: Set of assertions for SELECT queries within Loki

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

Table 5.4: Set of assertions for ASK queries within Loki

| type | parameter | Description |
|-------------|--------------------|--|
| asserttrue | Empty ⁴ | 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 [66]. Within Loki they take the form similar to the SELECT queries (see Listing 5.2), but instead of listing the selected pages (`?country` in the example), they return all attributes that characterize the selected pages, i.e. they return all triples where given page is a subject. This characteristic gives the possibility to provide additional type of assertion (`attributecount`) for checking the number of occurrences of given attribute, e.g. to

⁴One can simply omit the parameter: `[[unittest_assert_asserttrue:?!Checks for True]]`.

⁵XX should be replaced with one of the assertions types: `lessequal`, `less`, `greater`, `greaterequal`, `equal`, `notequal`.

Table 5.5: Set of assertions for DESCRIBE queries within Loki

| type | parameter | Description |
|-------------------------------|---|---|
| anyequal | Attribute name | Checks whether any page / all pages / no pages contain the given value of specified attribute |
| allequal | | |
| noneequal | | |
| attributecountXX ⁵ | | |
| pagecount | lessequal, less, greater, greaterequal, equal, notequal | Checks the number of pages returned |

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 requested value is available and do not bother with any other values.

```

1 <pl format="sparql">
2 PREFIX wiki: <>
3 DESCRIBE ?page
4 WHERE {
5   ?page a "city" .
6   ?page wiki:name ?name .
7   ?page wiki:largestSettlementOf wiki:organisation:eu .
8 }
9 </pl>

```

Listing 5.2: Sample query that describes all cities marked as the largest settlement of EU

The *reasoning unit tests* module is implemented as a separate PHP file with a separate class `loki_utl_test` that fits into other Loki classes model (see Figure 5.2). Thanks to the ability to register hooks for events in Loki (DokuWiki) controller, it is possible to trigger the unit tests function every time when the page is saved. The existing `loki_utl_sparql` plugin is used to execute the necessary SPARQL queries. Full interaction of module with the system is described by the following steps:

1. Module collects all test information from the corresponding directory.
2. Tests in the given directory are executed.
3. Results for each test are collected in the `/lib/plugins/loki/unittestresults/{test_page_name}/{yyyymmddHHMMSS}.txt` file.
4. If all tests in the directory succeed, all subdirectories are opened and for each of them steps 1-4 are repeated.
5. Tests summary on the `unittestsoverview` page is updated. If the graph changelog plugin is available, tests' statistics are also saved in it (see Section 6.2).

To be compatible with “the wiki way” of knowledge creation, especially with the *tolerant* principle (see Section 3.1; page 30), it should be possible to save a wiki page even if it contains errors. Two most potential errors were taken into consideration:

- if a user specifies an assertion that does not exist or uses the parameters that are not valid, the test is

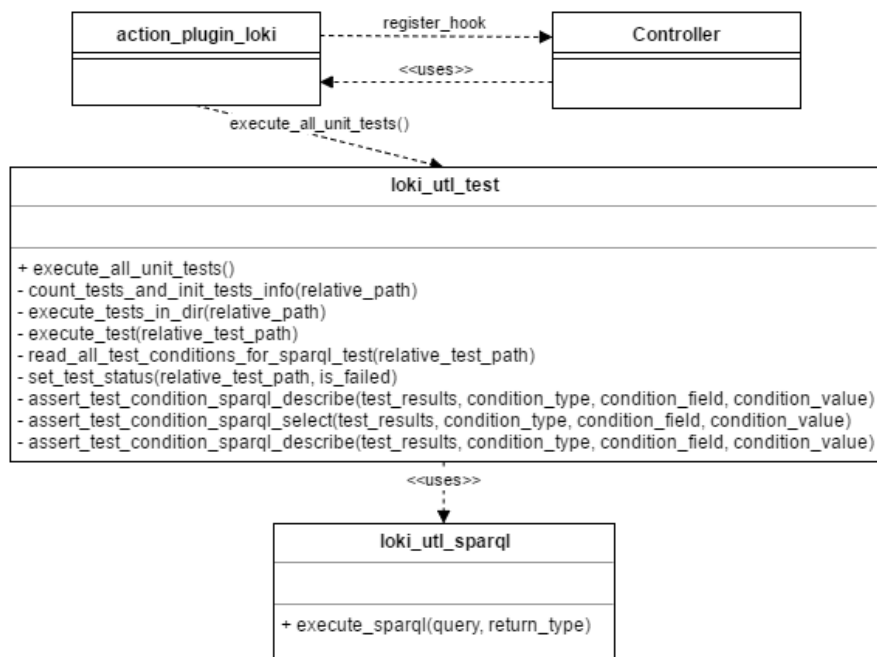


Figure 5.2: Reasoning unit tests module for Loki. Classes diagram

considered as a failure,

- if the test does not contain any assertions, it is considered as a success.

European Wiki

Within the *European Wiki* project (see Section 4.5), seven tests were prepared. Definition of each of them consists of a SPARQL query and a set of assertions (see Listings 5.3-5.9). As the *European Wiki* project already started and there is no knowledge within the wiki yet, the first tests failed. Thanks to the tree structure, deeper tests were not executed, which is summarized on a dedicated page (see Figure 5.3).

Project then goes through the *implementation* phase and some changes are made in the wiki (see Sec-

Reasoning unit tests results:

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

Figure 5.3: The tests summary at the very beginning of the project

tion 5.2 for changes description and Listings 5.13-5.15 for the final KB). Tests are executed after each change. Finally, *implementation* phase is closed and there is a time for *evaluation*. During this step, results saved within the wiki (see Figures 5.4-5.8) and summarized on a dedicated page (see Figure 5.9) are consulted. Tree structure of the tests set with an indication whether the test was passed, failed or not executed is presented on Figure 5.1. Used set consists of the following tests:

1. `citytest` (see Listing 5.3) is a simple ASK query that checks whether there are any cities in the wiki. If not, the test will fail and more specific tests placed deeper in the hierarchy will not be executed. In the example, there are three cities in the KB, so the test is passed (see Figure 5.4).

```

1 ===== Cities test =====
2 <pl format="sparql">
3 PREFIX wiki: <>
4 ASK {
5   ?page a "city".
6 }
7 </pl>
8
9 [[unittest_assert_asserttrue:?!Are there any cities?]]

```

Listing 5.3: The `unittest:citytest` test

2. `contradictiontest` (see Listing 5.4) is a test with two contradictory assertions. As a result, one can see that if there are failed and passed assertions, the whole test fails, which is an expected behaviour (see Figure 5.5).

```

1 ===== Contradictions test =====
2 <pl format="sparql">
3 PREFIX wiki: <>
4 SELECT ?page
5 WHERE {
6   ?page a "city" .
7 }
8 </pl>
9
10 [[unittest_assert_rowcount:?greater:3|More than 3 pages?]]\ \
11 [[unittest_assert_rowcount:?equal:3|Exactly 3 pages?]]

```

Listing 5.4: The `unittest:eu:contradictiontest` test

3. `largesttest` (see Section 5.5) presents the `any/noneequal` assertions for SELECT query. It also shows that two passed assertions result in a passed test (see Figure 5.6).

```

1 ===== Largest test =====
2 <pl format="sparql">
3 PREFIX wiki: <>

```

```

4 SELECT ?page ?name
5 WHERE {
6   ?page a "city" .
7   ?page wiki:name ?name .
8   ?page wiki:largestSettlementOf wiki:organisation:eu .
9 }
10 </pl>
11
12 [[unittest_assert_anequal:?name:Paris|Is Paris one of the largests?]]\
13 [[unittest_assert_noneequal:?name:Cracow|Cracow is not one of the largests!]]

```

Listing 5.5: The `unittest:eu:largeststest` test

4. `asktest` (see Listing 5.6) and `selectedcapitalstest` (see Listing 5.7) are not executed as they are deeper in the hierarchy than the failed `contradictionstest` test. As they were not executed, there are no results.

```

1 ===== Ask test =====
2 <pl format="sparql">
3 PREFIX wiki: <>
4 ASK {
5   ?page a "city" .
6   ?page wiki:capitalOf ?country .
7   ?country a "country" .
8 }
9 </pl>
10
11 [[unittest_assert_asserttrue:?!Are there any cities that are capitals?]]

```

Listing 5.6: The `unittest:eu:capitals:asktest` test

```

1 ===== Capitals names test =====
2 <pl format="sparql">
3 PREFIX wiki: <>
4 DESCRIBE ?page {
5   ?page a "city" .
6   ?page wiki:capitalOf ?country .
7 }
8 </pl>
9
10 [[unittest_assert_attributecountgreaterequal:?name:1|Each capital has at least
    one name?]]

```

Listing 5.7: The `unittest:eu:capitals:others:selectedcapitalstest` test

5. `emptytest` (see Listing 5.8) does not have any assertions and is marked as passed accordingly to

Test results

Query result

yes

Test: asserttrue; Result: PASSED

Figure 5.4: The results of the `unittest:citiestest` test

Test results

Query result

| page |
|--------|
| cracow |
| london |
| paris |

Test: rowcount; Field greater; Value: 3 Result: FAILED

Test: rowcount; Field equal; Value: 3 Result: PASSED

Figure 5.5: The results of the `unittest:eu:contradictionstest` test

Test results

Query result

| page | name |
|--------|-----------|
| london | Londinium |
| london | London |
| paris | Paris |

Test: anyequal; Field name; Value: Paris Result: PASSED

Test: noneequal; Field name; Value: Cracow Result: PASSED

Figure 5.6: The results of the `unittest:eu:largesttest` test

the specification (see Figure 5.7). It is also worth to note that failed `contradictionstest` does not affect this test, placed in a different branch of the hierarchy.

```

1 ===== Empty test (no assertions) =====
2 <pl format="sparql">
3 PREFIX wiki: <>
4 ASK {
5   ?page a "city".
6 }
7 </pl>

```

Listing 5.8: The `unittest:names:emptytest` test

6. `manynametest` (see Listing 5.9) has an assertion with a typo. As it is not recognized, the assertion is by default considered as a failure. Also, in this test one can observe that two failures lead to the failed test (see Figure 5.8).

```

1 ===== Many names test =====
2 <pl format="sparql">
3 PREFIX wiki: <>
4 SELECT ?page
5 WHERE {
6   ?page a "city" .
7   ?page wiki:name ?name1 .
8   ?page wiki:name ?name2 .
9   FILTER(?name1 != ?name2)
10 }
11 </pl>
12
13 [[unittest_assert_aneqwal:?name1:Cracow|Cracow has many names? (assertion with a
14   typo)]]\
15 [[unittest_assert_rowcount:?less:2|Less than 2 cities with many names?]]

```

Listing 5.9: The `unittest:names:detailed:manynametest` test

Based on presented example, it can be stated that *reasoning unit tests* module fulfills its requirements. It provides a simple way to define, modify and remove test cases based on regular wiki pages edition mechanism. The framework takes care of “the wiki way” principles and allows to save a broken test, i.e. a test without assertions or with errors. Results of tests are placed as separate clearly formatted wiki pages, easily accessible by users. And last, but not least, to facilitate testing, there are no configuration options, as they are not needed. Only on/off switch for the whole module is available.

Test results

Query result

yes

Figure 5.7: The results of the `unittest:names:emptytest` test

Test results

Query result

| page |
|--------|
| cracow |
| cracow |
| london |
| london |

Test: anyeqwal; Field name1; Value: Cracow Result: ERROR: Assertion type unknown!

Test: rowcount; Field less; Value: 2 Result: FAILED

Figure 5.8: The results of the `unittest:names:detailed:manynametest` test

Reasoning unit tests results:

| Test | Status |
|--|--------------|
| <code>citytest</code> | PASSED |
| <code>eu:capitals:asktest</code> | NOT EXECUTED |
| <code>eu:capitals:others:selectedcapitalstest</code> | NOT EXECUTED |
| <code>eu:contradictionstest</code> | FAILED |
| <code>eu:largesttest</code> | PASSED |
| <code>names:detailed:manynametest</code> | FAILED |
| <code>names:emptytest</code> | PASSED |

Figure 5.9: The summary of all tests used for validation

5.2 Metrics of Changes

Idea and Specification

The more the knowledge base is expanded, the more data it contains and the more difficult it becomes to manage its quality. One method that can help with this issue are reasoning unit tests described previously. They can be supplemented by metrics of changes, which are different ontological metrics (see Table 2.3) calculated after each change and then compared with the previous value. The difference between them shows the actual contribution of a given change to the quality of the KB.

As there are many metrics, comparison to evaluate which changes are better than others and which are the worst may be difficult. *Weighted average* metric is proposed to overcome this problem. It is a single value that combines many other metrics to provide a simple way of representing quality change. *Weighted average* is based on results of many metrics, i.e. on the set of different values with various ranges. Sometimes they represent percentage scores, another time they include negative values, etc., so there is a need to normalize them. The simplified scale of 1-5⁶ is proposed [42], as it is rather intuitive for all. The normalization could be done by providing 5 equal ranges or by more sophisticated function that matches the meaning of represented values. After such standardization, there is the possibility to compare or combine different metrics. Due to the fact that in the large KB the change in metric values is not relatively high (e.g. manipulation of 5 edges in a KB build of 5000 edges is a small difference), normalization at the metric value level can often cause the value before and after the change to be reduced to the same grade. As a result, the difference between them will often be 0, making this a useless approach. A more practical and interesting approach is to first calculate the difference in the metric values and then normalize it to scale 1-5. Example of such an approach prepared for a developed prototype is presented in Table 5.6.

There is no single metric that is useful in every situation. The selection of proper weighted average depends on three factors:

- *System capabilities*, especially knowledge representation and hardware performance. The former sets restriction on possible quality metrics, e.g. there is no possibility to calculate specific metric if current representation does not support it. The attention should be also paid to the fact that calculation of metrics requires resources and can slow down the system.
- *Use case*. Each of the cases is different and requires attention to specific characteristics.
- *Project phase*. The whole *CKE agile process* consists of many phases. As it was previously indicated (see Section 4.4), different tasks are done in the first iterations than in the last ones. This difference may be reflected in different metrics.

⁶In fact, it can be any scale. To correctly calculate the weighted average, it is important to normalize each of the metrics to the same scale.

As a part of this dissertation, use cases presented in Section 2.1.1 were analyzed. An attempt to group them into collections with similar properties was made, but unfortunately the cases were selected to show the variety of CKE problems and thus the analysis did not lead to the distinction of any groups. Such use case-specific customization should be done by experienced knowledge engineers as a part of the system preparation and maintenance. The phases of projects were also investigated. Based on author's literature research and conducted experiments (see Section 8.1), two main metrics are proposed:

1. the first one to be used when a KB is built from scratch. This metric promotes the largest base growth in terms of number of classes, instances and relations to provide a lot of material in the KB.
2. the second metric is used when existing KB has enough data, and the emphasis is put on the KB coherence. This metric promotes new instances addition and the limitation of the number of classes and relationships. This is based on the assumption that during the first phase some classes were redundant or obsolete, so to prepare the proper KB they should be cleared.

Due to the lack of proper CKE models, and, as a result, due to the lack of proper phases identification, more general phase-specific metrics are the object of future research.

Original Contributions

Metrics of ontologies that were presented were not authored as a part of this dissertation. They have been used to propose the idea of calculation of the *metrics of change*. The original contribution is the idea of calculation of *only one* value for each change, based on available metrics, to reflect how good or bad this particular change was for the whole KB evolution. Implementation of the *metrics of change* functionality for Loki was also done by the author.

Implementation in Loki

Four metrics from the whole review presented in Table 2.3 (see page 19) were implemented in Loki. They were arbitrarily selected as good quality indicators for *European Wiki* project example, as well as for wikis developed with students during experiments described in Section 8.1. They are presented together with the intuition behind the justification for their choice.

Attribute Richness is defined as an average number of attributes *att* per class *C* [155]:

$$AR = \frac{|att|}{|C|} \quad (5.1)$$

Intuition: ontologies with higher *AR* values have more classes with many attributes, which results in a richer knowledge base.

Average Population depicts the ratio of instances I to classes C [155]:

$$AP = \frac{|I|}{|C|} \quad (5.2)$$

Intuition: the low AP value means that there are many classes that have only a few instances. In a general case, it means problems with the knowledge base project, e.g. probably the class hierarchy is too specific for the project.

Size Of Vocabulary counts the vocabulary used in the ontology and is defined as a sum of classes C , instances I and attributes att [182]:

$$SOV = |C| + |I| + |att| \quad (5.3)$$

Intuition: SOV simply depicts the size of a KB, which can be used e.g. to determine the current project phase. It should be noted that the interpretation of this metric is strongly project-specific. In some projects, $SOV = 500$ is reached at the very end (e.g. students' toy ontologies), while in others it is a negligible value.

Edge Node Ratio is a ratio of all edges E in the KB (the number of all triples) to all nodes N (subjects and objects, both named and literals) [182]:

$$ENR = \frac{|E|}{|N|} \quad (5.4)$$

Intuition: ENR evaluates the density of connections in the KB. The higher the indicator is, the more dense the KB is, i.e. instances have more attributes and there are more interconnections between different concepts in the KB.

Implementation of presented metrics in Loki is saved as a separate PHP file `metrics.php`, which contains the code needed for evaluation of metrics⁷. This code is placed within the *semantic changelog* module (see Section 6.2) and triggered by the code responsible for creating changelog. Metrics code heavily uses the functions provided by Loki core, e.g. `list_attributes()`. Technical details of metrics implementation are provided below.

- *Attribute Richness* metric is based on two Loki core functions that return the list of all categories and the list of all attributes. The number of list items can be easily counted and divided to provide the final metric value, as presented on Listing A.1 on page 133.
- *Average Population* metric may be calculated thanks to Loki `list_category_uses(category)` function that lists all instances of given category.

⁷Metrics module is a part of prov plugin available at <http://loki.ia.agh.edu.pl/>

When combined with the categories listing function, it provides a way to calculate the requested value, as on Listing A.2 on page 133.

- *Size of Vocabulary* metric simply sums up the number of all attributes and all instances provided by Loki. See Listing A.3 on page 134.
- *Edge Node Ratio* is calculated as a ratio of all relations (edges) to all nodes, where nodes are all pages that have at least one triple and all literals. The current implementation iterates on all categories and then on all their attributes and relations (see Listing A.4 on page 134).

Table 5.6: Rules for normalization of differences in metric values to 1-5 scale

| Metric | 1 | 2 | 3 | 4 | 5 |
|------------|-------------------|----------------|---------------|--------------|-----------------|
| <i>AR</i> | $(-\infty; -0.4)$ | $[-0.4; -0.1)$ | $[-0.1; 0.1]$ | $(0.1; 0.4]$ | $(0.4; \infty)$ |
| <i>AP</i> | $(-\infty; -0.4)$ | $[-0.4; -0.1)$ | $[-0.1; 0.1]$ | $(0.1; 0.4]$ | $(0.4; \infty)$ |
| <i>SOV</i> | $(-\infty; -4)$ | $[-4; -1)$ | $[-1; 1]$ | $(1; 4]$ | $(4; \infty)$ |
| <i>ENR</i> | $(-\infty; -0.4)$ | $[-0.4; -0.1)$ | $[-0.1; 0.1]$ | $(0.1; 0.4]$ | $(0.4; \infty)$ |

Calculated metric values before and after each change are then compared, and the resulting difference is normalized to the 1-5 scale accordingly to the rules presented in Table 5.6. Based on these basic metrics, two proposed weighted average metrics may be calculated:

$$\begin{aligned}
 WA_1 &= \frac{AR + AP + 5 \cdot SOV + ENR}{8} \\
 WA_2 &= \frac{5 \cdot AR + 5 \cdot AP + SOV + 3 \cdot ENR}{14}
 \end{aligned} \tag{5.5}$$

The first one that promotes fast KB expansion should be based primarily on *Size of Vocabulary*, which directly measures the database growth. The second one that puts emphasis on KB coherence should combine primarily the values of *Attribute Richness*, *Average Population* and *Edge Node Ratio*. In this case, classes with more attributes and instances are promoted, as they are indicators of richer KB.

European Wiki

The *European Wiki* project starts with an empty KB (KB_0 state). Three pages are then created:

- in change Ch_1 city:london page is added (see Listing 5.10),
- in change Ch_2 city:paris page is created (see Listing 5.11),
- finally in change Ch_3 city:cracow page is added (see Listing 5.12).

```

1 ===== London =====
2
3 [[name:=London]] is the capital and most populous [[category:city|city]] of [[capitalOf::country:
4   england|England]] and the [[capitalOf::country:uk|United Kingdom]].
5
6 Standing on the River Thames in the south east of the island of Great Britain, London has been a
7   major settlement for two millennia. It was founded by the Romans, who named it [[name:=

```

```
Londinium]].
```

```
6
7 London has a diverse range of people and cultures, and more than 300 languages are spoken in the
region. Its estimated mid-2016 municipal population (corresponding to Greater London) was [[
population:=8787892|8,787,892]], the largest of any city in the [[largestSettlementOf::
organisation:eu|European Union]], and accounting for 13.4% of the UK population. London's
urban area is the second most populous in the EU, after Paris, with 9,787,426 inhabitants at
the 2011 census. The city's metropolitan area is the most populous in the EU with 13,879,757
inhabitants, while the Greater London Authority states the population of the city-region (
covering a large part of the south east) as 22.7 million. London was the world's most
populous city from around 1831 to 1925.
```

Listing 5.10: The city:london page created during Ch_1 change

```
1 ===== Paris =====
2
3 [[name:=Paris]] is the capital and most populous [[category:City|city]] of [[capitalOf::country:
france|France]], with an administrative-limits area of 105 square kilometres (41 square miles
) and a 2015 population of [[population:=2229621|2,229,621]]. The city is a commune and
department, and the capital-heart of the 12,012-square-kilometre (4,638-square-mile) Ile-de-
France region (colloquially known as the 'Paris Region'), whose 12,142,802 2016 population
represents roughly 18 percent of the population of France. By the 17th century, Paris had
become one of Europe's major centres of finance, commerce, fashion, science, and the arts, a
position that it retains still today.
4
5 [[largestSettlementOf::organisation:eu| ]]
6 [[category:city| ]]
7 [[category:town| ]]
```

Listing 5.11: The city:paris page created during Ch_2 change

```
1 ===== Cracow =====
2
3 [[name:=Krakow]], also [[name:=Cracow]], is the second largest and one of the oldest [[category:
city|cities]] in Poland. Situated on the Vistula River in the Lesser Poland region, the city
dates back to the 7th century. Krakow has traditionally been one of the leading centres of
Polish academic, cultural, and artistic life and is one of Poland's most important economic
hubs.
4
5 The city has grown from a Stone Age settlement to Poland's second most important city. It began
as a hamlet on Wawel Hill and was already being reported as a busy trading centre of Slavonic
Europe in 965. With the establishment of new universities and cultural venues at the
emergence of the Second Polish Republic in 1918 and throughout the 20th century, Krakow
reaffirmed its role as a major national academic and artistic centre. The city has a
population of approximately [[population:=760000|760,000]], with approximately 8 million
additional people living within a 100 km (62 mi) radius of its main square.
```

Listing 5.12: The city:cracow page created during Ch_3 change

These page creations aimed at fast KB growth are then followed by three more changes aimed at KB coherence:

- in change Ch_4 city:london page is extended by a definition of a new attribute onRiver and by

- a new relation `directFlightTo` that connects it with two other wiki pages (see Listing 5.13),
- then in change Ch_5 , users agree that the category for cities will be named `city`, so on `city:paris` two obsolete categories (`town` and `City`) are removed (see Listing 5.14),
 - finally, in change Ch_6 one short paragraph without semantic annotations was added to `city:cracow` page (see Listing 5.15).

```

1 ===== London =====
2
3 [[name:=London]] is the capital and most populous [[category:city|city]] of [[capitalOf::country:
4   england|England]] and the [[capitalOf::country:uk|United Kingdom]].
5
6 Standing on the River Thames[[onRiver:=Thames]] in the south east of the island of Great Britain,
7   London has been a major settlement for two millennia. It was founded by the Romans, who
8   named it [[name:=Londinium]].
9
10 London has a diverse range of people and cultures, and more than 300 languages are spoken in the
11   region. Its estimated mid-2016 municipal population (corresponding to Greater London) was [[
12   population:=8787892|8,787,892]], the largest of any city in the [[largestSettlementOf::
13   organisation:eu|European Union]], and accounting for 13.4% of the UK population. London's
14   urban area is the second most populous in the EU, after Paris, with 9,787,426 inhabitants at
15   the 2011 census. The city's metropolitan area is the most populous in the EU with 13,879,757
16   inhabitants, while the Greater London Authority states the population of the city-region (
17   covering a large part of the south east) as 22.7 million. London was the world's most
18   populous city from around 1831 to 1925.
19
20 London has the largest airport in Europe: London Heathrow. It connects the capital of the Great
21   Britain with 194 destinations in 82 countries. Among them there are: \[\[directFlightTo::city:
22   paris|Paris\]\] and \[\[directFlightTo::city:cracow|Cracow\]\].

