Implement Agile Development Process with Business Rules on the Semantic Web

by

Hilde A.J. Van Gysel

to be defended publicly on Wednesday November 29, 2017 at 14:00.

Student number: 838691113

BUSINESS RULES

Implement Agile Development Process with Business Rules on the Semantic Web

by

Hilde A.J. Van Gysel

Open University of the Netherlands, faculty of Management, Science and Technology
Master Software Engineering

to be defended publicly on Wednesday November 29, 2017 at 14:00.

Student number:	838691113		
Course code:	IMA0002		
Thesis committee:	Chairman	Prof. dr. ir. S.M.M. Joosten (chairman)	Open University
	Supervisor	Dr L.W. Rutledge (supervisor)	Open University

CONTENTS

Acknowledgment

My hunger for knowledge and learning could not be fulfilled by the education system in Belgium. After searching for several months, I acquainted with the Open University of the Netherlands and saw the possibility to combine full time work, the care of the family and studying. Astrid Nys gave me a lot of information and tips.

What started with 'I want to know a bit more of' ends several years later in a bachelor degree. But the hunger for more knowledge and learning was not yet met. Thus, I continued studying the mandatory courses to graduate, complemented by courses of pure interest.

You do not flounder all these years without support.

Lloyd gave me the opportunity to express my frustrations when one of the last courses for the bachelor degree discouraged me. He suggested me to consider both sides, seeing the glass no longer half empty but half filled. I was pleased to know that he would guide me through this dissertation.

Geert and Gert were always available to review my texts. Without them I was still struggling how to express my thoughts.

I want to thank my parents, sisters, brother, uncles and aunts to understand my absence, in body and/or in thoughts, at meetings, sometimes to celebrate happy events, sometimes the sad parts of life.

Last but not least I thank my family. My husband who was always ready to give me support, to handle my tasks where possible to mitigate the pressure on me. Who searched for opportunities to get me away from books and in to nature to let the wind blow through my hair and to relax. My children who had to be self-reliant and did not have the luxury of 'taxi mama' but had to ride the bicycle to arrive at the destination for school and hobby. Who saw their mam more reading books than cooking. It was the baking that they missed most, no more pies, cakes or waffles.

Abstract

This work implements business rule logic for scale agile development processes documented in a wiki by using Semantic MediaWiki technologies. We analyzed scale agile development processes and we determined business rules to support this process. Our work adapts a scrum software development management ontology to define the semantics of the terms. This ontology facilitates the use of business rules, the input of information by using forms. We combined business rules and Semantic Web on a semantic wiki platform, which allows adaption of the information, so the business rules hold. We formulated business rules on the role of the user for the agile development process. The wiki gives an overview of the sprints and displays the status of the current sprint, focusing on the interest of the role of the user. Forms avoid errors by the input of data and make changes only possible for roles that are responsible for the information. Further on, the outputs of the implemented business rules inform teams, managers and stakeholders of rules that do not hold. This work examines how output of these business rules can help to introduce and assist scale agile Development.

1 Introduction

Our work proposes a supported scale agile process by having business rules control the process. A wiki containing all project documentation, such as backlogs, sprints and user stories, allows an open communication among the team members and the Semantic Web technology controls the agile development process by business rules.

The rate of failed IT-projects, around 20–30% (Taherdoost & Keshavarzsaleh 2015), remains a negative factor in software engineering. This rate does not decrease by better education, new knowledge or technologies. Van Cauter (Van Leemputten 2016) suggested the use of agile methodology as a solution. The product owner functions as a facilitator for the collaboration of the business to the team, representing IT. Furthermore, the software artifacts produced in short iterations, creating business value, could be deployed to production when wanted. Hence, misunderstanding between business and IT pops up within a few iterations or sprints.

The processes and tools to follow the progress of an entire project with just a few milestones by managers, is not feasible for agile, where teams determine the value or product to deliver. For management, mostly hierarchical oriented, accepting team autonomy is a true challenge since they are left without a clear vision on project progress. In contrast to waterfall methodology, where people with specific skills do the job individually, agile development can only succeed through teamwork. The whole team has to take ownership and complete the job. Therefore, team spirit, cohesion and open communication are more important than the sum of individual skills.

Additionally, Business Rule Management is a crucial part of business-IT alignment (Soundararajan et al. 2012), thus combining agile methodology and business rules could avoid failures in IT-projects. However, introducing a new Development Methodology requires education and training. Agile methodology is not part of education programs (Soundararajan et al. 2012), principles can be taught but main skills like communication, mentoring and cooperating can only be learned by practice. On the other hand, the tools study concluded that agile tools fall in two extremes, they are either too basic or too difficult to use (Azizyan et al. 2011). By implementing an application that supports the agile development process through business rules, we tackle the reason for IT-failures mentioned above and close the gap of tools.

Spreeuwenberg (Spreeuwenberg & Gerrits 2006), a prominent business rule expert in the Netherlands, guided us from business rules via **A**rtificial **I**ntelligence (AI) to Semantic Web. The combination of Semantic Web and the fact that product development needs descriptions of the product itself, discussions and the history of both, created the requirement for respectively structured data and texts. Kleiner (Kleiner 2015) already analyzed this combination and opted for semantic wiki. Because of the similarity of his work and ours, we decided to continue from his conclusions and opted for Semantic MediaWiki as a platform. This choice eliminated the limitation of the 'one room' requirement for the team. Today, social media proved the feasibility of communication via the Web. Given the fact that wiki provides collaboration and stimulates discussions, we found a strong platform to develop products for which we wanted to support the process. Furthermore, wiki contains the actual documentation of the product, including the history of the documentation and discussions on the documentation. Hence, wiki solves the documentation gap between traditional and agile product development.

Another issue that arises by introducing agile is that it does not support the milestones and deadlines that were traditionally used to show the progress of product development. In traditional product development, the project manager determines the milestones. These milestones measure the progress of the development. The agile development process works with short iterations to deliver value, which makes the survey of the product development complex. Without a solution for the survey of progress, mainly hierarchical structured organizations would have problems to introduce agile methodologies. Managers need to know the status of the project. The need for this information has two aspects: (1) Managing their organization or project; (2) Contact with stakeholders, for which actual information on progress and potential risk is crucial.

Earlier studies of our faculty, done by Bos (Bos 2013) and Slootweg (Slootweg 2016), examined business rules in Semantic Web. Furthermore, another study of our faculty implemented Page Forms, formerly named Semantic Forms, by generating Fresnel Styles from OWL to build information boxes (Rutledge et al. 2016). Our work extended the knowledge by combining business rules and Semantic Web through the use of Semantic MediaWiki.

The research question for this work is:

> 'How and to what extent can business rules implemented with Semantic Web reasoning processed with a semantic wiki interface support the scale agile development process?'

In other words, the goal of this dissertation was the implementation of a wiki to support the use of a scale agile development framework in software engineering and the display of progress of product development, available for all stakeholders by the use of business rules in the Semantic Web. We used business rules, expressed in terms of software development ontologies, in order to facilitate the agile process for all parties involved.

Section 2 of this paper outlines our motivation for the research, where section 3 focuses on the research question and the way we handled it. A summary of papers that guided us to the result of our research and the way these papers were used is explained in section 3. The description of the platform (section 4) and ontology (section 5) helps to understand the proofs of concept for which the details are given from section 6 to 8. In section 9 the results and our conclusions are presented, followed by suggestions for future research in section 10.

2 Research

2.1 Research question

The question introduced in the previous section is

> 'How and to what extent can business rules be implemented with Semantic Web reasoning processed with a semantic wiki interface support the scale agile development process?'

The question contains the items: (1) business rules, (2) Semantic Web reasoning; (3) scale agile development process, (4) interface and more hidden under the word interface (5) the input and output. For each of the items we defined one or more sub questions.

2.1.1 Business rules

a) How and to what extent can business rules guide the team through the scale agile development process?

2.1.2 Semantic Web reasoning

Semantic Web reasoning needs an ontology. In our literature study, we searched for an existing ontology and our case study showed how and to what extent this ontology was usable.

b) How and to what extent can we use an existing ontology for agile methodology?

2.1.3 Scale agile development

Multiple scale agile development frameworks exist and these frameworks have huge differences. Hence, we searched for the most basic to implement our solution. Our study of the existing literature answered the sub question:

c) Which scale agile development framework can we implement in our case study?

2.1.4 Interface

Our proof of concept needs an interface to interact with the user. Our literature study helped us in this choice and thus the answer for sub question:

d) Which interface can we use in our case study to allow the interaction with the user?

2.1.5 Input and output

Each implementation needs input from which output is derived. The product owner writes the stories and starts with the acceptance criteria. The team describes more details to the latter, extends the information with story points, and adapts the status of the story. However, input and output requires data structure, which is described in an ontology, hence sub question b) is linked to the following sub questions.

e) How and to what extent can Semantic MediaWiki technology provide an interface to display the progress of product development?

2.2 Research plan

The first step in our research was the search, comparison and selection of the scale agile framework to use. For the Development Process, used by the selected scale agile framework, we searched for an existing ontology. Once both found, we examined the techniques and tools to use for the implementation of the scale agile development process, managed by business rules, focused on the role of the user. Before we could use terms, these had to be defined by an ontology. Hence, we needed research to check if all terms are available in the K-CRIO ontology (Lin et al. 2011). Based on the extended ontology, defined and developed in Semantic MediaWiki, we translated the business rules of LeSS.

To allow teams, spread cross-country, to collaborate on a common platform, where all team members can create and edit texts, we set up a semantic wiki. We created forms with the help of Page Forms (Koren et al. 2017) to check the format of the input of specific attributes as story points and statuses.

The key activity of our research focused on the interface by changing the view of the wiki pages based on the role of the user logged in in the wiki and business rules based on roles. The output helps the user to accomplish the tasks for his role and informs the user of possible violations of the rules.

A schematic view is given in Figure 1.

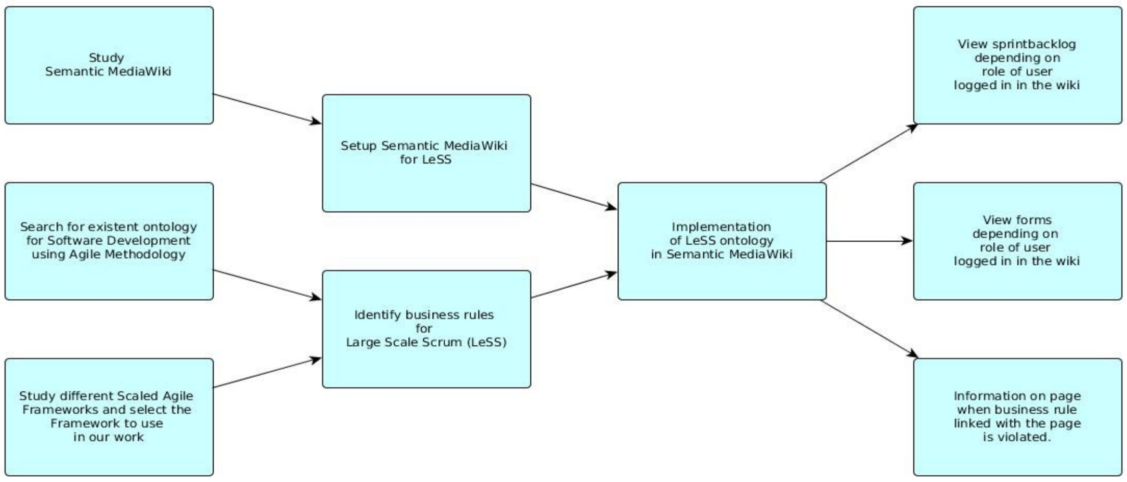

Study Semantic MediaWiki
Search for existent ontology for Software Development using Agile Methodology
Study different Scaled Agile Frameworks and select the Framework to use in our work
Setup Semantic MediaWiki for LeSS
Identify business rules for Large Scale Scrum (LeSS)
Implementation of LeSS ontology in Semantic MediaWiki
View sprintbacklog depending on role of user logged in in the wiki
View forms depending on role of user logged in in the wiki
Information on page when business rule linked with the page is violated.

Figure 1: Schematic view of our research

The combination of Semantic MediaWiki, agile development methodology and business rules is not studied. Hence, we opted for a proof of concept (poc) to study this combination. To limit our research the focus was set on roles within the agile methodology. The tasks to perform were:

- Selection of an agile scale framework
- Studying Semantic MediaWiki
- Searching for existence of a scrum ontology
- Identify business rules for Large Scale Scrum
- Setup a Semantic MediaWiki to implement the poc
- Develop a Large Scale Scrum in Semantic MediaWiki
- Implement sprint interfaces which are role dependent
- Implement forms which are role dependent
- Implement business rules linked on roles
- Analyze the results of the different pocs

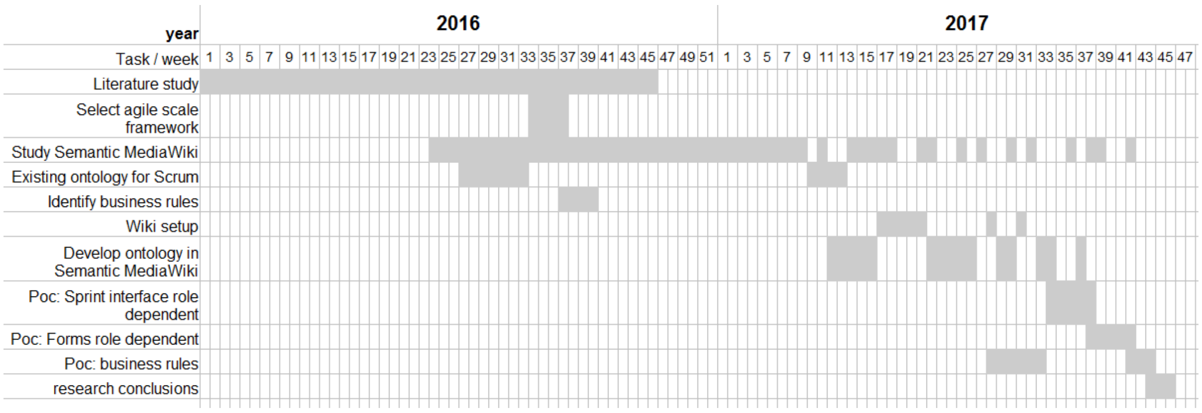


Figure 2: Planning dissertation

3 Related Work

3.1 Introduction

In this paragraph, we summarized the papers that guided us to answer sub question b*) How and to what extent can we use an existing ontology for agile methodology?* partly (3.3.2) and sub question c*) Which scale agile development framework can we implement in our case study?* completely (3.2.3).

Based on literature we explained our choice for agile (3.2), and for our case study the choice of the scale agile framework LeSS (3.2.3) and the problems we tackled. The thesis of Kleiner (Kleiner 2015), explaining the use of Semantic MediaWiki to manage IT Services (3.3.1), has similitude with our study, which made it obvious to include in this related work. We found an ontology for software projects, K-CRIO (3.3.2), ready to model the basic terms and relations of scrum. Semantic wiki, allowing to combine structured and unstructured data (3.4.1), Page Forms (3.4.2) to support input of data in the wiki and rule modeling in Semantic MediaWiki (3.4.3) are discussed in 3.4. Business rules in general and specific for the Semantic Web are described respectively in 3.5.1 and 3.5.2. Our conclusion can be found in 3.6.

3.2 Agile

3.2.1 Solution for IT failures?

One of the most studied subjects within informatics science is the causes of IT-project failures. A recent study (Taherdoost & Keshavarzsaleh 2015) made a review of earlier studies of success and failure of IT-projects. This study proved that the use of the agile methodology is mentioned everywhere as a success factor. By defining the knowledge and know-how of agile processes as a success factor, and considering the continued high failure rate of IT projects, we can conclude that this knowledge is substandard today. Our work will guide teams, scrum masters and product owners through the agile process by implementing business rules on the agile process linked with the differences of information needed based on the role of the user.

An ongoing study of IT-projects for governments (Van Leemputten 2016) criticizes the lack of involvement of the government officer. Van Cauter stated that the gap can be closed by using agile methodologies (Van Leemputten 2016). These methodologies oblige the customer in the role of product owner to be directly involved in the development of the application. The product owner manages the user stories and he is the one who decides if the user story holds its given criteria. This involvement excludes the problem of an application developed over a long period of time that does not comply with the business requirements.

3.2.2 Education

Not only the work of Taherdoost (Taherdoost & Keshavarzsaleh 2015) stated the increase of agile methodology. The Department of Computer Science at Virginia Tech organized a course on agile. More than half the number of students had software engineering industry experience and were reluctant to the agile methodology (Soundararajan et al. 2012). One of the reasons for reluctance was the agile concept of minimal documentation (Soundararajan et al. 2012).

The course objectives were (Soundararajan et al. 2012):

- Introducing the agile philosophy, its values, principles, and practices.

- Understanding the principles of agile and the differences between agile and conventional software engineering.
- Critical thinking using the agile methodology.

To achieve these objectives, the instructor explained the philosophy, values and principles and guest speakers presented their experience with the agile methodology. The teaching assistant and a student evaluated the course by writing a resume and all students evaluated the course by a questionnaire. A remarkable observation of this study was that with the progress of the course, the students changed from reluctance to embrace (Soundararajan et al. 2012).

Our implementation supports documentation of user story items by using a wiki (see chapter 7) and guides the teams through the agile development process, focused on sprints (chapter 6). This support facilitates the transition to the agile development process.

3.2.3 Scale agile

Topicus, a software house for financial packages, examined the best way to scale agile between several teams working on the same product. The case study (van Leeuwen 2015) compared three scale agile frameworks using the scrum methodology to find a solution to the agile scaling challenge. It showed that none of the frameworks fit their requirements. Further findings are that migrating from traditional to agile development methodologies not only requires changes from the team members, but also of the organization. (van Leeuwen 2015)

The main purpose of the study (van Leeuwen 2015), comparison of the scale agile frameworks, is discussed here in more detail to help us choose the framework to use in our work.

All of the documentation opportunities and some of the execution issues, i.e. one product backlog, no further meetings and warning signals, mentioned by Van Leeuwen (van Leeuwen 2015) apply to our work. The criteria used to select the framework to examine (van Leeuwen 2015) correspond with our criteria. Hence, we can copy van Leeuwens conclusion and continue with **L**arg**e** **S**caled **S**crum (LeSS), **S**caled **A**gile **F**ram**e**work® (SAFe®) and **D**isciplined **A**gile **D**elivery (DAD). The description of each of these frameworks can be found in (van Leeuwen 2015).

Table 1: Comparison of scale agile frameworks

LeSS	SAFe®
One single backlog for the product to develop	Up to four backlogs
One single Definition of Done for all teams	
Each ended sprint is ready to be deployed in production and the sprint adds value for the Business	Deployment of ended sprint after the Program Increment, hence after circa six sprints
One product owner	A product owner per team under a program owner, under epic owner, under program portfolio management
product owner prioritizes the items of the backlog and product owner decides which items will be developed first	Prioritization and the decision of the next item to develop are determined by the Weighted Shortest Job First logic.
Up to eight cross-functional teams	No teams limit, with team specialization
One sprint over all teams	One sprint over all teams
In case of linked items over multiple teams, a member of the other team can observe the daily scrum meeting of the other team.	Scrum masters participate in the scrum of the Scrum meeting.
flat structure	Hierarchical structure
Existence of LeSS Rules[1]	

1https://less.works/less/rules/index.html

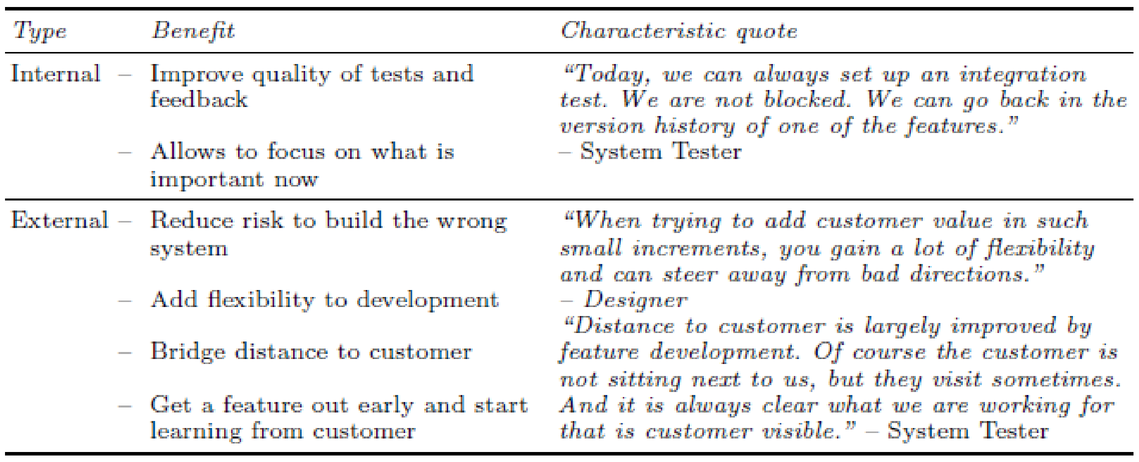
We had some knowledge of LeSS and SAFe®. These scale agile development frameworks correspond best with the scrum ideology of short iterations, which deliver value to the business. For these reasons, we only compared LeSS and SAFe® as possibilities to use in our work. LeSS (The LeSS Company B.V. 2016a) focuses on the coordination on team level, SAFe® (SAFe 2015) focuses on organization level by adding program and portfolio (van Leeuwen 2015).

LeSS uses the scrum terms, focuses on team level and last but not least has some described business rules (The LeSS Company B.V. 2016c). Hence, we opted to implement LeSS as scale agile framework. Thus, our sub question *c) Which scale agile development framework can we implement in our case study?* is answered:

> Our study uses LeSS because business rules exist.

Kasauli researched the added value in large scale scrum and studied the effects of adding value every sprint (Kasauli et al. 2017). He summarized the benefits of adding value in a table by distinguishing between two types of benefits. The internal benefits, which are only visible within the teams and the external which have a direct impact on the product and customers.

Table 2: Benefits of adding value every sprint (Kasauli et al. 2017)



Type	*Benefit*	*Characteristic quote*
Internal	– Improve quality of tests and feedback – Allows to focus on what is important now	*"Today, we can always set up an integration test. We are not blocked. We can go back in the version history of one of the features."* – System Tester
External	– Reduce risk to build the wrong system – Add flexibility to development – Bridge distance to customer – Get a feature out early and start learning from customer	*"When trying to add customer value in such small increments, you gain a lot of flexibility and can steer away from bad directions." – Designer* *"Distance to customer is largely improved by feature development. Of course the customer is not sitting next to us, but they visit sometimes. And it is always clear what we are working for that is customer visible."* – System Tester

Even taking into account the fact that only product builders and not the ancillary people defined by Lin (Lin et al. 2011) are cited, a clear difference is shown in the way a sprint is seen. These different views have an impact on the interface, because each role has its own interests in the sprint. Hence, different roles ask for different views.