```

Listing 5.13: The `city:london` page extended during Ch_4 change

```

1 ===== Paris =====
2
3 [[name:=Paris]] is the capital and most populous [[category:city|city]] of [[capitalOf::
4   country:france|France]], with an administrative-limits area of 105 square kilometres (41
5   square miles) and a 2015 population of [[population:=2229621|2,229,621]]. The city is a
6   commune and department, and the capital-heart of the 12,012-square-kilometre (4,638-square-
7   mile) Ile-de-France region (colloquially known as the 'Paris Region'), whose 12,142,802 2016
8   population represents roughly 18 percent of the population of France. By the 17th century,
9   Paris had become one of Europe's major centres of finance, commerce, fashion, science, and
10   the arts, a position that it retains still today.
11
12 [[largestSettlementOf::organisation:eu| ]]
13 [[category:city|city]]
14 [[category:town|town]]

```

Listing 5.14: The `city:paris` page fixed during Ch_5 change

```

1 ===== Cracow =====
2

```

3 `[[name:=Krakow]], also [[name:=Cracow]], is the second largest and one of the oldest [[category:`
`city|cities]] in Poland. Situated on the Vistula River in the Lesser Poland region, the city`
`dates back to the 7th century. Krakow has traditionally been one of the leading centres of`
`Polish academic, cultural, and artistic life and is one of Poland's most important economic`
`hubs.`

4

5 The city has grown from a Stone Age settlement to Poland's second most important city. It began
 as a hamlet on Wawel Hill and was already being reported as a busy trading centre of Slavonic
 Europe in 965. With the establishment of new universities and cultural venues at the
 emergence of the Second Polish Republic in 1918 and throughout the 20th century, Krakow
 reaffirmed its role as a major national academic and artistic centre. The city has a
 population of approximately `[[population:=760000|760,000]]`, with approximately 8 million
 additional people living within a 100 km (62 mi) radius of its main square.

6

7 [In 2000, Krakow was named European Capital of Culture. In 2013 Krakow was officially approved as a UNESCO City of Literature. The city hosted the World Youth Day in July 2016.](#)

Listing 5.15: The `city:cracow` page extended during Ch_6 change

For each of the seven KB states all available metrics were calculated. Based on them, metrics' values for changes were calculated as a simple subtraction. These differences were then normalized to 1-5 scale accordingly to the rules presented in Table 5.6, and then used to calculate Weighted Average metrics WA_1 and WA_2 (see Equation 5.5). All values are summarized in Table 5.7. Values in brackets represent values after normalization.

Table 5.7: Metrics values for subsequent states and changes in a sample KB

| KB state | Change | AR | AP | SOV | ENR | WA_1 | WA_2 |
|----------|--------|-----------|-----------|--------|----------|-------------|-------------|
| KB_0 | | 0.00 | 0.00 | 0 | 0.00 | – | – |
| | Ch_1 | 4.00 (5) | 1.00 (5) | 6 (5) | 0.88 (5) | 5.00 | 5.00 |
| KB_1 | | 4.00 | 1.00 | 6 | 0.88 | – | – |
| | Ch_2 | -2.67 (1) | -0.34 (2) | 3 (4) | 0.12 (4) | 3.38 | 2.21 |
| KB_2 | | 1.33 | 0.66 | 9 | 1.00 | – | – |
| | Ch_3 | 0.00 (3) | 0.34 (4) | 1 (3) | 0.00 (3) | 3.13 | 3.36 |
| KB_3 | | 1.33 | 1.00 | 10 | 1.00 | – | – |
| | Ch_4 | 0.67 (5) | 0.00 (3) | 2 (4) | 0.11 (4) | 4.00 | 4.00 |
| KB_4 | | 2.00 | 1.00 | 12 | 1.11 | – | – |
| | Ch_5 | 4.00 (5) | 2.00 (5) | -2 (2) | 0.01 (3) | 2.88 | 4.36 |
| KB_5 | | 6.00 | 3.00 | 10 | 1.12 | – | – |
| | Ch_6 | 0.00 (3) | 0.00 (3) | 0 (3) | 0.00 (3) | 3.00 | 3.00 |
| KB_6 | | 6.00 | 3.00 | 10 | 1.12 | – | – |

First three changes were aimed at quick KB expansion. It is properly reflected by the WA_1 scores ≥ 3.00 . Then, changes Ch_4 and Ch_5 were aimed at KB consistency, which was correctly identified by high WA_2 values. These two metrics also properly evaluate change Ch_2 that results in a database growth, but at the cost of consistency (introduction of new unnecessary concepts): good WA_1 score, but low WA_2 indicator and the Ch_5 , where unnecessary concepts were removed leading to more consistent KB, but with smaller

vocabulary. Finally, neutral change Ch_6 that does not affect semantic layer of a KB, results in a neutral note 3.0 in both metrics. Based on this example, it can be stated that introduction of *metrics of changes* module with the *weighted average* metric provides an easy way to assign each change a single value that describes how good or bad this change is in the context of current needs (stated by the proper definition of *WA* metric).

5.3 Opinions and Discussion

Idea and Specification

The process of knowledge base evaluation needs opinions of the experts who create and use such a system. It is proposed here that they should have two types of mechanisms for expressing their opinions: subjective assessment of changes and a place for discussion.

The first part is a numerical evaluation of the changes. If it requires only one click to choose the right value from the list, it is easy to do for every user, so it is more probable that they will be keen to do so. This is another numerical value (next to the previously mentioned unit tests and metric values) that determines the quality of knowledge. They can all be used in a system for evaluating user and knowledge credibility or for other purposes (see Sections 4.3 and 6.2). However, it must be implemented on the level of specific changes, since in order to evaluate users' credibility there is a need to get information about each contribution. A note given on the level of a concept does not allow to tell who is responsible: whether all the users who edited the concept up until now, or only the last one. The simplest of such numerical evaluation can be binary: $-1, 1$ (dislike and like often found in social networks), or triple-value if the ability to express neutral evaluation (0) will be introduced. However, this does not exhaust the problem complexity, because all changes are brought to the same level: good or bad. On the other hand, the introduction of 11 or more levels ($[-5, 5]$) is too varied, since, as the research indicates, the users are still using only a few selected values and the reliability of such scales is poor [127]. In this dissertation, introduction of the 5-level scale ($[-2, 2]$ or $[1, 5]$) is proposed, as it is simple and reliable. Other scales may be introduced, should there be any reasonable justification.

The numerical rating indicates the *quality* of the change, but it is not enough as a meaningful feedback. This should be described in a module that provides a possibility to enter comments with more specific feedback. Unlike numerical evaluation, these comments should be done per concept, not per change. Separation of comments per each change would make reading all of them very difficult. Also, the introduction of a single place for comments about each concept enable the discussion between experts leading to fine-tuning process of KB. Such an external place for discussion may reduce the number of conflicting changes in the database, which results in less destabilization (cf. oscillating knowledge mentioned in Section 2.3.1). Depending on the selected CKE system and the current application it is used for, it may be a comment page

that any user who has access to the concept can edit, or it can be handled using a specific protocol [124].

Original Contributions

Discussion of concepts is available in many tools, e.g. in Semantic MediaWiki. The original contribution is the mechanism of ratings for specific change. The *opinions and discussion* module for Loki is based on existing plugins for DokuWiki system. The author's work was limited to their proper configuration and combination (functionalities from two plugins were merged into one).

Implementation in Loki

In Loki, a separate wiki page is dedicated for each concept described in a system. Therefore, it was decided to introduce a discussion page for each regular wiki page. These pages are related to each other by the same name, but with `talk:` prefix in the discussion one, e.g. for regular page `sci:phd:chapter5` there is a page `talk:sci:phd:chapter5` for comments. Such a naming convention is consistent with the one that is used in MediaWiki. This additional page in MediaWiki gives users a place for comments. Within Loki it also has the second part: the list of all revisions with the possibility to evaluate them on a [1, 5] scale.

The prepared Loki module is a combination of existing DokuWiki modules:

*Discussion*⁸ – a plugin that allows discussions grouped in threads. They can be moderated by selected users.

*Talkpage*⁹ – a simple plugin that creates a link to `talk:name` page for each wiki name page.

*Rater*¹⁰ – a module that supports creating polls using [1, 5] stars scale or simple up-vote/down-vote buttons.

In addition, the plugin has many additional options such as IP address blocking or voting for non-logged users.

*Changes*¹¹ – a plugin that generates a list of recent page changes.

Talkpage and Discussion plugins are used as they were prepared by their developers. Selected functionalities of Rater and Changes have been merged into one plugin¹² to answer the need for evaluation of each revision. It lists all revisions (as in Changes plugin), and then for each of them it gives user the possibility to evaluate the revisions and see the current score (as in Rater plugin). All entered pieces of information are saved in plugin files. Obsolete code and customization possibilities were removed from both plugins to make it as simple as possible.

⁸See: <https://www.dokuwiki.org/plugin:discussion>.

⁹See: <https://www.dokuwiki.org/plugin:talkpage>.

¹⁰See: <https://www.dokuwiki.org/plugin:rater>.

¹¹See: <https://www.dokuwiki.org/plugin:changes>.

¹²*revisionsrater* plugin is available at <http://loki.ia.agh.edu.pl/>

Finally, to start the discussion one has to do the following steps:

- The regular wiki page is created in the CKE process.
- The discussion page can be accessed using the Talkpage option in the page's context menu (see Figure 5.10) or using the link created manually (using the macro `~~TALKPAGE~~`) anywhere on the page.
- On the discussion page, the code responsible for generation of voting mechanism and discussion should be specified. E.g. for page `sci:phd:chapter5`, the discussion page `talk:sci:phd:chapter5` is opened, and there the code for ratings `{revisionsrater>sci:phd:chapter5}` and for discussions `~~DISCUSSION~~` is placed.
- Finally, the discussion page is rendered as on Figure 5.11. On the top there is a possibility to evaluate all revisions, and below there is a discussion view with the possibility to add new comments.

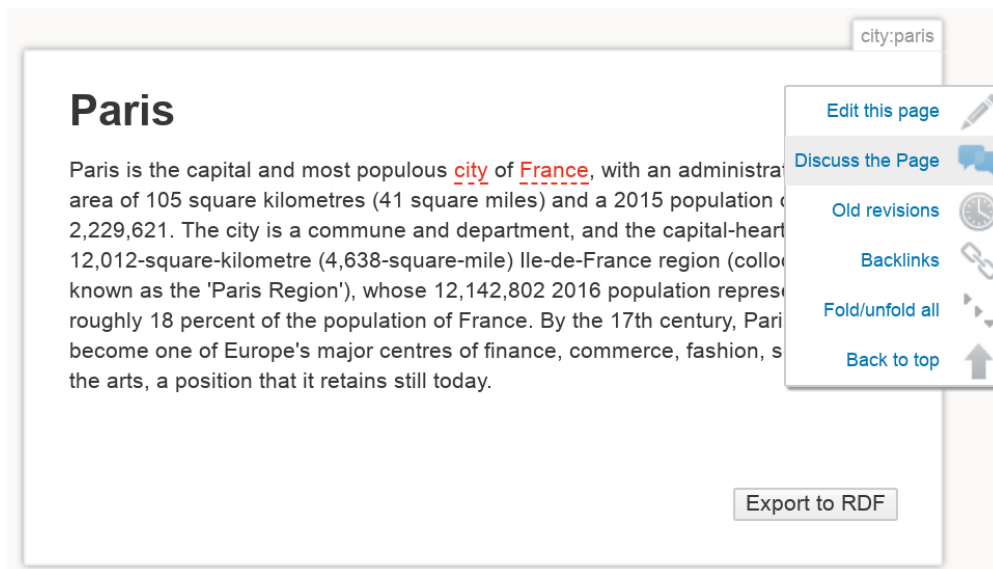


Figure 5.10: Discussion button in page menu

European Wiki

Let us focus on the second change made in the *European Wiki* project (see Section 5.2). Before this change, two tests were passed, and after it – three were passed (see Listing B.3). WA_1 metric for this change was calculated as 3.38. Automatic ways of evaluation indicate that this change is good. However, it is not, as user `kkutt` spotted. He opened the page `talk:city:paris` related to the `city:paris` page, gave the change note 3 of 5 and commented it. He noted that the prepared text is good, but there are some obsolete parts. Users then discussed on this topic (see Figure 5.11). Finally user `yoda` fixed the page, and this change was evaluated by `kkutt` as 5 of 5 (see Figure 5.12).

Based on this example, it can be stated that the proposed module fulfills the requirements described in

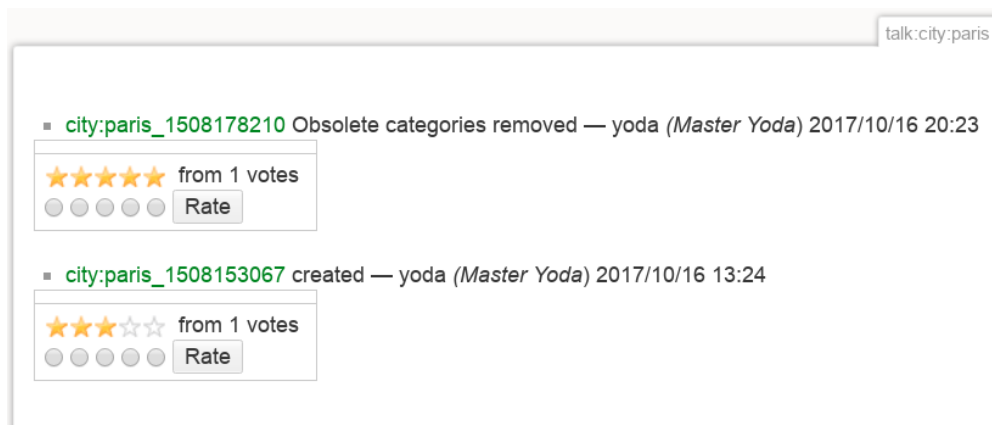


Figure 5.12: Evaluation form for two subsequent changes made to `city:paris` page

more sophisticated characteristics of current system state. Finally, the *opinions and discussion* mechanisms are introduced for all other issues that are not covered by automatic modules and have to be controlled by the users themselves.

Chapter 6

Change Management

The *CKE agile process* (see Section 4.4) should have a robust change management system to properly keep up with modifications that are taking place. In particular, various types of changes can be considered, because, as previously noted, different users may focus on different types of change, and therefore their expectations towards the system or its usability will vary (see Section 2.3.4). Changes and dependencies between their various types can be represented by ontology, as described in Section 6.1. Also version control system should be properly designed – not only to provide a log of all changes in the system, but also to enable the possibility to process it robustly. Such a graph-based VCS is proposed in Section 6.2.

6.1 Change Ontologies

Idea and Specification

To identify changes and their meaning properly, it is proposed to prepare an ontology that will depict the relations between them. Especially, it is proposed that two aspects of change should be taken into account: the factual change (What was done? e.g. typos or other small bugs fixed or new content added) and the goal (Why it was done? e.g. errors fixing or knowledge database expansion). It also should be noted that an ontology may be created for a specific task, e.g. one ontology for preparing conference papers and the other for developing system specification. In the following part of this chapter, a general high-level CKE ontology will be proposed. It was created in a process consisting of multiple iterations. Here, the three the most important milestones will be described.

The first version of the ontology was based on the analysis of literature and the analysis of the process of introducing changes in the KB. Specifically, two main sources of knowledge used were author's experience in preparing knowledge bases gained e.g. during ontology development in Prosecco project¹, and a list of types of contributions prepared by [97]. Based on them, the first ontology version in a form of a set of

¹See: <https://geist.re/pub/projects/prosecco:start>.

concepts was prepared:

1. Change:
 - Typos or other small bugs fixed,
 - New content added,
 - Content connected with other concepts,
 - Existing content expanded,
 - Refactoring,
 - Other (input from user).
2. Goal:
 - Errors fixing,
 - Knowledge base expansion,
 - Knowledge coherence improving,
 - Other (input from user).

This ontology was implemented and evaluated in an experiment (see Section 8.1.2). The results were used as an input during the preparation of second version of the ontology. Furthermore, the analysis of Git logs from repositories of two open source projects was taken into consideration as an attempt to look at the problem more broadly²:

*SlickGrid*³ is an advanced JavaScript library that allows to dynamically create arrays of large amounts of data. Messages from 677 available commits made in this project were analysed. Due to the repetitive nature of the messages, it was possible to isolate main groups of change. In the Table 6.1, examples of commit messages along with a description of actual changes made are described.

*Framework 7*⁴ is a HTML framework for creating hybrid mobile applications and web applications for iOS and Android. Messages from about 1900 commits were analysed. Examples are described in Table 6.1.

Finally, the second version of the ontology was extended by four options:

1. Change:
 - Content deleted.
2. Goal:
 - Improve code readability,
 - Appearance improvement,
 - Usage quality improvement.

²Analysed projects were from SE field, but they were used as an inspiration for CKE ontology enhancements. Projects were selected arbitrarily – they were used by the author in the time of analyses.

³See: <https://github.com/mleibman/SlickGrid>.

⁴See: <https://github.com/nolimits4web/Framework7>.

Table 6.1: Sample commit messages from analysed projects

| Project | Commit message | Description |
|-------------|--------------------------|--|
| SlickGrid | Code cleanup | Improved readability, comments removed, names of global variables corrected |
| | Missed a couple of typos | Variable names' typos fixed |
| | Appearance improving | Code indentation improved |
| Framework 7 | Typos | Variable names and punctuation improved, reference to badly named variable fixed |
| | Added better comment | More relevant comments added |
| | Rename icons and funcs | Icons and functions names changed |

This version was also evaluated in the experiment (see Sections 8.1.3-8.1.4).

The final (third) version is a result of transformation of the dictionaries presented above into a real ontology. Two main classes: `Change` and `Goal` were established to reflect the concepts of `What` and `Why` was done. Within the `Change` group two subclasses have been isolated, accordingly to the analyses presented in [97]: `AdditionChange` and `SynthesisChange` (see also Section 2.3.4). In turn, the `Goals` were divided arbitrarily into `VisualGoals` that cover all changes done in order to improve the visual side of the system and `KnowledgeGoals` that represent all goals concerning the KB's core. Final version of the ontology is depicted on Figure 6.1 and serialized on Listing 6.1. Concepts from this ontology may be linked with specific changes made in the KB (see Section 6.2) and then used e.g. to determine whether user is `Adder` or `Synthesizer` in order to update the interface properly.

Original Contributions

The general CKE change ontology presented on Figure 6.1 was developed by the author. Separation of `Change` and `Goal` concepts was motivated by further processing possibilities, e.g. the ones related to the adjustment of user interface. Two main change classes, i.e. `Addition` and `Synthesis` change, were taken from the previously cited source [97]. Finally, all other concepts in the ontology are results of analyses undertaken by the author.

Implementation in Loki

Considering the construction of `Loki`, it was decided to expand the page edition form in order to give the user the possibility to annotate the revision with `Change` and `Goal` information. To achieve this, the proper plugin was developed⁵ and the method that injects specified html code into the edit form was implemented. The first evaluation (see Section 8.1.2) indicated that when one option was selected by default, it was often not changed by the user, leading to the falsifications of the annotations. In the second version, it was decided to force the user to choose the option via a proper JavaScript code, which resulted in more differentiated

⁵Presented functionality is a part of prov plugin available at <http://loki.ia.agh.edu.pl/>

```

1 @prefix : <http://loki.ia.agh.edu.pl/wiki/change#> .
2 @prefix owl: <http://www.w3.org/2002/07/owl#> .
3 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
4 @prefix xml: <http://www.w3.org/XML/1998/namespace> .
5 @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
6 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
7
8 <http://loki.ia.agh.edu.pl/wiki/change#> rdf:type owl:Ontology .
9
10 ### Classes
11
12 :Change rdf:type owl:Class .
13 :Goal rdf:type owl:Class .
14
15 :AdditionChange rdf:type owl:Class ; rdfs:subClassOf :Change .
16 :OtherChange rdf:type owl:Class ; rdfs:subClassOf :Change .
17 :SynthesisChange rdf:type owl:Class ; rdfs:subClassOf :Change .
18
19 :KnowledgeGoal rdf:type owl:Class ; rdfs:subClassOf :Goal .
20 :OtherGoal rdf:type owl:Class ; rdfs:subClassOf :Goal .
21 :VisualGoal rdf:type owl:Class ; rdfs:subClassOf :Goal .
22
23 ### Data properties
24 :changeDescription rdf:type owl:DatatypeProperty ; rdfs:domain :OtherChange ;
25                                     rdfs:range xsd:string .
26 :goalDescription rdf:type owl:DatatypeProperty ; rdfs:domain :OtherGoal ;
27                                     rdfs:range xsd:string .
28
29 ### Individuals
30
31 :ExistingContentExpanded rdf:type owl:NamedIndividual , :AdditionChange .
32 :NewContentAdded rdf:type owl:NamedIndividual , :AdditionChange .
33 :TyposOrOtherSmallBugsFixed rdf:type owl:NamedIndividual , :AdditionChange .
34
35 :ContentConnectedWithOtherConcepts rdf:type owl:NamedIndividual , :SynthesisChange .
36 :ContentDeleted rdf:type owl:NamedIndividual , :SynthesisChange .
37 :Refactoring rdf:type owl:NamedIndividual , :SynthesisChange .
38
39 :ErrorsFixing rdf:type owl:NamedIndividual , :KnowledgeGoal .
40 :KnowledgeCoherenceImprovement rdf:type owl:NamedIndividual , :KnowledgeGoal .
41 :KnowledgeBaseExpansion rdf:type owl:NamedIndividual , :KnowledgeGoal .
42
43 :ApperanceImprovement rdf:type owl:NamedIndividual , :VisualGoal .
44 :CodeReadabilityImprovement rdf:type owl:NamedIndividual , :VisualGoal .
45 :UsageQualityImprovement rdf:type owl:NamedIndividual , :VisualGoal .

```

Listing 6.1: Full version of general CKE change ontology proposed in this dissertation



Figure 6.1: Full version of general CKE change ontology proposed in this dissertation

option selection (see Section 8.1.3). Final form is presented on Figure 6.2. Change ontology option selection is highlighted with the blue frame. If “Other...” option is chosen, then additional box for user input appears. The form is more comprehensively described in Section 6.2.

European Wiki

Assignment of a specific concept to a change requires only to select a proper option from a list in the wiki edition form (see Figure 6.2). Options selected by users in *European Wiki* project along with the comments entered by them are presented in Table 6.2. Based on presented example, it can be stated that *change ontologies* may be used to successfully gather information about changes’ types and goals from users. Forced choice introduced in the second version of the plugin led to more varied statistics, probably due to reflection on available options. As a result, they can be treated as more reliable ones.

(a) Default DokuWiki edition form

(b) DokuWiki edition form extended by the prov plugin

Figure 6.2: Default and extended DokuWiki/Loki edition form

6.2 Semantic Changelog

Idea and Specification

Semantic changelog is a graph-based way of describing changes made in the knowledge base. “Semantic” is here understood as changelog elements meaning and the ability to automatically analyze them. The difference between the changelog and the semantic changelog is thus comparable to the difference between the Wiki and the Semantic Wiki: semantic changelog is developed by adding Semantic Web technology elements (like concepts, ontologies, use of RDF syntax) to the “classical” changelog. Such a *semantic changelog*, which is de facto RDF-grounded knowledge base, should be separated from the KB that represents the expert knowledge. Here, the semantic changelog plays the role of a meta-layer describing what changes were made in the actual KB.

The overview of the *semantic changelog* is presented on Figure 6.3. The base here is the KB created in the CKE process. Particularly, on the figure KB_{n-1} state is presented. It represents the whole KB status at time $n - 1$. Then, a set of changes $Change_n$ is applied, resulting in the KB_n state. *Semantic changelog* meta-layer is here introduced to describe the KB states and the changes:

- Each KB state is characterised by the results of unit tests (see Section 5.1), a set of metrics (see

Table 6.2: Factual change and goal annotations selected by users in *European Wiki* project

| Change | Comment | Factual change | Goal |
|--------|-----------------------------------|------------------------------------|-------------------------------|
| Ch_1 | created | New content added | Knowledge base expansion |
| Ch_2 | created | New content added | Knowledge base expansion |
| Ch_3 | created | New content added | Knowledge base expansion |
| Ch_4 | onRiver and direct-FlightTo added | Content connected with other pages | Knowledge base expansion |
| Ch_5 | Obsolete categories removed | Content deleted | Knowledge coherence improving |
| Ch_6 | Added a little curiosity | New content added | Knowledge base expansion |

Section 5.2), the votes given by the users (see Section 5.3) and some metadata (described below).

- As a KB is built up from many concepts, a KB state consists of specific concepts' revisions. E.g. on Figure 6.3 one can see that the changes were applied only to the $C2$ concept, resulting in different revisions of this concept in subsequent KB states (revision $m - 1$ in KB_{n-1} and revision m in KB_n).
- Finally, each change made is described by the unit tests' and metrics' difference between two subsequent KB states, as well as a change type and a goal gathered from change ontology (see Section 6.1).

Changes are also summarised by semantic statistics and some metadata (described below).