3.2.4 Tools

Little literature is found on tool usage for agile development, even the search of less scientific sources gave no satisfactory results (Azizyan et al. 2011). To achieve a correct view on the usage of tools, complementary to few papers found, a survey was published on social media. The responses gave input from 120 companies, spread over 35 countries (Azizyan et al. 2011).

The analysis of the received information showed that most companies use the scrum methodology. Half of the teams use the wall, including post-its, to follow the progress of the sprint. The survey showed a remarkable, but not surprising, difference in the tool usage between collocated and distributed teams. The latter use tools more frequently (Azizyan et al. 2011), because the wall alternative forms no possible communication platform for distributed teams.

The satisfaction of the tools in ease of use is high, but integration with other systems and customized reports scored low. The conclusion of the study is that the tool is either too basic or too complex. Teams need something in the middle. (Azizyan et al. 2011)

A group of respondents (42%) specified features needed in tools for agile development. These comments were grouped as follows (Azizyan et al. 2011):

- Reporting
- Integration with other systems
- Virtual task board
- Project status tracking

The TargetProcess white-paper documents the use of agile software and differentiates the old-school, agile project management tools and the whiteboard. The white-paper concludes that the tools are still in try out modus. (Dubakov & Stevens 2008)

Comparing the numbers of the survey of both papers, the use of tools decreases. The survey of 2008 mentions 34% of the teams using spreadsheets or no tool at all (Dubakov & Stevens 2008), compared with 49% of the teams in 2011 (Azizyan et al. 2011). Combing this observation and the education issue, see 3.2.2, the need of a tool with clear business rules is obvious.

3.3 Semantic Web Technology

3.3.1 Similar studies on Semantic Web-based platforms

To implement a process in the Semantic Web, the knowledge of the process to implement is needed. The thesis of Kleiner (Kleiner 2015) gave us an insight in the method to use. He looked for the fundamentals of his work, being IT Service Management, **IT I**nfrastructure **L**ibrary (ITIL), ontologies, Semantic Web, wikis and semantic wikis (Kleiner 2015). The explanations given on these fundamentals provide the reader the knowledge to understand the work. Once the basics are known, the analysis of the environment of the IT Service Management, the configuration management and the existing tools started, described in (Kleiner 2015) in chapter 3. The next step formed the design of the semantic wiki platform, which included the selection of the technical platform, the requirements for IT Service Management within a semantic wiki and the data model, designed as ontologies. Based on this information the components of the system are designed and implemented. The implementation was evaluated by validating the fulfillment of the requirements and a user study. This evaluation is followed by the conclusions of the work. (Kleiner 2015)

Interesting for our work is the motivation for Semantic Web technology and the approach of the study, which we both describe in more detail. IT Service Management requires (Kleiner 2015):

- Storage of structured information and unstructured information
- Web-based collaborative editing
- Reporting capabilities
- Customizability and extensibility
- Adaptability to changes
- License

This requirement list created two options from which the option semantic wiki (see 3.3.1) was chosen (Kleiner 2015). The motivation of the choice is the allowance of storage of all data, structured and unstructured, the processing of structured data and the capacity of reasoning (Kleiner 2015).

Reasoning can make implicit data explicit, a functionality we need to implement the LeSS business rules (see 3.5.2).

Once the choice for semantic wiki was made, the question became which platform to use, as different semantic wiki platforms exist. Kleiner opted for Semantic MediaWiki (see 3.4) because the platform has a GPL license, has the technical advantages of stability, flexibility, free from dependencies, is actively used and last but not least implements the academic structured approach by Page Forms (see 3.4.2) (Kleiner 2015).

The approach followed to choose the platform and semantic wiki framework to use, consisted of the following steps (Kleiner 2015):

- Defining the goal
- Gathering the requirements
- Searching possible options
- Analyzing the options
- Evaluation and choosing the best fitting option.

The design and implementation of the components was described in more detail by following steps (Kleiner 2015):

- Motivation
- Requirement analysis
- Use Cases
- Relevant information
- Design of the component
- Implementation
- Representation of the information in wiki

Our work will follow the approach used in (Kleiner 2015), by following the analysis as described above and by implementing the needed functionalities for our project as separate components.

3.3.2 Ontology

The **W**orld **W**ide **W**eb (WWW), a vast source of information, links information by the use of hypertext and hypermedia. This makes information not accessible for automated tools, hence, searching or browsing for information forms the main shortcoming of the Web. (Horrocks 2008)

Annotations help to structure data in a labeled directed graph or **R**esource **D**escription **F**ramework (RDF) by triples. Each triple consists of a subject, predicate and object. A binary relationship between subject and object links the information via the predicate. (Horrocks 2008)

The **W**eb **O**ntology **L**anguage (OWL) extends RDF with cardinality constraints, conjunction and disjunction of classes (Horrocks 2008). Hence, an analogy between OWL and databases exists, but more important are the differences. Databases handle missing information as false, while OWL treats this missing data as unknown (Horrocks 2008). The literature described this issue under the terms open-world assumption and closed-world assumption. Further differences are (Horrocks 2008):

- Use of unique names: Databases use the **U**nique **N**ame **A**ssumption (UNA), and OWL does not.
- Schema considered at query time for OWL and not for databases.

The existing implementations of business rules are based on the use of databases. Hence, they consider a closed world, in which the information is available. We have to keep in mind the impact of the open-world assumption for the implementation of the business rules.

An ontology defines the terms of the domain. Our domain is the large scale agile process, which can be seen as part of software project management. This domain is not new. Hence, we can assume that an ontology already exists. A review, published in 2014 (Fitsilis et al. 2014), criticized the use of prototypes as ontologies and the lack of standardization. Fitsilis searched for ontologies related to project management in general, project management knowledge areas, project management methodologies, software process models and software life cycle phases (Fitsilis et al. 2014). The found ontologies were split in two categories, namely project management and software process models (Fitsilis et al. 2014). Fitsilis described each of them in more detail and made lists to create an overview of the existing ontologies (Fitsilis et al. 2014). This literature study showed a large amount of ontologies, hence Fitsilis concluded that "*the most important issue in ontology usage in the area of software project management is standardization*" (Fitsilis et al. 2014).

Starting from this point of view, we searched for an existing scrum ontology and found the K-CRIO ontology. This ontology is based upon the meta-model for organizations, CRIO (Lin et al. 2011). K-CRIO is studied for scrum in OWL by using Protégé (Lin et al. 2011), a well-known tool in our faculty.

The study followed an ontological approach to describe the processes to design a product (Lin et al. 2011). The main concepts of the ontology are organization, design object, role, capacity, ontology, interaction, state, ontology element and time. Relations link these concepts or classes. An organization X can be part of another organization Y, denoted by `isSubOrganizationOf` (X, Y). Relations can have cardinality restrictions. An example for the cardinality restriction is that an Organization must include one Role. (Lin et al. 2011)

The study gives an overview of the scrum context, to build an ontology based on K-CRIO for scrum (Lin et al. 2011). The result of this work is captured in Figure 3 (Lin et al. 2011).

Our work needs an ontology, which defines the scrum and LeSS terms, and relations among these terms. In order to meet the requirement for more standardization, we start from the K-CRIO and extend this ontology where needed. Hence, a part of the sub question *b) How and to what extent can we use an existing ontology for agile methodology?* is answered, namely we can reuse an existing ontology named K-CRIO.

There is no team or work that can be done without involving people. A well-know and studied ontology on the web is Friend of a Friend, also know as FoaF. Chen described FoaF as (Chen et al. 2005):

> *'FOAF: This ontology allows the expression of personal information and relationships, and is a useful building block for creating information systems that support online communication.... use FOAF ontologies to express and reason about a person's contact profile and social connections to other people in their vicinity.'*

Within agile people, and thus persons, are a prime factor. To allow Semantic Web reasoning, and to strengthen the view of Lee-Timers for linked data (Bizer et al. 2009), the used ontology in our work has to be linked with FoaF.

Protégé, a tool developed at Stanford University to create ontologies for bio-medicine, was used to investigate the creation of worldview profiles and to identify the correlation between worldview discussions (Jakobsen 2016). Jakobsen interviewed 19 representatives from three religions in Denmark (Jakobsen 2016). The information of the interviews was translated to selected attributes,

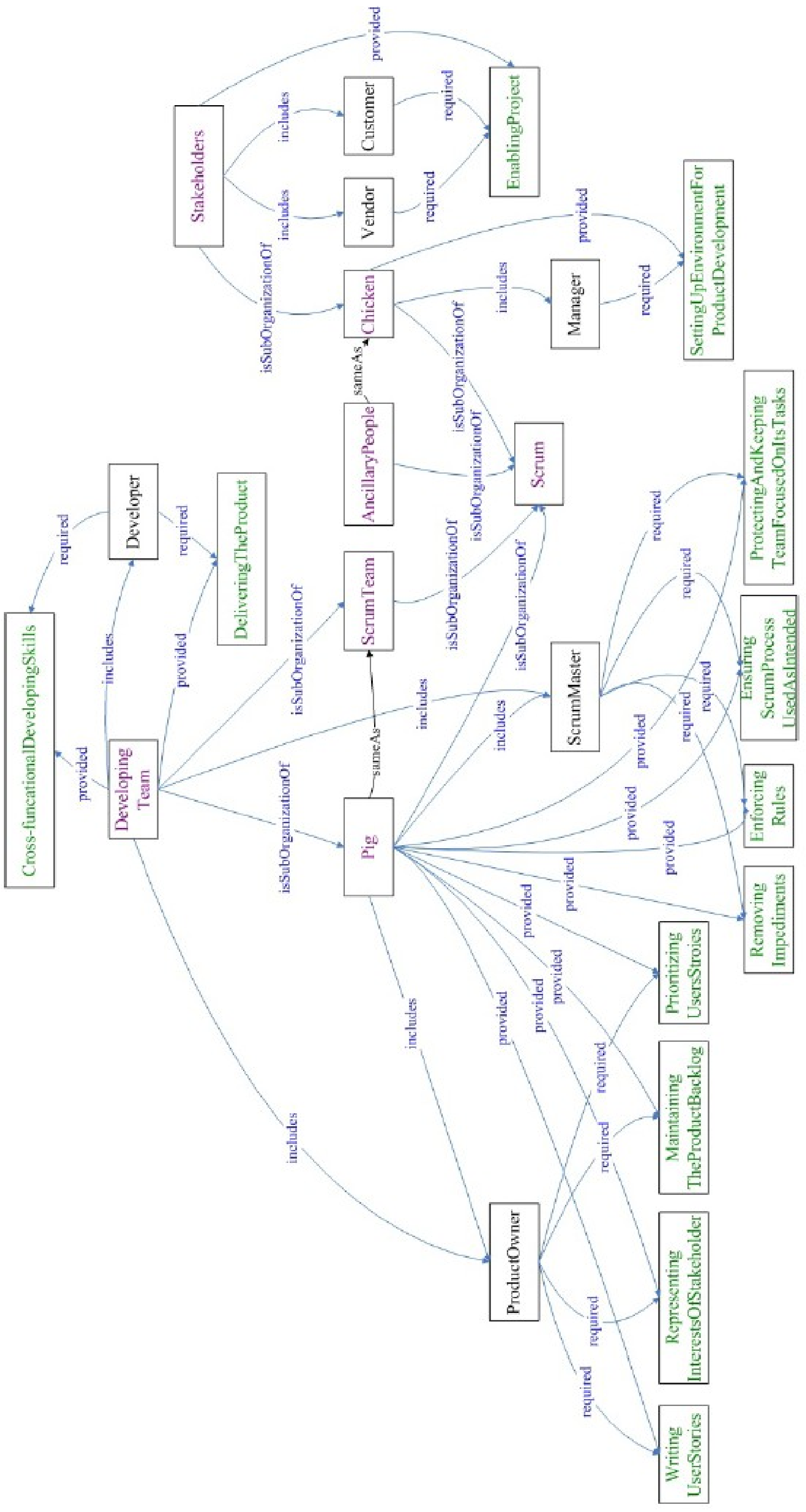


Figure 3: Scrum with K-CRIO (Lin et al. 2011)

namely relevant to aspects of their life stories, on beliefs and views, on objective characteristics of the interviewees and on (inter)religious dialogues (Jakobsen 2016). The study explored this information by categorizing the data and logical deduction, this revealed unexpected aspects and oddities (Jakobsen 2016).

The study proved that although Protégé was developed for bio-medicine purposes, it is not at all limited to this discipline.

3.4 Semantic wiki

3.4.1 Semantic MediaWiki

As mentioned above (section 3.3.1) there is a similarity between the work of Kleiner and ours. One of the similarities is the need for both structured and unstructured information. Kleiner's research opted for Semantic MediaWiki. Hence, we follow this choice. Thus, we need information on Semantic MediaWiki.

The main goal of **S**emantic **M**edia**W**iki (SMW) is to create a Semantic Wikipedia. To reach this goal, the following wiki problems must be solved:

- consistency of content
- access of knowledge
- reuse of knowledge.

Annotating pages enables the wiki to give semantic context, and thus structured information, to the text-centric content, being unstructured information, of the wiki page by the use of categories, relations and attributes. Querying and searching take advantage of the structured information given by annotations. Furthermore, the query functionality allows embedding of dynamic content into pages. (M. Krötzsch et al. 2007)

These functionalities are all based on Semantic Web technology (M. Krötzsch et al. 2007).

3.4.2 Page Forms

Semantic wiki closes the gap between the Semantic Web technology and wiki. Semantic wiki adds data input and processing to the originally text oriented wikis (Rutledge et al. 2016). Hence, the unstructured text-centric wiki is extended by the structured data-centric semantic wiki.
The most used wiki platform, MediaWiki, has an extension for semantic wiki, namely Semantic MediaWiki. Semantic MediaWiki enriches MediaWiki with Semantic Web functionalities (Rutledge et al. 2016):

- Data annotation
- Data querying
- Data export in Semantic Web form, directly on the Linked Data Cloud if wanted

Page Forms, based on the formerly Semantic Forms, another extension of MediaWiki, allows (Rutledge et al. 2016):

- Data derivation from table displays
- Form based user entry of data
- Interface for creating tables and forms;

Rutledge created the functionality that allows a quick start for creating a wiki with the possibility to enter data and to post the data on the Semantic Web. Using the plug in for Fresnel Forms in Protégé allows the user to generate an interface that can be stored as an RDF file of Fresnel triples. Fresnel Forms use a default styling, but customized styling is also possible. By importing the Fresnel code, an XML file, in a wiki, the Page Forms assist the user to enter data. The case study tested the Fresnel Forms on Wikipedia info-boxes by creating the info-box of Tim Berners-Lee. (Rutledge et al. 2016)

Our work needs the input of structured data by the users. Our case study examined the use of Fresnel Forms to speed up our implementation.

3.4.3 Rule modeling in Semantic MediaWiki

A limitation of Semantic MediaWiki is its lack of support for modeling rules. Semantic MediaWiki needs the following extensions to enable rule modeling (Bao et al. 2009):

- Ask Query as simple query language
- Semantic Template to enable parameters
- Parser Function for **E**vent-**C**ondition-**A**ctions (ECA)
- Arraymap, a loop function.

OWL enables entailment rules and logic programs enable to perform integrity constraint checking (Bao et al. 2009).

The way of modeling rules in Semantic MediaWiki is used when we implement these in the semantic wiki.

3.5 Business Rules

3.5.1 Business Rules

Joosten demonstrated that '*business rules alone can serve as functional business requirements'* (Joosten & Joosten 2005) because *'rules can be used to describe the modeling techniques and methods themselves'* (Joosten & Joosten 2005). **A D**escription **L**anguage (ADL), using relational algebra, can represent business rules. The Plan-Do-Check-Act principle triggers actions when a violation of a business rule occurs. The rule engine traces which business rule is violated to determine the alternatives to resolve the violation. (Joosten & Joosten 2005)

The principle is explained by the example of an invoice that has to be either paid or returned. The moment an invoice arrives, the rule is violated. An action of a user or system is needed to either pay or return the invoice. Once this action is executed, the rule holds. The principle prescribes neither the order nor the way to act, so maximum flexibility is given to users to resolve the violation. When the business process goes wrong, new rules need to be added or incorrect rules need to be adapted. (Joosten & Joosten 2005)

The principle Plan-Do-Check-Act checks for violations and triggers actions. Users will handle most of the actions to resolve violations for the agile development process in our implementation. The flexibility to resolve problems is important and guaranteed by the use of this principle.

In another work (Joosten 2010), Joosten distinguished two types of rules. On the one hand, there are the action rules, driven by activities. These have an order for the execution of activities and no flexibility, which frustrates knowledge workers. The literature described action rules as **E**vent-**C**ondition-**A**ction (ECA) or Condition-Action rules, when the rule is not triggered by an event. On the other hand, there are the invariant rules, driven by policies and agreements. The business creates the rules by and for themselves. The strength of invariant rules is that one business rule maps to one invariant rule, which facilitates traceability. Many Event-Condition-Action or Condition-Action rules control the invariant rule. Hence, changing a business rule does not need a scan over all Event-Condition-Action rules but only over a limited number of ECA rules, because adapting the linked invariant rule satisfies the business change. It is not surprising that the Business Rule Manifesto opts for the use of policies or invariant rules. In literature this principle is cited as '*Separate the know from the flow'* (Joosten 2010). Contrasting the Manifesto and the principle, the software packages on the market support action driven rules. (Joosten 2010)

To satisfy the principle of the Manifesto, and thus the advice given in (Joosten 2010), our work uses business rules mapped to invariant rules.

Our faculty did research on the applicability of business rules between relational algebra and Semantic Web. Given the fact that the study to implement legislative drafting in relational algebra was done earlier, Bos limited his study to the use of **S**emantic **W**eb **R**ule **L**anguage (SWRL) and compared the results of his conclusions and these of the implementation using relational algebra. His conclusions are presented in Table 3 (Bos 2013).

Table 3: comparison relational algebra and SWRL (Bos 2013)

Description	Relational algebra	SWRL
Type checking	No type checking	Type checking
comparison	Impossible due to the lack of type checking (comparing apples to oranges)	possible
Open/close world assumption	Close world by returning true or false for each rule	Open world because a lot of information is not available, so the rule returns true, false or unknown.
Object integrity	Two or more occurrences of a unique reference throw an error.	SWRL considers two or more occurrences of a unique reference as being the same object, thus no error is thrown.
Reasoning	Relational algebra does not reason, it uses constraints.	SWRL derives new information, which is reusable, by the use of information assertion rules. Reasoning takes time due to number of possible paths and the huge amount of data.
Object properties	Interference between object and key element	Clear definition of object properties.
Separation of rules	Interference between **U**ser **I**nterface (UI), session and business rules.	No user interface, neither session rules.

It seems obvious that the conclusions of Bos (Bos 2013) offer instructions for the use of business rules, an important part of our work.

The **M**anchester **B**usiness **R**ules **M**anagement (MBRM), an approach to a business rule centric development, classifies business rules using three views: intentional view, operational view and information systems view. For the analysis of each of these views, business rules are treated differently. (Kardasis & Loucopoulos 2004)

The rules as seen from a business perspective are intentional rules. Operational rules express business processes and information system architecture rules impact the information system implementation. (Kardasis & Loucopoulos 2004)

Comparing this information with the classification given by Joosten (Joosten 2010) we see the similitude between on the one hand, policy rules and intentional rules and, on the other hand the action rules and operational rules, respectively named by Joosten (Joosten 2010) and Kardasis (Kardasis & Loucopoulos 2004). Hence, we need the operational rules to implement the intentional ones.

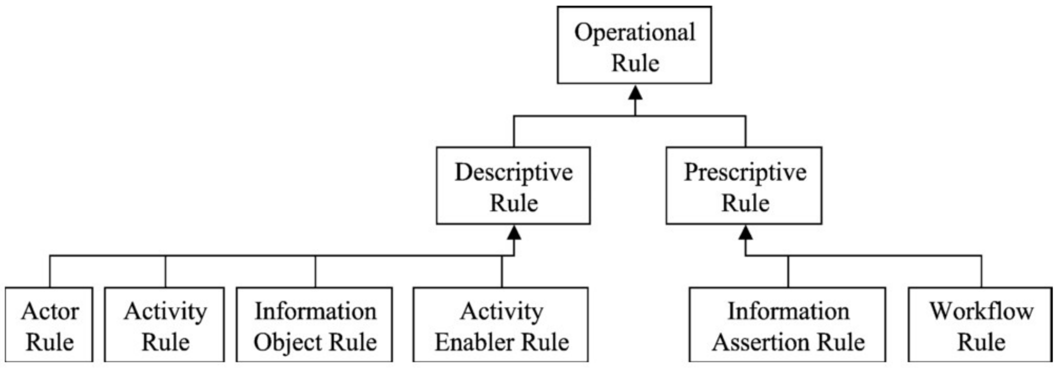


Figure 4: High level classification for operational rules (Kardasis & Loucopoulos 2004)

Descriptive rules describe the structure, which means that they need always to be true or a violation occurs. Prescriptive rules describe Event-Condition-Action rules. The paper describes in detail the structure for the creation of the rules and explains each of the operational rules by creating the rules for the electronic procurement system. (Kardasis & Loucopoulos 2004)

The Manchester Business Rules Management (Kardasis & Loucopoulos 2004) is used for the creation of rules in our case study.

3.5.2 Business Rules and Semantic Web

Spreeuwenberg searched for differences between business rules and Semantic Web. They both have **A**rtificial **I**ntelligence (AI), the study of knowledge representation, as an ancestor. Today, business rule engines are named expert systems and Semantic Web uses the term knowledge management. The main differences between the business rule and Semantic Web languages are on the one hand the focus on respectively humans and machines. On the other hand, business rules uses the closed world assumption where Semantic Web works with the open world assumption. Both languages win when they become interpretable for humans as well as for machines. The survey by Spreeuwenberg (Spreeuwenberg & Gerrits 2006) stated that the focus of ontology builders on decreasing complexity is seen as hopeful for closing the gap by domain experts designing the ontology and machines being able to understand and exploit the ontology.

In our work, we paid attention to use models in a way that is comprehensible for domain experts, thus humans, and it is evident that the ontology had to be understood by machines.

SWRL allows querying and reasoning about an OWL-ontology. The latter enables the creation of new knowledge from explicit data. By reasoning about OWL instances, SWRL extends OWL with Horn clauses. The form of the rule is an implication between antecedent and consequent, which means that if the conditions in the antecedent are met, the conditions in the consequent must also be met. The classic example is: if x has parent y and y has brother z, then x has uncle z. (Ruiz-Bertol et al. 2011)

The reasoning of SWRL enables the use of business rules, written in Horn clauses, a functionality needed to accomplish our work.

Ontology-driven Business Rule Specification is a domain oriented -thus understandable for the business- approach to discover and specify business rules. This approach, described for an accounting information system, consists of the following steps (Gailly & Geerts 2013):

- Classify the enterprise model in a domain ontology,
- Match the enterprise model with the ontology model
- Determine the business rule patterns associated with the enterprise model
- Use of semantic annotations to instantiate the business rule patterns

Given the fact that our work needs an ontology, to define terms and their relationships for human and machine, and business rules to describe business requirements, the combination of both, and thus the approach described in (Gailly & Geerts 2013) seems promising and was used in this work (see 3.3.2, 3.5.1 and 3.5.2).

Slootweg described in a case study the implementation of Hohfeldian legal concepts using the Semantic Web technology. The first step was the creation of an ontology, followed by defining business rules. The Semantic Web technologies SWRL and SPARQL validated these rules. (Slootweg 2016)

In this work, we have taken advantage of the n-ary relation pattern described and used in (Slootweg 2016) to create the relationship between on the one hand user stories and on the other hand story points and story priorities.

3.6 Conclusion

We found a lot of studies mentioning agile as one of the solutions for the high rate of IT-project failures, from which we selected (Taherdoost & Keshavarzsaleh 2015). The idea is that the close collaboration of the customer with the team delivering the product will help teams to focus on the right issues, and thus the needs of the business. In addition to the problem of IT-project failures, the knowledge of the agile methodology is not widely spread among software engineers working in the industry (Soundararajan et al. 2012). The reluctance to go agile is linked with the disappearance of the focus on documentation and the focus for Plan Driven Development. Agile focuses on value creation by stating ‘working software over comprehensive documentation’ (Beck 2001).

Little scientific literature is available on the agile methodology. For this reason we limited our description to scale agile frameworks as described in a recent master thesis (van Leeuwen 2015). The work describes three scale agile frameworks in detail, namely LeSS, SAFe® and DAD. We used their comparison of the frameworks, due to the resemblance of requirements and concluded that the use of LeSS in our work is the best option to choose.

The study on agile tooling had to be included in our work. Remarkable was the observation of a decrease in the use of tools based on surveys done in 2008 (Dubakov & Stevens 2008) and 2011 (Azizyan et al. 2011). For the tool to be implemented, we filtered the requirements of users in (Azizyan et al. 2011) and copied these to the requirements for the tool to implement.

The thesis of Kleiner (Kleiner 2015) explained the advantage of a semantic wiki, being the use of the combination of structured and unstructured information and the collaboration needed to exchange information over the teams. The existence of study and implementations by our faculty with Page

Forms (Rutledge et al. 2016) and business rule modeling in Semantic MediaWiki (Bao et al. 2009) creates the opportunity to develop components for business rules in the Semantic Web technology. The basis for the scrum ontology to use is given by K-CRIO (Lin et al. 2011). This knowledge gave us a hint for sub question *b) How and to what extent can we use an existing ontology for agile methodology?* by knowing that an existing ontology can be used. The questions 'how' and 'to what extent' remained unanswered in our literature study.

Implementing business rules means that the knowledge for the creation of rules has to be available. Earlier studies in our faculty by Joosten (Joosten & Joosten 2005),(Joosten 2010), Slootweg (Slootweg 2016) and Bos (Bos 2013) extended our knowledge. The Manchester Business Rules Management (Kardasis & Loucopoulos 2004) was studied in more detail to implement their approach for identifying and creating operational rules.

Based on our research, we concluded that the development of a platform for the use and control of a scale agile development framework is achievable.

4 Platform

4.1 Introduction

MediaWiki is the software platform on which Wikipedia runs, and thus one of the most used wikis (Kleiner 2015). The development of the core of MediaWiki is under supervision of the Wikimedia Foundation[2]. To develop functionalities without making the core code more complex, the framework is open to extensions. Semantic MediaWiki is such an extension of MediaWiki. The manual to extend functionalities of the MediaWiki core code by developing extensions documents the possibilities, namely tag extensions, parser functions, hooks, special pages, skins, magic words and APIs (Wikimedia Foundation 2017). Our proof of concept implements business rules based on the role of the user for LeSS within Semantic MediaWiki and thus without implementing a new extension.

4.2 Semantic MediaWiki

For our proof of concept we needed a platform with following specifications:

- Easy to use for all people working on the project, going from stakeholders and business users to software developers and architects. Otherwise the wiki would not be used and cannot help in the process of product development.
- Share and exchange knowledge on the product, the architecture of the solution and technical items within and between the teams.
- Collaborative so that users not located at the same place can interact. For example a development team needs to interact with other development teams, with the product owner and the subject experts.
- History to show the changes made and the author of these changes.
- Rollback to allow to go to one of the previous versions if the changes are not accepted.
- Not limited to the number of users because the LeSS methodology does not limit the number of teams and each team consists of five to nine members.
- License of free use to avoid high costs for large products, thus large number of users.
- Handle unstructured information such as the description of the user story; the reason why the story is needed and acceptance criteria. The content of these items is meant to be used by human beings. There is no possibility to structure this information.
- Handle structured information such as status, story points and development teams to allow querying the information.
- Use of triples to allow Semantic Web output and reasoning.

The requirements above resemble these of Kleiner in his selection of the technical platform (Kleiner 2015). Furthermore, Krötzsch stated (Markus Krötzsch et al. 2007):

> *'Wikis have become popular tools for collaboration on the web, and many vibrant online communities employ wikis to exchange knowledge. For a majority of wikis, public or not, primary goals are to organise the collected knowledge and to share this information. Wikis are usually viewed as tools to manage online content in a quick and easy way, by editing some simple syntax known as wikitext.'*

2 https://wikimediafoundation.org/wiki/Our_projects

So, wiki stands for collaboration, collection and exchange of knowledge, sharing information and easy to use. These are parts of the requirements we mentioned above.

Based on this analysis of our literature review, we opted for Semantic MediaWiki as platform and interface for the user. We describe our decision in more detail by describing the architecture of Semantic MediaWiki (4.2) and the extensions used (4.3).

4.2.1 Architecture of Semantic MediaWiki

Semantic MediaWiki is an extension on MediaWiki (Kleiner 2015), the platform on which the well-known encyclopedia Wikipedia runs (Markus Krötzsch et al. 2007). The architecture of the main components of Semantic MediaWiki related to MediaWiki (Markus Krötzsch et al. 2007) is shown in Figure 5.

A lot of information on Semantic MediaWiki is available online, but finding the right and current information is not straightforward. We discussed the architecture and working in more detail so that an overview is available and accelerates the learning process.

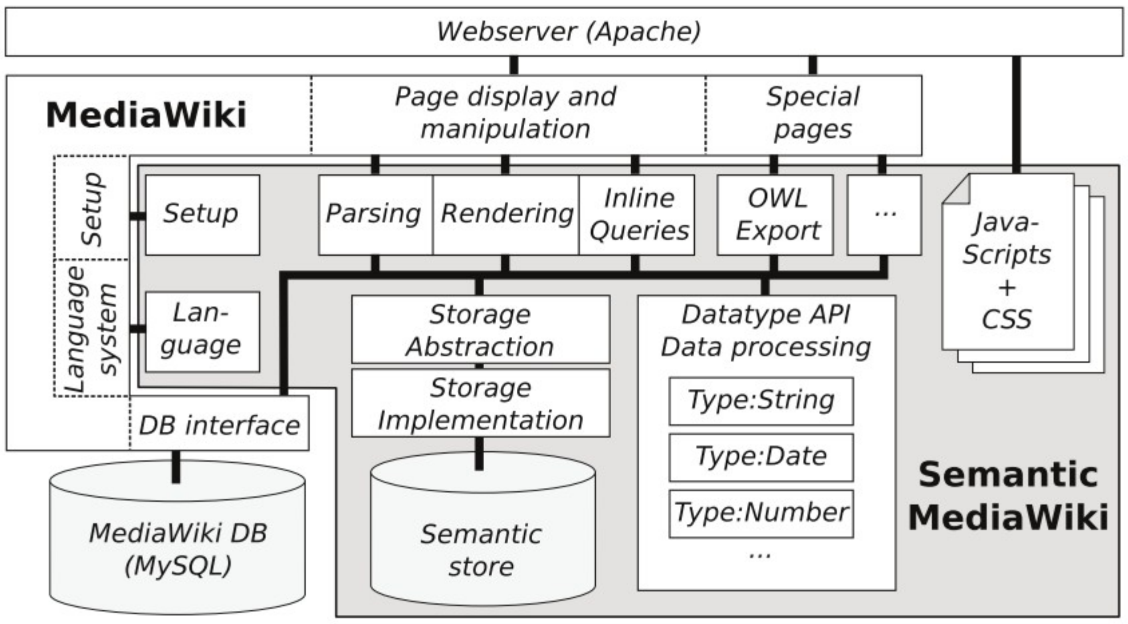
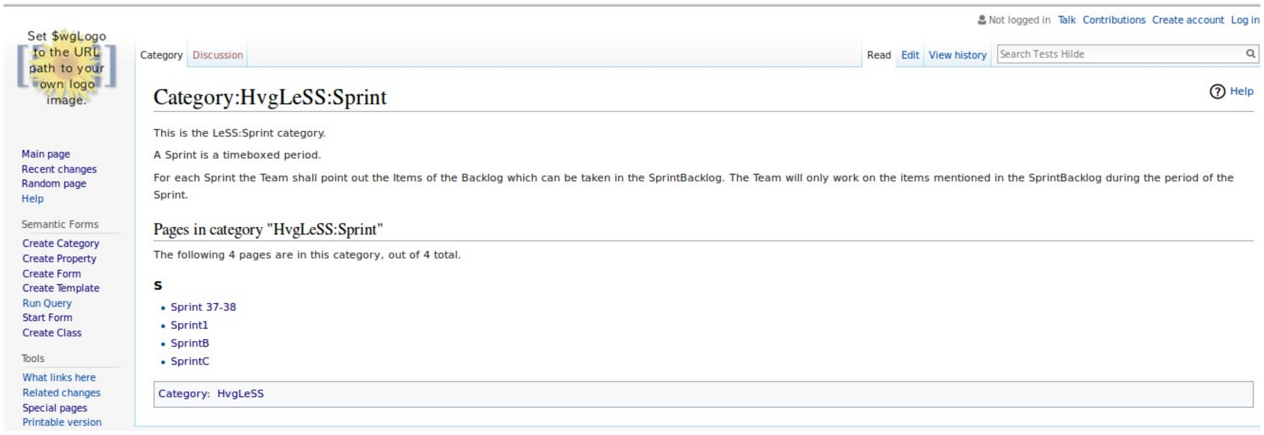


Figure 5: Semantic MediaWiki architecture (Markus Krötzsch et al. 2007)

MediaWiki uses pages, hyperlinks, namespaces, categories and templates to structure information (Markus Krötzsch et al. 2007).

- Pages: The wiki pages in which content is organized with each page describing a subject. So each word existing in Wikipedia has its page, describing that word.
- Hyperlink: A page related to another page is linked by a hyperlink.
- Namespaces: A page belongs to just one namespace, recognizable by using a prefix in the page name, e.g. `User:Hilde`, `Category:HvgLeSS` or `Help:MediaWiki`.
- Categories: A page can have no, one or more categories, recognizable by the categories shown at the bottom of the wiki page (Figure 6).
- Templates: A template makes reuse of text on several pages possible. The content of templates is designed to be embedded inside pages. Each template belongs to the template namespace. To insert a template on a page, call the template by its name enclosed by double

braces. For example to call the `Template:Hello` on the page `Intro`, insert the code `{{Hello}}` on the `Intro` page where the text of the template is wanted.



Category:HvgLeSS:Sprint

This is the LeSS:Sprint category.

A Sprint is a timeboxed period.

For each Sprint the Team shall point out the Items of the Backlog which can be taken in the SprintBacklog. The Team will only work on the items mentioned in the SprintBacklog during the period of the Sprint.

Pages in category "HvgLeSS:Sprint"

The following 4 pages are in this category, out of 4 total.

S

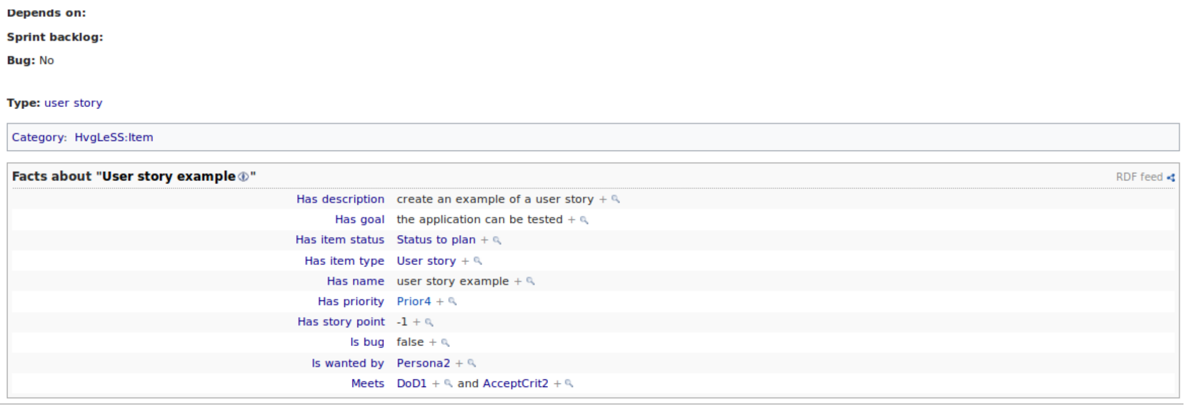
- Sprint 37-38
- Sprint1
- SprintB
- SprintC

Category: HvgLeSS

Figure 6: Display of categories to which a page belongs to

But these structures do not solve the problems of consistency of content, accessing and reuse of knowledge (Markus Krötzsch et al. 2007). Semantic MediaWiki makes annotations available. The underlying conceptual framework uses a binary relationship between the wiki page and an entity or data value (Markus Krötzsch et al. 2007). This relationship is described by a so named property.

The property itself is set for the subject by using the property name and its value on the page that describes the subject. On the wiki page only the value is visible for the user in read mode, so the reader will only see the value. For example the notation `[[Has item type:: user story]]` set on the page `All information accessible` that describes an item of the backlog, gives the property `Has item type` the value `user story` for the page `All information accessible`. If the property `Has item type` does not exist, a new page will be created with its name in the property namespace, thus the page name will be `Property:Has_item_type`. The default value for the type of the property is set on Page, but can be overwritten by another data type such as String, Date or Number mentioned in Figure 5 under Datatype API. When a page is chosen, the property value will have a hyperlink to the page `User_story`. The information of the properties used on a page is shown in the fact-box (Figure 7).



Depends on:

Sprint backlog:

Bug: No

Type: user story

Category: HvgLeSS:Item

Facts about "User story example" RDF feed

Has description	create an example of a user story +
Has goal	the application can be tested +
Has item status	Status to plan +
Has item type	User story +
Has name	user story example +
Has priority	Prior4 +
Has story point	-1 +
Is bug	false +
Is wanted by	Persona2 +
Meets	DoD1 + and AcceptCrit2 +

Figure 7: Example factbox for a user story item

Another important aspect of Semantic MediaWiki needed for our work and mentioned in Figure 5 are inline queries. An inline query allows to create dynamic content by displaying the result of queries on the page (Markus Krötzsch et al. 2007). This query can be called by the use of the tag-function, enclosing the wiki text of the query by `<ask>` and `</ask>` or by the use of the parser function, enclosing the ask-parser function and the wiki text of the query by double braces, thus `{{#ask:` *`wiki text of the query`* `}}`.

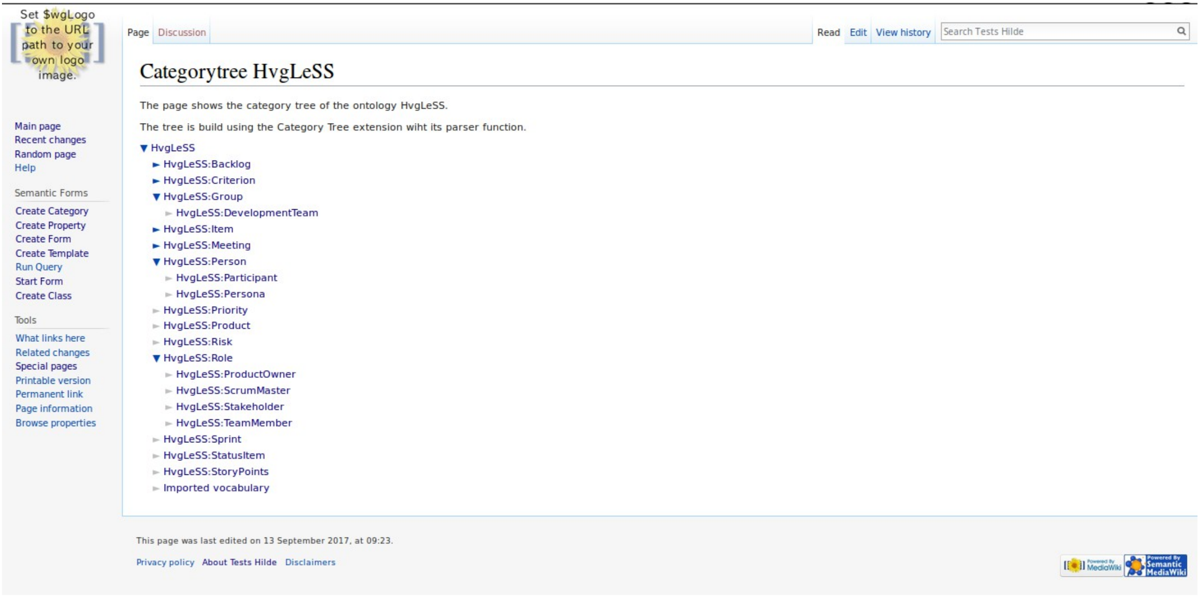
4.3 Used MediaWiki extensions

In addition to the Semantic MediaWiki extension, we used some others. We give a brief description of the used extensions.

4.3.1 Extension CategoryTree

An overview of the categories is very useful during the development of an ontology within Semantic MediaWiki. We could use an inline query but the function is available in the CategroryTree extension (Kinzler 2017). This extension provides as the name suggests a dynamic tree of categories. Hence, we decided to not implement queries but to use the extension.

The extension allowed us to have at each moment an overview of the categories created for our proof of concept and their subcategories Figure 8. The categories itself are explained under 5.3.



Categorytree HvgLeSS

The page shows the category tree of the ontology HvgLeSS.

The tree is build using the Category Tree extension wiht its parser function.

```
▼ HvgLeSS
   ► HvgLeSS:Backlog
   ► HvgLeSS:Criterion
   ▼ HvgLeSS:Group
      ► HvgLeSS:DevelopmentTeam
   ► HvgLeSS:Item
   ► HvgLeSS:Meeting
   ▼ HvgLeSS:Person
      ► HvgLeSS:Participant
      ► HvgLeSS:Persona
   ► HvgLeSS:Priority
   ► HvgLeSS:Product
   ► HvgLeSS:Risk
   ▼ HvgLeSS:Role
      ► HvgLeSS:ProductOwner
      ► HvgLeSS:ScrumMaster
      ► HvgLeSS:Stakeholder
      ► HvgLeSS:TeamMember
   ► HvgLeSS:Sprint
   ► HvgLeSS:StatusItem
   ► HvgLeSS:StoryPoints
   ► Imported vocabulary
```

This page was last edited on 13 September 2017, at 09:23.

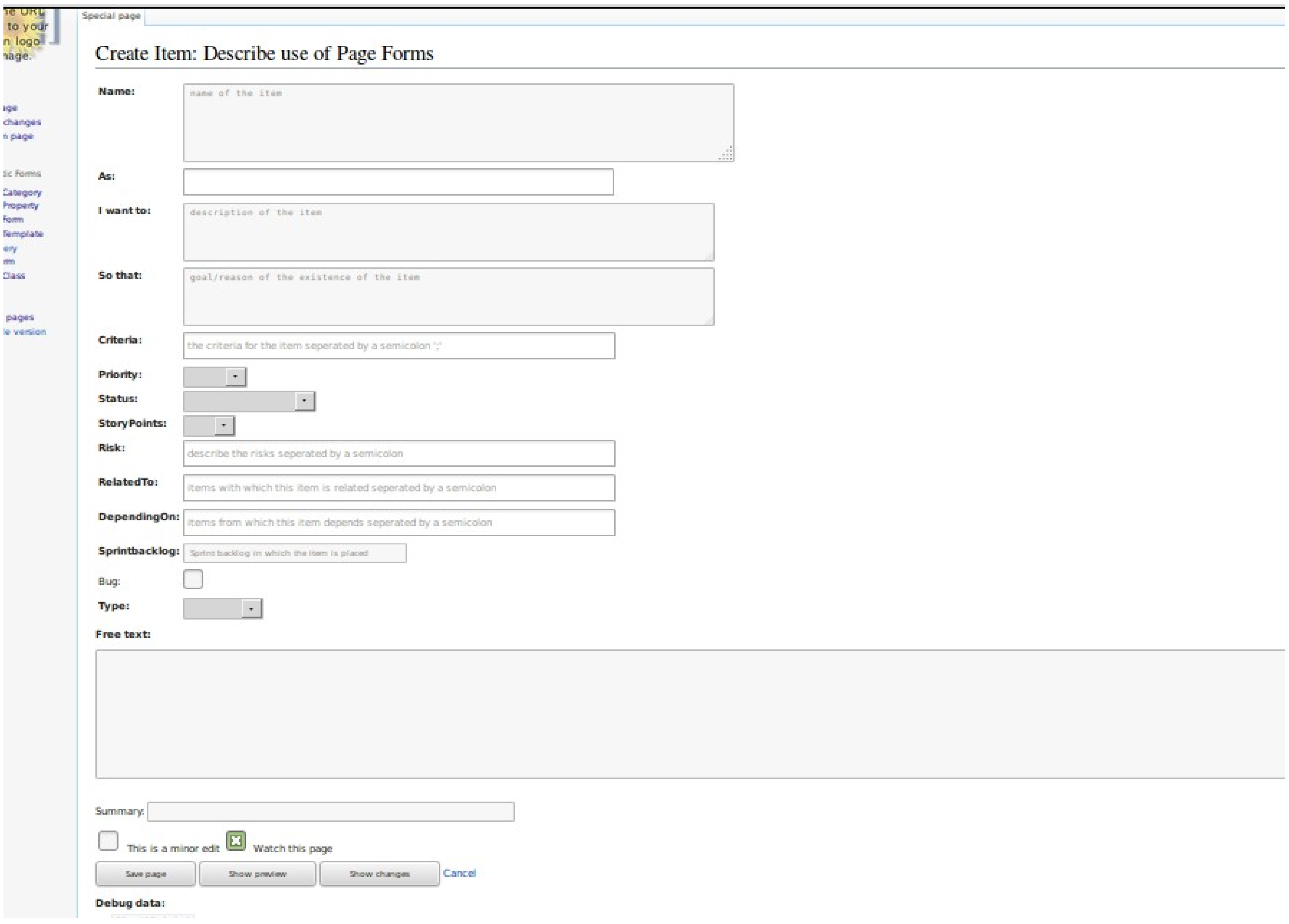
Figure 8: Category tree for the root category HvgLeSS

4.3.2 Extension GetUserName

The GetUserName extension (Ejcaputo 2010) obtains the current user-name from the MediaWiki global variable `wgUser`. The user-name of the current user is called by the parser function `{{#username:}}`. The function allowed us to change the page content based on the role the user has.