Semantic changelog is based on the PROV standard developed by W3C Consortium to describe the data provenance (see Section 2.3.2 and Figure 2.3). This standard is actually an RDF dictionary that specifies the vocabulary useful for this task. Thanks to being grounded in RDF, it is easily serializable to one of the RDF syntaxes, and then processable by the use of SPARQL query language (see Section 3.2).

To properly describe the metadata, there is a need to develop the *semantic changelog* skeleton that represents the KB, its states, concepts' revisions, changes made and relations between them. Here, change presented on Figure 6.3 will be used as an example (KB states are represented by red and black elements; C nodes represent concepts, L – literals; *semantic changelog* elements are green). The `:C1` and `:C2` terms are introduced to represent specific concepts from the KB being developed. Each of them has its own revisions (`:C1_1`, `:C2_m-1`, `:C2_m`) connected with the main terms by the relation `prov:specializationOf`. Subsequent revisions of the same concept are connected using `prov:wasRevisionOf` property. There is also a `:KB` element, as well as KB states `:KB_n-1` and `:KB_n` that group (`prov:Collection`) the specific concepts' revisions. Transition between KB states is done by changes, which is indicated by the fact that `:KB_n` is connected with the `:Change_n` by the `prov:wasGeneratedBy` property. Due to the fact that each KB state is interlinked with all specific concepts' revisions, it can be easily processed to indicate which concepts were affected by the given change. Final changelog skeleton is presented on Figure 6.4. To simplify, only $C1$ and $C2$ concepts from KB and their revisions are represented.

Such a skeleton may be then filled by all previously mentioned characteristics of changes and KB states:

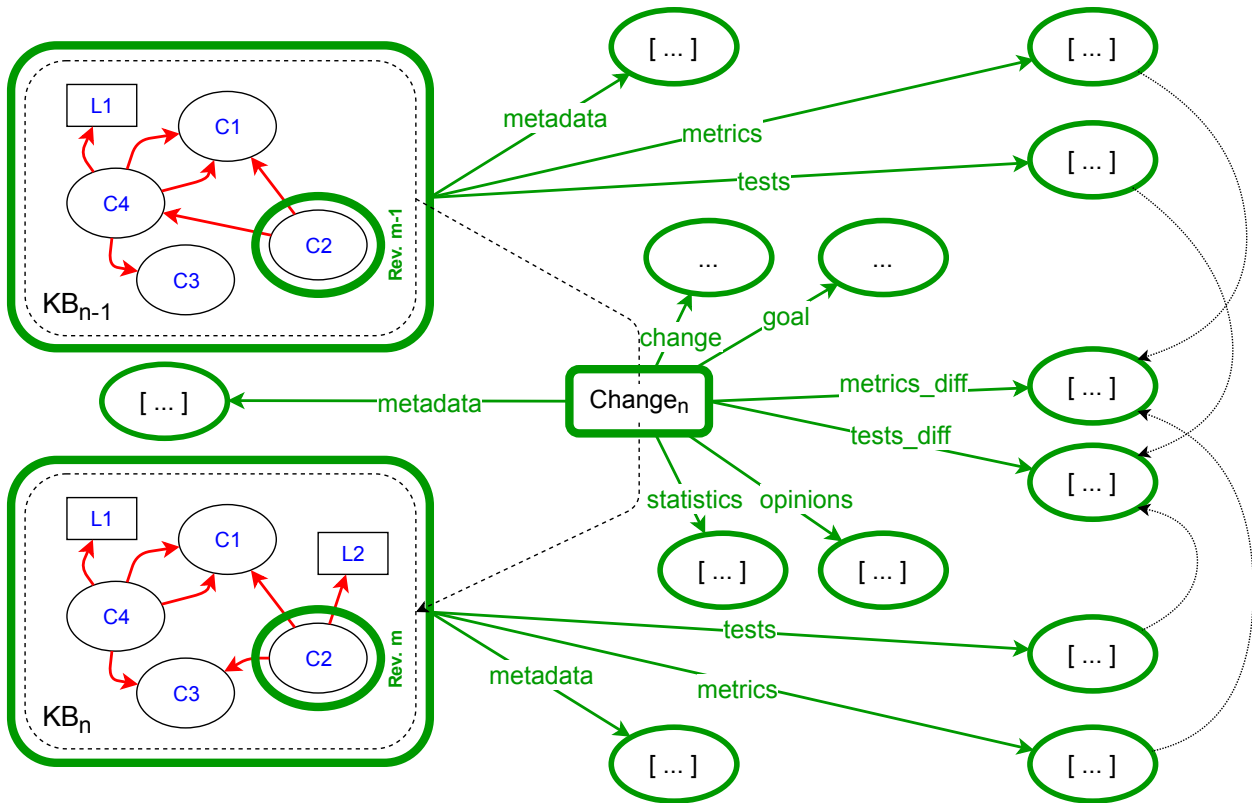


Figure 6.3: *Semantic changelog* provides a meta-layer for subsequent states of KB

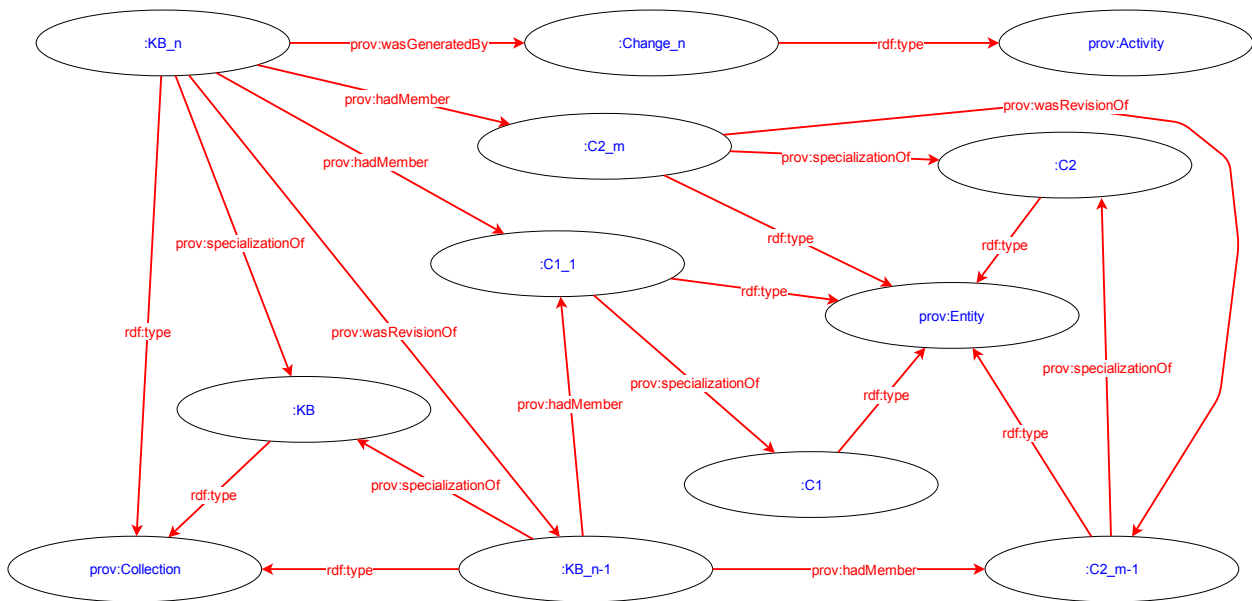


Figure 6.4: *Semantic changelog* skeleton that describes the change presented on Figure 6.3

- Both “classic” *metadata* available in all changelog systems (author, timestamp, short comment) and more rich data: internal (other concepts description) and external (e.g. books, web pages, ...) sources of knowledge, as well as users’ characteristics. In addition to the prov vocabulary (prov:wasAssociatedWith, prov:used), other dictionaries that may also be considered useful are Friend-of-a-Friend (e.g. foaf:Person foaf:name, foaf:mbox) and Dublin Core (e.g. dc:description) (see Figure 6.5; :Internal_resource and :External_resource may be any valid URI).

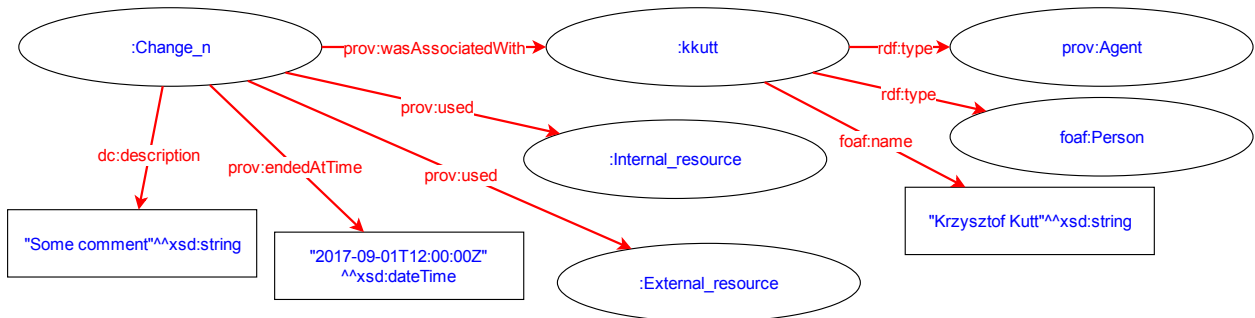


Figure 6.5: Metadata representation within the *semantic changelog*

- Basic *semantic statistics* describing how many class, object property and data property statements were changed (added, removed, corrected) (see Figure 6.6).

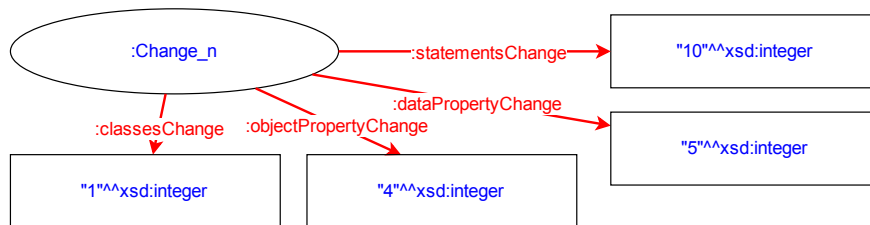


Figure 6.6: Semantic statistics saved in the *semantic changelog*

- *Reasoning unit tests’ results* and *metrics’ values*. These values are collected for specific KB states and then the change impact may be calculated as a weighted average metric (as described in Section 5.2) (see Figure 6.7).

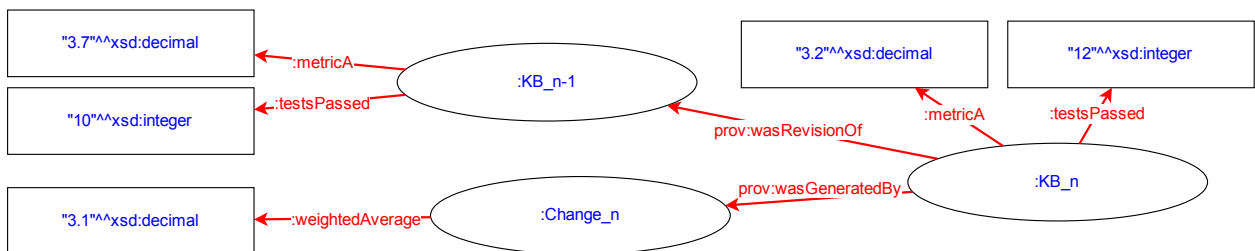


Figure 6.7: Number of reasoning tests passed and metrics in the *semantic changelog*

- Users’ *opinions* for specific concepts’ revisions in a form of votes (see Figure 6.8).

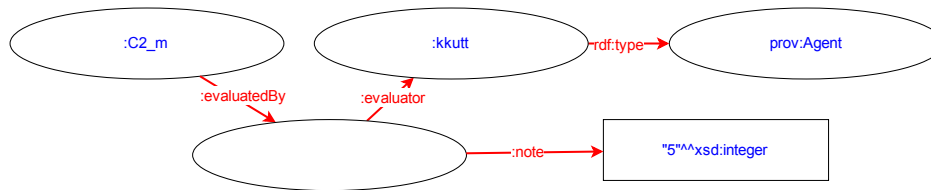


Figure 6.8: Vote for specific concept's revision within the *semantic changelog*

- *Change* and *goal* selected from the change ontology or entered by hand (see Figure 6.9). There is a possibility to use existing concepts from change ontology, as in `:change`, or to create own descriptions using `:OtherChange` and `:OtherGoal` concepts, as in `:goal` (compare with the Figure 6.1).

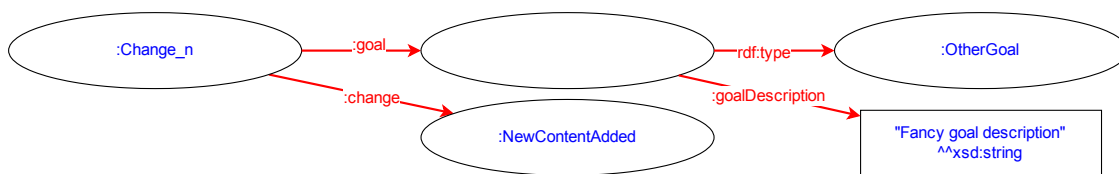


Figure 6.9: Change type and goal assignment to the specific change in the *semantic changelog*

Specification of a changelog in a semantic way gives a possibility to process it in various ways. Here, five use cases will be indicated:

Get rid of changes with poor statistics. With tests' statistics, one can quickly identify which changes are bad⁶ and should be further examined. In a more rigorous case, a system administrator can block the ability to save changes if a new revision is worse than the previous one (less tests were passed).

Sources analysis. By combining tests' statistics, metrics of changes, users opinions and sources lists, one can determine which sources are of low quality and should not be used in the future.

User types identification. Thanks to placing the solution on the top of changes ontology and on the top of all metrics, one can identify different kinds of users, e.g. good users (with good notes, e.g. as a result of more tests passed after editions) and bad users (who introduce changes with poor statistics and conceivably should be automatically banned) or creators (users that add a lot of text), annotators (the ones that provide many new relations for existing text) and proofreaders (users that mainly fix spelling and small bugs).

Underdeveloped pages indication. Semantic statistics provide information about how a specific concept is rich with the semantics (e.g. how many relations are stated for the concept). If it is not, perhaps it would be advisable to pay users' attention to the particular concept.

⁶In this thesis, bad changes are the ones with less tests passed and weightedAverage value below 3.0. On the other hand, good changes are the ones with more tests passed and weightedAverage value above 4.0.

Motivation by gamification. Accurate metrics allow for awarding points to users, giving them badges for different tasks (e.g. a badge for adding 5 new relations to the KB) and creating leaderboards. This strategy motivates them to create better knowledge base, which is a main goal of CKE.

These use cases scenarios were also tested within the prototypical *semantic changelog* implementation for Loki. Specific queries that search for values needed will be provided in the next part of this section.

Original Contributions

The idea of semantic changelog, i.e. to store the KB change metadata along with additional characteristics in a graph-based way that can be further processed, is based on W3C PROV standard. It was extended and aligned by the author to the CKE setting to provide a way of KB states and changes representation. Also, the author developed sample use cases scenarios. Finally, development of prov plugin for Loki that implements both *change ontologies* and *semantic changelog* functionality was done by the author.

Implementation in Loki

Due to Loki's specificity, some modifications were made to general changelog semantics presented above. First of all, in Loki, each page (concept) change generates a new version of the KB. As a consequence, there is no need for isolation of the KB state concept that groups different concepts' revisions. Secondly, a change within the wiki represents a specific page change and is identified by a timestamp. This is a consequence of wiki design, where each page revision is named and archived using the timestamp. Incorporation of timestamp into the change makes further processing easier. It also gives the possibility to list all KB states – such an operation needs only to list all changes of all pages and sort them by the timestamp.

The *semantic changelog* plugin⁷ developed consists of three logical parts:

- First one is the extended wiki edition form presented on Figure 6.2. It consists of four main components:
 1. Change ontology option selection (blue frame) described in Section 6.1.
 2. List of URIs used during page edition (red frame): both external and internal, i.e. other pages within the wiki, shortened by the use of `lokipage:` prefix.
 3. Helper list of all current wiki pages with simple filtering capabilities (yellow frame).
 4. Classical DokuWiki comment field (black frame).

This form was created by the HTML code injection into the wiki code through the provided plugin mechanism.

- Second part is the semantic changelog skeleton generated by the `prov` plugin. It defines pages, their revisions, changes and relations between all of them. It also calculates the semantic statistics and save

⁷Semantic changelog is implemented as a prov plugin available at <http://loki.ia.agh.edu.pl/>

the description and resources entered in the extended edition form. Files are generated using the Turtle syntax of RDF and saved in the separate `data/prov` directory within the wiki installation. One file is generated for each wiki page.

- The third one is a set of additional information generated by the external plugins, e.g. reasoning test statistics calculated by the `loki` plugin or users' votes gathered by the `revisionsrater` plugin. Proper methods for triples generation are placed in the `prov` plugin to enable easy management of the whole semantic changelog in one place. These methods are public and accessible by the external plugins that put something inside the changelog. If the `prov` plugin is not available in the system, external plugins simply do not create the changelog and only store their information in their own files.

Semantic changelog is generated according to the specification presented in the first part of this section (see Figures 6.4-6.9), with small changes related to the wiki specificity. Full changelog template is presented on the Listing B.1 on page 135. All its parts will be discussed here. Sample changelog generated during the *European Wiki* project (see Section 6.2) is available on Listings B.2-B.4.

Namespaces In all changelog files, two static namespaces are used: `change` that represents the change ontology (the one presented on Listing 6.1) and `loki` which groups all changelog terminology. Also, three more namespaces are generated for the specific wiki instance: `lokipage` as a shortcut for all pages within the wiki, `lokievent` that provides unique URIs for all change events and `lokiuser` which is a shortcut to special user-related wiki pages. `lokipage` and `lokiuser` URIs represent the URLs of existing pages that can be accessed through the HTTP. To make it possible, namespaces are handled according to the one of wiki's URL formats that is defined in the wiki's settings. Let us consider wiki instance available at: `http://home.agh.edu.pl/~kkutt/wiki/`. By default, settings pages can be accessed through the `http://home.agh.edu.pl/~kkutt/wiki/doku.php?id=page` addresses and the namespace `@prefix lokipage:` `<http://home.agh.edu.pl/~kkutt/wiki/doku.php?id=>` is generated. If the "nice URLs" option is selected, the pages are available at `http://home.agh.edu.pl/~kkutt/wiki/page` addresses and the namespace is generated accordingly as `@prefix lokipage:` `<http://home.agh.edu.pl/~kkutt/wiki/>`.

Skeleton Three main concepts within the changelog skeleton are: a page, a revision of the page and a change. A page is simply represented by its URL: `lokipage:page1`. Each page revision is represented by the name of the page and the revision's timestamp: `lokipage:page1_timestamp`. Finally, due to the fact that wiki differentiates three main types of events (creation, edition, deletion), three types of changes are differentiated in the changelog. Each change within the wiki is related to the specific page revision, which finally gives the URI of the change: `lokievent:edited_page1_timestamp`. In the Listing B.1 two values are used as a timestamp: the `$newRev` indicates the timestamp of the revision created

by the current change and the `$oldRev` represents the previous revision of changed page.

Metadata All resources entered in the extended page edit form are changed into a list of values for `prov:used` predicate. To simplify the notation, when user selects internal resources from the available selection box, they appear with the `loki:page:` prefix discussed above.

Semantic statistics Four basic statistics are calculated for classes assertions (`[[category:class]]`), object properties (`[[relation::other_page]]`) and data properties (`[[attribute:=value]]`). Change is calculated as a sum of added and corrected assertions, reduced by the number of removed assertions. Also, the number of all semantic assertions changes is provided as a `loki:statementsChange`.

Reasoning unit tests Results of the tests are gathered before and after the change and saved within the changelog. For further processing, the change impact can be calculated by simple subtraction of these two values.

Metrics As it was indicated in the Section 5.2, four metrics are currently available in Loki. They are saved in the same way as unit tests results. Also, weighted average is calculated and saved as a separate property in the changelog.

Opinions All users' votes are saved along with the evaluator login. If the user is not logged in, a special concept `loki:guest` is used instead.

Change ontologies For both Change and Goal there is a possibility to select a value from the change ontology or to enter it by hand. Changelog is then updated according to the specification presented on the Figure 6.9.

Each wiki page has its own changelog file. They can be easily combined to provide a possibility to process the whole graph, i.e. ask queries against the whole set of changelog files. Several examples tested with the data collected during experiments will be discussed below.

Identification of changes with poor statistics Simply speaking, bad changes are the ones that result in the fall in the reasoning unit tests passed number and in the weighted average below the 3.0 value. These are considered jointly to omit the changes that only worsened one of these values – such changes probably were on the expected path to the better knowledge quality. To select all such changes one can ask a query against the semantic changelog⁸:

⁸All queries presented assume that prefixes defined in Listing B.1 are available. They are here omitted to simplify the code.

```

1 SELECT ?change ?user
2 WHERE {
3     ?change a prov:Activity ;
4         loki:testsPassed [ loki:valueBefore ?testsBefore ;
5                             loki:valueAfter ?testsAfter ] ;
6         loki:weightedAverage ?metric ;
7         prov:wasAssociatedWith ?user .
8     FILTER (?testsBefore > ?testsAfter) .
9     FILTER (?metric < 3.0) .
10 }

```

The results of this query will allow the administrator to restore a previous, better-quality page revision, and identify a user who may potentially harm the KB. Such a query may be also used as a condition that is checked before the change is applied. It is important to note that such a practice will be against the wiki principles, as it does not allow to save the page at any time with any content. One can also use more sophisticated queries to more precisely select bad changes of interest.

Identification of users associated with low quality changes Aggregation of changes listed in previous query may lead to preparation of TOP 5 potentially least useful users. It only needs to be extended by the count of all found changes per user:

```

1 SELECT ?user (COUNT(?user) as ?maliciousChanges)
2 WHERE {
3     ?change a prov:Activity ;
4         loki:testsPassed [ loki:valueBefore ?testsBefore ;
5                             loki:valueAfter ?testsAfter ] ;
6         loki:weightedAverage ?metric ;
7         prov:wasAssociatedWith ?user .
8     FILTER (?testsBefore > ?testsAfter) .
9     FILTER (?metric < 3.0) .
10 }
11 GROUP BY ?user
12 ORDER BY DESC(?maliciousChanges)
13 LIMIT 5

```

This is also a kind of gamification technique, as it provides a TOP 5 list (a concept borrowed from games) and may motivate users to perform better in the future.

Another approach to identifying potentially least useful users is to find those who have performed the most significant deletion actions. This approach will help to prevent the competition among users, which may deliberately negatively affect the content richness. The query uses regular expressions to check the

change URI and change type URI from change ontology. It is also sorted by the number of deletions and limited to TOP 5 users, as page deletion is a natural operation within wiki, and only done in a too frequent manner is probably malicious to the KB:

```

1 SELECT ?user (COUNT(?user) as ?deletions)
2 WHERE {
3     ?change loki:whatWasDone ?changeType ;
4     prov:wasAssociatedWith ?user .
5     FILTER (regex(str(?change), "deleted")
6             || regex(str(?changeType), "ContentDeleted") ) .
7 }
8 GROUP BY ?user
9 ORDER BY DESC(?deletions)
10 LIMIT 5

```

Sources analysis. Resources entered by users as a sources of knowledge may be analysed in order to select the good ones that can be proposed for further changes, e.g. for inexperienced users. Also, the worst resources may be identified, and as a result further changes with them may cause an alert to the user, suggesting to rethink the changes made. To achieve the goal of the good resources selection, one can execute the following query which counts how many good changes (more tests passed and weightedAverage value above 4.0) are connected with each of resources:

```

1 SELECT ?resource (COUNT(?resource) as ?resourceStats)
2 WHERE {
3     ?change a prov:Activity ;
4     loki:testsPassed [ loki:valueBefore ?testsBefore ;
5                       loki:valueAfter ?testsAfter ] ;
6     loki:weightedAverage ?metric ;
7     prov:used ?resource .
8     FILTER (?testsBefore < ?testsAfter
9             || ?metric > 4.0) .
10 }
11 GROUP BY ?resource
12 ORDER BY DESC(?resourceStats)
13 LIMIT 5

```

Users' types identification. As it was stated before (see Section 6.1), users can be ascribed to two main categories: Adders and Synthesizers. Such identification may be then used e.g. for the customization of user interface, with respect to the specific group needs. To properly identify whether user should be categorized as the former or as the latter, the following query can be executed:

```

1 SELECT ?user ?type
2 WHERE {
3   {
4     SELECT ?user (COUNT(?user) AS ?additions)
5     WHERE {
6       ?change rdf:type prov:Activity ;
7       loki:whatWasDone ?changeType ;
8       prov:wasAssociatedWith ?user .
9       ?changeType rdf:type change:AdditionChange .
10    }
11    GROUP BY ?user
12  }
13  UNION
14  {
15    SELECT ?user (COUNT(?user) AS ?syntheses)
16    WHERE {
17      ?change rdf:type prov:Activity ;
18      loki:whatWasDone ?changeType ;
19      prov:wasAssociatedWith ?user .
20      ?changeType rdf:type change:SynthesisChange .
21    }
22    GROUP BY ?user
23  }
24  BIND(if(( ?additions > ?syntheses ), "Adder", "Synthesizer") AS ?type)
25 }

```

European Wiki

Semantic changelog module gathers various system characteristics, generated by previously mentioned modules, and combines them in RDF files. One file is prepared for each wiki page, according to the scheme presented on Listing B.1. During the first iteration of the *European Wiki* project, six changes were made, leading to the preparation of three wiki pages. All changes were summarized in three RDF files, one for each of pages (see Listings B.2-B.4).