4.3.3 Extension Page Forms

The extension Page Forms (Koren et al. 2017) uses a framework to design and store forms included in this extension. It provides special pages by using forms to add and edit articles.


Special page

Create Item: Describe use of Page Forms

Name:	name of the item
As:	
I want to:	description of the item
So that:	goal/reason of the existence of the item
Criteria:	the criteria for the item seperated by a semicolon ';'
Priority:	
Status:	
StoryPoints:	
Risk:	describe the risks seperated by a semicolon
RelatedTo:	items with which this item is related seperated by a semicolon
DependingOn:	items from which this item depends seperated by a semicolon
Sprintbacklog:	Sprint backlog in which the item is placed
Bug:	☐
Type:	

Free text:

Summary:

☐ This is a minor edit ☒ Watch this page

Debug data:

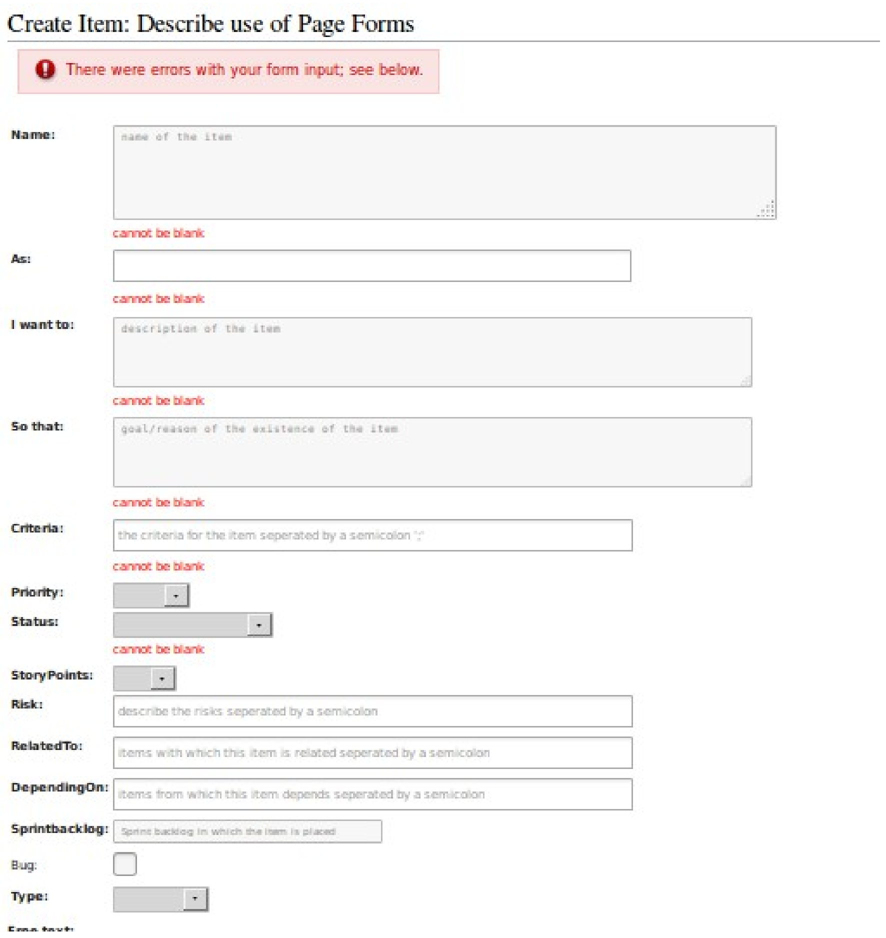
Figure 9: View of the item creation or edit form.

The form helps users to input information and allows to give information on the content wanted in the fields (Figure 9). The input can be mandatory or limited by the value through use of for example a type of the input or drop-down-boxes, available when the property allows value is used. The error handling for these limitations is done by the extension itself (Figure 10).

The details for the implementation are described under 7.4.

Help can also be given to the user by the auto complete function, default values or the description of the needed content for the field (Figure 9 and Figure 10).

Besides the help for completing information, Page Forms helps to build new forms. The form is created on one or more templates, mostly based on categories. The template sets the fields, a label and the property to define the relationship with the page the field is set on (Figure 11). By defining the form, the input type, restrictions and added information is set (Figure 12). Both of these actions are supported by the extension.



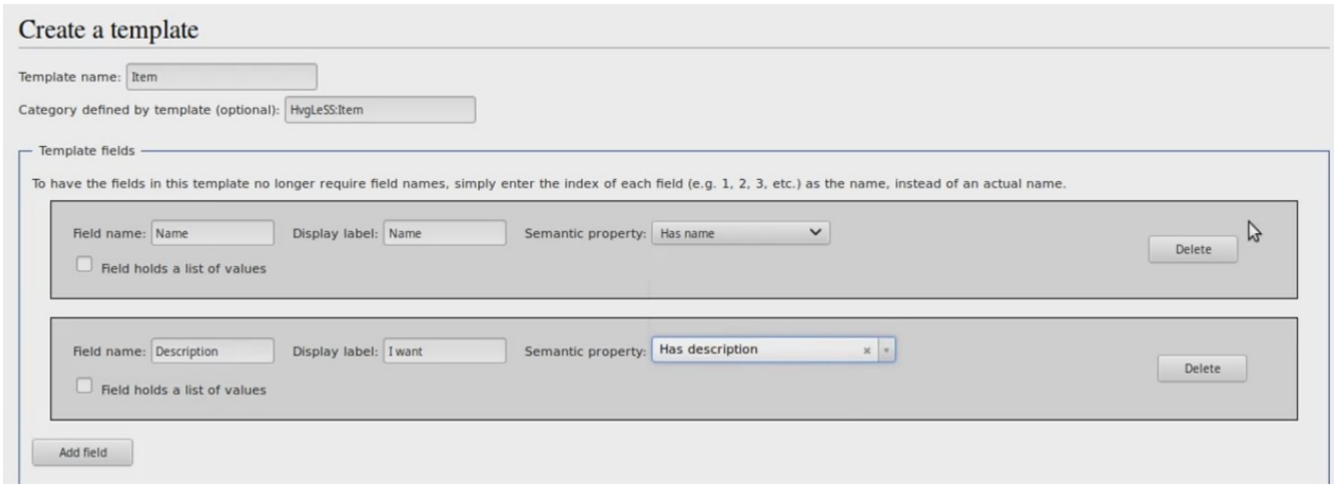
Create Item: Describe use of Page Forms

There were errors with your form input; see below.

Name:	name of the item
	cannot be blank
As:	
	cannot be blank
I want to:	description of the item
	cannot be blank
So that:	goal/reason of the existence of the item
	cannot be blank
Criteria:	the criteria for the item seperated by a semicolon ";"
	cannot be blank
Priority:	
Status:	
	cannot be blank
StoryPoints:	
Risk:	describe the risks seperated by a semicolon
RelatedTo:	items with which this item is related seperated by a semicolon
DependingOn:	items from which this item depends seperated by a semicolon
Sprintbacklog:	Sprint backlog in which the item is placed
Bug:	☐
Type:	

Figure 10: Error given by the extension Page Forms

Page Forms is a redesign of the former Semantic Forms. The latter were based on the Semantic MediaWiki extension and its way to storage data. This dependence on Semantic MediaWiki is cleared by Page Forms through the use of forms markup code. For storage of structured information, collected by Page Forms, the MediaWiki extension Cargo or Semantic MediaWiki is required. Our work is based on Semantic MediaWiki and thus used the latter extension.



Create a template

Template name: Item

Category defined by template (optional): HvgLeSS:Item

Template fields

To have the fields in this template no longer require field names, simply enter the index of each field (e.g. 1, 2, 3, etc.) as the name, instead of an actual name.

Field name: Name Display label: Name Semantic property: Has name
☐ Field holds a list of values

Field name: Description Display label: I want Semantic property: Has description
☐ Field holds a list of values

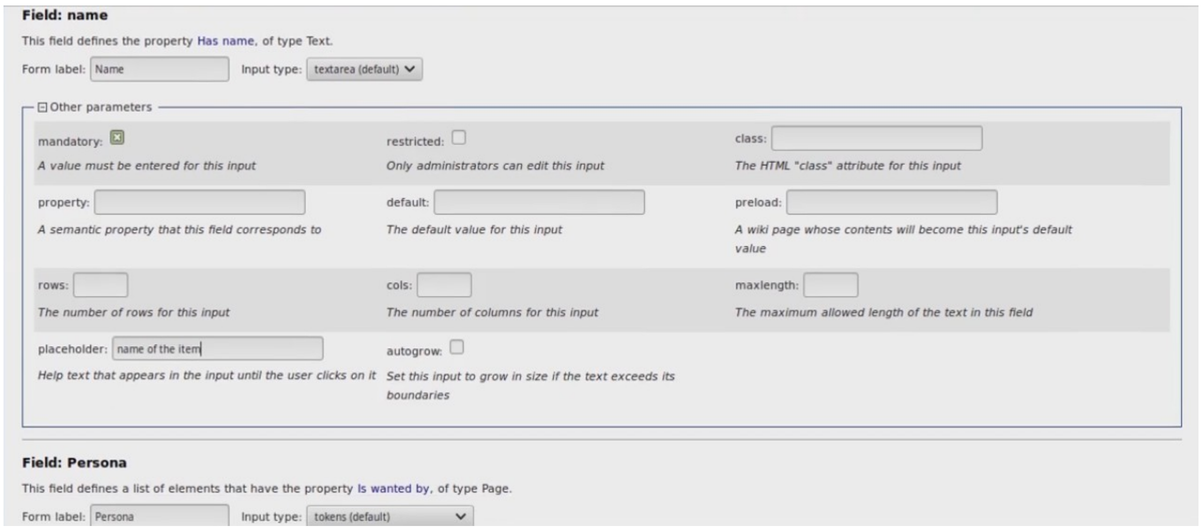
Figure 11: Create template provided by Page Forms

Additionally, help can be given to the user by the auto complete function, default values or the description of the needed content for the field (Figure 12).

The above description of Page Forms is comparable with the working of Fresnel. Rutledge defined Fresnel as (Rutledge et al. 2016):

> *'Fresnel is a Semantic Web ontology for the presentation of data from given Semantic Web ontologies'.*

In Page Forms the menu options of the fields for properties of the category are comparable with the external Semantic Web ontology of the class to be presented by the Fresnel declaration.



Field: name

This field defines the property Has name, of type Text.

Form label: Name Input type: textarea (default)

⊟ Other parameters

mandatory: ☒ *A value must be entered for this input*	restricted: ☐ *Only administrators can edit this input*	class: *The HTML "class" attribute for this input*
property: *A semantic property that this field corresponds to*	default: *The default value for this input*	preload: *A wiki page whose contents will become this input's default value*
rows: *The number of rows for this input*	cols: *The number of columns for this input*	maxlength: *The maximum allowed length of the text in this field*
placeholder: name of the item *Help text that appears in the input until the user clicks on it*	autogrow: ☐ *Set this input to grow in size if the text exceeds its boundaries*	

Field: Persona

This field defines a list of elements that have the property Is wanted by, of type Page.

Form label: Persona Input type: tokens (default)

Figure 12: Create form provided by Page Forms

4.4 Used Semantic MediaWiki extensions

4.4.1 Extension Semantic Result Formats

Semantic Result Formats (SRF) (De Dauw et al. 2017) makes a lot of formats available to use in inline queries of searches.

Within LeSS the sum of story points has to be calculated. SRF allowed us to calculate the sum in one single query. Another possibility is the use of the expression function which needs querying for each item the number of story points conjunct by the expression symbol +.

Further on, the sprint burn-down chart could be created by using SRF. We did not include this burn-down in our work.

4.4.2 Extension RDFIO

The extension RFDIO (Lampa et al. 2017) provides the import and export of RDF triples in Semantic MediaWiki. The extension follows the implementation of MediaWiki by the use of PHP/MySQL based triple store. The SPARQL endpoint allows RDFIO to write operations using the ARC2 library.

> *'The RDF import stores the original URI of all imported RDF entities in the `Equivalent URI` property, which can later be used by the SPARQL endpoint, instead of SMW's internal URIs, which thus allows to expose the imported RDF data "in its original formats", with its original URIs. This allows to use SMW as a collaborative RDF editor, in workflows together with other semantic tools, from which it is then possible to "export, collaboratively edit, and import again", to/from SMW.' (Lampa et al. 2017)*

The property `Equivalent URI` is used in our work to link the ontology of HvgLeSS with the K-CRIO and FoaF ontology.

4.5 Semantic MediaWiki and Fresnel Forms

As demonstrated by Rutledge (Rutledge et al. 2016) info-boxes can be designed by the use of Fresnel Forms. The information to show is defined by a Fresnel lens, defining the properties to display and the ordering of these. How properties are rendered is respecified by the Fresnel format[3].

Within Semantic MediaWiki the similar functionality as Fresnel Forms can be achieved by the use of inline queries for the selections defined by lenses and the skins and style-sheets for the rendering. The output will mostly be wiki text, but can be set for example to a graph by using the Semantic Result Formats. Our proofs of concepts described in chapters 6 and 7 use lenses in a Semantic MediaWiki-way.

4.6 Conclusion

The MediaWiki Platform, extended with the extensions CategoryTree, GetUserName, Page Forms, Semantic MediaWiki, Semantic Result Formats and RDFIO, covers our requirements, stipulated under 4.2. The extensions allow us to query the user (4.3.2), format results (4.4.1), link the semantic content of our proof of concept with known ontologies (4.4.2) and create an overview of the added categories to complete the ontology for LeSS (4.3.1). Last but not least, the extension Page Forms, comparable with the Semantic Web Fresnel ontology for presenting Semantic Web classes, (4.3.3) helps the user to input needed information. We even found a similarity between Fresnel Forms and the rendering of inline queries within Semantic MediaWiki (4.5). Thus, the MediaWiki platform with its extensions covers the requirements we defined for our proof of concept.

3 https://www.w3.org/2005/04/fresnel-info/manual/

5 LeSS ontology

5.1 Introduction

In the literature we found one ontology for agile, named K-CRIO (Lin et al. 2011) and described it under 3.3.2. Besides the study for reuse of this ontology, the check had to be made if our work could be linked with other ontologies, such as FoaF.

In this section we searched for the answer for the question *b) How and to what extent can we use an existing ontology for agile methodology?* by searching for the equivalence between the terms used in LeSS and in the K-CRIO ontology. Therefore we selected the terms which have the same meaning for scrum and LeSS (5.2) and defined the ontology used in our work (5.3) by designing a global LeSS ontology, named HvgLeSS. With the idea of Berners-Lee to create the Semantic Web (Lassila, T Berners-Lee, J Hendler 2001) with linked data (Bizer et al. 2009), we searched for other ontologies which are related to ours (5.4). At the end of this section, we summarized our findings in a conclusion (5.5).

5.2 Reuse of K-CRIO

Earlier studies at the Open University of the Netherlands proved that the import of an ontology with its style definitions, defined by the use of Fresnel Forms, is possible (Rutledge et al. 2016). Further on, Lin developed a K-CRIO ontology for scrum (Lin et al. 2011), described under 3.3.2 and available as OWL-file created using Protégé (see Annex 12.1). The ontology can be imported using the RDFIO extension, discussed under 4.4.2.

Lin based the K-CRIO ontology on the organization which includes roles. These roles can be performed by a person, an activity or a service. Each role requires capabilities. The organization has to provide all these capabilities to accomplish the tasks (Lin et al. 2011).

Translated to scrum, which is an organization, she defined two sub organizations namely `Pig` and `Chicken`. The organization `Pig` contains the product owner, the scrum master and a development team, containing itself five to nine members. The other organization, `Chicken`, contains the managers and stakeholders. The latter is split into the group of customers and vendors (Lin et al. 2011).

Starting from the schema given by Lin and copied in our work under Figure 3, we have drawn a Venn diagram of the described roles and organizations in Figure 13.

All groups are described as organizations and people in the organization have roles. The root organization in the diagram is scrum, which has sub organizations, drawn as sub diagrams and marked in Figure 3 as `isSubOrganizationOf` predicate. The relation between organizations and roles is given by the predicate `includes` and these between roles and tasks or skills by `provided`.

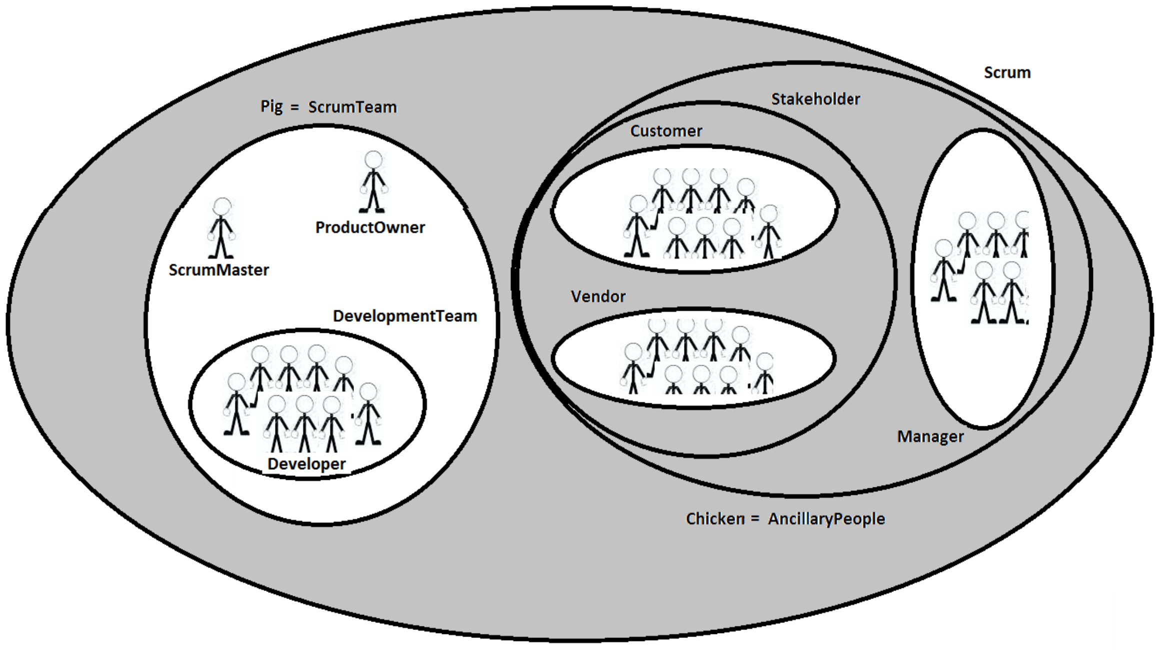


Figure 13: Venn diagram of organizations and roles described by Lin (Lin et al. 2011)

The roles of the so-called Ancillary people can be used without any adaptation in our work, but not for the scrum team. Per definition a scrum team contains just one development team, one scrum master and one product owner. The LeSS methodology has also just one product owner per product, but the number of scrum masters and development teams differs. Thus LeSS requires (The LeSS Company B.V. 2016a):

- Just one product owner
- One or more scrum masters
- Two or more development teams, because when there is only one we do not need a large scale scrum.

By creating a sub organization scrum master team and allowing multiple development teams, the roles are LeSS compliant. But changing the cardinality or the definition of a term is not wanted because at that moment semantic reasoning will compare apples to oranges.

The tasks and skills described by Lin (Lin et al. 2011) are scrum and thus also LeSS compliant. Using only one predicate, namely `includes,` to describe the different roles in the organization, makes it difficult to reason because the predicate is not significant. The relationship between the role and the tasks, `provided`, creates the same problem.

Based on the research above, we had to conclude that terms used in the K-CRIO ontology and in LeSS have a slight difference in meaning which makes the reuse of the K-CRIO ontology as such in our work not possible. For this reason we opted to describe a new ontology that has to be linked to K-CRIO.

Hereby we found a partial answer to the second part of the question *b How and to what extent can we use an existing ontology for agile methodology?*. An ontology exists, namely K-CRIO. We could reuse only parts of the K-CRIO ontology defined by Lin (Lin et al. 2011). Slight differences showed that a total reuse is not possible, so we searched for the answer how and to what extent we could reuse parts of existing ontologies in the paragraphs below.

5.3 HvgLeSS ontology

As our work searched to answer the questions *d) Which interface can we use in our case study to allow the interaction with the user?*, *e) How and to what extent can Semantic MediaWiki technology provide an interface to display the progress of product development?* and the not yet totally answered question *b) How and to what extent can we use an existing ontology for agile methodology?*, we described here the categories directly linked in our research.

An overview of the created categories is shown under 4.3.1 in Figure 8. To illustrate the relations between the different categories, we have drawn an information model (Figure 14) for the LeSS ontology.

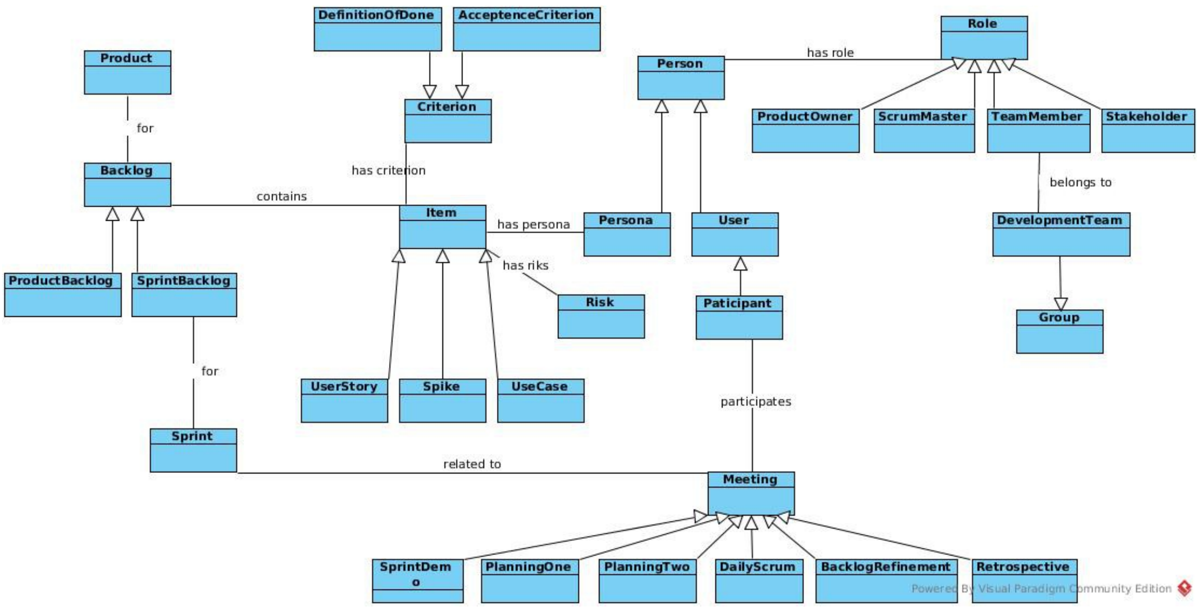


Figure 14: Domain Model HvgLeSS

Under this paragraph we discussed the role (5.3.1), sprint (5.3.2) and item (5.3.3) category which offered all components for a proof of concept on the view of the sprint for different roles.

5.3.1 HvgLeSS:Role

Under 5.2 we concluded that `AncillaryPeople` could be reused for the HvgLeSS ontology. This work only needed the role of stakeholder, so `Stakeholder` is the only `AncillaryPeople` class used from K-CRIO to implement our proof of concept.

The definition of members of the team is the same for both agile methodologies, scrum and LeSS. For the development team, the only difference is the cardinality (see 5.2). For these reasons we could conclude that the terms have the same meaning for scrum and LeSS.

In contrast to the equality of the definition for members, product owner has a different meaning in scrum and for LeSS. On the one hand, we could state that for product owner there is the same difference as for development team because only the cardinality changes. On the other hand, we generalized development team by making more teams possible. For the product owner a specification is needed to limit the number to only one per product. We did not find this specification in the K-

CRIO ontology. This subtle difference of cardinality makes a lot of difference to keep the overview of the product. Hence, with the idea in mind that we have to avoid comparing apples to oranges, we decided to create a specific definition for the role of product owner in LeSS. This decision did not allow us to link `HvgLeSS:ProductOwner` to `K-CRIO:ProductOwner` by an `equivalent URI` property. So we created only one triple being `HvgLeSS:ProductOwner` is a `HvgLeSS:Role` by making the category `HvgLeSS:ProductOwner` a subcategory of `HvgLeSS:Role`.

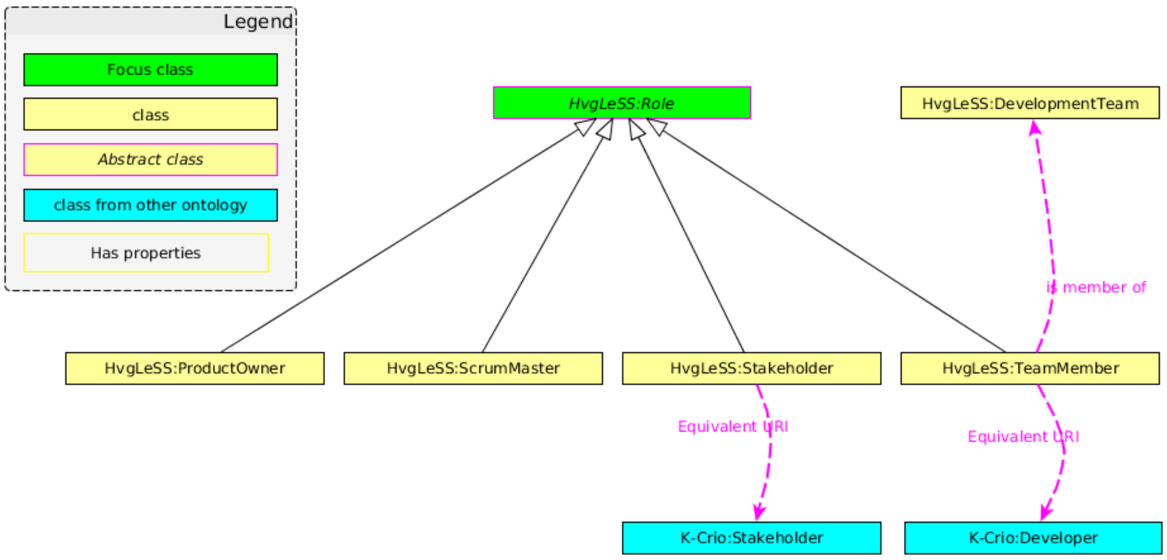


Figure 15: HvgLeSS:Role

For the role of scrum master we saw a similar problem. Scrum works with one team and thus with only one scrum master. The link of the scrum master and the team exists in LeSS, each team has one specific scrum master. But for LeSS a scrum master is not limited to only one team. He or she can be scrum master over different teams, working for the same product. Generalizing scrum master, as we did with development team, would make it impossible to relate a specific scrum master to one or more development teams. Besides, scrum allows a scrum master to be part of the team, LeSS does not. For this reason we concluded that the definitions for scrum and LeSS are not equivalent and opted to not relate `HvgLeSS:ScrumMaster` to `K-CRIO:ScrumMaster`.

Our conclusions allowed for some of the classes defined by Lin (Lin et al. 2011) to be reused and others not. To avoid confusion and fault interpretations, the creation of a new ontology was needed. The idea to name the ontology 'LeSS' is dropped because MediaWiki uses the style-sheet languages Less[4]. Using the same term for style-sheets and the new ontology could lead to misunderstanding. Thus, we added our initials to the name of the ontology and got the name HvgLeSS.

Figure 15 gives a summary on the above description. We described the role of LeSS as `HvgLeSS:Role`, discussed the different roles needed in our work and linked these where possible with the K-CRIO ontology described by Lin (Lin et al. 2011). These items are linked via the RDFIO property `Equivalent URI`, which will be translated into Semantic Web predicate `sameAs`. In this way the link of the ontology for scrum by Lin (Lin et al. 2011) and the created ontology HvgLeSS are related and allow semantic reasoning.

4 https://www.mediawiki.org/wiki/Extension:TemplateStyles

5.3.2 HvgLeSS:Sprint

One of the main categories for LeSS is the sprint, because it contains all the work to do during the current period. Not all users look in the same way at a sprint.

- A team member wants to know which issue or task to pick up next.
- A development team wants to know how the work progresses during the sprint.
- A scrum master needs to know if there are any blocking issues or when instructions, help or interactions with other teams or the product owner have to be initiated.
- A stakeholder wants to know what will be delivered and how the delivered product can help increase the productivity.
- A product owner wants to see the progress and uses it to define the items to discuss during the refinement meeting for items to be developed in the next sprints.

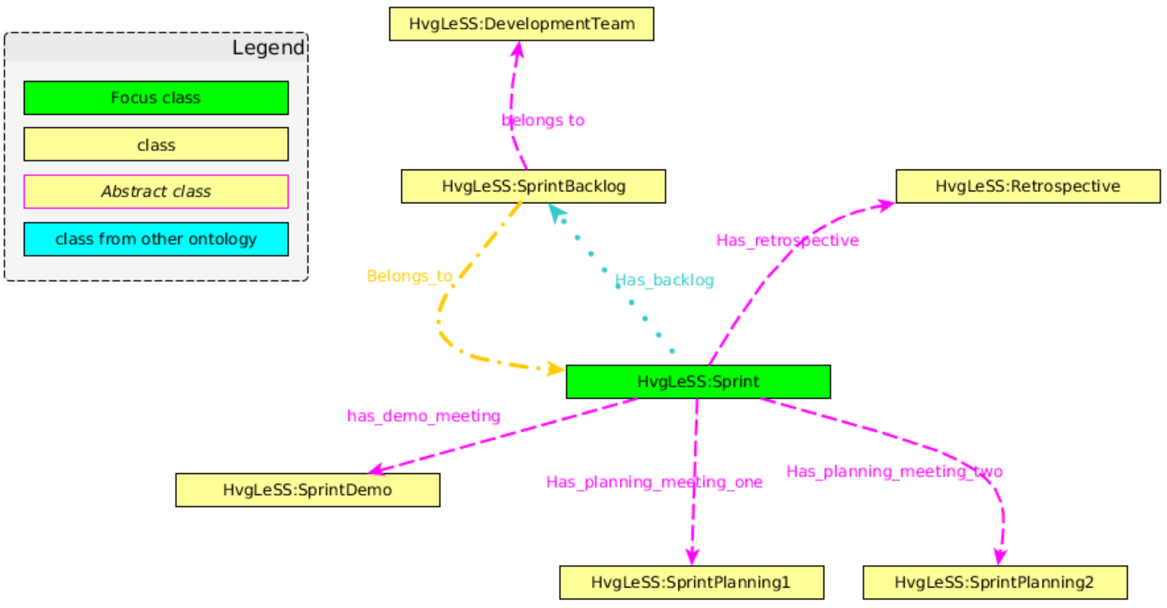


Figure 16: Relationships of HvgLeSS:Sprint

The properties of the `HvgLeSS:Sprint` and the main links with other categories are shown in Figure 16. The triples created and used in chapter 6 are:

- `HvgLeSS:Sprintbacklog Belongs_to HvgLeSS:DevelopmentTeam`
- `HvgLeSS:Sprint Has_backlog HvgLeSS:SprintBacklog`

5.3.3 HvgLeSS:Item

The category `HvgLeSS:Item`, shown as central category in Figure 17, represents one of the most used categories of our application. The `HvgLeSS:Item` category has three subcategories, namely `HvgLeSS:Spike`, `HvgLeSS:UseCase` and `HvgLeSS:UserStory`. We considered the abstraction of `HvgLeSS:Item` and the type-specific properties (i.e. the properties of the three subcategories) defined on this parent class out of scope. Considering that the implementation is a proof of concept, we opted for the solution described above over the possibility to only create one type of Item.

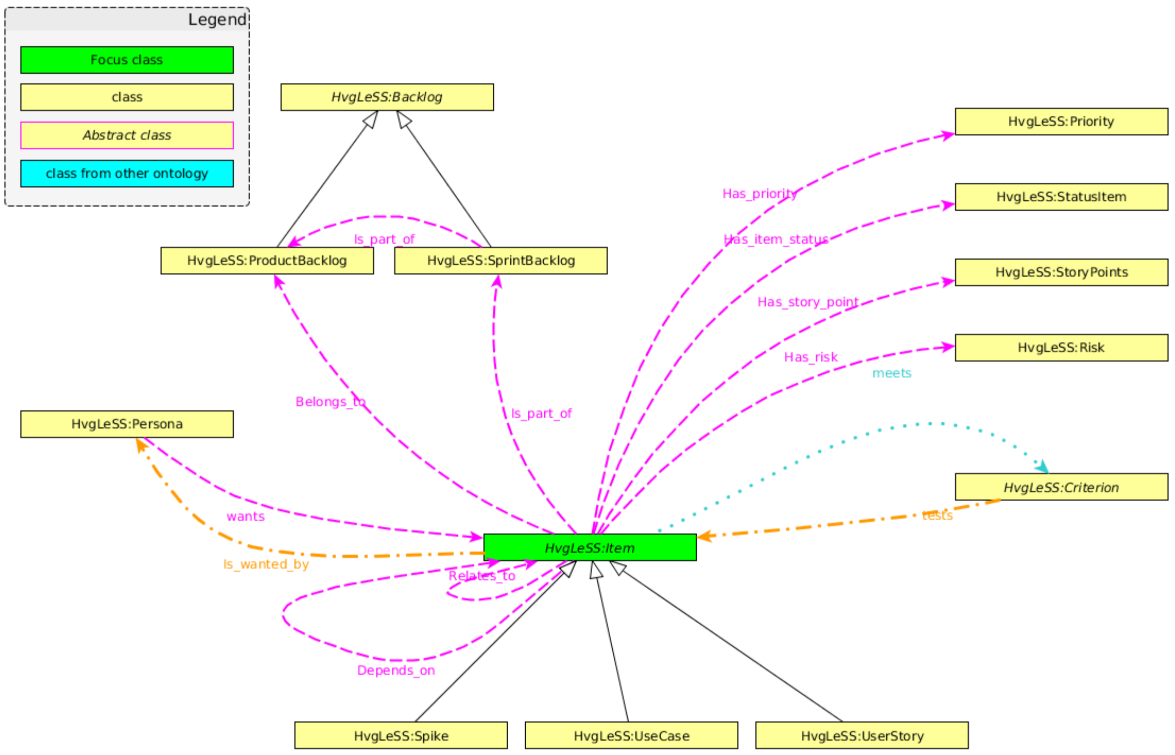


Figure 17: Category HvgLeSS:Item

The K-CRIO ontology (Lin et al. 2011) does not describe an item or a subclass of it which made linking `HvgLeSS:Item` to this ontology impossible.

5.4 Other linked ontologies

The Friend of a Friend (FoaF) ontology is a well-known ontology which describes people and links people with information using the Web by social networks, friendship, and representational networks (Brickley & Miller 2014). One of the main classes within the FoaF ontology is `Person` (Chen et al. 2005), a class our work also needs. By limiting the properties of person to name and mbox, we could use the `foaf:Person` class for our category `HvgLeSS:Person` by importing the FoaF vocabulary into Semantic MediaWiki.

Linking our categories with the FoaF ontology helped us to make some decisions:

- Our category Persona describes an archetypical user. Due to the fact that imaginary persons are included in the `foaf:Person` class, we opted to make Persona a subcategory of our category Person.
- Our category FeatureTeam defines a group of persons who work together on one product feature. The FoaF ontology has a class `foaf:Group` which represents a collection of individual agents[5], with Agent being a super-class of Person. Hence, we created a new category Group, defined it as imported from `foaf:Group` and made it a super-category of `DevelopmentTeam`. In addition to the creation of the category we also needed the property `foaf:member`, so we created a subcategory of the property `is part of` and named it `Member` and made it equivalent with the `member` property of the FoaF ontology (see also Figure 18).

5 http://xmlns.com/foaf/spec/#term_Group

- Class `foaf:Project` is not used in our work. First of all, LeSS works on products, not on projects and second the project class is still in test.
- The best way to identify persons in a unique way is by use of email addresses. The same email address can never occur more than once. Hence, the predicate `mbox`, containing the email address, used within the FoaF ontology is used as property for the category `HvgLeSS:Person`. Thus reasoning will allow concluding that FoaF and HvgLeSS are talking about one and the same person.

By the use of the property `imported from` we linked our ontology to existing ones and expanded these with information of people working together to create a product using the LeSS framework.

We picked the category `HvgLeSS:Person` to illustrate here in more detail the categories with links to existing and published ontologies. We could not use the same technique for the reuse of K-CRIO because this ontology is not published. Fortunately, Semantic MediaWiki provided us the RDFIO extension (Lampa et al. 2017) which allows the use of the property `equivalent URI` to link categories with vocabularies that could or are not imported into Semantic MediaWiki (see Figure 18). By the use of these two properties we could link the information of our ontology with other ones, and so help build the Semantic Web (Lassila, T Berners-Lee, J Hendler 2001).

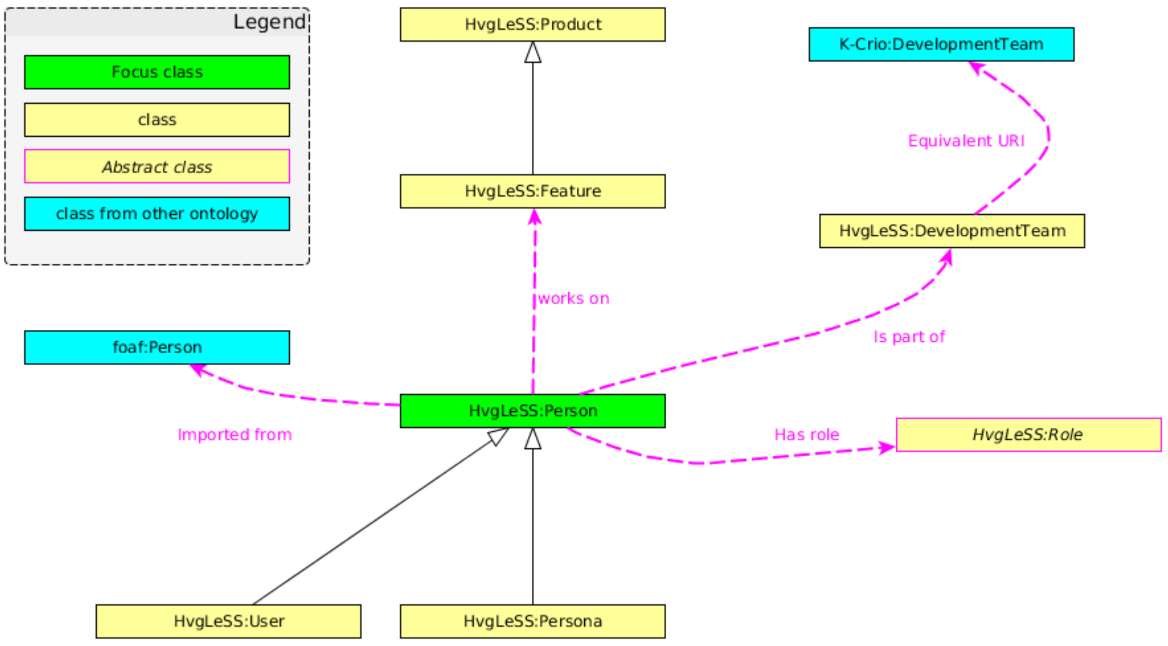


Figure 18: Category HvgLess:Person

The structure of the categories in Figure 8 (see 4.3.1) gives only an overview of the used categories. Figure 18 describes the properties of the categories related to the category `HvgLeSS:Person`. Not only the links and properties between our own created categories are described by this view, but also

- the `Equivalent URI`-links with the K-CRIO ontology, provided by the RDFIO Semantic MediaWiki extension (Lampa et al. 2017) and translated in Semantic Web predicates as `sameAs`.
- The imported from property provided by Semantic MediaWiki to import an existing vocabulary, in our case Friend of a Friend, with the definitions given in its own ontology.

In this way we linked our ontology to already existing ones. The links to the FoaF ontology are described above and these to the K-CRIO ontology in the previous sub paragraphs of 5.3.

We summarized the link made to the K-CRIO ontology (Lin et al. 2011) shown in Figure 18.

- The terms, and therefore also the classes, stakeholder, development team and developer have the same definition in scrum and LeSS. Hence we can conclude that the URIs are equivalent.
- Product owner is described in the K-CRIO ontology as a member of the Developing Team. Due to the fact that the LeSS framework is designed to work with multiple teams on the same product and that there is only one single product owner responsible for the product, we do not have an equivalency. Hence, `K-Crio:ProductOwner` is not present in Figure 18 because there is no equivalent URI with `HvgLeSS:ProductOwner`.
- For scrum master, a similar reasoning can be made. LeSS uses multiple development teams while scrum uses only one. So we decided to not use the `equivalent URI` property.
- For product we opted to not link the terms as equivalent between the K-CRIO and HvgLeSS ontology because in K-CIO the product is provided by a developing team, where in LeSS a product is split in features and each feature is handled by one or more development teams. By not making the product equivalent we created the ability to capture differences and the possibility to let both the ontologies expand without interfering with one another. For this case, the HvgLeSS ontology has to be expanded by including feature team between `HvgLeSS:DevelopmentTeam` and `HvgLeSS:Group` categories. In our work we did not need and thus skipped the feature team.

5.5 Conclusion

In this paragraph we documented some of the used categories by explaining the reason of their creation when reuse of the existing K-CRIO ontology was not possible. We started with the study of the K-CRIO ontology to search for possible reuse. This study taught us that we could only use a part of the classes defined by Lin (Lin et al. 2011). Because of this conclusion we opted to create a new ontology that links to existing ones and thus contributes to link the data, available on the web (Bizer et al. 2009). Hence, the HvgLeSS ontology is not only linked to K-CRIO -via the `Equivalent URI` property, provided by RDFIO (Lampa et al. 2017), but also to FoaF -via the `Imported from` property provided by Semantic MediaWiki-, by person and group. This way we could deliver a modest contribution to the Semantic Web and its capability for reasoning.

Hence, we found an answer for the question b) *How and to what extent can we use an existing ontology for agile methodology?*. We used the existing ontologies indirectly, by the use of the RDFIO property `Equivalent URI`. This decision facilitated to answer the second part of the question because wanted relationships can be added. We decided to be prudent and thus not create an equivalent URI property when doubts arose.

6 Role-dependent page view

6.1 Introduction

Different roles in LeSS focus on different views on the information. For instance, a product owner needs an overview of the progress for all sprint backlogs for the product, whereas a team member is most of the time only interested in the progress of his team and wants to see which item has to be picked next. Within Semantic Web Fresnel creates views by lenses on classes, but we have not found any view depending on the user information. We studied the possibility to create a page view which depends on the role of the user who is logged in the wiki by the use of Semantic MediaWiki.

In this chapter we first discussed the implementation of the categories, introduced in the previous chapter, in more detail under section 6.2, by describing the implementation for the categories role (6.2.1) and person (6.2.2). The used extensions are explained: Semantic Result Formats (6.3) and UserGetName (6.4). Combining the ontology, the extensions and Semantic MediaWiki, a page, determining the view based on the role of the user, is created in the wiki (6.5). This chapter gave partially answers for the questions *a) How and to what extent can business rules guide the team through the scale agile development process?* and *e) How and to what extent can Semantic MediaWiki technology provide an interface to display the progress of product development?*, which are formulated under the conclusions (6.6).

6.2 HvgLeSS Category

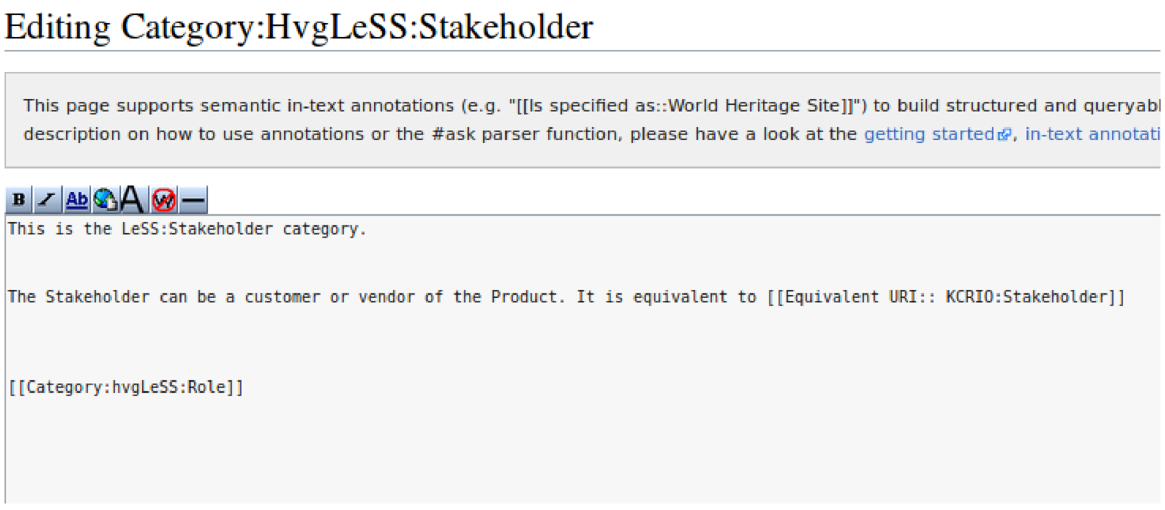
Under this paragraph we described the categories and properties created within Semantic MediaWiki to allow the implementation of our proof of concept.