Generated files may be validated in two ways. Firstly, the values may be compared to the ones presented in previous sections of evaluation (see Tables 5.7-6.2 and Figures 5.9, 5.12). As one can see, they are identical. Secondly, files may be syntactically checked for compatibility with the RDF standard. Two online services for RDF validation were used: W3C RDF Validation Service⁹ and Apache Anything To Triples¹⁰. Both of them lead to the same result, stating that the files are correct.

⁹The service is available at <https://www.w3.org/RDF/Validator/>.

¹⁰The service is available at <http://any23.org/>.

Presented evaluation allows to state that *semantic changelog* module properly gathers many important KB characteristics and saves them in syntactically correct files. This allows further CKE process analyses, e.g. the ones presented in Section 6.2, where sample use cases were given.

6.3 Summary

In this chapter, the idea of *change ontologies*, that help to organize the factual changes made and their goals, was presented. Then, the *semantic changelog*, the main contribution of this thesis, was described. Thanks to the changelog, all CKE modules mentioned so far (*reasoning unit tests, metrics of changes, users' opinions* and *change ontologies*) are assembled into one. It is described as “semantic”, because it adds the meaning to the data which it represents, using the Semantic Web techniques.

Chapter 7

User Involvement

CKE process is not only related to technical challenges as presented in Chapters 5 and 6 where quality and change management were discussed. It is also a matter of people who are involved in it. One of the issues regarding the participants is their motivation, especially on the knowledge acquisition step which is considered the bottleneck of the whole process [124]. This issue becomes more critical when group is working voluntarily, as considered in this dissertation (see Chapter 4 for scope definition). Here, two motivational areas will be discussed. Firstly, the gamification ideas adaptation to the CKE process will be briefly presented in Section 7.1. Then, usability issues will be described in Section 7.2. As these are strongly tool-specific aspects, they will be discussed with respect to Loki. General discussion was given in Section 2.3.4.

7.1 Gamification

Idea and Specification

Gamification is an idea of game mechanics elements' incorporation into the regular work process (see Section 2.3.4). It is becoming popular in many fields, but still lacks a proper representation and study within the CKE setting. Among existing solutions, only the simple elements may be enumerated, like list of Top 10 active groups in ICD-11 project [159] or "Contribution Scores" defined as the result of a simple mathematical operation on the number of edits and the number of pages changed in Semantic MediaWiki¹. In this section, more complex ideas for incorporation of gamification into the CKE process will be discussed.

Gamification seems to be a natural consequence of the quality and change management methods presented in this dissertation. They are all gathered and summarized in a semantic changelog graph, giving the possibility to both offline analyses of e.g. sources quality (see examples in Section 6.2) and to online instant feedback in a form of a gamification. Such feedback may be used as an incentive to achieve both main goals

¹See: https://www.mediawiki.org/wiki/Extension:Contribution_Scores.

that can be specified within the CKE setting: the need for a large amount of knowledge to cover the domain of interest (connected with the Knowledge Acquisition bottleneck mentioned before) and the need to ensure the quality of this knowledge. The former one can be achieved by focusing on number of concepts created or edited and by promoting regular work on daily basis, as it indirectly influences the amount of work done. On the other hand, the knowledge quality can be maintained by referring to the notion of user and knowledge credibility. It can be calculated using the quality management tools proposed within this dissertation.

To make it more concrete, a list of mechanisms is proposed. They are intended to use with a general CKE case as the ones presented in Section 2.1.1. They were created on the basis of use cases analysis², as well as literature studies [170]:

- The simple *points* (pts) system:
 - Login to the system – 5 pts once a day,
 - New concept added – 25 pts,
 - Concept edition – 5 pts.
- Besides this, one can get points and *badges* as a result of achievements:
 - For logging in for X days in a row:

| X | Points | Badge ³ |
|-----|--------|-------------------------|
| 1 | 10 | Welcome to the Y |
| 2 | 15 | Y is Fun |
| 3 | 15 | Y is Really Fun |
| 7 | 25 | Y for a Whole Week |
| 14 | 40 | Two Weeks in a Row? |
| 30 | 40 | You Have to be Addicted |

- For creating X new concepts:

| X | Points | Badge |
|-----|--------|--------------------------------|
| 1 | 10 | My First Concept |
| 10 | 50 | Growing the KB |
| 25 | 75 | Builder |
| 50 | 100 | KB Expansion Specialist |
| 100 | 100 | The Holy Order of the Concepts |

- For X editions (in this category X is much bigger than in the previous one to reflect the fact that editions are done more often):

²E.g. the analysis of mechanisms available at: <http://pathofexile.gamepedia.com>.

³ Y is a placeholder for system name.

| X | Points | Badge |
|------|--------|------------------------|
| 1 | 10 | Going Down in History |
| 10 | 20 | Look What I Can Do! |
| 50 | 40 | Keep Up the Good Work |
| 100 | 50 | Serious Business |
| 250 | 100 | The Sky is the Limit |
| 1000 | 100 | So... Many... Edits... |

- Points lead to achieving higher *levels*, that are calculated according to the equation⁴:

$$P_{L+1} = 40L^{\frac{5}{3}} + 10 \quad (7.1)$$

where P_L defines the number of points on level L . Calculated value is rounded to the tens. Presented formula leads to the definition of minimum points for each level: level 2 – 50, level 3 – 140, level 4 – 260, ..., level 10 – 1570, level 20 – 5420, and so on.

Methods presented so far are responsible for motivating users to use the system and to enter the knowledge (motivation especially for Adders group). To encourage participants to create good quality KB, other mechanisms are proposed (for Synthesizers):

- *Bounty* system that allows users to create challenges, e.g. “create an X concept” or “parse the source Y into the KB” and assign the rewards to them (50, 100 or 200 pts). When the challenge is finished, its author accepts the change made and the reward goes to the change maker.
- *Natural tests* mechanism that shows user the random changes and concepts from the KB and asks him or her to mark them as good or bad (this mechanism is an extension to the opinions mechanism proposed in Section 5.3). Each rating is worth 2 points for the rater⁵.
- *User’s credibility* calculated using the objective elements: (a) ratio of reasoning unit tests passed to all tests and (b) weightedAverage metric, as well as the subjective ones: (c) mean value of votes given and (d) ratio of positive natural tests values to all ratings given. If the final credibility, calculated as a simple mean of all values, exceeds the threshold of 70%, the “Trusted user” badge is granted to the user and is visible in her or his profile as long as the appropriate value is maintained. Also, “Trusted adder” and “Trusted synthesizer” badges may be considered – they should be calculated the same way as “Trusted user” but using values only for changes defined as AdditionChange or SynthesisChange (see Change ontology in Section 6.1). As users may change during the project, it is proposed to calculate the credibility based on the values from last week, or another fixed time span adjusted to

⁴It is adopted from <http://pathofexile.gamepedia.com>.

⁵This mechanism is similar to the Google Translator Community available at: <https://translate.google.com/community>.

project needs. Such a time limitation will allow for a more accurate assessment of the current user status.

All discussed mechanisms are presented with the use of:

- *User profiles*, where all statistics, points, badges and finished challenges are stored,
- *Leaderboards* with TOP 10 users. Two lists are proposed: one that sums the points and reflects mainly the *quantity* of content added and the second that summarizes the user's credibility and reflects the *quality* of the content.

A set of gamification techniques presented in this section is a result of general CKE process analysis and is adapted to support its main needs, especially the motivation for knowledge acquisition and the motivation for providing good quality knowledge. It should be noted that this set should be thoroughly tested and tailored to the particular use case and specific group of people taking part in the CKE process. One also has to pay attention to the fact that not everyone is fond of games and, in consequence, gamification. With this in mind, a possibility to hide the gamification module for specific user should be guaranteed.

Original Contributions

Prepared *gamification* module is a set of techniques that were previously described in the gamification handbooks. The author contribution is the idea of incorporation of these ideas into the *CKE agile process* with the use of capabilities provided by the *semantic changelog* module. Resulting techniques may be considered as a set of involvement metrics, useful for e.g. scrum owner in CKE process. Implementation of wikigame module for Loki was done by the author.

Implementation in Loki

The *gamification* module for Loki⁶ is an use case for *semantic changelog* module. It takes the whole changelog graph, processes it and generates two outputs:

- the set of statistics for each user on a dedicated user page: level, sum of points, credibility details (as in the Table 7.2) and the list of badges,
- the TOP 10 leaderboard in a side panel of the wiki that summarizes: the number of points (used for sorting the whole list), the levels of the users (presented in brackets) and the credibility of the users (indicated with a “tick” when it exceeds the given threshold) (see Figure 7.1).

These two elements are updated in two different ways. User's statistics are calculated after each change made by her: number of points is updated, badges are granted. On the other hand, credibility and TOP 10 list are calculated every 10 minutes. This default time interval may be changed in system settings. Such

⁶Gamification is implemented as a wikigame plugin available in a Loki repository at <https://gitlab.geist.re/pro/loki/>.

a manner was selected to not overload the system with continuous calculations while still offering almost instant feedback.

Calculation of credibility is based on three elements that are available in RDF files prepared by the *semantic changelog* module (compare with the file skeleton on Listing B.1):

- average ratio of unit tests passed after each change made by the user. It is based on the `loki:testsPassed` relation from *semantic changelog* and on the total number of tests stored by Loki in a dedicated text file,
- an average *weighted average* score for changes of the user specified in the *semantic changelog* by the `loki:weightedAverage` relation,
- average value of votes given to user's changes indicated by the `loki:evaluatedByUser` construct.

The points, levels and badges system is implemented according to the specification presented in previous section.

European Wiki

Within the *European Wiki*, 6 changes were made by users. As a result, authors were given points and badges as described in Table 7.1. Finally, after the first project iteration users statuses were the same as presented in Table 7.3. Details about users credibility calculation are provided in Table 7.2. These values are the averages of all values related to specific user changes in the last week. As the first iteration takes only one day to complete, all 6 changes were used for calculations. Calculations were based on an assumption that for the first three changes WA_1 metric was used, and WA_2 metric for other changes. Summary of current status of users is always visible in a sidebar of the wiki (see Figure 7.1). Users are sorted according to the number of points. Trusted users, i.e. the ones with the credibility above the threshold of 70%, are marked with a tick sign. More detailed information, e.g. a list of badges, are available on users' pages.

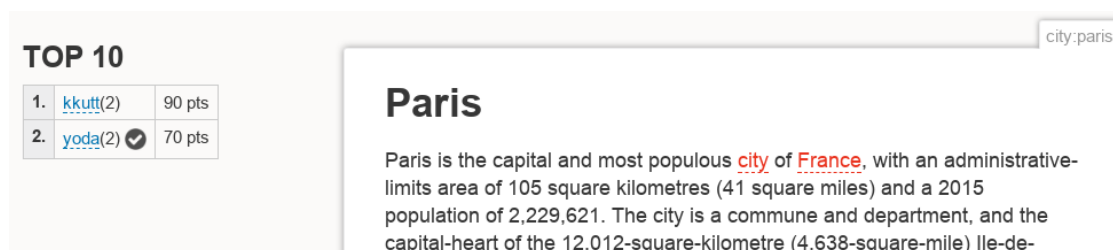


Figure 7.1: TOP 10 list in a sidebar of a wiki. It is consistent with values presented in Table 7.3

As it was shown, after the first activity in the system, users are rewarded with points and badges. Such a rapid success can be a good motivation and encouragement to work in the project. Over time, the difficulty level will increase, i.e. more activities are required to gain next badge or level. Still, it will go hand in hand with experience in project, helping in maintenance of user engagement at the right level. It was also shown

Table 7.1: Points and badges awarded for changes within *European Wiki* project

| Change | User | Achievements |
|--------|-------|--|
| Ch_1 | kkutt | Login to a system: 5 pts; New concept added: 25 pts; Logging in for 1 day: 10 pts + Badge “Welcome to the European Wiki”; Creation of 1 concept: 10 pts + Badge “My First Concept” |
| Ch_2 | yoda | Login to a system: 5 pts; New concept added: 25 pts; Logging in for 1 day: 10 pts + Badge “Welcome to the European Wiki”; Creation of 1 concept: 10 pts + Badge “My First Concept” |
| Ch_3 | kkutt | New concept added: 25 pts |
| Ch_4 | yoda | Concept edition: 5 pts; Edition of 1 concept: 10 pts + Badge “Going Down in History” |
| Ch_5 | yoda | Concept edition: 5 pts |
| Ch_6 | kkutt | Concept edition: 5 pts; Edition of 1 concept: 10 pts + Badge “Going Down in History” |

Table 7.2: Users’ credibility calculation after the first iteration of the *European Wiki* project

| User | Ratio of unit tests passed | Weighted average metric | Votes | Final Credibility |
|-------|----------------------------|-------------------------|---------|-------------------|
| kkutt | $\frac{2.33}{6}$ (44%) | 3.71 (74%) | N/A | 59% |
| yoda | $\frac{3}{6}$ (50%) | 3.91 (78%) | 4 (80%) | 70% |

that gamification is a kind of instant feedback for users. Especially, TOP 10 list is always visible and updated immediately after each change. It is also important to note that this list presents the second type of evaluation as well, i.e. the users’ credibility calculated every ten minutes. Combination of these characteristics in one list clearly promotes both addition of knowledge to the wiki (points) and good quality maintenance (trusted user mark). In conclusion, based on the presented evaluation, it can be stated that the *gamification* module introduces incentives to create a good quality knowledge base. As this is the goal of the CKE process, the module can be considered as important for the process.

7.2 Usability

Considering the Collaborative Knowledge Engineering, it is important to note that this process is related to processing of huge amounts of knowledge. When the task is to process a document with 2000 pages, gamification is good, but not enough to keep motivation of the user at the right level all the time. In such a scenario a useful and usable tool is needed.

Usability analysis in Loki has been done in two ways. Firstly, feedback was collected from the experiments participants using structured reports or a survey. Secondly, the current wiki use cases were analyzed to determine what kind of functionalities should be added. Ultimately, based on these analyzes, the decision to implement the four most anticipated features was made⁷:

⁷All were developed by the author of the thesis.

Table 7.3: Summary of status of users after the first iteration of the *European Wiki* project

| User | Sum of points | Level | Place in TOP10 | Credibility |
|-------|---------------|-------|-----------------|-------------|
| kkutt | 90 | 2 | 1 st | 59% |
| yoda | 70 | 2 | 2 nd | 70% |

- *SPARQL Endpoint*, which allows to execute SPARQL SELECT / ASK / DESCRIBE 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 Requests. This functionality is currently a standard that allows data to be retrieved from various RDF-based systems.
- *SPARQL Update Endpoint*, which allows to change contents of the wiki remotely. This solution was developed as a support for an SE project management tool, which stores project documentation within a wiki, and processes it partially (e.g. task statuses are changed) from the IntelliJ programming IDE.
- The ability to *store ontologies* in a wiki. Up until now, Loki did not allow explicit definition of ontology and its structure. It could be deduced only from statements on individual wiki pages. This hindered the big picture view on the knowledge base stored in the wiki.
- Functionality, which at each time was indicated as the most expected by the wiki users, is *code hint and completion* system, that helps to enter semantics into wiki pages. It was exceptionally important, because when a basic vocabulary was finally established in the team, there was no possibility to help users using it another than writing all important terms on a blackboard.

All of them are shortly described in this section.

SPARQL Endpoint is a web service that allows to execute SPARQL queries to the database. Its semantics and operating principle are described in a W3C Recommendation [173]. This mechanism is used by almost all RDF databases to share the data contained therein⁸. Thanks to the ability to combine results from different services in one query⁹, SPARQL enables easy manipulation of many data sources and creation of so-called semantic mashups (see Section 3.2).

Within Loki, SPARQL queries were already handled by the wiki core modules. Developing of the Endpoint required only creating a form that would take queries, delegate them to existing functions, and then return results in selected format¹⁰. Three output syntaxes are available: HTML table that is easily understood by humans and JSON and XML that are ready for machine processing (see Section 3.2 for RDF syntaxes presentation). Prepared form (see Figure 7.2) processes the arguments (the query and the results format) using the HTTP GET method in a form of an URL: `http://{wiki_address}/sparql/?query={HTTP_encoded_query}&format={html,json,xml}`,

⁸See: <https://old.datahub.io/dataset?tags=sparql-endpoint> for list of open datasets with SPARQL Endpoints.

⁹See SERVICE keyword description in [66].

¹⁰SPARQL Endpoint is a part of loki core plugin available at <http://loki.ia.agh.edu.pl/>.

providing a possibility to execute queries locally by writing them down in a form or remotely by the preparation of proper HTTP Request URL. SPARQL queries can be also executed using the XML-RPC mechanism implemented in the DokuWiki. Sample code snippet in PHP that query Loki is presented on Listing 7.1.

Loki SPARQL Endpoint

Ask queries against [this Loki instance](#): input query, set any options and press "Get Results".

Loki supports a limited version of SPARQL. On official Loki page [documentation is available](#) as well as [list of exemplary SPARQL Queries](#) against Loki.

```
PREFIX wiki: <>
SELECT ?page ?lang ?length
WHERE {
  ?page a "querytest".
  ?page wiki:test_lang ?lang.
  ?page wiki:test_length ?length.
  FILTER (?lang!='polish')
}
ORDER BY INC(?length)
LIMIT 5
```

Results Format: HTML ▾

Get Results

Figure 7.2: SPARQL Endpoint form for Loki

SPARQL Update Endpoint uses the same mechanism as presented in SPARQL (Query) Endpoint, but executes KB updates instead of queries. It is important to note that SPARQL Update operation may be difficult in Semantic Wiki, where data is primarily stored as wiki text pages and these have to be updated. Having this in mind, as well as considering an SE project management tool that has to remotely change the agile tasks descriptions stored within wiki, for which the Update Endpoint was created, a set of use cases was established:

1. New triple is added on a specified page (e.g. the task is assigned to the selected user).
2. Selected triple is removed (e.g. task was assigned incorrectly).
3. Triple object is changed (e.g. task status is changed from “In progress” to “Done”).
4. Presented changes are performed on many pages at once (e.g. all tasks from current sprint are assigned to the specific user).

To implement this functionality, SPARQL Update patterns were selected¹¹:

¹¹Current version of SPARQL Update Endpoint is available in a Loki repository at <https://gitlab.geist.re/pro/loki/>.

```

1 require_once('inc/init.php'); //DokuWiki files with XML-RPC functions (IXR_Client)
2
3 $client = new IXR_Client('http://{wiki_address}/lib/exe/xmlrpc.php');
4
5 $query = 'PREFIX wiki: <>
6 SELECT ?page ?lang ?length
7 WHERE {
8     ?page a "querytest".
9     ?page wiki:test_lang ?lang.
10    ?page wiki:test_length ?length.
11    FILTER (?lang!="polish")
12 }
13 ORDER BY INC(?length)
14 LIMIT 5';
15
16 if ($client->query('plugin.loki.querySparql', $query, 'json')
17 ) {
18     echo $client->getResponse();
19 }

```

Listing 7.1: Execution of SPARQL query against Loki using the XML-RPC mechanism

1. To add a new triple, e.g. to add statement about assignment of user to the task, the following scheme may be used:

```

1 INSERT DATA {
2     {page} {relation} {object} .
3     ##e.g. task:prepare_phd assigned_to user:kkutt .
4 }

```

New information is added at the bottom of a proper wiki page. Specifically, in the presented example, the page `task:prepare_phd` will be edited, and the annotation `[[assigned_to::user:kkutt]]` will be inserted.

2. To remove a piece of information, e.g. to purge the bad statement about user mistakenly assigned to a task, one can use a pattern:

```

1 DELETE DATA {
2     {page} {relation} {object} .
3     ##e.g. task:prepare_phd assigned_to user:gjn .
4 }

```

Specified wiki page is searched by the use of regular expression to find the given information which is then removed. When the example will be executed, the page `task:prepare_phd` will be opened and searched for the annotation `[[assigned_to::user:gjn]]`. Should it be found, it will be cleared from the text.

3. To change the value of the annotation, one can simply execute the **DELETE DATA** pattern to remove the old value and then the **INSERT DATA** scheme to add the new one.
4. When the update is requested for many pages, it could be done as a set of simple **DELETE DATA** and

INSERT DATA operations or using the scheme with **WHERE** clause:

```

1 DELETE { ?page task_status 'In Progress' . }
2 INSERT { ?page task_status 'Done' . }
3 WHERE {
4     ?page task_status 'Done' .
5 }

```

Ontologies storage for Loki was important due to two facts: firstly to provide an easy way to store the general idea of ontology used in one place (see Requirement R3; page 16), and secondly to help users with stating new annotations on 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. To help the users even more, the ontology storage was then used as a base for code hint and completion mechanism described below.

Requirements for such a storage was identified:

- Within the ontology: hierarchy of concepts, hierarchy of properties and the definition of domain and range for properties should be stored.
- There should be a possibility to store many ontologies within the system.
- The system should be easy to use and transparent to users.
- The system should be designed in such a way that in the future it can be extended to include a graphical editor and the ability to import and export ontologies.

Given the architecture of Loki and identified requirements, it was decided to store the ontologies on `special:ontology:{name}`¹² pages, where `{name}` represents the name of the ontology stored on a specific page: `default` for base ontology, `foaf` for Friend-of-a-Friend, etc. On these pages, ontologies are stored using XML syntax (see Listing 7.2). When connected with XSLT stylesheets¹³, it offers great visualisation capabilities. Two stylesheets were finally prepared: one with simple edition form, that gives a possibility to enter a new statement or delete an existing one (see Figure 7.3), and the second for viewing the ontology (see Figure 7.4)¹⁴.

Code hint and completion mechanism was selected as the most useful extension for Loki, as a solution for a problem that appears in all conducted experiments (see Section 8.1). Namely, typos or mistakes made by users 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.

¹² `special:` namespace is used in Loki to store special pages, e.g. the list of concepts, the URI resolve system, etc.

¹³ XSLT stylesheets may be used within Loki by the use of the `xslt` plugin available at: <https://www.dokuwiki.org/plugin:xslt>.

¹⁴ Current version of `lokionontology` plugin is available in a Loki repository at <https://gitlab.geist.re/pro/loki/>.

Ontology name:

Classes

| ID | Name | |
|---------------------------------|-----------------------------------|--|
| item | Multimedia Item | edit :: delete |
| book | Book | edit :: delete |
| movie | Movie | edit :: delete |
| human | Human being | edit :: delete |
| actor | Actor | edit :: delete |
| author | Author | edit :: delete |
| <input type="text" value="ID"/> | <input type="text" value="Name"/> | <input type="button" value="Save"/> |

Class relations

| Relation | Subject ID | Object ID | |
|---------------------------------------|---|--|-------------------------------------|
| subClassOf | book | item | delete |
| subClassOf | movie | item | delete |
| subClassOf | actor | human | delete |
| subClassOf | author | human | delete |
| <input type="text" value="Relation"/> | <input type="text" value="Subject ID"/> | <input type="text" value="Object ID"/> | <input type="button" value="Save"/> |

Object properties

| Property ID | Subject ID | Object ID | |
|--|---|--|-------------------------------------|
| hasAuthor | book | author | delete |
| hasActor | movie | actor | delete |
| isAssociatedWith | human | item | delete |
| <input type="text" value="Property ID"/> | <input type="text" value="Subject ID"/> | <input type="text" value="Object ID"/> | <input type="button" value="Save"/> |

Data properties

| Property ID | Domain | Range | |
|--|-------------------------------------|------------------------------------|-------------------------------------|
| hasTitle | item | xsd:string | delete |
| hasName | human | xsd:string | delete |
| <input type="text" value="Property ID"/> | <input type="text" value="Domain"/> | <input type="text" value="Range"/> | <input type="button" value="Save"/> |

Property relations

| Property ID | Subject ID | Object ID | |
|--|---|--|-------------------------------------|
| subPropertyOf | hasAuthor | isAssociatedWith | delete |
| subPropertyOf | hasActor | isAssociatedWith | delete |
| <input type="text" value="Property ID"/> | <input type="text" value="Subject ID"/> | <input type="text" value="Object ID"/> | <input type="button" value="Save"/> |

Figure 7.3: Ontology edition form for Loki. Presented ontology is saved in the XML file on Listing 7.2

To overcome discussed problems, code hint mechanism that follows the rules was implemented using the JavaScript¹⁵:

1. Mechanism works for the whole time that the edition form is opened.
2. When the user starts the annotation, i.e. uses the `[[]]` construct, hints are displayed.
3. If user enters `[[category:]]` construct, all available classes are displayed (see Figure 7.5a).
4. In the same way the plugin suggests the available object and data properties. To do it properly, current text in editor is scanned for category statements. Based on this list of classes, all properties that have defined them as domain are proposed for user. When user selects one of the properties, the possible object values are proposed using the range definition from the ontology.