6.2.1 HvgLeSS:Role

For our proof of concept we focused on the following sub-categories of the HvgLeSS:Role category Figure 15:

- product owner
- scrum master (Figure 19)
- team member
- stakeholder

As decided in 5.3.1, we declared that stakeholder and team member in our ontology are the same as respectively stakeholder and developer defined in K-CRIO. This relationship is created by the use of the `Equivalent URI` property for the mentioned categories and set by the MediaWiki text notation `[[Equivalent URI:: KRCIO:Stakeholder]]` (Figure 19).


Editing Category:HvgLeSS:Stakeholder

This page supports semantic in-text annotations (e.g. "[[Is specified as::World Heritage Site]]") to build structured and queryabl description on how to use annotations or the #ask parser function, please have a look at the getting started, in-text annotati

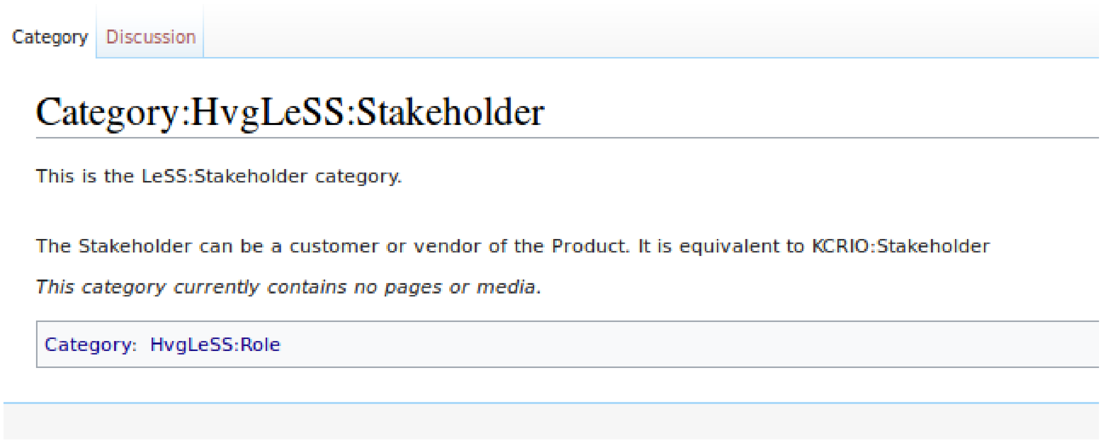
```
This is the LeSS:Stakeholder category.

The Stakeholder can be a customer or vendor of the Product. It is equivalent to [[Equivalent URI:: KCRIO:Stakeholder]]


[[Category:hvgLeSS:Role]]
```

Figure 19: Editing category HvgLeSS:Stakeholder

Depending on the skin chosen in MediaWiki the view of the category page can change. Our wiki uses the default Vector skin. This skin shows the information of the category in a box at the bottom of the wiki page (Figure 20).



Category:HvgLeSS:Stakeholder

This is the LeSS:Stakeholder category.

The Stakeholder can be a customer or vendor of the Product. It is equivalent to KCRIO:Stakeholder

This category currently contains no pages or media.

Category: HvgLeSS:Role

Figure 20: View page category HvgLeSS:Stakeholder

6.2.2 HvgLeSS:Person

The above discussed role is given to a person. In LeSS, all mentioned roles are dedicated full time roles (The LeSS Company B.V. 2016b). Hence, a person can only have one role in the development of the product.

The category `HvgLeSS:Person` with its related categories and its properties is shown in Figure 21. We skipped the feature category in our proof of concept and decided to relate the person directly with the product to decrease the complexity of the categories to create and use. To demonstrate this decision we did not drop the category `HvgLeSS:Feature` but skipped it. Thus, the indirect relation between `HvgLeSS:Person` and `HvgLeSS:Product` (Figure 18) became a direct one, named `works on` (Figure 21).

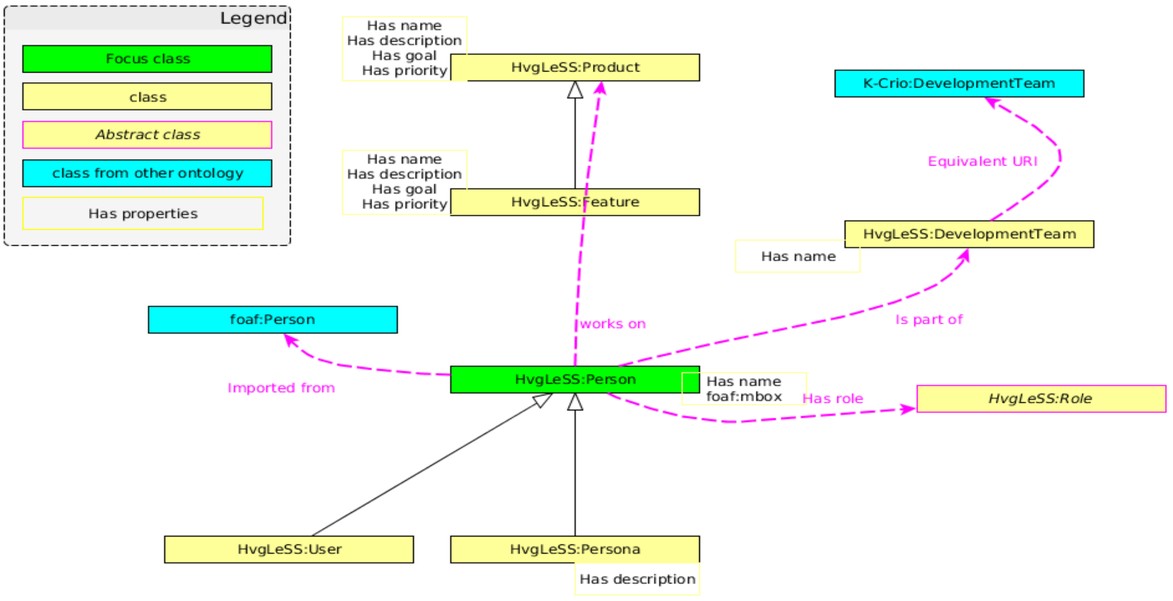
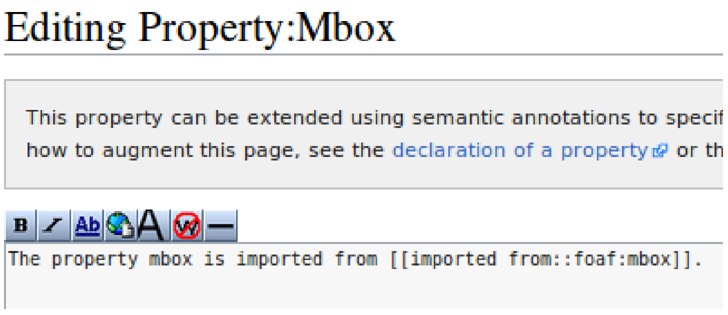


Figure 21: Properties and category HvgLeSS:Person

The properties of HvgLeSS:Person (Figure 21) are

- `Has subcategory` is used twice. Once for `HvgLeSS:User` and once for `HvgLeSS:Persona`. The former is used to hold the information of the users of the wiki in our proof of concept. It is this user who has a role. The latter is created to describe user stories and is considered as a fictive person. We opted to let this fictive person inherit the properties of the person and added the property `Has description`. So, the possibility to describe the persona and its context is created as part of the wiki and our proof of concept.
- `Has name`, containing the name of the person and having a text type.
- `mbox`, imported from the FoaF ontology[6]: The property is added to allow to find the same person based on the unique value of the mailbox address (Brickley & Miller 2014). The creation or edit view for this property is given in Figure 22 and the normal view on the page in Figure 23
- `Has role` by linking him with the category `HvgLeSS:Role`, the category already discussed above.
- `Works on` with the short cut made to the category `HvgLeSS:Product` to determine for which product the person works. Mostly a person will only work on one product to allow the main focus for scrum (Deemer et al. 2012), but excluding the possibility will narrow the scope, which is not wanted.
- `Is part of` the development team, thus the category `HvgLeSS:DevelopmentTeam`. This property will only be used for team members. The relationship can also be defined by the use of the property is member of with subject the team member and object the development team. We doubted to drop the latter relation because of the uniqueness of the role for a person. But we decided to let both relations exist to allow the possibility for a product change.
- `Imported from` the `foaf:Person` is made possible because the FoaF ontology is loaded into the wiki. To import the ontology we followed the steps described in the user manual of MediaWiki[7]

6 http://xmlns.com/foaf/spec/#term_mbox
7 https://www.semantic-mediawiki.org/wiki/Help:Import_vocabulary_(Example_of_importing_FOAF_vocabulary)

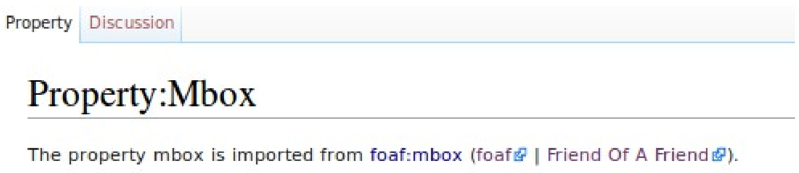

Editing Property:Mbox

This property can be extended using semantic annotations to specif
how to augment this page, see the declaration of a property or th

```
The property mbox is imported from [[imported from::foaf:mbox]].
```

Figure 22: Property mbox imported from FoaF

The use of the `imported from` property is shown for the `Mbox` in the property editing page (Figure 22). The effect is given in the view mode of this page (Figure 23).



Property:Mbox

The property mbox is imported from foaf:mbox (foaf | Friend Of A Friend).

Figure 23: View on property Mbox page

6.3 Use of Semantic Result Formats

Semantic Result Formats (De Dauw et al. 2017) supports the sum function. So, in our work the extension is used to calculate the total story points for the items of the sprint backlog. The calculation is based on an inline query, the way of querying data in Semantic MediaWiki, which receives the format sum[8].

6.4 Use UserGetName

Our research focused on the roles in the LeSS methodology and each person involved in the development of the product has his role. Hence, we needed a way to capture the role of the user logged in in the wiki.

First, the user-name of the login has to be known and based on this name, we can determine which role the user has. This means that we used the MediaWiki user-name from the namespace User, captured in the variable $wgUser. The MediaWiki information can only be captured using PHP-code and that is what the extension UserGetName does (Ejcaputo 2010). The limitation is that the name used in our category `HvgLeSS:User` has to have the same name as the wiki user, camel cases included. The role of the user is set as property in our `HvgLeSS:User` category, so this can be used in inline queries (Figure 24).

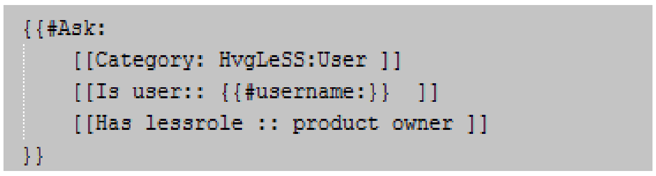
6.5 Interface Sprint dependent on role of user

By combining the parser functions if and ask, we can select values depending on the role of the user to show the information of the sprint backlog in our wiki.

8 https://www.semantic-mediawiki.org/wiki/Help:Sum_format

The condition to generate the correct output is written as an inline query and checks if the user:

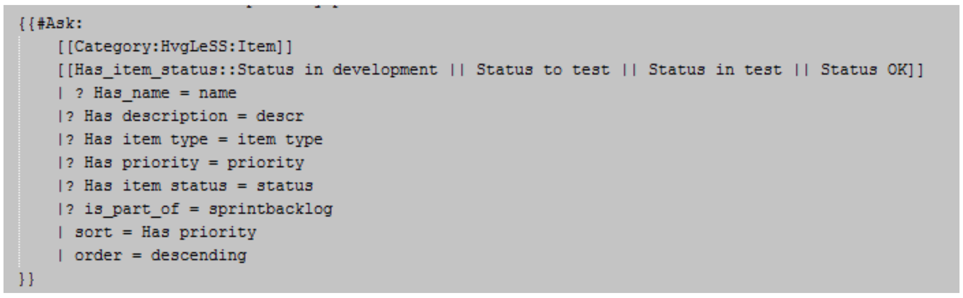
- exists in the wiki, thus belonging to the category `HvgLeSS:User`
- has the property name with the value equal to the name of the user who is logged in, which we get by using the user-name parser function of the GetUserName extension
- has the role 'product owner' as value for the `Has_lessrole` property. (Figure 24)



```
{{#Ask:
    [[Category: HvgLeSS:User ]]
    [[Is user:: {{#username:}}  ]]
    [[Has lessrole :: product owner ]]
}}
```

Figure 24: Inline query to meet condition of product owner role

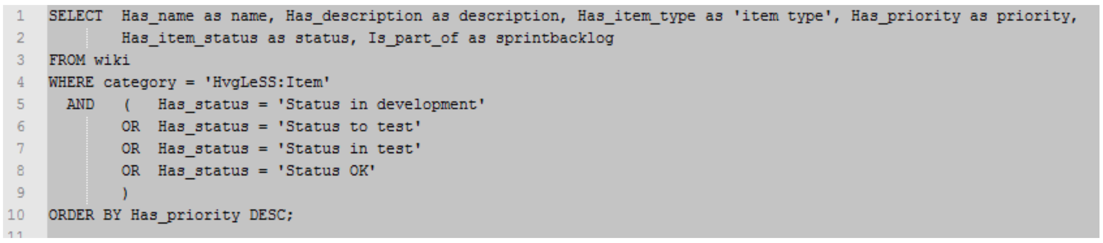
When the logged in user has the LeSS role 'product owner', the title 'View on the current sprint by product owner' is given. To create the view of the sprint, an inline query (Figure 25) is used to select the name, the description, the item type, the priority, the status and the sprint backlog by checking the property of the item. The where-clause limits the selection for item, by specifying that it has to belong to the category `HvgLeSS:Item` and for its status by only selecting the status with value 'Status in development', 'Status to test', 'Status in test' or 'Status OK'. The information is ordered by the value of the property `Has_priority` in a descending way. The selection of the item is similar to a Fresnel selector, developed within MediaWiki and thus does not use SPARQL to query the information but a Semantic MediaWiki inline query.



```
{{#Ask:
    [[Category:HvgLeSS:Item]]
    [[Has_item_status::Status in development || Status to test || Status in test || Status OK]]
    | ? Has_name = name
    |? Has description = descr
    |? Has item type = item type
    |? Has priority = priority
    |? Has item status = status
    |? is_part_of = sprintbacklog
    | sort = Has priority
    | order = descending
}}
```

Figure 25: Ask query for the creation of the view for the product owner

As documentation we translated the inline query to a SQL statement (Figure 26). We opted for a translation to SQL because it is commonly well understood by IT professionals. Another option would be to show a translation to a Fresnel lens.



```
SELECT  Has_name as name, Has_description as description, Has_item_type as 'item type', Has_priority as priority,
        Has_item_status as status, Is_part_of as sprintbacklog
FROM wiki
WHERE category = 'HvgLeSS:Item'
  AND   (   Has_status = 'Status in development'
        OR  Has_status = 'Status to test'
        OR  Has_status = 'Status in test'
        OR  Has_status = 'Status OK'
        )
ORDER BY Has_priority DESC;
```

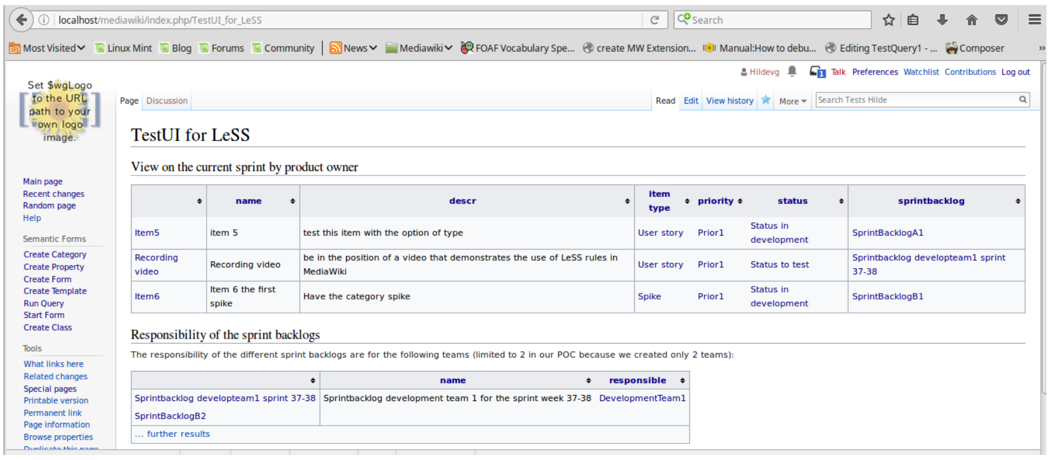
Figure 26: SQL code for inline query

The product owner has to have a view on the sprint backlogs of all teams and needs the information which item has to be developed by which team. So, we used a second query in this view (Figure 27). The limit set is 2 because in our proof of concept we only created two teams and each team has its backlog, so there will be two backlogs in the sprint. Another possibility to achieve the result is by using the sprint or the current time-stamp.

```
{{#Ask:
    [[Category:HvgLeSS:SprintBacklog]]
    |? Has name = name
    |? is responsibility of = responsible
    | order = descending
    | limit = 2
}}
```

Figure 27: Inline query to show sprint backlogs

Our proof of concept for the view implemented the product owner and stakeholder roles. If the user does not have one of these roles, the information is given that this implementation is not yet done.



TestUI for LeSS

View on the current sprint by product owner

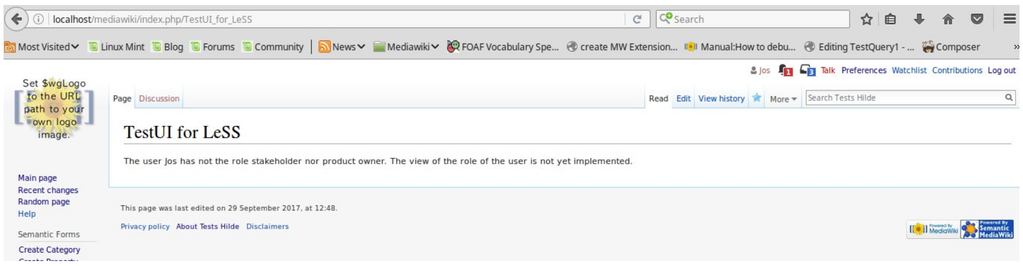
	name	descr	item type	priority	status	sprintbacklog
Item5	item 5	test this item with the option of type	User story	Prior1	Status in development	SprintBacklogA1
Recording video	Recording video	be in the position of a video that demonstrates the use of LeSS rules in MediaWiki	User story	Prior1	Status to test	Sprintbacklog developteam1 sprint 37-38
Item6	Item 6 the first spike	Have the category spike	Spike	Prior1	Status in development	SprintBacklogB1

Responsibility of the sprint backlogs

The responsibility of the different sprint backlogs are for the following teams (limited to 2 in our POC because we created only 2 teams):

	name	responsible
Sprintbacklog developteam1 sprint 37-38	Sprintbacklog development team 1 for the sprint week 37-38	DevelopmentTeam1
SprintBacklogB2		
… further results		

Figure 28: User Interface of current Sprint viewed by product owner

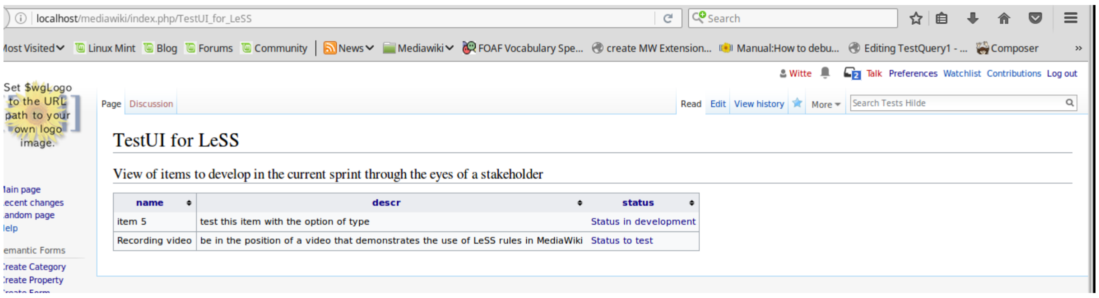


TestUI for LeSS

The user Jos has not the role stakeholder nor product owner. The view of the role of the user is not yet implemented.

This page was last edited on 29 September 2017, at 12:48.

Figure 29: User Interface of current Sprint viewed by scrum master



TestUI for LeSS

View of items to develop in the current sprint through the eyes of a stakeholder

name	descr	status
item 5	test this item with the option of type	Status in development
Recording video	be in the position of a video that demonstrates the use of LeSS rules in MediaWiki	Status to test

Figure 30: User Interface of current Sprint viewed by stakeholder

6.6 Conclusion

By the use of the MediaWiki if parser function and the extension GetUserName (Ejcaputo 2010), the view of the page is made dependent on the role the user has within the product. An inline query checks for the result on a given role for the user and when a page, meeting the given conditions, is found, the view for the role is created. If the condition is not met, another view on the page is shown. The inline query querying the information of the item and backlog could also be done by Fresnel simple selector[9], but we did not find information on making these lenses dependent on the role of the logged in user. To implement the functionality we created an ontology within Semantic MediaWiki. The data for our proof of concept is set partly by the use of Page Forms (Koren et al. 2017) and queried by Semantic MediaWiki inline queries, using the categories and properties from our ontology.

This section gave a partial answer to our question *a) How and to what extent can business rules guide the team through the scale agile development process?*. Semantic MediaWiki can use the information of the user logged in in the wiki to adapt the information given on a page, allowing the user to keep focused on the items important for his role in the product development. We did not find a solution in Semantic Web by the use of Fresnel in the literature. Hence, for the how part of the question we found that within Semantic MediaWiki a business rule based on the user role can be used to help the team through the development process. The focus, one of the main principles of scrum (Deemer et al. 2012), is surely an important help that we achieved here by using a business rule based on the role of the user in the product development.

Furthermore, the sprint backlog gives an overview of the progress of the sprint. Thus, the section describes also a partial answer to the question *e) How and to what extent can Semantic MediaWiki technology provide an interface to display the progress of product development?*. The extension Semantic Result Formats (De Dauw et al. 2017) allows the use of graphs. So not only the textual interface used in our proof of concept is available, but also the creation of a burn-down graph to show the progress of the development for the sprint backlogs.

Linking these findings with our knowledge of Fresnel, a limitation for Fresnel is found because a way to let lenses depend on the logged in user's role was not found. Further research on this dependency is needed. Also the limits of format has to be researched. Semantic Result Formats gave good results for our proof of concept, but we did not compare these with the possibilities given by Fresnel formats. On the other hand, for the output, it is clear that Fresnel Forms delivers more possibilities, because Semantic MediaWiki is limited to its output in a wiki.

9 https://www.w3.org/2005/04/fresnel-info/manual/#selectors

7 Role-dependent form

7.1 Introduction

By combing the terms forms and MediaWiki, Fresnel Forms popped up in our thoughts. Due to the focus on Semantic MediaWiki as a development platform, we could not use these Fresnel Forms as such. By using Page Forms, the Semantic MediaWiki translation of Fresnel Forms, we were sure to not break the link with Fresnel.

In this chapter we researched how Semantic MediaWiki can help in the management of items, which have properties that are under the responsibility of the product owner, of the development team and scrum master, and even shared responsibilities. The timeline of a user story will be as follows: First the product owner creates the item and gives it a name, description, indicates for which persona the story is needed and defines the priority. During the refinement meeting the item will be discussed and risk or related items can be added. When the item is understood by the team, the story points for the item are scored. These latter actions are mainly the responsibility of the scrum master and the team. So having two different forms for the same item could make it easier to input the information for which the role is responsible.

First we described how Page Forms were created (7.2), followed by the limitations of Page Forms (7.3), a possible extension (7.4) and a found solution by using sub pages (7.5). The conclusion of this chapter is discussed under 7.6

7.2 Page Forms

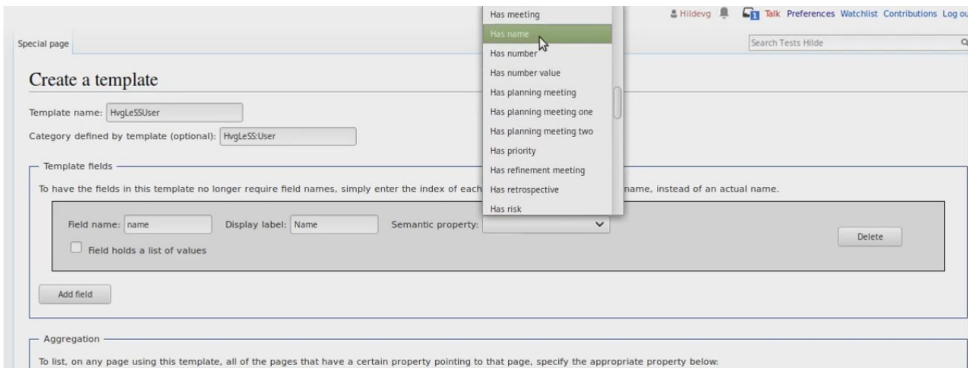
We used the Page Forms extension for several categories. The category `HvgLeSS:User` was chosen to explain the implementation because this category is one of the most important in our work.

The first step is the creation of a template. The template will define which information is used. One way to define the information, and the only one we used, is based on the category.

The template explains itself. The information to set is

- the name for the template to create
- Optional the name of the category the template is created for
- the field name
- the label to display on the wiki for the field
- the semantic property to set the input value to
- if the field can hold a list, the check-box for the list of values has to be checked.

The creation of the template is demonstrated in Figure 31 We named the template `HvgLeSSUser` and specified that it defines the category `HvgLeSS:User`. The field name, display label and semantic property is set. This latter can be selected from the list of properties, what makes typos impossible. If a next field is wanted, it can be set by clicking the add field button.


Special page

Create a template

Template name: HvgLeSSUser

Category defined by template (optional): HvgLeSS:User

Template fields

To have the fields in this template no longer require field names, simply enter the index of each … name, instead of an actual name.

Field name: name Display label: Name Semantic property:

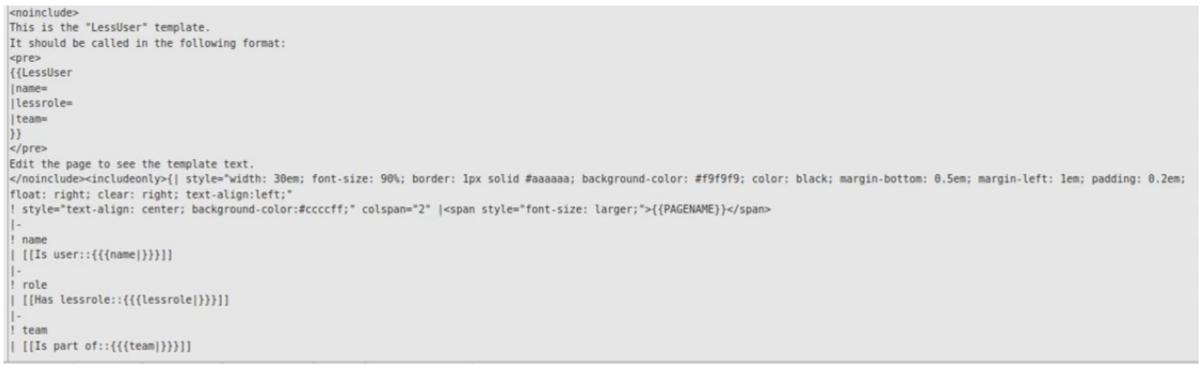
Field holds a list of values

Aggregation

To list, on any page using this template, all of the pages that have a certain property pointing to that page, specify the appropriate property below:

Figure 31: Creation of the user template

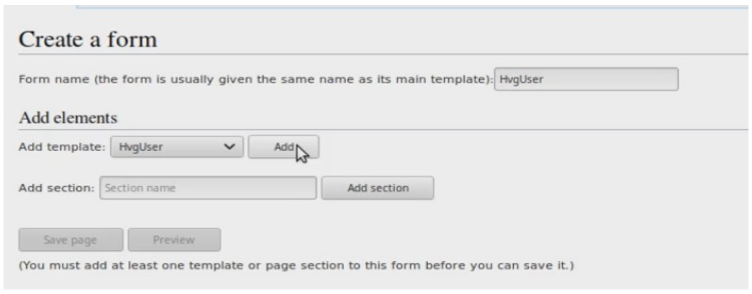
Once the template is created, modifications have to be done in the code of the template, which consists of five parts. (1) The intro text, (2) fields to show, named by the name of the field, (3) the information that the template text is only visible in edit mode, (4) followed by the format code and at last (5) the detail of the information to show.



```
<noinclude>
This is the "LessUser" template.
It should be called in the following format:
<pre>
{{LessUser
|name=
|lessrole=
|team=
}}
</pre>
Edit the page to see the template text.
</noinclude><includeonly>{| style="width: 30em; font-size: 90%; border: 1px solid #aaaaaa; background-color: #f9f9f9; color: black; margin-bottom: 0.5em; margin-left: 1em; padding: 0.2em;
float: right; clear: right; text-align:left;"
! style="text-align: center; background-color:#ccccff;" colspan="2" |<span style="font-size: larger;">{{PAGENAME}}</span>
|-
! name
| [[Is user::{{{name|}}}]]
|-
! role
| [[Has lessrole::{{{lessrole|}}}]]
|-
! team
| [[Is part of::{{{team|}}}]]
```

Figure 32: Editing template code for user

The view on the edit page of the template Figure 32 shows that the information visible for the user in read mode is set between the `pre` tags, referred as (2) above. The detailed information -referred as (5)- contains the label, followed by the property to set and its value, set between triple braces. The text between the `includeonly` tags is only visible in edit mode and this information is given as text in view mode, referred as (3) above.


Create a form

Form name (the form is usually given the same name as its main template): HvgUser

Add elements

Add template: HvgUser

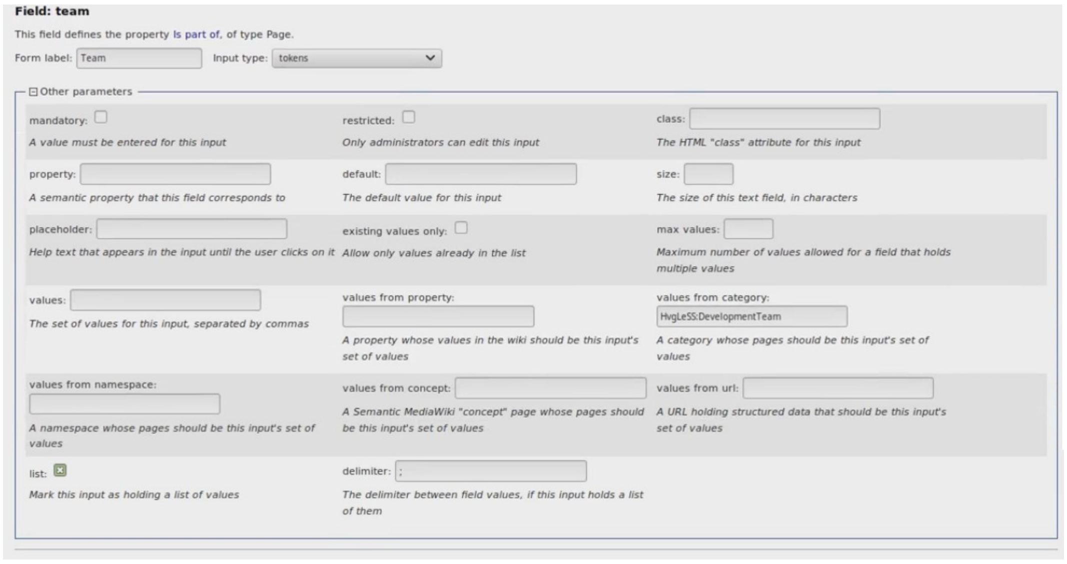
Add section: Section name

(You must add at least one template or page section to this form before you can save it.)

Figure 33: Create the user form

Once the template is created, the second step, being the creation of the form, starts. The form is created based on one or more templates. Hence, saving a form without selecting a template is not possible as mentioned at the bottom of Figure 33. The elements of the template will be added to the form. For each element the settings have to be completed. So in our case we set the name as mandatory, because we want to compare the name of the person with the user-name of the wiki. Also the role has to be mandatory because otherwise the situation that a `HvgLeSS:User` has no role can exist, which is unwanted.

In contrast with user-name and role, the team cannot be set as mandatory because a product owner does not belong to any of the teams. Hence, in the form team, the check-box mandatory is unchecked. The property used to define the team to which a user belongs is `Is_part_of`. This property is not limited to only one category, so to avoid inconsistency we defined in the form that the value has to be of the category `HvgLeSS:DevelopmentTeam`. At the bottom of Figure 34 we set the list option and defined the delimiter as being a semi colon, to allow a scrum master facilitating the work of more than one team. The form creation delivered by Page Forms helps in designing the form (Figure 34).


Field: team

This field defines the property Is part of, of type Page.

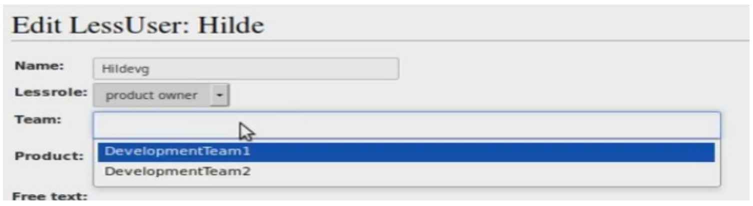
Form label: Team Input type: tokens

Other parameters

mandatory: ☐ *A value must be entered for this input*	restricted: ☐ *Only administrators can edit this input*	class: *The HTML "class" attribute for this input*
property: *A semantic property that this field corresponds to*	default: *The default value for this input*	size: *The size of this text field, in characters*
placeholder: *Help text that appears in the input until the user clicks on it*	existing values only: ☐ *Allow only values already in the list*	max values: *Maximum number of values allowed for a field that holds multiple values*
values: *The set of values for this input, separated by commas*	values from property: *A property whose values in the wiki should be this input's set of values*	values from category: HvgLeSS:DevelopmentTeam *A category whose pages should be this input's set of values*
values from namespace: *A namespace whose pages should be this input's set of values*	values from concept: *A Semantic MediaWiki "concept" page whose pages should be this input's set of values*	values from url: *A URL holding structured data that should be this input's set of values*
list: ☒ *Mark this input as holding a list of values*	delimiter: ; *The delimiter between field values, if this input holds a list of them*	

Figure 34: Detail creation form User for field team

For team we chose tokens as import type and the limitation of the values from a category (Figure 34) results in the use of a combo-box in the form where the possible values are shown (Figure 35).


Edit LessUser: Hilde

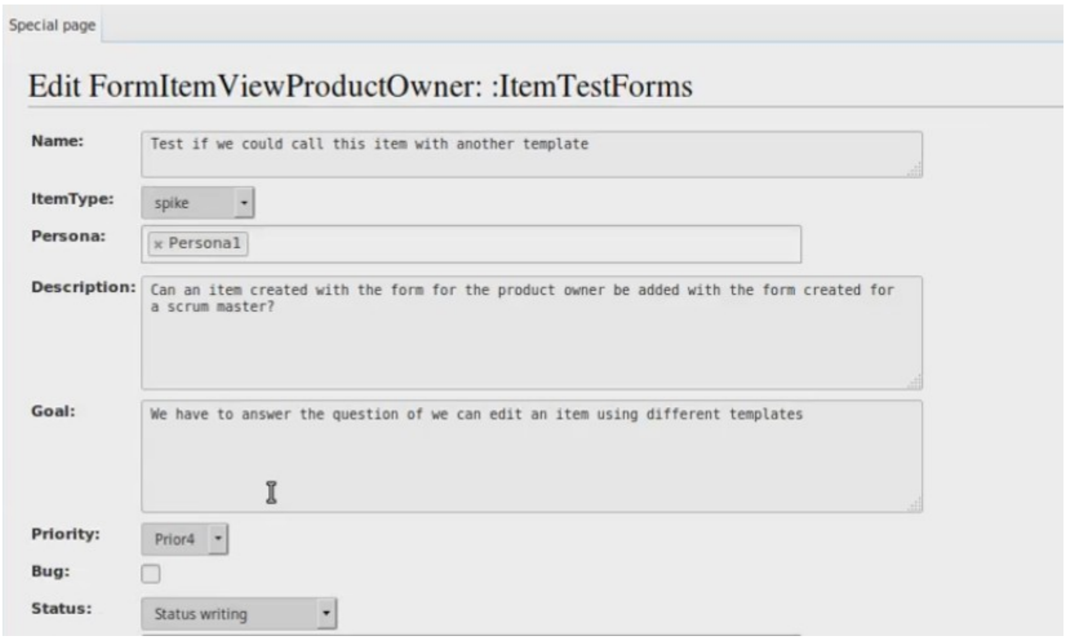
Name:	Hildevg
Lessrole:	product owner
Team:	
Product:	DevelopmentTeam1 DevelopmentTeam2
Free text:	

Figure 35: Result of form for the options chosen for the field team

7.3 Changing Page Forms based on user role

The creation of a user story is done by a product owner, but his responsibility is limited to define the name, description, goal and priority. But when a view on an item is given, the product owner wants to see all the information available. To make the difference between the information to edit and the information to view, we created two templates, one for each action. We combined these templates in a form by the use of Page Forms and created a new item.

The scrum master has another focus on an item and other information to complete. Thus, two new templates were created to help the scrum master execute his tasks. Based on these templates a second form for the input of the item was made.

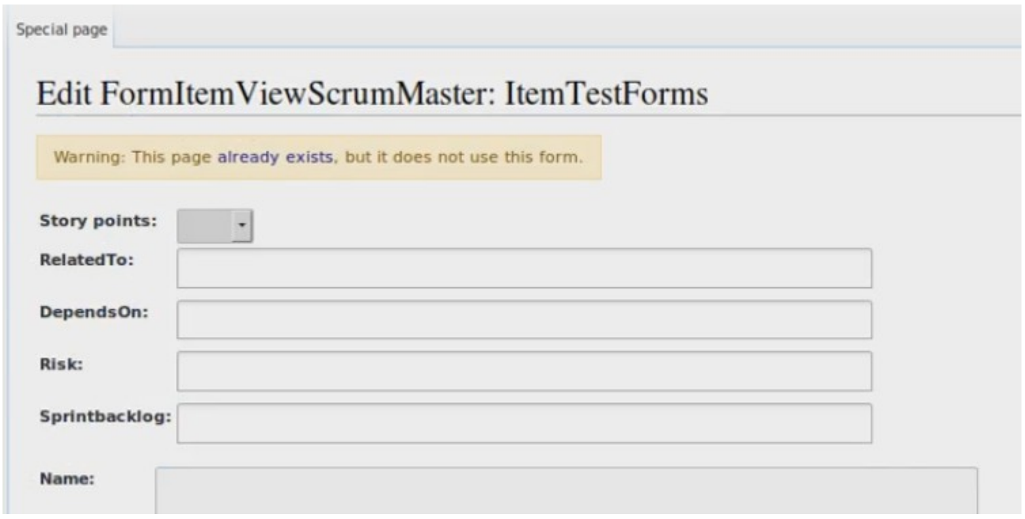

Special page

Edit FormItemViewProductOwner: :ItemTestForms

Name:	Test if we could call this item with another template
ItemType:	spike
Persona:	× Persona1
Description:	Can an item created with the form for the product owner be added with the form created for a scrum master?
Goal:	We have to answer the question of we can edit an item using different templates
Priority:	Prior4
Bug:	☐
Status:	Status writing

Figure 36: Create new item by using the product owner form

Creating the page for a new item by the product owner did not give any difficulties (Figure 36), but the information inputted in this page could not be shown by using the form created for the scrum master (Figure 37). Page Forms gave an error because the original page had been created with another form and in Figure 37 the information inputted in the name field by the use of the form created for the product owner, is not visible in the form created for the scrum master.


Special page

Edit FormItemViewScrumMaster: ItemTestForms

Warning: This page already exists, but it does not use this form.

Story points:	
RelatedTo:	
DependsOn:	
Risk:	
Sprintbacklog:	
Name:	

Figure 37: Editing the item using the form of the scrum master

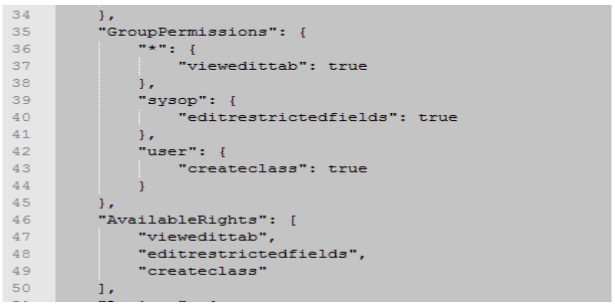
Regarding the name given at the extension, being Page Forms, we were not surprised by the impossibility to make the form user dependent. The form declares the page and the page is given by its name. So using the same page name and two different page forms does not seem logical.

7.4 Page Forms extension

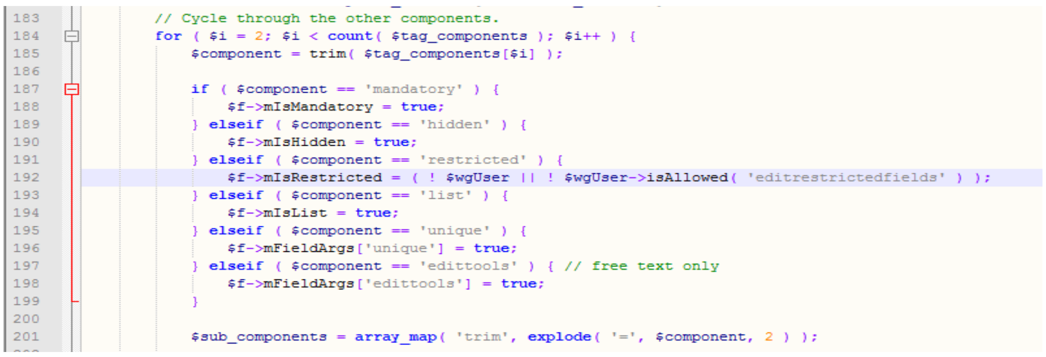
As concluded above, the use of the Page Forms does not allow different views for one and the same page. But other solutions can be investigated.

Page Forms uses a restriction, being the second check-box in the section of other parameters shown in Figure 34. By checking this restriction, only the administrators or sysops can change the value of the field, or the restriction can be set to a group name (Koren et al. 2017). Sysops and groups belong to the MediaWiki User namespace and thus are not available via Semantic MediaWiki properties. The question to reuse the MediaWiki namespace User in our work arose already at the moment we needed a user category and we decided to not reuse the User namespace. Here, we stuck to our decision and did not opt for the creation of a new group.