For example, let us consider the ontology presented on Figure 7.4. On the current page, user already put the `[[category:book]]` statement. As a result, `hasAuthor` and `hasTitle` properties are proposed. If user selects the former, all `author` class instances are proposed as objects. If the latter is selected, user has to write the string value, so nothing is proposed.

¹⁵Presented mechanism is a part of `lokiontology` plugin used also for ontology storage within Loki. It is available in a Loki repository at <https://gitlab.geist.re/pro/loki/>.

Multimedia

Classes

- **Multimedia Item** (ID: item)
- **Book** (ID: book)
- **Movie** (ID: movie)
- **Human being** (ID: human)
- **Actor** (ID: actor)
- **Author** (ID: author)

Class relations

- **Book** -> subClassOf -> **Multimedia Item**
- **Movie** -> subClassOf -> **Multimedia Item**
- **Actor** -> subClassOf -> **Human being**
- **Author** -> subClassOf -> **Human being**

Object properties

- hasAuthor(**Book**, **Author**)
- hasActor(**Movie**, **Actor**)
- isAssociatedWith(**Human being**, **Multimedia Item**)

Data properties

- hasTitle(**Multimedia Item**, xsd:string)
- hasName(**Human being**, xsd:string)

Property relations

- hasAuthor -> subPropertyOf -> isAssociatedWith
- hasActor -> subPropertyOf -> isAssociatedWith

Figure 7.4: Ontology visualisation for Loki. Presented ontology is saved in the XML file on Listing 7.2

5. All annotations are validated online and highlighted using red color for errors (see Figure 7.5b).
6. Finally, when the user wants to save the page, a warning is displayed if the page contains errors.

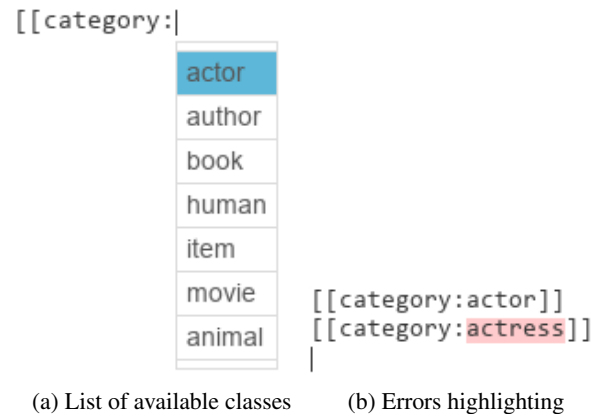


Figure 7.5: Code hint and highlight mechanism for Loki

7.3 Summary

In this chapter, issues related to user involvement were discussed. Firstly, an idea of incorporating game elements into CKE process was presented. Proposed framework consists of scores and badges that motivate users to work regularly and introduce large amounts of knowledge into the system, as well as bounty

```
1 <ontology>
2   <name>Multimedia</name>
3   <classes>
4     <class id='item'>Multimedia Item</class>
5     <class id='book'>Book</class>
6     <class id='movie'>Movie</class>
7     <class id='human'>Human being</class>
8     <class id='actor'>Actor</class>
9     <class id='author'>Author</class>
10  </classes>
11  <classRelations>
12    <relation type='subClassOf' subject='book' object='item' />
13    <relation type='subClassOf' subject='movie' object='item' />
14    <relation type='subClassOf' subject='actor' object='human' />
15    <relation type='subClassOf' subject='author' object='human' />
16  </classRelations>
17  <objectProperties>
18    <property id='hasAuthor' subject='book' object='author' />
19    <property id='hasActor' subject='movie' object='actor' />
20    <property id='isAssociatedWith' subject='human' object='item' />
21  </objectProperties>
22  <dataProperties>
23    <property id='hasTitle' domain='item' range='xsd:string' />
24    <property id='hasName' domain='human' range='xsd:string' />
25  </dataProperties>
26  <propertyRelations>
27    <relation type='subPropertyOf' subject='hasAuthor' object='isAssociatedWith' />
28    <relation type='subPropertyOf' subject='hasActor' object='isAssociatedWith' />
29  </propertyRelations>
30 </ontology>
```

Listing 7.2: XML representation of simple multimedia ontology

and natural tests mechanisms, which together with user's credibility calculation encourage participants to create good quality KB. Then, the need to consider usability issues was introduced. Four most anticipated functionalities for Loki were implemented and depicted.

Evaluation of the framework proposed in Chapter 4 and comprehensively described in Chapters 5-7 is the matter of the next chapter.

Chapter 8

Experiments and Evaluation

In Chapters 5-7, a set of methods and tools for Collaborative Knowledge Engineering was presented, as a response to the CKE Requirements outlined in this dissertation. Their general idea was presented in Chapter 4. It consists of *CKE agile process* complemented with description of related issues: (a) *reasoning unit tests, metrics of changes, opinions and discussion* mechanisms for quality management, (b) *change ontologies* and *semantic changelog* for change management, as well as (c) *gamification* and usability considerations. Presented methods are supplemented by the description of prototypical framework implementation for Loki called BiFröST.

There is a need for proper evaluation of prepared modules. Within this work, two complementary types of evaluation were performed. Firstly, it was checked whether each of the modules meet its requirements. By the use of a simple use case called European Wiki, the module behaviour was presented. The results obtained for the sample input data are verified. This was presented in previous chapters together with the modules description. That evaluation is supported by four mid-term experiments conducted to test specific hypotheses during modules' development, to gather some data about real modules usage and to receive some feedback from the users. These experiments are described in Section 8.1. Final evaluation of the whole framework is provided in Section 8.2. Modules coverage in conducted experiments is presented in Table 8.1. Section 8.3 concludes this chapter.

8.1 Conducted Experiments

Modules described in this thesis are supported by four experiments conducted during their development. The first one (see Section 8.1.1) was designed to evaluate whether a methodology in which a team of experts and a team of knowledge engineers work independently (alternately) is worth considering. The second and the third, described in Sections 8.1.2-8.1.3, were conducted to check current modules versions, and to test selected elements from the *CKE agile process*. Goals of the fourth experiment (see Section 8.1.4), except

Table 8.1: Modules (and plugins' versions) coverage in conducted experiments

| Module (plugin) / Feature | Experiment | | | | |
|--|----------------|----------------|----------------|----------------|----------------|
| | 1st | 2nd | 3rd | 4th | Final |
| Loki | + (2011-04-14) | + (2015-12-01) | + (2016-10-26) | + (2016-10-26) | + (2017-09-19) |
| Reasoning unit tests (Loki) | | | | | + (2017-09-19) |
| Metric of changes (prov) | | | | + (2017-01-25) | + (2017-09-19) |
| Opinions and discussion (revisionsrater) | | | | | + (2017-09-19) |
| Change ontologies (prov) | | + (2015-12-01) | + (2016-12-04) | + (2017-01-25) | + (2017-09-19) |
| Semantic changelog (prov) | | + (2015-12-01) | + (2016-12-04) | + (2017-01-25) | + (2017-09-19) |
| Gamification (wikigame) | | | | | + (2017-11-14) |
| Usability extensions (Loki) | | | + (2016-10-26) | | + (2017-09-19) |
| CKE agile process | + | + | + | + | + |
| Usability evaluation (SUMI) | | | | | + |
| Comparison with SMW | | | | + | |

these of previous ones, also included an attempt to gather feedback about usability, which was further used to select solutions that were developed for Loki (see Section 7.2), and to compare current process in Loki to the one in Semantic MediaWiki.

8.1.1 First Experiment: Pokemons, Simpsons, et al.

The first experiment was conducted during the winter term of 2014/2015 academic year. Eight students of computer science at AGH-UST, participants of “Semantic Web Technologies” course, took part in the *CKE process* in which a team of experts and a team of knowledge engineers work alternately and independently. Specifically, there were four independent processes led by four two-person expert teams. Each of them was aimed at creating a wiki about the subject chosen by the team (Pokemons, Simpsons, Dragon Ball Z and Drinks). During the course of the experiment, the teams exchanged wiki systems between themselves, simulating the appearance of new actors – knowledge engineers. Finally, for the evaluation phase of the experiment, the second exchange was carried out. The course of experiment is summarized in Table 8.2.

The whole experiment lasted six weeks. The first three were devoted to the first phase aimed at creation.

Table 8.2: Summary of the course of the first experiment

| | Phase 1: Creation | Phase 2: Annotation | Phase 3: Evaluation |
|--------|-------------------|---------------------|---------------------|
| Team 1 | Pokemons | Simpsons | Dragon Ball Z |
| Team 2 | Drinks | Dragon Ball Z | Simpsons |
| Team 3 | Simpsons | Pokemons | Drinks |
| Team 4 | Dragon Ball Z | Drinks | Pokemons |

Teams took on the role of domain experts. Based on their knowledge of the chosen topic and on the publicly available materials, they prepared “classic” wikis containing text, links and media files. After the first wikis exchange, the second phase began and participants shifted to the roles of knowledge engineers. They were introduced to KE and Semantic Web techniques during the lectures and laboratories preceding the experiment. Within the second phase, their task was to identify the right vocabulary and to use it to annotate the content. As a result, the “classic” wikis were turned into the “semantic” ones. Finally, during the sixth week, evaluation phase took place. Teams were granted access to systems that they had not seen before. Their task was to evaluate the content of the wiki, both text and semantic annotations, and perform final fixes and cleaning.

Each CKE process was conducted with the use of DokuWiki (version 2014-09-29b “Hrun”) combined with Loki plugin (version 2011-04-14). During 6 weeks of experiment, eight participants made 3000 changes, which resulted in creation of 1891 RDF triples saved on 345 wiki pages (for detailed summary see Table 8.3). The whole experiment was concluded by a discussion with participants. The most important observations are summarized in two points.

First of all, attention has been paid to the difficulty of involving engineers in the project. In the experiment, wikis were created on simple topics, but it took some time to come up with a hierarchy of concepts that would correspond to the knowledge. Participants noted that in the case of a specific topic in the field that they did not know, it would be impossible or at least very resource intensive task.

Secondly, the task of the members of the group was to work simultaneously. It was requested to make the interactions between them susceptible to observation. Unfortunately, work was often done in completely different time periods, and on the other hand, occasionally, it was performed by two people sitting with one device and logged as one of the participants. As a result, the analyses of the interactions based on the systems logs, were impossible to perform, due to the lack of data.

The first experiment finally led to two conclusions. Firstly, during the *CKE process*, domain experts and knowledge engineers must collaborate at every stage. Secondly, experiments with students should be conducted under more strictly controlled conditions. Due to this observation, the further experiments were done in a class with an author available for the whole time as a product owner, iteration master and more advanced knowledge engineer that supported the whole team.

8.1.2 Second Experiment: CSP Library

The second experiment was a part of “Semantic Web Technologies” course offered during the winter term of 2015/2016 academic year. Thirteen students of computer science at AGH-UST involved in the project aimed at transferring the knowledge available in CSPLib¹ to the wiki system and preparing relevant semantic

¹Constraint Satisfaction Problem Library. See: <http://www.csplib.org/>.

Table 8.3: Statistics of CKE processes within the first experiment

| | Changes | Wiki pages | Categories | Relations | Triples |
|---------------|--|------------|------------|-----------|---------|
| Dragon Ball Z | Phase 1: 250, Phase 2: 187, Phase 3: 22, Sum: 459 | 49 | 5 | 31 | 214 |
| Drinks | Phase 1: 411, Phase 2: 539, Phase 3: 14, Sum: 964 | 121 | 4 | 9 | 742 |
| Pokemons | Phase 1: 544, Phase 2: 249, Phase 3: 18, Sum: 811 | 104 | 10 | 21 | 503 |
| Simpsons | Phase 1: 232, Phase 2: 532, Phase 3: 2, Sum: 766 | 71 | 7 | 36 | 432 |

annotations. The experiment had several goals:

1. To evaluate current version of Loki and *semantic changelog* plugins.
2. To verify the first version of *change ontology*.
3. To implement simple *CKE process* based on Ontology 101 (see Section 2.3.3) and gather feedback from participants.
4. To observe whether any roles will emerge during the process.

Experiment started with short introduction to Loki. Participants created simple wiki pages about themselves to get familiar with the syntax and capabilities of the wiki. Then, the CSP Library project started. It was done in an iterative manner. Each of them lasted 2 hours and consisted of three stages: (a) Competency Questions, (b) Implementation, (c) Evaluation. There were 5 iterations:

1. During the first iteration, first set of competency questions to CSP Library was developed. After that, ontology prototypes were created. In order to give everyone the same opportunity to participate in the ontology creation process, this was done in two-person groups. When prototypes were created, 6 groups were merged to form two larger ones, the task of each was to develop a common ontology based on the ones created before. Finally, during a discussion moderated by the author of the dissertation, one common version of the ontology was established.
2. Second iteration started with implementation phase. First wiki pages were created and annotated. Thereafter, evaluation phase started. Great attention was devoted to testing during this iteration. The whole team was divided into 5 groups. Each of them was assigned a different item from a set of competency questions created before. The task was to formalize the question into a SPARQL query, run it against the wiki and interpret the result.
3. Third, fourth and fifth iterations were run using the same scheme. They started with 10 minutes discussion about competency questions update. Then, implementation phase took place, when new wiki pages were created and ontology was updated if needed. Each iteration was summarized by 15 minutes long evaluation phase, with the use of SPARQL queries.

At the end of experiment, each of participants prepared the report accordingly to the given scheme (see

Appendix C).

Project was developed in the DokuWiki (version 2014-09-29b “Hrun”) combined with Loki (version 2015-12-01) and prov (version 2015-12-01) plugins. Thirteen users made 1186 changes. Eventually, the knowledge base consisted of 821 triples, 9 relations and 5 categories, saved on 265 wiki pages. Subjective author observations and analysis of reports prepared by participants led to the following conclusions:

1. Several issues with the wiki and *semantic changelog* plugins were identified and further analysed after the experiment (see the first goal). Especially, the lack of proper code hint and completion mechanism was identified by users as a very important fact (it was then implemented as a part of this thesis, see Section 7.2).
2. Usage statistics of the first version of *change ontology* were gathered (see Figures 8.1-8.2 for summary of change and goal descriptions selected by users). As it can be seen, 845 changes are options “Typos or other small bugs fix” and “Error fixing”, which were the default settings for drop-down lists. It can be concluded that the vast majority of users either forgot to complete these fields or did not consider it important. This observation resulted in changes made in the second version of *change ontology* module, where the edition form was modified to force the user to choose an option before saving changes.

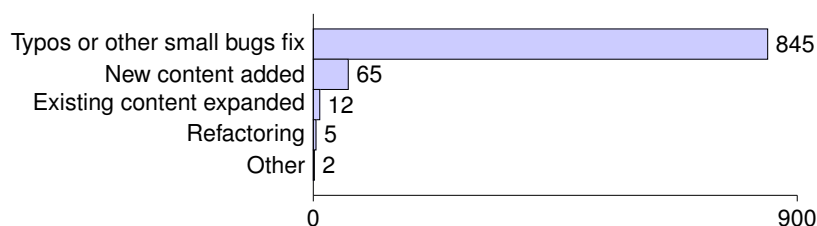


Figure 8.1: Summary of factual change descriptions selected from the first ontology version

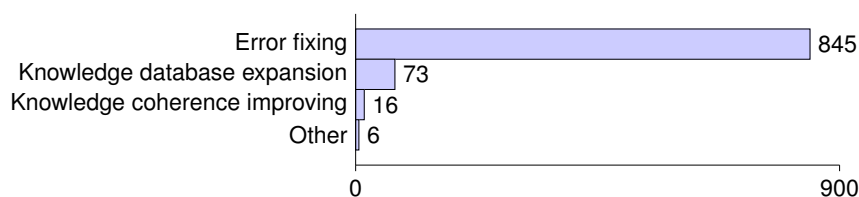


Figure 8.2: Summary of goal descriptions selected from the first ontology version

3. The process was evaluated by the participants as intuitive. It was easy to implement and every step was clear (see the third goal). Three improvements were proposed:
 - (a) Users proposed to devote the second iteration of the process to preparation of sample pages schemes. As the project tasks were very repetitive, such proper schemes significantly simplified the work. However, due to the fact that they appear later in the project, it took some time to apply them to all existing pages.

- (b) Formalization of competency questions to SPARQL queries at the end of the second iteration is too late. Although some practical issues with prepared ontology are repaired immediately, due to the fact that during the second iteration many instances were created – they have to be corrected. In the final version of the *CKE process* presented in Section 4.4, this issue is addressed by incorporation of *reasoning unit tests* definition (in a form of SPARQL queries), as a part of *competency questions* step.
- (c) It was also noted that tasks assignment may be done in a more controlled way. During the experiment, participants asked aloud each other what they were doing and in that way they determined which tasks were free. Final *CKE process* proposed in this dissertation incorporates an iteration board to address this issue.
4. Three roles were observed during the process (see the fourth goal): a leader who takes design decisions and solves disputes, a scheme designer who creates the wiki pages schemes and watches whether they are properly implemented, and a developer who performs the regular tasks. Especially the selection of the former was indicated as important due to many disputes that arose during the process. A leader may be chosen by a group, or it may be a person who is going to make decisions at some point and the group starts to comply to them, as during the experiment. Within the *CKE process* proposed in the dissertation, the leadership duties may be taken by the iteration master. It also should be noted that product owner was also available during the project – the author of the thesis takes this role.
5. Finally, selection of the task for the project was rated negatively. As the whole knowledge was already in CSPLib system, the team was responsible for copy-paste of many similar pages and for their annotation. As a result, the team performed only the tasks of the knowledge engineers, without the part of domain experts. This can be accomplished successfully by a proper web crawler, which makes the task boring for many participants.

In conclusion, the experiment was a next step in development of *change ontologies* and *semantic changelog* modules. Feedback from participants was used especially to identify important steps and roles in the *CKE process*, as well as a motivation for quality considerations in a form of reasoning unit tests.

8.1.3 Third Experiment: Pubs in Cracow

The third experiment was conducted during the winter term of 2016/2017 academic year. Fifteen students of computer science at AGH-UST, participants of “Semantic Web Technologies” course, took part in a project aimed at *CKE process* implementation. From a methodological point of view, it was de facto a repetition of the second experiment: *CKE process* was done the same way as before, but with the use of new versions of Loki and prov plugins (2016-10-26 and 2016-12-04, respectively) installed in current version of DokuWiki (2016-06-26a “Elenor of Tsort”). Also, first version of *ontologies storage* plugin was used. The goal of the

experiment was to evaluate modules in action and to gather feedback about the *CKE process*.

The project was chosen by the participants. It was decided to create a wiki about pubs located in the center of Cracow. During five iterations, lasting approximately 2 hours each (see description of previous experiment, as it was done exactly the same way), users created a wiki consisting of 1031 triples, 11 relations and 6 categories. Everything was saved on 202 wiki pages, as a result of 665 changes made. The experiment was concluded by reports prepared by participants, according to the given scheme (see Appendix C). Among the most important comments that did not appear in the previous experiment, attention should be paid to the following ones:

1. Edition form with forced selection of change and goal was rated as annoying and tiring. Nevertheless, compared to the previous version (see Figures 8.1-8.2), one can see a more varied choice of options during this experiment (see Figures 8.3-8.4). Users, forced to choose, begun to choose other options than the ones previously set as default. The need to annotate changes with these characteristics was clearly evaluated by the participants as important, especially in larger systems. However, when a lot of small changes were made, the way in which user choose elements from change ontology is not very user-friendly.

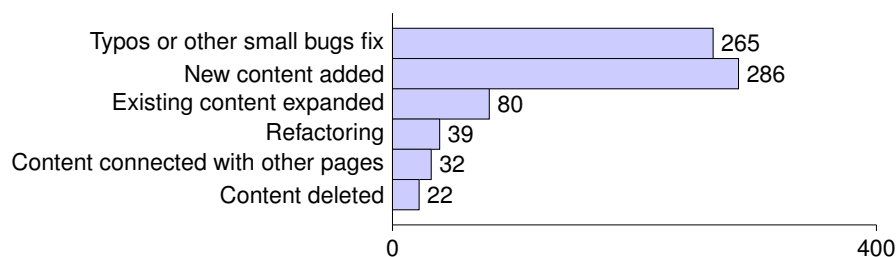


Figure 8.3: Summary of factual change descriptions selected from the second ontology version

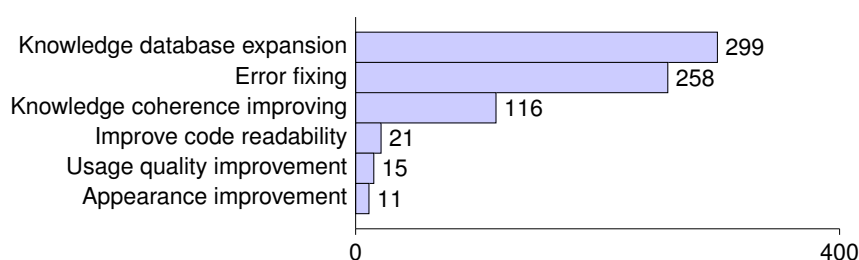


Figure 8.4: Summary of goal descriptions selected from the second ontology version

2. Edition form does not support multiple selection. Users reported that all important cases are covered by available options, but actual changes made are not as simple as these options. Often one change incorporates new content addition and content connection with other pages and so on, making it hard to select the one option that describes such a change.
3. Except the roles identified in a previous experiment (a leader, a scheme designer, a developer), participants within the third experiment have also found that one person was responsible for quality

assurance: watching all created pages and checking whether the vocabulary (categories and property names) is correctly spelled and used in a proper way. Such a task may be automated, which was done in a final version of framework proposed within the dissertation by introduction of code hint and completion mechanism that also checks the annotations and gives instant feedback of all errors found. As a result, person who was responsible for quality assurance, may work as a regular developer, contributing to a faster project evolution.

4. As in previous experiment, the lack of task assignment mechanism was found as problematic. The group itself organized an online spreadsheet, where the list of tasks (list of pubs in city centre) was placed. At the beginning of each iteration, user selected a task for herself and marked it as “in progress”. Such a group initiative was formalized in a final *CKE process* as an *iteration planning* step that uses an iteration board.

In summary, this and previous experiment gave a possibility to evaluate current version of modules. Also, implementation of a simple *CKE process* gives a possibility to observe the issues and deficiencies, and to gather feedback that was used further to provide the final version of the process proposed in this thesis.

8.1.4 Fourth Experiment: Artificial Intelligence Class

The fourth experiment was conducted during the spring term of 2016/2017 academic year. 105 computer science students, participants of “Artificial Intelligence Foundations” course (pl. “Podstawy Sztucznej Inteligencji”) at AGH-UST, created a wiki about Artificial Intelligence topics covered during the course. To motivate the students to work with the project, the activity within the wiki had impact on a final course note. Also, to motivate to not only do one’s own part, but also to take care about other wiki pages, the wiki content was used as a knowledge source for the final test. Among the main goals of the experiment there were:

1. To verify whether there are differences in difficulty levels of wiki usage between Loki (without a set of modules introduced in this dissertation) and Semantic MediaWiki.
2. To test a *CKE process* conducted by a bigger group than in previous experiments.
3. To gather feedback about gamification techniques and usability improvements that will support a *CKE process* within wiki.

Due to the specific setting, i.e. the fact that the project was done as a part of a course, the process was adjusted (it is a modification of a process presented in [129]):

1. Students were domain experts. They had a short semantic wiki training that lasted 1 hour.
2. Then, the first version of an ontology, consisting of 3 classes and 2 relations, was proposed by the author of the thesis (a project coordinator) and discussed with students.
3. Actual work was divided into seven phases that corresponded to seven laboratory classes during the course. Within each phase:

- (a) List of tasks for a phase was prepared by a project coordinator and placed online in a form of a spreadsheet. Each task had a goal (e.g. “describe the Semantic Web Stack”) and a primary source of knowledge (e.g. https://en.wikipedia.org/wiki/Semantic_Web_Stack).
 - (b) At the beginning of a phase, students form 2-person groups. Each group then selected one task, assigned themselves to it and changed the status from “to do” to “in progress”.
 - (c) Afterwards, the first iteration started. Participants had 1 week to implement the tasks. Finished tasks were marked by “done” status. It was followed by the first evaluation done by a coordinator. Each task was given a binary note based on the content of a wiki, and judged whether the given topic in the wiki is described properly. The result was indicated by a status: “accepted” or “needs correction”.
 - (d) Laboratory class took place right after the first iteration. During the class, all ambiguities could be explained.
 - (e) Then, the second iteration started. It was devoted to semantics adjustment: grouping pages into categories, making connections between pages, annotating data properties. Evaluation of prepared annotations was done at the end of this iteration by the coordinator.
 - (f) Right after the end of one phase, the new one started with a new list of tasks.
4. The whole experiment was concluded by a survey about wiki systems and the whole CKE process (see Appendix D).

In fact, there were two independent wiki instances within the experiment. The first one, Wiki A, was developed with the use of DokuWiki (version 2016-06-26a “Elenor of Tsort”) with Loki (current version from git repository) and prov (version 2017-01-25) plugins installed. The second, Wiki B, was an instance of MediaWiki (version 1.26.4) with Semantic MediaWiki module (version 2.4.6). 56 students were assigned to the Wiki A and 49 students – to Wiki B. In summary, they made 5046 changes leading to creation of 285 wiki pages. Detailed statistics are presented in Table 8.4. Final survey was filled by 34 students from Wiki A and by 32 students from Wiki B.

Table 8.4: Statistics of CKE processes within the fourth experiment

| | Changes | Wiki pages | Categories | Relations | Triples |
|---------------|---------|------------|------------|-----------|---------|
| Wiki A (Loki) | 2417 | 177 | 71 | 7 | 1113 |
| Wiki B (SMW) | 2629 | 108 | 54 | 36 | 1422 |

To address the first goal, a two-sided Mann-Whitney U test was used to check whether there were any differences between difficulty levels for Loki and SMW reported in survey. This statistical test was selected, as it is designed for determining whether two samples represent different distributions and it does not require the assumption of normal distributions. The test results² do not allow the null hypothesis of no difference

²Results consists of two values. U is a Mann-Whitney test statistic, whereas p represents the probability of the null hypothesis of

between Loki and SMW to be rejected for each of four measured wiki aspects (see Appendix D): an 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 process and feedback gathered from users led to the following observations (see the second goal):

1. A 1 hour introductory training at the beginning was not sufficient for the purposes of the project. Some participants reported that they do not know how to annotate content or do not understand the notion of concept/category. Previous experiments, apart from exactly the same training, were preceded by a seven laboratory classes, when students built their own ontologies, which was very important for their later understanding of semantic annotations.
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. However, until the very end, a strong leader or subgroup who would make any decision, had not emerged. There were no such problems in a group of 10-15 people, as in previous experiments.
3. Some users were not convinced of wiki technology. The most frequently pointed problems were: a small page editing window, no typos highlighting, fear of no internet access and data loss. None of the indicated problems were reported earlier during the project. All of them have simple solutions: editor window is scalable, typos checking is provided by a web browser, wiki auto-saves pages every few minutes.

What is interesting, 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. Lastly, motivation feedback was analysed. Unfortunately, the participants focused on incentives related to the educational process itself, primarily on the greater influence of honest work within the wiki on the high grade of the subject.

To conclude, there are no significant differences between reported difficulty for Loki and SMW. This result suggests that current state of the art semantic wikis represent roughly the same level.

8.2 Final Evaluation: Cookbook and Movies KB

Four experiments described in previous section were conducted along with the modules development as a way to verify current versions of modules and to gather some data about the *CKE agile process*. Finally, when the whole set of modules was described and the BiFröST framework was developed, the final evaluation was conducted. It was done in two parallel ways:

- the fifth experiment was conducted: during the winter term of 2017/2018 academic year, sixteen computer science students enrolled in “Semantic Web Technologies” course, took part in a *CKE agile*

no difference to be true. If p score is lower than 0.05, one can reject a null hypothesis and state that there is a difference.

process aimed at cookbook development within Loki,

- online call for evaluation was announced on Loki web page³ and sent to KE researchers at AGH-UST, as well as graduates of “Engineering of Intelligent Systems”: their task was to add some concepts (wiki pages) to movies KB in Loki.

Among the goals of this study there were:

1. To check whether all modules are compatible with each other.
2. To conduct an usability study of the whole framework.
3. To verify whether the proposed framework improves the *CKE process*.

During the experiment, the topic of the wiki was selected by participants. It was decided to create a student cookbook. The whole project lasts for 5 iterations, each about 1 hour long. In particular, during the first iteration the base ontology was created. Five main concepts were identified: meal, typeOfMeal (soup, cake, ...), ingredient, equipment and typeOfCuisine (Polish, vegetarian, ...), along with the list of object and data properties. The second iteration was aimed primarily at preparation of the first set of unit tests. Then, the third, fourth and fifth iterations were conducted according to the *CKE agile process* described in Section 4.4. They started with the review of motivating scenarios and competency questions. Then, the main implementation phase took place and the iteration was concluded by the review of unit tests results.

Both the experiment and the call for evaluation were developed with the use of BiFröST framework, i.e. in DokuWiki (version Release 2017-02-19e “Frusterick Manners”) combined with the set of plugins developed as a part of this dissertation⁴:

- Loki (version 2017-09-19),
- lokiontology (version 2017-11-19),
- prov (version 2017-09-19),
- revisionsrater (version 2017-09-19),
- wikigame (version 2017-11-14).

The experiment was also supported by a simple iteration board developed with the use of Google Spreadsheet, where participants specified the list of tasks, assigned them to themselves and marked their current status (to do, in progress, done). At the end of the experiment, as well as at the end of tasks connected with call for evaluation, all participants filled the Software Usability Measurement Inventory (SUMI)⁵ [84] (see Section 2.3.4).

As a result of evaluation, the following observations can be stated:

1. All modules of BiFröST framework smoothly interact with each other. No major flaws were noticed (see the first goal).

³See: <http://loki.ia.agh.edu.pl/wiki/evaluation2017>.

⁴Plugins are available at <http://loki.ia.agh.edu.pl/wiki/>.

⁵See: <http://sumi.uxp.ie/>.

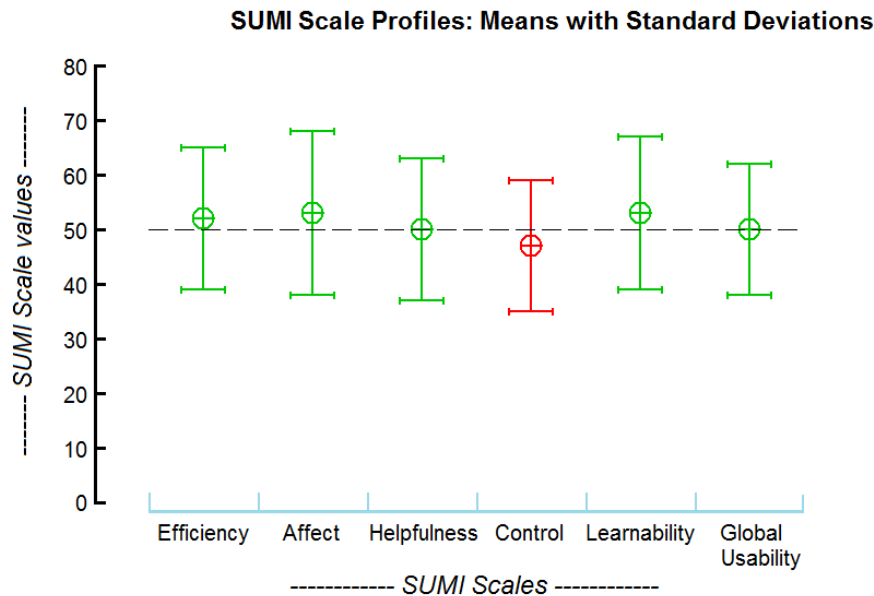


Figure 8.5: Summary of SUMI scales

2. 18 users filled out the SUMI inventory. The results are presented on Figure 8.5. The inventory indicates that users generally felt satisfied with the use of the BiFröST framework (see the second goal). The lowest score was achieved on the Control scale. Detailed analysis shows that it is related mainly to the fact that users feel safer if they use only a few familiar functions.
3. Proposed framework improved the *CKE process* (see the third goal):
 - (a) In contrast to previous experiments, conducted according to the “classic” Ontology 101 process (see Section 2.3.3), during this experiment no one reported that she 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. As a result, with the same team size, one more person was involved in knowledge engineering.
 - (b) These improvements did not prevent problems from occurring, though, as conflicts still appeared. All participants of the experiment reported the major problem: whether there should be “egg” or “eggs” concept. However, these problems could be resolved thanks to the *discussion*.
 - (c) 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. This table also proved to be a countermeasure for stagnation, because when one task was complete, one could simply pick another one from the list. What is more, if someone found missing items in the wiki, one did not have to patch them herself. There was a possibility to create new tasks, visible for all and anyone from the team could take care of them. It was proposed that more advanced iteration boards, like Trello⁶ or JIRA⁷, may be used in the future, giving more possibilities, e.g. to group tasks.

⁶See: <https://trello.com/>.

⁷See: <https://www.atlassian.com/software/jira>.

8.3 Summary

Within this dissertation, a *CKE process* and a set of modules to support the *CKE process* was proposed. Prototypical implementation called BiFröST was also developed. Based on presented modules evaluation (see “European Wiki” sections in Chapters 5-7), one can state that all of them work properly, i.e. they generate proper values for given inputs, as was depicted with the use of simple example of *European Wiki* project. It was also highlighted that all of them fulfill their requirements. Evaluation was supported by five experiments conducted during the course of the thesis. Especially the last one is important, as the whole BiFröST framework was used in it. Results indicate that all modules of the framework smoothly interact with each other. Also, usability evaluation shows that users generally felt satisfied with the use of the BiFröST framework. Finally, it can be stated that *proposed set of modules improved the CKE agile process*, which was the objective of the thesis.

Chapter 9

Concluding Remarks

9.1 Conclusions

The main objectives of the research presented in this dissertation were twofold: firstly, to describe a Collaborative Knowledge Engineering process to provide a general framework that defines roles, that should be identified in a group, and steps, that should be taken in this process; and secondly, to propose methods and tools that support the defined CKE process, leading to creation of good quality KB in reasonable time, through means that will be convenient for target users.

The analysis of state-of-the-art allowed to provide precise definition of the area of interest, exemplified with several use cases in Section 2.2. Issues and challenges that appear in the Collaborative Knowledge Engineering were then analysed in Section 4.1, leading to the distinction of six fields of CKE support. Preparation of the solutions for selected four of them was a research objective of the thesis:

- *CKE agile process* – the need for definition of robust agile process that takes care of different kinds of users' and their expectations.
- *Quality management* – the need for assuring the proper level of knowledge quality within the KB. Here, issues related to review process and automatic knowledge checking, as well as knowledge consistency and credibility, are considered. Also, handling of users' conflicts is an important issue.
- *Change management* – management of factual changes in the database, i.e. the need for a proper versioning mechanism.
- *User involvement* – as participation in CKE process is not spontaneous, various incentives should be considered to motivate people. They may be provided in a form of gamification, i.e. the usage of game elements in a serious tool. Also usability issues, i.e. taking care of compatibility with established practices, providing proper visualisation and automation capabilities and taking into account specific users' characteristics are here discussed.

Within these fields of interests, several modules were proposed that respond to both the set of require-

ments and the need for preparation of tools for domain users that are not KE specialists: *reasoning unit tests, metrics of changes, opinions and discussion, hange ontologies, graph-based changelog, gamification*. Proposed methods were supported by a prototypical implementation of modules for Loki, called BiFröST framework (see Figure 4.1). Both methods specification and prototype implementation were described in Chapters 5-7. Developed framework fulfills more CKE support requirements than any other existing semantic wiki solutions, as it is presented in Table 9.1 (see also state-of-the-art wikis comparison in Table 3.1).

Table 9.1: BiFröST framework compared to other semantic wikis with regard to the CKE requirements

| | SMW | KnowWE | OntoWiki | BiFröST |
|---|-------------|------------------------------|---------------------------|------------------------------|
| R1: Shared repository | Yes | Yes | Yes | Yes |
| R2: Create in the own way | Yes | Yes, many possibilities | Rather not | Yes, many possibilities |
| R3: Ontology as a big picture | Yes | Yes | It depends | Yes |
| R4: Bottom-up and top-down manner | Yes | Yes | Yes | Yes |
| R5: Access rights | Yes | Yes | Yes | Yes |
| R6: Approval, disagreement and discussion | Yes | No | Yes | Yes |
| R7: Consistency tracking | No | No | No | Yes |
| R8: Compatibility | RDF, SPARQL | RDF, SPARQL, domain-specific | RDF, SPARQL, vocabularies | RDF, SPARQL, BPMN, SBVR, SMW |
| R9: Simple maintenance cycle | Yes | Yes | Yes | Yes |
| R10: Experts as owners | Yes | Yes | Yes | Yes |
| R11: Adaptation to use case | Yes | Yes | Yes | Yes |
| R12: Visualization | Yes | Yes | Yes | Yes |
| R13: Credibility determination | No | No | No | Yes |
| R14: Usability | Yes, tested | Not tested | Not tested | Yes, tested |
| R15: Conflicts identification | No | No | No | Yes, limited to XTT2 rules |
| R16: Automation | No | No | No | No |
| R17: Experts in the process | Yes | Yes | Yes | Yes |

Continued on next page

Table 9.1 – Continued from previous page

| | SMW | KnowWE | OntoWiki | BiFröST |
|---|--|-------------|-------------|--------------------------|
| R18: Agile development | This requirement is related to CKE process, not specific tools | | | |
| R19: Mainstream and domain specific tools | Yes | Yes | Yes | Yes |
| R20: Automatic tests | No | Yes | No | Yes |
| R21: Versioning control | Yes, simple | Yes, simple | Yes, simple | Yes, robust |
| R22: Current state visualization | No | Yes | No | No |
| R23: Stable version at any time | No | Yes | No | No |
| R24: Sources credibility | No | No | No | Partially |
| R25: Users conflicts | Yes | Yes | Yes | Yes |
| R26: User's kinds | No | No | No | Yes, only identification |
| R27: Gamification | Yes, very simple | No | No | Yes |
| Requirements fulfilled | 17 | 17 | 14 | 23 |

The following results are considered to be the most important *original contributions* of this thesis:

1. *Formulation of distributed KE taxonomy* – analysis of concepts used in literature to describe the area of interest and formulation of their clear definitions was given in Section 2.2.
2. *Analysis of issues and challenges for systems that support CKE process* – a set of requirements that should be addressed in order to provide proper solutions for CKE was formulated in Section 2.3. It was then grouped into fields of CKE support in Chapter 4.
3. *Definition of CKE agile process* – the description of a general process for CKE, alongside with the specification roles, their responsibilities, the series of steps and some good practices were given in Section 4.4.
4. *Conceptualization of change ontology* – a formal representation of factual changes made in knowledge base during the CKE process along with their goals was given in Section 6.1.
5. *Proposal of graph-based semantic changelog* – a meta-layer that describes all changes being made in knowledge base, and is de facto a meta-base that can be queried, and processed was described in Section 6.2.
6. *Formulation of involvement metrics* – preparation of gamification module led to definition of a set of involvement metrics, useful for e.g. scrum owner in CKE agile process.
7. *Implementation of prototypical toolkit that support CKE process within wikis* – a set of modules in

a form of plugins for Loki that support CKE process in the quality management, change management and user involvement fields was described in Chapters 5-7.

This thesis also proposed *improvements* of existing methods and tools. The following ones are considered to be the most important:

1. *Definition of hierarchy for reasoning unit tests* – an idea of adopting unit tests from SE into the KE was already described in literature. In Section 5.1 the hierarchical structure is proposed as an extension to provide a possibility to group the tests based on their specificity and to execute only the subset of them.
2. *Proposal of method of summarizing characteristics of changes into synthetic metric* – a weighted average metric that provides one simple value, calculated based on many metrics, and also captures various characteristics of changes being made in knowledge base was proposed in Section 5.2.
3. *Design of gamification module for CKE process* – a selection of gamification techniques and their adaptation to CKE needs was given in Section 7.1.

The work in this dissertation was supported by the AGH UST research grants for young scientists in 2016 and 2017.

9.2 Future Work

As it was presented in Table 9.1, current version of framework fulfills the greatest number of identified requirements among available Semantic Wikis, but it still has not addressed all of them, which gives the primary motivation for the future work. Three aspects are considered:

1. Automation (R16) – within such a simple system as wiki, identification of automation possibilities is a difficult task. However, one of such cases may be the (semi-)automation of change ontology. It can be achieved in two ways. First of all, relations between specific types of change and their goals may be introduced, e.g. `:TyposOrOtherSmallBugsFixed` are almost always done in order to do `:ErrorsFixing`. Secondly, some features of current change may be evaluated before it is saved to propose the most suitable actual change and goal description, e.g. if knowledge was not changed, it is very likely that the goal was `:ApperanceImprovement`. This could be probably supported by some machine learning model.
2. Continuous integration capabilities (R22, R23) – as reasoning unit tests module was developed for BiFröST framework, there is a possibility to determine whether current KB state is stable or not, i.e. whether all tests are passed or not, and to provide a possibility to export its latest stable version.
3. Sources credibility (R24) – prepared graph-based versioning mechanism generates powerful knowledge base in terms of further analyses of the CKE process. One of possibilities is sources credibility evaluation, as presented in Section 6.2, which can be calculated in the framework in an automatic way.

Also, other use cases for semantic changelog should be considered.

Moreover, it is proposed to apply the presented CKE process in various fields. Among the considered areas there are:

- Digital humanities and cultural heritage – these disciplines are related to the processing of large amounts of knowledge, and as a result their interest in CKE techniques is growing. At the same time, most experts in these fields are not so advanced in technology, making these areas a good test for the framework proposed in this work.
- Software engineering – within this area especially the system specification task is strongly related to KB preparation. It could be done using various knowledge representations: text, pictures, UML use case diagrams, business processes, or business rules. As Loki has the ability to handle all these representations thanks to the prepared plugins, it becomes one of natural fields for BiFröST framework usage.

It is assumed that this will help the domain users in the knowledge engineering process, as well as it will result in gathering feedback from them that will help in further CKE process tuning. As it was stated, description of the process is getting better as more cases are taken into account in its preparation.

Appendix A

Code for Metrics Calculation

Implementation of *metrics of change* module requested to implement some metrics. Prepared code is presented in this appendix in Listings A.1-A.4. For more information see Section 5.2.

```
1 public function attributeRichness() {
2     include_once(DOKU_INC."/lib/plugins/loki/utl/loki_utl_special.php");
3     $utls = new LokiUtlSpecial;
4     $attributes = $utls->list_attributes();
5     $categories = $utls->list_categories();
6     $ar = count($attributes)/count($categories);
7
8     return $ar;
9 }
```

Listing A.1: Code responsible for calculation of *Attribute Richness* metric.

```
1 public function averagePopulation() {
2     include_once(DOKU_INC."/lib/plugins/loki/utl/loki_utl_special.php");
3     $utls = new LokiUtlSpecial;
4     $categories = $utls->list_categories();
5     $instanceNumber = 0;
6     foreach($categories as $category) {
7         $instance = $utls->list_category_uses($category);
8         $instanceNumber += count($instance);
9     }
10    $ap = $instanceNumber/count($categories);
11
12    return $ap;
13 }
```

Listing A.2: Code responsible for calculation of *Average Population* metric.

```

1 public function sov() {
2     include_once(DOKU_INC."/lib/plugins/loki/utl/loki_utl_special.php");
3     $utls = new LokiUtlSpecial;
4     $attributes = $utls->list_attributes();
5     $categories = $utls->list_categories();
6     $instanceNumber = 0;
7     foreach($categories as $category) {
8         $instance = $utls->list_category_uses($category);
9         $instanceNumber += count($instance);
10    }
11    $sov = count($attributes)+$instanceNumber;
12
13    return $sov;
14 }

```

Listing A.3: Code responsible for calculation of *Size of Vocabulary* metric.

```

1 public function enr() {
2     include_once(DOKU_INC."/lib/plugins/loki/utl/loki_utl_special.php");
3     $utls = new LokiUtlSpecial;
4     $relations = $utls->list_relations();
5     $attributes = $utls->list_attributes();
6     $categories = $utls->list_categories();
7     $page_with_3 = [];
8     $amount_3 = 0;
9     $amout_attr_3 = 0;
10    foreach($categories as $category) {
11        $instance = $utls->list_category_uses($category);
12        foreach($instance as $ins) {
13            $amount_3 = 0;
14            $amout_attr_3 = 0;
15            foreach($attributes as $attribute) {
16                $attr_uses = $utls->list_attribute_uses($attribute);
17                $amount_3 += count($attr_uses);
18                $amout_attr_3 += count($attr_uses);
19                foreach($attr_uses as $value) {
20                    if($value[0] == $ins) {
21                        $page_with_3[$value[0]] = 1;
22                    }
23                }
24            }
25            foreach($relations as $relation) {
26                $rel_uses = $utls->list_relation_uses($relation);
27                $amount_3 += count($rel_uses);
28                foreach($rel_uses as $value) {
29                    if($value[0] == $ins) {
30                        $page_with_3[$value[0]] = 1;
31                    }
32                }
33            }
34        }
35    }
36    $result = $amount_3/(count($page_with_3)+$amout_attr_3);
37    return $result;
38 }

```

Listing A.4: Code responsible for calculation of *Edge Node Ratio* metric.

Appendix B

Semantic Changelog Files

In this appendix *semantic changelog* files generated by prov plugin are placed (see Section 6.2 for more information). A general scheme for these files is presented on Listing B.1. Three actual files generated during the *European Wiki* project (see Sections 4.5 and 6.2) are presented on Listings B.2-B.4.

```
1 ### Head of the file:
2
3 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
4 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
5 @prefix prov: <http://www.w3.org/ns/prov#> .
6 @prefix dc: <http://purl.org/dc/elements/1.1/> .
7 @prefix change: <http://loki.ia.agh.edu.pl/wiki/change#> .
8 @prefix loki: <http://loki.ia.agh.edu.pl/wiki/ns#> .
9 @prefix lokipage: <{$wiki_address}> .
10 @prefix lokievent: <{$wiki_address}special:lokievent#> .
11 @prefix lokiuser: <{$wiki_address}user:> .
12
13 ### Page creation:
14
15 lokipage:{$page} a prov:Entity .
16 lokipage:{$page}_{$newRev} a prov:Entity ;
17     prov:specializationOf lokipage:{$page} ;
18     prov:wasGeneratedBy lokievent:created_{$page}_{$newRev} .
19 lokievent:created_{$page}_{$newRev} a prov:Activity .
20
21 ### Page edition:
22
23 lokipage:{$page}_{$newRev} a prov:Entity ;
24     prov:specializationOf lokipage:{$page} ;
25     prov:wasRevisionOf lokipage:{$page}_{$oldRev} ;
26     prov:wasGeneratedBy lokievent:edited_{$page}_{$newRev} .
```

```

27 lokievent:edited_{$page}_{$newRev} a prov:Activity .
28
29 ### Page deletion:
30
31 lokievent:deleted_{$page}_{$newRev} a prov:Activity .
32
33 ### Common to all events:
34
35 lokievent:{$event}_{$page}_{$newRev}
36   prov:wasAssociatedWith lokiuser:{$author} ;
37   dc:description "{$comment}"^^xsd:string ;
38   loki:whatWasDone change:{$change} ;      ## if option from ontology is selected
39   loki:whyWasDone [ a change:OtherGoal ;    ## if own description is provided
40                   change:goalDescription "{$goalDescription}"^^xsd:string ] ;
41   prov:used {$link1} , {$link2} ;
42   loki:classesChange "{$val}"^^xsd:integer ;
43   loki:relationsChange "{$val}"^^xsd:integer ;
44   loki:attributesChange "{$val}"^^xsd:integer ;
45   loki:statementsChange "{$val}"^^xsd:integer ;
46   loki:testsPassed [ loki:valueBefore "{$metric_val}"^^xsd:integer ;
47                   loki:valueAfter "{$metric_val}"^^xsd:integer ] ;
48   loki:attributeRichness [ loki:valueBefore "{$metric_val}"^^xsd:decimal ;
49                           loki:valueAfter "{$metric_val}"^^xsd:decimal ] ;
50   loki:averagePopulation [ loki:valueBefore "{$metric_val}"^^xsd:decimal ;
51                           loki:valueAfter "{$metric_val}"^^xsd:decimal ] ;
52   loki:sizeOfVocabulary [ loki:valueBefore "{$metric_val}"^^xsd:decimal ;
53                           loki:valueAfter "{$metric_val}"^^xsd:decimal ] ;
54   loki:edgeNodeRatio [ loki:valueBefore "{$metric_val}"^^xsd:decimal ;
55                       loki:valueAfter "{$metric_val}"^^xsd:decimal ] ;
56   loki:weightedAverage "{$weightedAverage_val}"^^xsd:decimal ;
57   loki:evaluatedByUser [ loki:note "{$note}"^^xsd:integer ;
58                       loki:evaluator lokiuser:{$evaluator} ] .

```

Listing B.1: Semantic changelog template for Loki wiki.

```

1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
3 @prefix prov: <http://www.w3.org/ns/prov#> .
4 @prefix dc: <http://purl.org/dc/elements/1.1/> .
5 @prefix loki: <http://loki.ia.agh.edu.pl/wiki/ns#> .
6 @prefix lokipage: <http://127.0.0.1/~krzysiek/european-wiki/doku.php?id=> .
7 @prefix lokievent: <http://127.0.0.1/~krzysiek/european-wiki/doku.php?id=special:
   lokievent#> .
8 @prefix lokiuser: <http://127.0.0.1/~krzysiek/european-wiki/doku.php?id=user:> .

```

```

9
10 loki:page:city:london a prov:Entity .
11
12 loki:page:city:london_1508151452 a prov:Entity ;
13     prov:specializationOf loki:page:city:london ;
14     prov:wasGeneratedBy loki:event:created_city:london_1508151452 .
15
16 loki:event:created_city:london_1508151452 a prov:Activity ;
17     prov:wasAssociatedWith loki:user:kkutt ;
18     dc:description "created" ;
19     loki:whatWasDone "New content added" ;
20     loki:whyWasDone "Knowledge database expansion" ;
21     prov:used <https://en.wikipedia.org/wiki/London> ;
22     loki:attributeRichness [ loki:valueBefore "0"; loki:valueAfter "0" ] ;
23     loki:averagePopulation [ loki:valueBefore "1"; loki:valueAfter "1" ] ;
24     loki:sizeOfVocabulary [ loki:valueBefore "1"; loki:valueAfter "1" ] ;
25     loki:edgeNodeRatio [ loki:valueBefore "0"; loki:valueAfter "0" ] .
26
27 loki:page:city:london_1508178135 a prov:Entity ;
28     prov:specializationOf loki:page:city:london ;
29     prov:wasRevisionOf loki:page:city:london_1508151452 ;
30     prov:wasGeneratedBy loki:event:edited_city:london_1508178135 .
31
32 loki:event:edited_city:london_1508178135 a prov:Activity ;
33     prov:wasAssociatedWith loki:user:yoda ;
34     dc:description "onRiver and directFlightTo added" ;
35     loki:whatWasDone "Content connected with other pages" ;
36     loki:whyWasDone "Knowledge database expansion" ;
37     prov:used loki:page:city:london_1508151452 ;
38     loki:attributeRichness [ loki:valueBefore "0.5"; loki:valueAfter "0.5" ] ;
39     loki:averagePopulation [ loki:valueBefore "2.25"; loki:valueAfter "2.25" ] ;
40     loki:sizeOfVocabulary [ loki:valueBefore "11"; loki:valueAfter "11" ] ;
41     loki:edgeNodeRatio [ loki:valueBefore "0"; loki:valueAfter "0" ] .

```

Listing B.2: Semantic changelog file generated for city:london page.

```

1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
3 @prefix prov: <http://www.w3.org/ns/prov#> .
4 @prefix dc: <http://purl.org/dc/elements/1.1/> .
5 @prefix loki: <http://loki.ia.agh.edu.pl/wiki/ns#> .
6 @prefix loki:page: <http://127.0.0.1/~krzysiek/european-wiki/doku.php?id=> .
7 @prefix loki:event: <http://127.0.0.1/~krzysiek/european-wiki/doku.php?id=special:
    loki:event#> .

```



```

8 @prefix lokiuser: <http://127.0.0.1/~krzysiek/european-wiki/doku.php?id=user:> .
9
10 lokipage:city:paris a prov:Entity .
11
12 lokipage:city:paris_1508153067 a prov:Entity ;
13     prov:specializationOf lokipage:city:paris ;
14     prov:wasGeneratedBy lokievent:created_city:paris_1508153067 .
15
16 lokievent:created_city:paris_1508153067 a prov:Activity ;
17     prov:wasAssociatedWith lokiuser:yoda ;
18     dc:description "created" ;
19     loki:whatWasDone "New content added" ;
20     loki:whyWasDone "Knowledge database expansion" ;
21     prov:used <https://en.wikipedia.org/wiki/Paris> ;
22     loki:attributeRichness [ loki:valueBefore "1"; loki:valueAfter "1" ] ;
23     loki:averagePopulation [ loki:valueBefore "1"; loki:valueAfter "1" ] ;
24     loki:sizeOfVocabulary [ loki:valueBefore "4"; loki:valueAfter "4" ] ;
25     loki:edgeNodeRatio [ loki:valueBefore "0"; loki:valueAfter "0" ] .
26
27 lokievent:edited_city:paris_1508153067 loki:evaluatedByUser [loki:note "3"; loki:
    evaluator lokiuser:kkutt].
28
29 lokipage:city:paris_1508178210 a prov:Entity ;
30     prov:specializationOf lokipage:city:paris ;
31     prov:wasRevisionOf lokipage:city:paris_1508153067 ;
32     prov:wasGeneratedBy lokievent:edited_city:paris_1508178210 .
33
34 lokievent:edited_city:paris_1508178210 a prov:Activity ;
35     prov:wasAssociatedWith lokiuser:yoda ;
36     dc:description "Obsolete categories removed" ;
37     loki:whatWasDone "Content deleted" ;
38     loki:whyWasDone "Knowledge coherence improving" ;
39     prov:used lokipage:city:paris_1508153067 ;
40     loki:attributeRichness [ loki:valueBefore "0.75"; loki:valueAfter "0.75" ] ;
41     loki:averagePopulation [ loki:valueBefore "2.25"; loki:valueAfter "2.25" ] ;
42     loki:sizeOfVocabulary [ loki:valueBefore "12"; loki:valueAfter "12" ] ;
43     loki:edgeNodeRatio [ loki:valueBefore "0"; loki:valueAfter "0" ] .
44
45 lokievent:edited_city:paris_1508178210 loki:evaluatedByUser [loki:note "5"; loki:
    evaluator lokiuser:kkutt].

```

Listing B.3: Semantic changelog file generated for city:paris page.

```

1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

```

```

2 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
3 @prefix prov: <http://www.w3.org/ns/prov#> .
4 @prefix dc: <http://purl.org/dc/elements/1.1/> .
5 @prefix loki: <http://loki.ia.agh.edu.pl/wiki/ns#> .
6 @prefix lokipage: <http://127.0.0.1/~krzysiek/european-wiki/doku.php?id=> .
7 @prefix lokievent: <http://127.0.0.1/~krzysiek/european-wiki/doku.php?id=special:
   lokievent#> .
8 @prefix lokiuser: <http://127.0.0.1/~krzysiek/european-wiki/doku.php?id=user:> .
9
10 lokipage:city:cracow a prov:Entity .
11
12 lokipage:city:cracow_1508166776 a prov:Entity ;
13     prov:specializationOf lokipage:city:cracow ;
14     prov:wasGeneratedBy lokievent:created_city:cracow_1508166776 .
15
16 lokievent:created_city:cracow_1508166776 a prov:Activity ;
17     prov:wasAssociatedWith lokiuser:kkutt ;
18     dc:description "created" ;
19     loki:whatWasDone "New content added" ;
20     loki:whyWasDone "Knowledge database expansion" ;
21     prov:used <https://en.wikipedia.org/wiki/Krak%C3%B3w> ;
22     loki:attributeRichness [ loki:valueBefore "0.5"; loki:valueAfter "0.5" ] ;
23     loki:averagePopulation [ loki:valueBefore "1.75"; loki:valueAfter "1.75" ] ;
24     loki:sizeOfVocabulary [ loki:valueBefore "9"; loki:valueAfter "9" ] ;
25     loki:edgeNodeRatio [ loki:valueBefore "0"; loki:valueAfter "0" ] .
26
27 lokipage:city:cracow_1508179476 a prov:Entity ;
28     prov:specializationOf lokipage:city:cracow ;
29     prov:wasRevisionOf lokipage:city:cracow_1508166776 ;
30     prov:wasGeneratedBy lokievent:edited_city:cracow_1508179476 .
31
32 lokievent:edited_city:cracow_1508179476 a prov:Activity ;
33     prov:wasAssociatedWith lokiuser:kkutt ;
34     dc:description "Added a little curiosity" ;
35     loki:whatWasDone "New content added" ;
36     loki:whyWasDone "Knowledge database expansion" ;
37     prov:used lokipage:city:cracow_1508166776 ;
38     loki:attributeRichness [ loki:valueBefore "1.5"; loki:valueAfter "1.5" ] ;
39     loki:averagePopulation [ loki:valueBefore "2"; loki:valueAfter "2" ] ;
40     loki:sizeOfVocabulary [ loki:valueBefore "7"; loki:valueAfter "7" ] ;
41     loki:edgeNodeRatio [ loki:valueBefore "0"; loki:valueAfter "0" ] .

```

Listing B.4: Semantic changelog file generated for city:cracow page.

Appendix C

Report Scheme Used in Experiments

In this appendix the report scheme used in the second and third experiment is presented (see Sections 8.1.2-8.1.3). During experiments, Polish version was used. English adaptation was prepared for the dissertation.

1. Describe your experience with wiki:
 - (a) wiki page text editor,
 - (b) semantic annotations,
 - (c) queries.
2. Describe your experience with page edition form:
 - (a) was it comfortable? Why?
 - (b) did the available options exhaust the issue of change description? What options should be added / removed to make the form better?
3. If you used the documentation on the <http://loki.ia.agh.edu.pl/wiki/> webpage, let me know:
 - (a) what is good?
 - (b) what is not explained clearly?
 - (c) what is missing?
4. How do you rate group work on the knowledge base? How do you rate your contribution?
 - (a) Do you think that the task generally went smoothly for the group as a whole?
 - (b) What was your role? What did you do during the project?
 - (c) What other roles did you see in the group?
 - (d) How do you rate communication in a group? Did you know what other people are doing and how your work is related to their?
5. Knowledge engineering process:
 - (a) Did you know what to do at the moment? Were you able to find it out quickly?
 - (b) Do you think the process by which we went through the project is fine? What would you change

in it?

6. Do you have any other observations or comments?

Appendix D

Survey Used in the Fourth Experiment

Survey was prepared with the use of Google Forms platform. During experiments, the polish version was used. English adaptation was prepared for the dissertation.

1. Rate the difficulty of using the wiki (1 – very easy, 5 – very difficult).
 - (a) Overall wiki usage,
 - (b) Wiki text editor,
 - (c) Semantic annotations,
 - (d) Queries.
2. Describe your experience with the wiki.
3. Do you have ideas what modules would be useful in a wiki?
4. Should gamification elements appear in a wiki? If so, which ones?
5. Are there any other ways that would motivate you to work more reliably in a wiki?
6. Did you know what to do at the moment? Were you able to find it out quickly?
7. How did you work? Have you used wiki directly or created pages elsewhere? If outside the wiki, where and why?¹
8. Should creation of an artificial intelligence wiki appear in the next edition of the course? (yes, no)
9. Do you have any other observations or comments?

¹This question has come up with the observation that not everyone was using the wiki system, even though it was a task.

Bibliography

- [1] Sunitha Abburu and G Suresh Babu. Survey on ontology construction tools. *International Journal of Scientific & Engineering Research*, pages 1748–1752, 2013.
- [2] Ben Adida and Mark Birbeck. RDFa primer. W3C working draft, W3C, June 2008. <http://www.w3.org/TR/2008/WD-xhtml-rdfa-primer-20080620/>.
- [3] Weronika T. Adrian, Szymon Bobek, Grzegorz J. Nalepa, Krzysztof Kaczor, and Krzysztof Kluza. How to reason by HearT in a semantic knowledge-based wiki. In *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2011*, pages 438–441, Boca Raton, Florida, USA, November 2011.
- [4] Weronika T. Adrian and Grzegorz J. Nalepa. Loki – presentation of logic-based semantic wiki. In Joaquin Canadas, Grzegorz J. Nalepa, and Joachim Baumeister, editors, *7th Workshop on Knowledge Engineering and Software Engineering (KESE2011) at the Conference of the Spanish Association for Artificial Intelligence (CAEPIA 2011): November 10, 2011, La Laguna (Tenerife), Spain*, page 52, 2011.
- [5] Weronika T. Adrian, Grzegorz J. Nalepa, and Antoni Ligeza. On potential usefulness of inconsistency in collaborative knowledge engineering. In *Proceedings of the 8th International Conference on Knowledge, Information and Creativity Support Systems*, 2013.
- [6] Emhimed Alatrish. Comparison some of ontology. *Journal of Management Information Systems*, 8(2):018–024, 2013.
- [7] Dean Allemang and James A. Hendler. *Semantic Web for the Working Ontologist - Effective Modeling in RDFS and OWL, Second Edition*. Morgan Kaufmann, 2011.
- [8] Sören Auer, Sebastian Dietzold, and Thomas Riechert. OntoWiki - A Tool for Social, Semantic Collaboration. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *The Semantic Web - ISWC 2006, 5th International*

- Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, volume 4273 of *Lecture Notes in Computer Science*, pages 736–749, Berlin / Heidelberg, 2006. Springer.
- [9] Sören Auer, Ali Khalili, and Darya Tarasowa. Crowd-sourced open courseware authoring with slidewiki.org. *iJET*, 8(1):62–63, 2013.
- [10] Amelia Badica, Bogdan Trawinski, and Ngoc Thanh Nguyen. Preface. In *Recent Developments in Computational Collective Intelligence*, pages V–VI. Springer, 2014.
- [11] J. Baumeister and F. Puppe. Web-based knowledge engineering using knowledge wikis. In *Proc. of the AAAI 2008 Spring Symposium on "Symbiotic Relationships between Semantic Web and Knowledge Engineering"*, pages 1–13, Stanford University, USA, 2008.
- [12] Joachim Baumeister. *Agile development of diagnostic knowledge systems*. PhD thesis, Julius Maximilians University Würzburg, Germany, 2004.
- [13] Joachim Baumeister and Grzegorz J. Nalepa. Verification of distributed knowledge in semantic knowledge wikis. In H. Chad Lane and Hans W. Guesgen, editors, *FLAIRS-22: Proceedings of the twenty-second international Florida Artificial Intelligence Research Society conference: 19–21 May 2009, Sanibel Island, Florida, USA*, pages 384–389, Menlo Park, California, 2009. FLAIRS, AAAI Press.
- [14] Joachim Baumeister and Jochen Reutelshoefer. Developing knowledge systems with continuous integration. In *I-KNOW 2011, 11th International Conference on Knowledge Management and Knowledge Technologies, Graz, Austria, September 7-9, 2011*, page 33, 2011.
- [15] Joachim Baumeister, Jochen Reutelshoefer, Volker Belli, Albrecht Striffler, Reinhard Hatko, and Markus Friedrich. Knowwe—a wiki for knowledge base development. In *Knowledge Engineering and Software Engineering (KESE8)*, 2012.
- [16] Joachim Baumeister, Jochen Reutelshoefer, and Frank Puppe. Engineering intelligent systems on the knowledge formalization continuum. *International Journal of Applied Mathematics and Computer Science (AMCS)*, 21(1), 2011.
- [17] Joachim Baumeister, Jochen Reutelshoefer, and Frank Puppe. Knowwe: A semantic wiki for knowledge engineering. *Applied Intelligence*, pages 1–22, 2011. 10.1007/s10489-010-0224-5.
- [18] Joachim Baumeister, Albrecht Striffler, Marc Brandt, and Michael Neumann. Collaborative decision support and documentation in chemical safety with knowsec. *Journal of Cheminformatics*, 8(1):21, 2016.

- [19] Ariane Ben Eli and Jeremy Hutchins. Intelligence after intellipedia: improving the push pull balance with a social networking utility. Technical report, DTIC Document, 2010.
- [20] Mike Bergman. A brief survey of ontology development methodologies, Aug 2010.
- [21] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, May 2001.
- [22] Geraldo Boz Jr, Milton P Ramos, Gilson Yukio Sato, Júlio Nievola, and Emerson C Paraiso. Noctua: A tool for knowledge acquisition and collaborative knowledge construction with a virtual catalyst. In *Computer Supported Cooperative Work in Design (CSCWD), 2011 15th International Conference on*, pages 222–229. IEEE, 2011.
- [23] Steve Bratt. Semantic web, and other technologies to watch. <https://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/>, 2007. Accessed: 19.04.2017.
- [24] François Bry, Sebastian Schaffert, Denny Vrandečić, and Klara A. Weiland. 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*, pages 329–369, 2012.
- [25] D Callahan. *The Cheating Culture: Why More Americans are Doing Wrong to Get Ahead*. Harcourt, 2004.
- [26] Ruth Cobos. Knowledge crystallisation supported by the knowcat system. In Huei-Tse Hou, editor, *New Research on Knowledge Management Technology*. InTech, 2012.
- [27] Collaborative. In Margaret Deuter, Jennifer Bradbery, and Joanna Turnbull, editors, *Oxford Advanced Learner's Dictionary (9th Edition)*. Oxford University Press, 2015. <http://www.oxfordlearnersdictionaries.com/definition/english/collaborative>.
- [28] Collective. In Margaret Deuter, Jennifer Bradbery, and Joanna Turnbull, editors, *Oxford Advanced Learner's Dictionary (9th Edition)*. Oxford University Press, 2015. http://www.oxfordlearnersdictionaries.com/definition/english/collective_1.
- [29] C. B. Comendador, P. J. Muñoz-Merino, and C. Delgado Kloos. A smartphone application for the collaborative knowledge creation based on reputation. In *2015 IEEE International Conference on Consumer Electronics (ICCE)*, pages 84–85, 2015.
- [30] Cooperative. In Margaret Deuter, Jennifer Bradbery, and Joanna Turnbull, editors, *Oxford Advanced Learner's Dictionary (9th Edition)*. Oxford University Press, 2015. http://www.oxfordlearnersdictionaries.com/definition/english/cooperative_1.

- [31] George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. *Distributed Systems: Concepts and Design (5th Edition)*. Pearson, 2011.
- [32] Kimiz Dalkir. *Knowledge Management in Theory and Practice*. MIT press, 2011.
- [33] Aldo de Moor, Pieter De Leenheer, and Robert Meersman. DOGMA-MESS: A meaning evolution support system for interorganizational ontology engineering. In *Conceptual Structures: Inspiration and Application, 14th International Conference on Conceptual Structures, ICCS 2006*, pages 189–202. Springer, 2006.
- [34] B. Decker, E. Ras, J. Rech, P. Jaubert, and M. Rieth. Wiki-based stakeholder participation in requirements engineering. *Software, IEEE*, 24(2):28–35, March 2007.
- [35] Frank Dengler and Hans-Jörg Happel. Collaborative modeling with semantic mediawiki. In *Proceedings of the 6th International Symposium on Wikis and Open Collaboration, WikiSym '10*, pages 23:1–23:2, New York, NY, USA, 2010. ACM.
- [36] Frank Dengler, Steffen Lamparter, Mark Hefke, and Andreas Abecker. Collaborative process development using semantic mediawiki. In *Fifth Conference Professional Knowledge Management: Experiences and Visions, March 25-27, 2009 in Solothurn, Switzerland*, pages 97–107, 2009.
- [37] Frank Dengler, Denny Vrandečić, and Elena Simperl. Comparison of wiki-based process modeling systems. In *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies, i-KNOW '11*, pages 30:1–30:4, New York, NY, USA, 2011. ACM.
- [38] Distributed system. In Margaret Deuter, Jennifer Bradbery, and Joanna Turnbull, editors, *Oxford Advanced Learner's Dictionary (9th Edition)*. Oxford University Press, 2015. <http://www.oxfordlearnersdictionaries.com/definition/english/distributed-system>.
- [39] Martin Doerr, Athina Kritsotaki, Vassilis Christophides, and Dimitris Kotzinos. Reference ontology for knowledge creation processes. In Anne Moen, Anders I. Mørch, and Sami Paavola, editors, *Collaborative Knowledge Creation: Practices, Tools, Concepts*, pages 31–52. SensePublishers, Rotterdam, 2012.
- [40] Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. Knowledge-based trust: Estimating the trustworthiness of web sources. *PVLDB*, 8(9):938–949, 2015.
- [41] Doina Alexandra Dumitrescu, Ruth Cobos, and Jaime Moreno-Llorena. A multi-agent system for extracting and analysing users' interaction in a collaborative knowledge management system. In

- Longbing Cao, editor, *Data Mining and Multi-agent Integration*, pages 93–102. Springer US, Boston, MA, 2009.
- [42] Astrid Duque-Ramos, Jesualdo Tomás Fernández-Breis, Robert Stevens, and Nathalie Aussenac-Gilles. Oquare: A square-based approach for evaluating the quality of ontologies. *Journal of Research and Practice in Information Technology*, 43(2):159–176, 2011.
- [43] J. Durkin. *Expert Systems: Design and Development*. Macmillan, 1994.
- [44] Romy Elze, Tom-Michael Hesse, and Michael Martin. Dispedia.de – a linked information system for rare diseases. In Andreas Holzinger and Klaus-Martin Simonc, editors, *Information Quality in e-Health: 7th Conference of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society*, pages 691–701. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [45] Timofey Ermilov, Norman Heino, Sebastian Tramp, and Sören Auer. Ontowiki mobile – knowledge management in your pocket. In Grigoris Antoniou, Marko Grobelnik, Elena Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Pan, editors, *The Semantic Web: Research and Applications: 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29-June 2, 2011, Proceedings, Part I*, pages 185–199. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [46] Amjad Farooq, Syed Ahsan, and Abad Shah. Web-ontology design quality metrics. *Journal of American Science*, 6(11):52–58, 2010.
- [47] Edward A. Feigenbaum and Pamela McCorduck. *The fifth generation - artificial intelligence and Japan's computer challenge to the world*. Addison-Wesley, 1983.
- [48] Dieter Fensel. *Tools*, pages 47–88. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
- [49] Mariano Fernández-López, Asunción Gómez-Pérez, and Natalia Juristo. Methontology: from ontological art towards ontological engineering. In *Proceedings of the Ontological Engineering AAAI-97 Spring Symposium Series*. American Association for Artificial Intelligence, 1997.
- [50] Giorgos Flouris, Dimitris Manakanatas, Haridimos Kondylakis, Dimitris Plexousakis, and Grigoris Antoniou. Ontology change: classification and survey. *Knowledge Eng. Review*, 23(2):117–152, 2008.
- [51] Martina Freiberg and Joachim Baumeister. A survey on usability evaluation techniques and an analysis of their actual application. Technical report, Institute of Computer Science, University of Würzburg, Germany, 2008.

- [52] Martina Freiberg, Joachim Baumeister, and Frank Puppe. The usability stack: Reconsidering usability criteria regarding knowledge-based systems. In *LWA 2009: Workshop-Woche: Lernen, Wissen, Adaptivität, Darmstadt, 21.-23. September 2009*, pages 1–9, 2009.
- [53] Martina Freiberg and Frank Puppe. Template-based extensible prototyping for creativity- and usability-oriented knowledge systems development. In Joaquin Canadas, Grzegorz J. Nalepa, and Joachim Baumeister, editors, *8th Workshop on Knowledge Engineering and Software Engineering (KESE2012) at the at the biennial European Conference on Artificial Intelligence (ECAI 2012): August 28, 2012, Montpellier, France*, pages 54–57, 2012.
- [54] Martina Freiberg, Albrecht Striffler, and Frank Puppe. Extensible prototyping for pragmatic engineering of knowledge-based systems. *Expert Syst. Appl.*, 39(11):10177–10190, 2012.
- [55] Philipp Frischmuth, Michael Martin, Sebastian Tramp, Thomas Riechert, and Sören Auer. Ontowiki—an authoring, publication and visualization interface for the data web. *Semantic Web*, 6(3):215–240, 2015.
- [56] Sebastian Furth and Joachim Baumeister. An ontology debugger for the semantic wiki KnowWE (tool presentation). In Grzegorz J. Nalepa and Joachim Baumeister, editors, *Proceedings of 10th Workshop on Knowledge Engineering and Software Engineering (KESE10) co-located with 21st European Conference on Artificial Intelligence (ECAI 2014), Prague, Czech Republic, August 19 2014*, 2014.
- [57] Sebastian Furth and Joachim Baumeister. TELESUP - textual self-learning support systems. In *Proceedings of the 16th LWA Workshops: KDML, IR and FGWM*, pages 277–286, 2014.
- [58] Yolanda Gil and Simon Miles. PROV model primer. W3C working group note, W3C, April 2013. <https://www.w3.org/TR/prov-primer/>.
- [59] Avelino J Gonzalez and Douglas D Dankel. *The engineering of knowledge-based systems: theory and practice*. Prentice-Hall, Inc., 1993.
- [60] Anna Goy, Diego Magro, Giovanna Petrone, Claudia Picardi, and Marino Segnan. Shared and personal views on collaborative semantic tables. In Pascal Molli, John G. Breslin, and Maria-Esther Vidal, editors, *Semantic Web Collaborative Spaces*, pages 13–32. Springer International Publishing, 2016.
- [61] Michael Gruninger and Mark S Fox. The design and evaluation of ontologies for enterprise engineering. In *Workshop on Implemented Ontologies, European Workshop on Artificial Intelligence, Amsterdam, The Netherlands, 1994*.

- [62] Nicola Guarino. Formal ontology and information systems. In *Proceedings of the First International Conference on Formal Ontologies in Information Systems*, pages 3–15, 1998.
- [63] Giovanni Guida and Carlo Tasso. *Design and development of knowledge-based systems: from life cycle to methodology*. John Wiley & Sons, Inc., 1995.
- [64] I Halatchliyski. Social network analysis of collaborative knowledge creation in wikipedia. In *Connecting Computer-Supported Collaborative Learning to Policy and Practice: CSCL 2011 Conf. Proc. - Community Events Proceedings, 9th International Computer-Supported Collaborative Learning Conf.*, pages 1254–1258, 2011.
- [65] Garrett Hardin. The tragedy of the commons. *Science*, 162:1243–1248, 1968.
- [66] Steve Harris and Andy Seaborne. SPARQL 1.1 query language. W3C recommendation, W3C, March 2013. <https://www.w3.org/TR/sparql11-query/>.
- [67] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011.
- [68] Norman Heino, Sebastian Dietzold, Michael Martin, and Sören Auer. Developing semantic web applications with the ontowiki framework. In Tassilo Pellegrini, Sören Auer, Klaus Tochtermann, and Sebastian Schaffert, editors, *Networked Knowledge - Networked Media*, pages 61–77. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [69] Martin Hepp. Hypertwitter: Collaborative knowledge engineering via twitter messages. In Philipp Cimiano and H. Sofia Pinto, editors, *Knowledge Engineering and Management by the Masses: 17th International Conference, EKAW 2010*, pages 451–461. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [70] Cindy E. Hmelo-Silver and Howard S. Barrows. Facilitating collaborative knowledge building. *Cognition and Instruction*, 26(1):48–94, 2008.
- [71] Rinke Hoekstra and Paul T. Groth. Prov-o-viz - understanding the role of activities in provenance. In *Provenance and Annotation of Data and Processes - 5th International Provenance and Annotation Workshop, IPAW 2014*, pages 215–220, 2014.
- [72] Clyde W. Holsapple and Kshiti D. Joshi. A collaborative approach to ontology design. *Commun. ACM*, 45(2):42–47, 2002.
- [73] Trung Dong Huynh and Luc Moreau. Provstore: A public provenance repository. In *Provenance and Annotation of Data and Processes - 5th International Provenance and Annotation Workshop, IPAW 2014*, pages 275–277, 2014.

- [74] Michele H. Jackson. Collaboration and cooperation. In Klaus Bruhn Jensen, Robert T. Craig, Jefferson D. Pooley, and Eric W. Rothenbuhler, editors, *The International Encyclopedia of Communication Theory and Philosophy*. John Wiley & Sons, Inc., 2016.
- [75] Maria Jakubik. Experiencing collaborative knowledge creation processes. *The learning organization*, 15(1):5–25, 2008.
- [76] Michael John and Ronald Melster. Knowledge networks – managing collaborative knowledge spaces. In Grigori Melnik and Harald Holz, editors, *Advances in Learning Software Organizations*, volume 3096 of *Lecture Notes in Computer Science*, pages 165–171. Springer Berlin Heidelberg, 2004.
- [77] Dean Jones, Trevor Bench-Capon, and Pepijn Visser. Methodologies for ontology development. In *Proceedings of IT&KNOWS Conference of the 15th IFIP World Computer Congress*, 1998.
- [78] Luiz Carlos L. Silva Jr., Marcos R. S. Borges, and Paulo Victor R. de Carvalho. 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, CSCWD 2009*, pages 167–172, 2009.
- [79] W. Junjie, M. Qian, M. Dongxia, and L. Su. Research for collaborative knowledge management based on semantic wiki technology. In *2010 Second International Workshop on Education Technology and Computer Science*, volume 3, pages 464–467, 2010.
- [80] Georgios Kahrmanis, Nikolaos M. Avouris, and Vassilis Komis. Interaction analysis as a tool for supporting collaboration: An overview. In *Technology-Enhanced Systems and Tools for Collaborative Learning Scaffolding*, pages 93–114. Springer, 2011.
- [81] Kensaku Kawamoto, Yasuhiko Kitamura, and Yuri A. Tijerino. Kawawiki: A semantic wiki based on RDF templates. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology - Workshops, Hong Kong, China, 18-22 December 2006*, pages 425–432, 2006.
- [82] Ali Khalili, Sören Auer, Darya Tarasowa, and Ivan Ermilov. Slidewiki: Elicitation and sharing of corporate knowledge using presentations. In *Knowledge Engineering and Knowledge Management - 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings*, pages 302–316, 2012.
- [83] Jurek Kirakowski. The software usability measurement inventory: background and usage. In Patrick W. Jordan, B. Thomas, Ian Lyall McClelland, and Bernard Weerdmeester, editors, *Usability evaluation in industry*, pages 169–178. Taylor & Francis, Inc., London, 1996.

- [84] Jurek Kirakowski and Mary Corbett. Sumi: The software usability measurement inventory. *British journal of educational technology*, 24(3):210–212, 1993.
- [85] Troy C. Kohwalter, Thiago Nazareth de Oliveira, Juliana Freire, Esteban Clua, and Leonardo Murta. Prov viewer: A graph-based visualization tool for interactive exploration of provenance data. In *Provenance and Annotation of Data and Processes - 6th International Provenance and Annotation Workshop, IPAW 2016*, pages 71–82, 2016.
- [86] Markus Krötzsch and Denny Vrandečić. Semantic mediawiki. In *Foundations for the Web of Information and Services - A Review of 20 Years of Semantic Web Research.*, pages 311–326, 2011.
- [87] Markus Krötzsch, Denny Vrandečić, and Max Völkel. Semantic mediawiki. In *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, pages 935–942, 2006.
- [88] Markus Krötzsch, Denny Vrandečić, Max Völkel, Heiko Haller, and Rudi Studer. Semantic wikipedia. *Web Semantics*, 5:251–261, 2007.
- [89] Steven Kuhn. Prisoner’s dilemma. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2017 edition, 2017.
- [90] Minna Lakkala, Auli Toom, Liisa Ilomäki, and Hanni Muukkonen. Re-designing university courses to support collaborative knowledge creation practices. *Australasian Journal of Educational Technology*, 31(5):521–536, 2015.
- [91] Bo Leuf and Ward Cunningham. *The Wiki way: quick collaboration on the Web*. Addison-Wesley Professional, 2001.
- [92] Peng Liang, Paris Avgeriou, and Viktor Clerc. Requirements reasoning for distributed requirements analysis using semantic wiki. In *2009 Fourth IEEE International Conference on Global Software Engineering (ICGSE 2009), 13-16 July 2009, Limerick, Ireland*, pages 388–393. IEEE, July 2009.
- [93] Jay Liebowitz, editor. *The Handbook of Applied Expert Systems*. CRC Press, Boca Raton, 1998.
- [94] Helena Lindgren and Peter Winnberg. Evaluation of a semantic web application for collaborative knowledge building in the dementia domain. In *Electronic Healthcare - Third International Conference, eHealth 2010, Casablanca, Morocco, December 13-15, 2010*, pages 62–69, 2010.
- [95] Fang Liu. Usability evaluation on websites. In *Computer-Aided Industrial Design and Conceptual Design. CAID/CD 2008. 9th International Conference on*, pages 141–144. IEEE, 2008.

- [96] Valeria Ludovici, Fabrizio Smith, and Francesco Taglino. Collaborative ontology building in virtual innovation factories. In *2013 International Conference on Collaboration Technologies and Systems, CTS 2013, San Diego, CA, USA, May 20-24, 2013*, pages 443–450, 2013.
- [97] Ann Majchrzak, Christian Wagner, and Dave Yates. Corporate wiki users: results of a survey. In *Proceedings of the 2006 international symposium on Wikis*, pages 99–104, 2006.
- [98] Abdelhamid Malki and Sidi Mohamed Benslimane. Building semantic mashup. In *ICWIT*, pages 40–49, 2012.
- [99] Deborah L. McGuinness. Ontologies for the modern age. ISWC2017 Keynote Speech, October 2017.
- [100] Michael Meder, Till Plumbaum, Ernesto William De Luca, and Sahin Albayrak. Gamification: A semantic approach for user driven knowledge conservation. In *Report of the symposium "Lernen, Wissen, Adaptivität 2011" of the GI special interest groups KDML, IR and WM, LWA 2011, Magdeburg, 28.-30.September 2011*, pages 265–268, 2011.
- [101] Luc Moreau and Paul T. Groth. *Provenance: An Introduction to PROV*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers, 2013.
- [102] G.J. Nalepa, M. Slazynski, K. Kutt, E. Kucharska, and A. Luszpaj. Unifying business concepts for smes with prosecco ontology. In *Computer Science and Information Systems (FedCSIS), 2015 Federated Conference on*, pages 1321–1326, Sept 2015.
- [103] Grzegorz J. Nalepa. PIWiki – a generic semantic wiki architecture. In Ngoc Thanh Nguyen, Ryszard Kowalczyk, and Shyi-Ming Chen, editors, *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems, First International Conference, ICCCI 2009, Wroclaw, Poland, October 5-7, 2009. Proceedings*, volume 5796 of *Lecture Notes in Computer Science*, pages 345–356. Springer, 2009.
- [104] Grzegorz J. Nalepa. Architecture of the HearT hybrid rule engine. In Leszek Rutkowski and [et al.], editors, *Artificial Intelligence and Soft Computing: 10th International Conference, ICAISC 2010: Zakopane, Poland, June 13–17, 2010, Pt. II*, volume 6114 of *Lecture Notes in Artificial Intelligence*, pages 598–605. Springer, 2010.
- [105] Grzegorz J. Nalepa. Collective knowledge engineering with semantic wikis. *Journal of Universal Computer Science*, 16(7):1006–1023, 2010.
- [106] Grzegorz J. Nalepa. Loki – semantic wiki with logical knowledge representation. In Ngoc Thanh Nguyen, editor, *Transactions on Computational Collective Intelligence III*, volume 6560 of *Lecture Notes in Computer Science*, pages 96–114. Springer, 2011.

- [107] Grzegorz J. Nalepa. *Modeling with Rules Using Semantic Knowledge Engineering*, volume 130 of *Intelligent Systems Reference Library*. Springer, 2018.
- [108] Grzegorz J. Nalepa and Szymon Bobek. Embedding the HeaRT rule engine into a semantic wiki. In Radoslaw Katarzyniak, Tzu-Fu Chiu, Chao-Fu Hong, and Ngoc Nguyen, editors, *Semantic Methods for Knowledge Management and Communication*, volume 381 of *Studies in Computational Intelligence*, pages 265–275. Springer Berlin, Heidelberg, 2011.
- [109] Grzegorz J. Nalepa, Krzysztof Kluza, and Urszula Ciaputa. 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 ETFA 2012, Kraków, Poland, 28 September 2012*, 2012.
- [110] Grzegorz J. Nalepa, Krzysztof Kluza, and Krzysztof Kaczor. Sbvwiki a web-based tool for authoring of business rules. In Leszek Rutkowski and [et al.], editors, *Artificial Intelligence and Soft Computing: 14th International Conference, ICAISC 2015: Zakopane, Poland, Lecture Notes in Artificial Intelligence*, pages 703–713. Springer, 2015.
- [111] Grzegorz J. Nalepa, Antoni Ligęza, and Krzysztof Kaczor. Formalization and modeling of rules using the XTT2 method. *International Journal on Artificial Intelligence Tools*, 20(6):1107–1125, 2011.
- [112] Akila Narayanan. *Gamification for Employee Engagement*. Packt Publishing Ltd, 2014.
- [113] Michael Negnevitsky. *Artificial Intelligence. A Guide to Intelligent Systems*. Addison-Wesley, Harlow, England; London; New York, 2002. ISBN 0-201-71159-1.
- [114] Piyush Nimbalkar. Semantic interpretation of structured log files. Master’s thesis, University of Maryland, Baltimore County, August 2015.
- [115] Piyush Nimbalkar, Varish Mulwad, Nikhil Puranik, Anupam Joshi, and Tim Finin. Semantic interpretation of structured log files. In *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*, pages 549–555, July 2016.
- [116] Natalya F. Noy and Deborah L. McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford University, 2001.
- [117] Natalya Fridman Noy, Abhita Chugh, and Harith Alani. The CKC challenge: Exploring tools for collaborative knowledge construction. *IEEE Intelligent Systems*, 23(1):64–68, 2008.
- [118] Natalya Fridman Noy, Abhita Chugh, William Liu, and Mark A. Musen. A framework for ontology evolution in collaborative environments. In *The Semantic Web - ISWC 2006*, pages 544–558, 2006.

- [119] Jukka K. Nurminen, Olli Karonen, and Kimmo Hätönen. What makes expert systems survive over 10 years - empirical evaluation of several engineering applications. *Expert Syst. Appl.*, 24(2):199–211, 2003.
- [120] Ontology metrics. https://www.bioontology.org/wiki/index.php/Ontology_Metrics. Accessed: 11.03.2017.
- [121] Eyal Oren, Renaud Delbru, Knud Möller, Max Völkel, and Siegfried Handschuh. Annotation and navigation in semantic wikis. In Max Völkel and Sebastian Schaffert, editors, *SemWiki*, volume 206 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.
- [122] Other uses for wiki software? <https://ask.slashdot.org/story/06/01/21/1958244/other-uses-for-wiki-software>, 2006. Accessed: 04.04.2017.
- [123] S. Otto. Caucasian spiders. a faunistic database on the spiders of the caucasus. version 1.4.3, 2015. <http://caucasus-spiders.info/>.
- [124] Emerson Cabrera Paraiso, Geraldo Boz Jr, Milton Pires Ramos, Gilson Y Sato, and Cesar Tacla. Improving Knowledge Acquisition in a Collaborative Knowledge Construction Tool with a Virtual Catalyst. *Computing and Informatics*, 35(4):914–940, 2016.
- [125] Erika Shehan Poole and Jonathan Grudin. A taxonomy of wiki genres in enterprise settings. In *Proceedings of the 6th International Symposium on Wikis and Open Collaboration, WikiSym '10*, pages 14:1–14:4, New York, NY, USA, 2010. ACM.
- [126] Pattarawan Prasarnphanich and Christian Wagner. The role of wiki technology and altruism in collaborative knowledge creation. *Journal of Computer Information Systems*, 49(4):33–41, 2009.
- [127] Carolyn C Preston and Andrew M Colman. Optimal number of response categories in rating scales: reliability, validity, discriminating power, and respondent preferences. *Acta Psychologica*, 104(1):1–15, 2000.
- [128] Eric S. Raymond. *The cathedral and the bazaar - musings on Linux and Open Source by an accidental revolutionary*. O'Reilly, 1999.
- [129] Jochen Reutelshoefer, Florian Lemmerich, Joachim Baumeister, Jorit Wintjes, and Lorenz Haas. Taking OWL to athens. In Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, editors, *The Semantic Web: Research and Applications*, volume 6088 of *Lecture Notes in Computer Science*, pages 333–347. Springer, 2010.

- [130] Debbie Richards. Ad-hoc and personal ontologies: A prototyping approach to ontology engineering. In *Advances in Knowledge Acquisition and Management, Pacific Rim Knowledge Acquisition Workshop, PKAW 2006*, pages 13–24, 2006.
- [131] Debbie Richards. Collaborative knowledge engineering: socialising expert systems. In *2007 11th International Conference on Computer Supported Cooperative Work in Design*, pages 635–640. IEEE, 2007.
- [132] Debbie Richards. A social software/web 2.0 approach to collaborative knowledge engineering. *Information Sciences*, 179(15):2515 – 2523, 2009.
- [133] Christoph Richter, Ekaterina Simonenko, Tsuyoshi Sugibuchi, Nicolas Spyrtos, Frantisek Babic, Jozef Wagner, Jan Paralic, Michal Racek, Crina Damşa, and Vassilis Christophides. Mirroring tools for collaborative analysis and reflection. In Anne Moen, Anders I. Mørch, and Sami Paavola, editors, *Collaborative Knowledge Creation: Practices, Tools, Concepts*, pages 117–140. SensePublishers, Rotterdam, 2012.
- [134] Thomas Riechert, Ulf Morgenstern, Sören Auer, Sebastian Tramp, and Michael Martin. Knowledge engineering for historians on the example of the catalogus professorum lipsiensis. In *International Semantic Web Conference*, pages 225–240. Springer, 2010.
- [135] Jeremy Roschelle and Stephanie D. Teasley. The construction of shared knowledge in collaborative problem solving. In Claire O’Malley, editor, *Computer Supported Collaborative Learning*, pages 69–97. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.
- [136] Kenneth S Rubin. *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley, 2012.
- [137] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 3rd edition, 2009.
- [138] Sebastian Schaffert. Ikewiki: A semantic wiki for collaborative knowledge management. In *WETICE ’06: Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 388–396, Washington, DC, USA, 2006. IEEE Computer Society.
- [139] Guus Schreiber and Yves Raimond. RDF 1.1 primer. W3C working group note, W3C, June 2014. <https://www.w3.org/TR/rdf11-primer/>.
- [140] Leslie F. Sikos. *Mastering Structured Data on the Semantic Web*. Apress, 2015.

- [141] Elena Simperl and Markus Luczak-Rösch. Collaborative ontology engineering: a survey. *Knowledge Eng. Review*, 29(1):101–131, 2014.
- [142] Aarti Singh and Poonam Anand. State of art in ontology development tools. *International Journal of Advances in Computer Science and Technology*, 2(7):96–101, 2013.
- [143] Katharina Siorpaes and Martin Hepp. Games with a purpose for the semantic web. *IEEE Intelligent Systems*, 23(3):50–60, 2008.
- [144] Katharina Siorpaes and Martin Hepp. Ontogame: Weaving the semantic web by online games. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, *The Semantic Web: Research and Applications: 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008 Proceedings*, pages 751–766. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [145] Hala Skaf-Molli, G r me Canals, and Pascal Molli. DSMW: distributed semantic mediawiki. In Lora Aroyo, Grigoris Antoniou, Eero Hyv nen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, editors, *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010*, pages 426–430, 2010.
- [146] Burrhus F Skinner. Are theories of learning necessary? *Psychological Review*, 57(4):193–216, 1950.
- [147] Ian Sommerville. *Software Engineering*. International Computer Science. Pearson Education Limited, 7th edition, 2004.
- [148] Gerry Stahl. *Group Cognition: Computer Support for Building Collaborative Knowledge*. The MIT Press, 2006.
- [149] Gerry Stahl. Traversing planes of learning. *International Journal of Computer-Supported Collaborative Learning*, 7(4):467–473, 2012.
- [150] Rudi Studer, V Richard Benjamins, and Dieter Fensel. Knowledge engineering: principles and methods. *Data & knowledge engineering*, 25(1-2):161–197, 1998.
- [151] York Sure, Michael Erdmann, J rgen Angele, Steffen Staab, Rudi Studer, and Dirk Wenke. Ontoedit: Collaborative ontology development for the semantic web. In *International Semantic Web Conference*, pages 221–235. Springer, 2002.
- [152] I. Suriarachchi, Q. Zhou, and B. Plale. Komadu: A capture and visualization system for scientific data provenance. *Journal of Open Research Software*, 3(1):e4, 2015.

- [153] Jeff Sutherland and JJ Sutherland. *Scrum: the art of doing twice the work in half the time*. Crown Business, 2014.
- [154] Tadeusz M Szuba. *Computational collective intelligence*. John Wiley & Sons, Inc., 2001.
- [155] Samir Tartir, I Budak Arpinar, Michael Moore, Amit P Sheth, and Boanerges Aleman-Meza. Ontoqa: Metric-based ontology quality analysis. In *Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources (KADASH)*, pages 45–53, 2005.
- [156] Gabriel Moser Torres. *Collaborative Knowledge Engineering*. LAP Lambert Academic Publishing, 2013.
- [157] Sebastian Tramp, Norman Heino, Sören Auer, and Philipp Frischmuth. Rdfauthor: Employing rdfa for collaborative knowledge engineering. In *Knowledge Engineering and Management by the Masses - 17th International Conference, EKAW 2010, Lisbon, Portugal, October 11-15, 2010. Proceedings*, pages 90–104, 2010.
- [158] Tania Tudorache, Natalya Fridman Noy, Samson W. Tu, and Mark A. Musen. Supporting collaborative ontology development in protégé. In *The Semantic Web - ISWC 2008, 7th International Semantic Web Conference*, pages 17–32, 2008.
- [159] Tania Tudorache, Csongor I Nyulas, Natalya F Noy, and Mark A Musen. Using semantic web in icd-11: three years down the road. In *International Semantic Web Conference*, pages 195–211. Springer, 2013.
- [160] Michael Uschold and Martin King. Towards a methodology for building ontologies. In *IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada, 1995*.
- [161] Mike Uschold and Michael Gruninger. Ontologies: Principles, methods and applications. *The knowledge engineering review*, 11(02):93–136, 1996.
- [162] Sharon Villines. Collaborative, collective, cooperative. <http://www.sociocracy.info/collaborative-collective-cooperative/>, September 2014.
- [163] Denny Vrandečić. Ontology evaluation. In *Handbook on Ontologies*, pages 293–313. Springer, 2009.
- [164] Denny Vrandečić and Aldo Gangemi. Unit tests for ontologies. In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, pages 1012–1020, 2006.
- [165] Denny Vrandečić, Helena Sofia Pinto, Christoph Tempich, and York Sure. The DILIGENT knowledge processes. *J. Knowledge Management*, 9(5):85–96, 2005.

- [166] Christian Wagner. Breaking the knowledge acquisition bottleneck through conversational knowledge management. *Inf. Resour. Manage. J.*, 19(1):70–83, January 2006.
- [167] Jörg Waitelonis, Nadine Ludwig, Magnus Knuth, and Harald Sack. Whoknows? evaluating linked data heuristics with a quiz that cleans up dbpedia. *Interact. Techn. Smart Edu.*, 8(4):236–248, 2011.
- [168] Yinglin Wang and Ming Chen. A collaborative knowledge production model for knowledge management in complex engineering domains. In *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics: The Hague, Netherlands, 10-13 October 2004*, pages 5050–5055, 2004.
- [169] D.A. Waterman. *A Guide to Expert Systems*. Teknowledge series in knowledge engineering. Addison-Wesley, 1986.
- [170] Kevin Werbach and Dan Hunter. *For the win: How game thinking can revolutionize your business*. Wharton Digital Press, 2012.
- [171] Richard Wettel and Michele Lanza. Codecity: 3d visualization of large-scale software. In *30th International Conference on Software Engineering (ICSE 2008)*, pages 921–922, 2008.
- [172] Wiki design principles. <http://wiki.c2.com/?WikiDesignPrinciples>, 2014. Accessed: 02.04.2017.
- [173] Gregory Todd Williams. SPARQL 1.1 service description. W3C recommendation, W3C, March 2013. <https://www.w3.org/TR/sparql11-service-description/>.
- [174] David Wood, Marsha Zaidman, and Luke Ruth. *Linked Data: Structured data on the Web*. Manning Publications Co., 2014.
- [175] Haoxiang Xia, Zhaoguo Xuan, Taketoshi Yoshida, and Zhongtuo Wang. Toward patterns for collaborative knowledge creation. In *Knowledge Science, Engineering and Management, Second International Conference, KSEM 2007, Melbourne, Australia, November 28-30, 2007, Proceedings*, pages 581–586, 2007.
- [176] Haining Yao, Anthony Mark Orme, and Letha Etzkorn. Cohesion metrics for ontology design and application. *Journal of Computer science*, 1(1):107–113, 2005.
- [177] Liyang Yu. *A Developer’s Guide to the Semantic Web*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.
- [178] Ivan Zaikin and Anatoly Tuzovsky. Owl2vcs: Tools for distributed ontology development. In *Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED 2013)*

- co-located with 10th Extended Semantic Web Conference (ESWC 2013), Montpellier, France, May 26-27, 2013.*, 2013.
- [179] Stefan Zander, Christian Swertz, Elena Verdú, María Jesús Verdú Pérez, and Peter Henning. A semantic mediawiki-based approach for the collaborative development of pedagogically meaningful learning content annotations. In Pascal Molli, John G. Breslin, and Maria-Esther Vidal, editors, *Semantic Web Collaborative Spaces - Second International Workshop, SWCS 2013, Third International Workshop, SWCS 2014*, pages 73–111, 2014.
- [180] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. Quality assessment for linked data: A survey. *Semantic Web*, 7(1):63–93, 2016.
- [181] Andreas Zeller. Yesterday, my program worked. today, it does not. why? In *Software Engineering - ESEC/FSE'99, 7th European Software Engineering Conference*, pages 253–267, 1999.
- [182] Hongyu Zhang, Yuan-Fang Li, and Hee Beng Kuan Tan. Measuring design complexity of semantic web ontologies. *Journal of Systems and Software*, 83(5):803–814, 2010.
- [183] Zhang Zhang, Jian Sang, Lina Ma, Gang Wu, Hao Wu, Dawei Huang, Dong Zou, Siqi Liu, Ang Li, Lili Hao, Ming Tian, Chao Xu, Xumin Wang, Jiayan Wu, Jing-Fa Xiao, Lin Dai, Ling-Ling Chen, Songnian Hu, and Jun Yu. Ricewiki: a wiki-based database for community curation of rice genes. *Nucleic Acids Research*, 42(Database-Issue):1222–1228, 2014.
- [184] Hasti Ziaimatin, Tudor Groza, Tania Tudorache, and Jane Hunter. Modelling expertise at different levels of granularity using semantic similarity measures in the context of collaborative knowledge-curation platforms. *J. Intell. Inf. Syst.*, 47(3):469–490, 2016.