Other possibilities are to adapt the code of the Page Forms, a possibility created by the open source of the code, or to extend the extension. First, the number of group permissions has to be extended with product owner, scrum master and team member as minimum. Then new available rights have to be created to give the wanted rights to each of the new groups. The group permissions and available rights used by Page Forms are shown in Figure 38.


```
    },
    "GroupPermissions": {
        "*": {
            "viewedittab": true
        },
        "sysop": {
            "editrestrictedfields": true
        },
        "user": {
            "createclass": true
        }
    },
    "AvailableRights": [
        "viewedittab",
        "editrestrictedfields",
        "createclass"
    ],
```
Figure 38: Permission settings of Page Forms from extension.json file

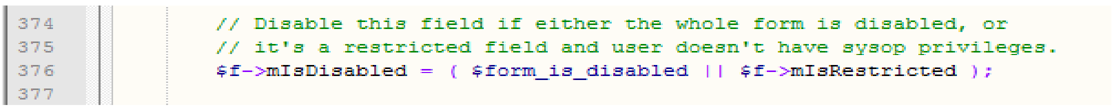
If `restricted` is checked, the code checks if the user is known, thus logged in the wiki and if he is, the permission is controlled based on the group(s) to which the user belongs by calling the `isAllowed` function of the user (Figure 39).


```
        // Cycle through the other components.
        for ( $i = 2; $i < count( $tag_components ); $i++ ) {
            $component = trim( $tag_components[$i] );

            if ( $component == 'mandatory' ) {
                $f->mIsMandatory = true;
            } elseif ( $component == 'hidden' ) {
                $f->mIsHidden = true;
            } elseif ( $component == 'restricted' ) {
                $f->mIsRestricted = ( ! $wgUser || ! $wgUser->isAllowed( 'editrestrictedfields' ) );
            } elseif ( $component == 'list' ) {
                $f->mIsList = true;
            } elseif ( $component == 'unique' ) {
                $f->mFieldArgs['unique'] = true;
            } elseif ( $component == 'edittools' ) { // free text only
                $f->mFieldArgs['edittools'] = true;
            }

            $sub_components = array_map( 'trim', explode( '=', $component, 2 ) );
```
Figure 39: Field component reading and setting the value mIsRestricted if restricted checked (PageForms/includes/PF_FormField.php)

Once the variable `mIsRestricted` for the field is set and having the value true, the field is disabled (Figure 40).



```
        // Disable this field if either the whole form is disabled, or
        // it's a restricted field and user doesn't have sysop privileges.
        $f->mIsDisabled = ( $form_is_disabled || $f->mIsRestricted );
```

Figure 40: Disable field due to restriction (PageForms/includes/PF_FormField.php)

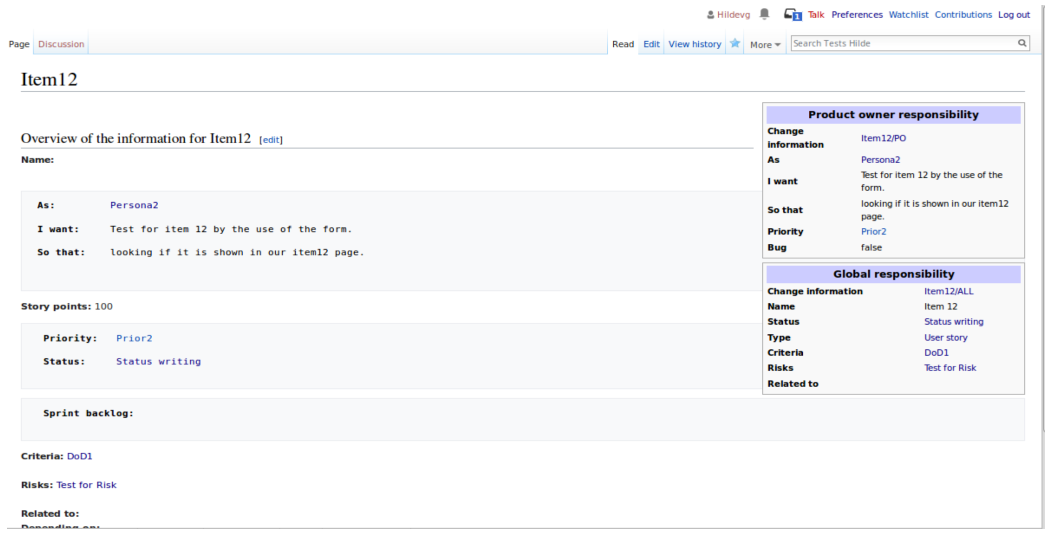
Based on the analysis of the Page Forms code, we concluded that Page Forms could be extended to allow only the responsible persons to change the content of fields. So, the privilege to create a new item could be set for the product owner and the content of story points could only be changed by a scrum master. This adaptation or extension is not implemented in our work, due to the fact that it focuses on Semantic MediaWiki and these adaptations impact (an extension of) MediaWiki.

7.5 Use of sub pages

MediaWiki works with sub pages, when these are enabled in the local settings of the wiki. Technically a sub page is a page, which solved the blocking format of our proof (see 7.3). Furthermore MediaWiki supports magic words for pages and sub pages[10], which offers the possibility to use page names as variables.

The data for item is set by the use of Page Forms, split into three forms based on the owner of the data. So is the description of the user story, the focus of our proof of concept, a responsibility of the product owner. The story points and the decision in which sprint it will be developed, is under the responsibility of the team and scrum master. A risk can be detected by product owner, scrum master or one of the team members and is considered as a shared responsibility.

The detail of the item is made role-dependent. A user who has not the role of product owner, nor scrum master, nor team member, sees all the available information of the item and no info-box. For the role of product owner, an info-box with the data for which he is responsible is added, followed by an info-box with the data under the responsibility of all roles working on the product (Figure 41). The scrum master and development team also see two added info-boxes, with data under their responsibility (Figure 42). By asking the detail of the information shown in the info-boxes, a page with form opens, allowing to manage the data.



Item12

Overview of the information for Item12 [edit]

Name:

As:	Persona2
I want:	Test for item 12 by the use of the form.
So that:	looking if it is shown in our item12 page.

Story points: 100

Priority:	Prior2
Status:	Status writing

Sprint backlog:

Criteria: DoD1

Risks: Test for Risk

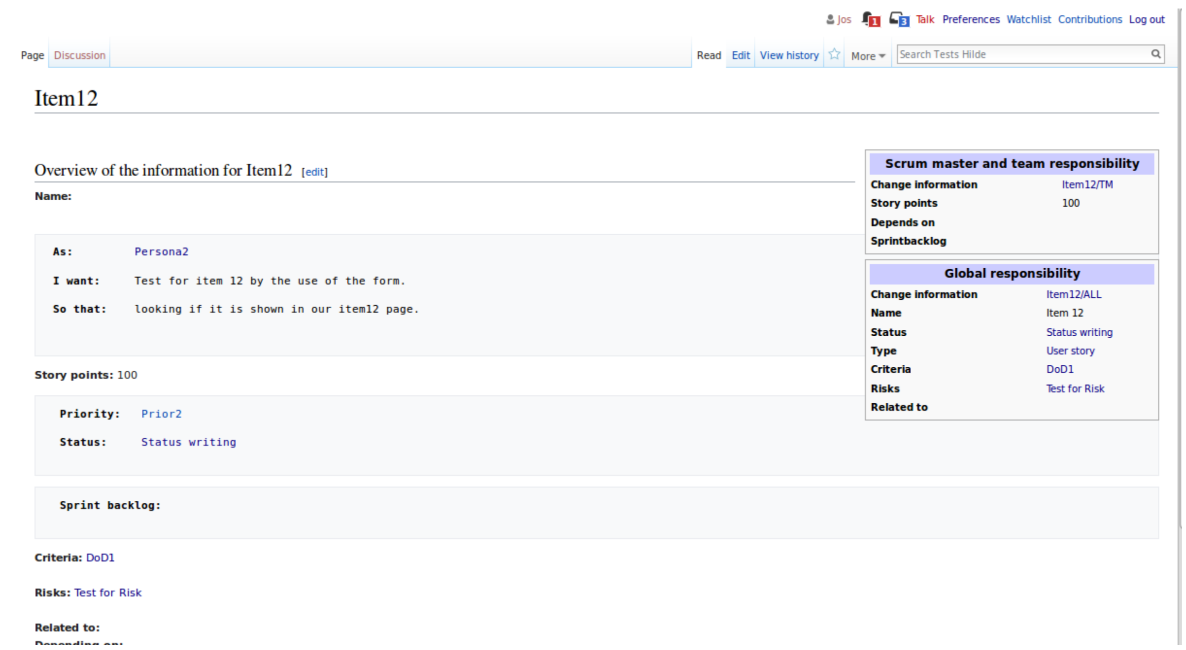
Related to:

Product owner responsibility	
Change information	Item12/PO
As	Persona2
I want	Test for item 12 by the use of the form.
So that	looking if it is shown in our item12 page.
Priority	Prior2
Bug	false

Global responsibility	
Change information	Item12/ALL
Name	Item 12
Status	Status writing
Type	User story
Criteria	DoD1
Risks	Test for Risk
Related to	

Figure 41: Item viewed by product owner

10 https://www.mediawiki.org/wiki/Help:Magic_words#Page_names


Jos
Talk Preferences Watchlist Contributions Log out
Page Discussion
Read Edit View history More
Search Tests Hilde

Item12

Overview of the information for Item12 [edit]

Name:

As:	Persona2
I want:	Test for item 12 by the use of the form.
So that:	looking if it is shown in our item12 page.

Story points: 100

Priority:	Prior2
Status:	Status writing

Sprint backlog:

Criteria: DoD1

Risks: Test for Risk

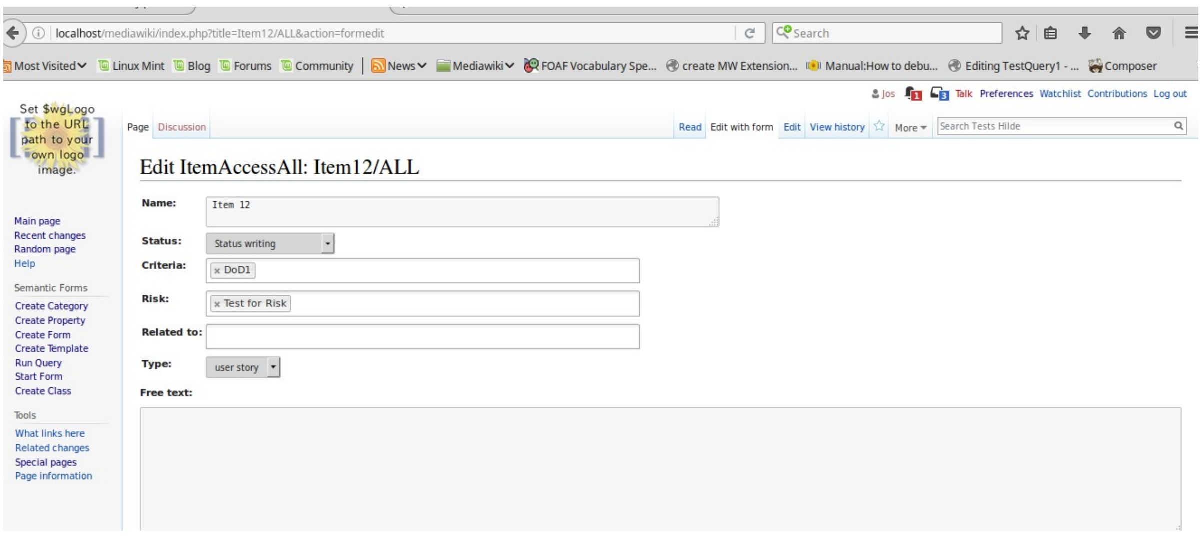
Related to:

Scrum master and team responsibility	
Change information	Item12/TM
Story points	100
Depends on	
Sprintbacklog	

Global responsibility	
Change information	Item12/ALL
Name	Item 12
Status	Status writing
Type	User story
Criteria	DoD1
Risks	Test for Risk
Related to	

Figure 42: Item viewed by scrum master or team member

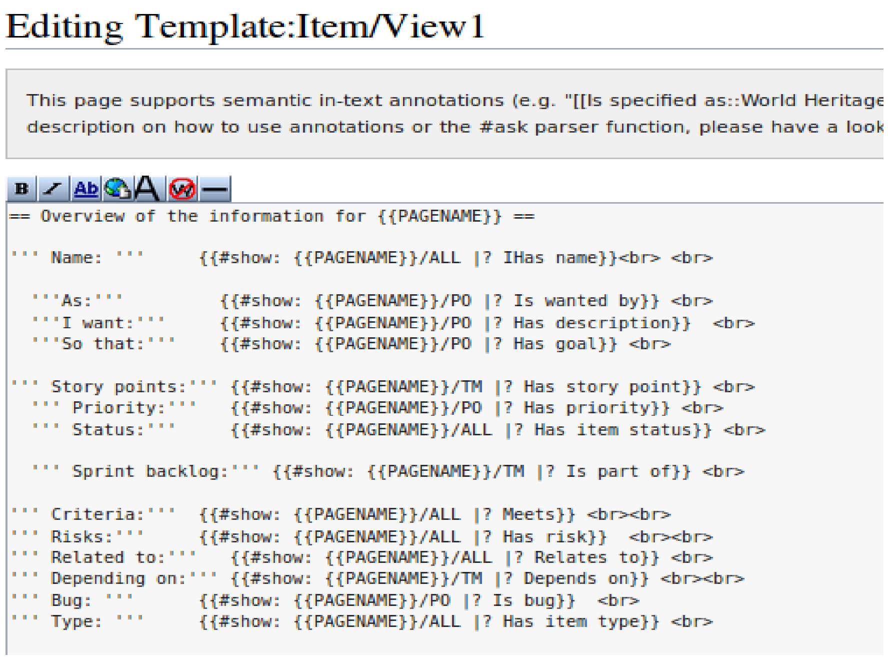
Technically, we created the page of the item and used the information set by Page Forms in sub pages to allow the view of the data in the main page. The existing magic words for the page name did not deliver the wanted results, so our proof of concept uses hard coded values for the implementation (Figure 44). Each info-box is linked with the Page Form that handles its data, which allowed us to always use the same page to manage the information. Figure 43 demonstrates the management of the data for which the whole product team is responsible. Hence, we could resolve the blocking described under 7.3 but only by the use of consistent and restricted page names. The page name of the main page is free, but the sub pages must have the suffix /PO for product owner, /TM for scrum master and team member, and /ALL for responsibilities combined by product owner, scrum master and team member.



Edit ItemAccessAll: Item12/ALL

Name:	Item 12
Status:	Status writing
Criteria:	× DoD1
Risk:	× Test for Risk
Related to:	
Type:	user story
Free text:	

Figure 43: Form with data for which whole product team is responsible


Editing Template:Item/View1

This page supports semantic in-text annotations (e.g. "[[Is specified as::World Heritage
description on how to use annotations or the #ask parser function, please have a look

```
== Overview of the information for {{PAGENAME}} ==

''' Name: '''      {{#show: {{PAGENAME}}/ALL |? IHas name}}<br> <br>

  '''As:'''          {{#show: {{PAGENAME}}/PO |? Is wanted by}} <br>
  '''I want:'''      {{#show: {{PAGENAME}}/PO |? Has description}}  <br>
  '''So that:'''     {{#show: {{PAGENAME}}/PO |? Has goal}} <br>

''' Story points:''' {{#show: {{PAGENAME}}/TM |? Has story point}} <br>
  ''' Priority:'''   {{#show: {{PAGENAME}}/PO |? Has priority}} <br>
  ''' Status:'''     {{#show: {{PAGENAME}}/ALL |? Has item status}} <br>

  ''' Sprint backlog:''' {{#show: {{PAGENAME}}/TM |? Is part of}} <br>

''' Criteria:'''   {{#show: {{PAGENAME}}/ALL |? Meets}} <br><br>
''' Risks:'''      {{#show: {{PAGENAME}}/ALL |? Has risk}}  <br><br>
''' Related to:'''   {{#show: {{PAGENAME}}/ALL |? Relates to}} <br>
''' Depending on:''' {{#show: {{PAGENAME}}/TM |? Depends on}} <br><br>
''' Bug: '''       {{#show: {{PAGENAME}}/PO |? Is bug}}  <br>
''' Type: '''      {{#show: {{PAGENAME}}/ALL |? Has item type}} <br>
```

Figure 44: Item information available for all roles, including stakeholder, with hard coded sub pages

7.6 Conclusion

In this chapter we examined the possibility to make forms dependent on the role of the user. Page Forms as such could not be used because each role has his responsibilities and Page Forms does not support to change one and the same data field by different forms. Further research is needed to examine if this limitation also holds for Fresnel Forms. If it does not, the impact of importing Fresnel Forms into Semantic MediaWiki should be explored.

We found two ways to resolve the problem. The first and most obvious solution is to adapt or extend Page Forms allowing fields to be enabled or disabled based on the role -above specified by group- and its available rights. But this was not the option we were looking for. We wanted to order the fields in different ways for different roles and found a solution through the use of sub pages. This solution does not give full satisfaction because of the lack of the possibility to use magic words for the page names. Further research is needed to solve this problem, allowing to use Page Forms with sub pages and dynamic page names.

8 Business rule based on roles

8.1 Introduction

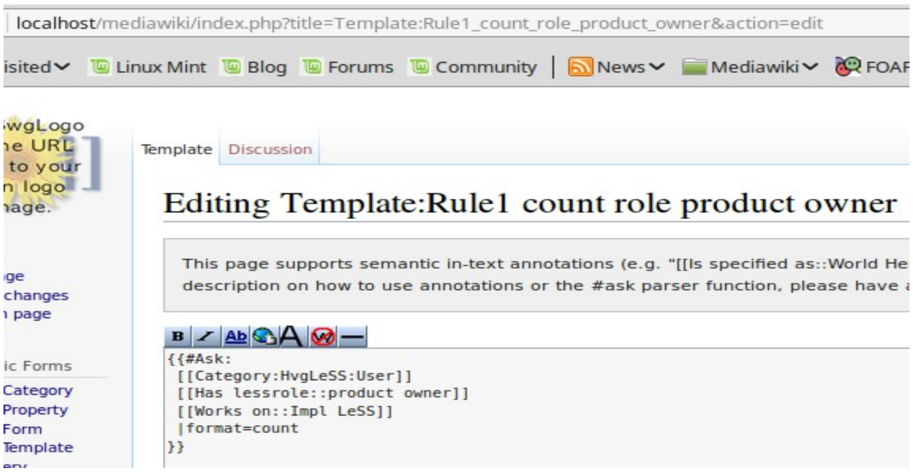
The moment a business rule is violated, the information of this violation has to be available. Fresnel offers a rendering of properties, but we found no indication that text output to report violations is available. The Open University of the Netherlands uses in the course business rules (Joosten et al. 2010) the data-driven tool Ampersand (Slootweg 2016) to describe and check business rules. Considering Ampersand as an example, we tried to get a similar result by using Semantic MediaWiki.

In this chapter we examined the possibility to create a business rule in Semantic MediaWiki, following the instructions given by Boa (Bao et al. 2009) by using templates (8.2.1, 8.2.2 and 8.2.3) and inline queries (8.2.1 and 8.2.2). Focusing on roles in LeSS, we decided to implement two business rules. The first checks that a product has one and only one product owner. The second counts the number of scrum masters, which has to be minimum one. Under 8.3 we concluded that Semantic MediaWiki supports business rules.

8.2 Implementation

8.2.1 Template containing the business rule

The first template counts the number of product owners by using an inline query with the count format (Figure 45) provided by Semantic Result Formats (De Dauw et al. 2017) and for another rule the number of scrum masters. The result of these queries is used in the following template.



Editing Template:Rule1 count role product owner

This page supports semantic in-text annotations (e.g. "[[Is specified as::World He
description on how to use annotations or the #ask parser function, please have

```
{{#Ask:
 [[Category:HvgLeSS:User]]
 [[Has lessrole::product owner]]
 [[Works on::Impl LeSS]]
 |format=count
}}
```

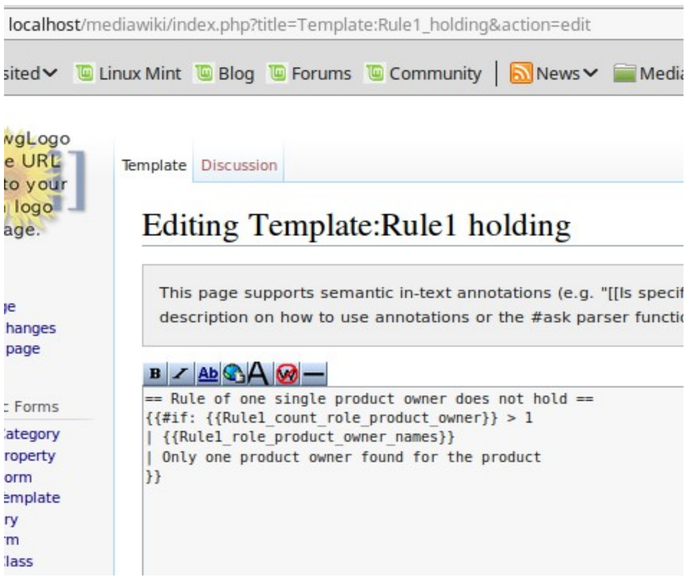
Figure 45: Counting the number of product owners for the product 'Impl LeSS'

By using count-functions in our templates, we were able to avoid the problem of the unknown open world assumption (Bos 2013). The limits of the use of business rules in the open world within Semantic MediaWiki needs further research.

8.2.2 Template containing the check if the rule holds

The inline query counts the users who have the role of product owner, and compares that with the maximum of product owners allowed by LeSS (The LeSS Company B.V. 2016c). If more than one is found, the information set in the template `Rule1_count_role_product_owner` is shown (Figure 46). The output of the check is given in a text, informing the user of the problem found and options to resolve the problem.

To allow the check of the fired rules, we also give the information in our proof of concept that the rule holds (Figure 46). Hence, the information on the check of different rules can be shown and no doubt exists if all wanted rules are checked.


localhost/mediawiki/index.php?title=Template:Rule1_holding&action=edit
Linux Mint
Blog
Forums
Community
News
Template
Discussion

Editing Template:Rule1 holding

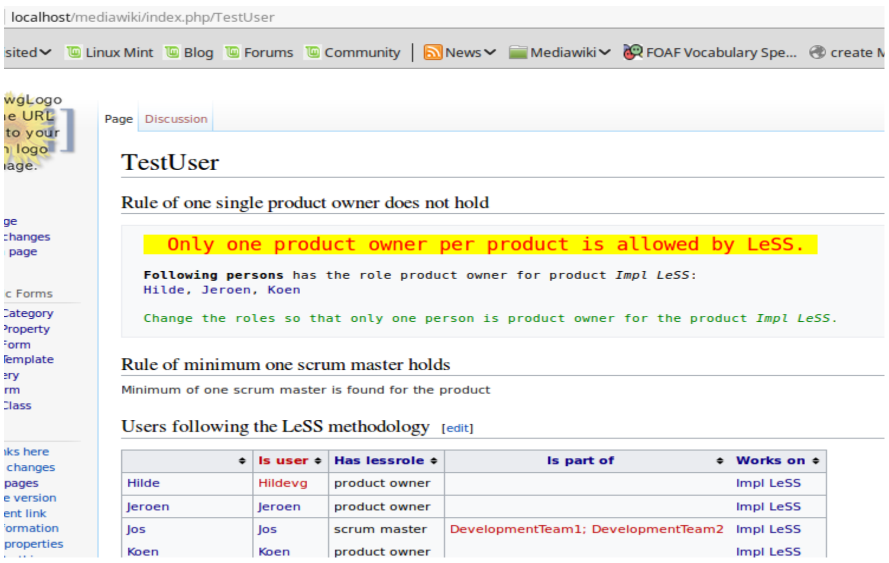
This page supports semantic in-text annotations (e.g. "[[Is specif
description on how to use annotations or the #ask parser functi

```
== Rule of one single product owner does not hold ==
{{#if: {{Rule1_count_role_product_owner}} > 1
| {{Rule1_role_product_owner_names}}
| Only one product owner found for the product
}}
```

Figure 46: Check if the rule holds

8.2.3 Interface inform the user of the holding of business rules

By calling the rule holding templates in any page, the check on the business rules is done. To allow a view and checking the result, we called these rules on a page querying the users (Figure 47).



TestUser

Rule of one single product owner does not hold

```
Only one product owner per product is allowed by LeSS.

Following persons has the role product owner for product Impl LeSS:
Hilde, Jeroen, Koen

Change the roles so that only one person is product owner for the product Impl LeSS.
```

Rule of minimum one scrum master holds

Minimum of one scrum master is found for the product

Users following the LeSS methodology [edit]

	Is user	Has lessrole	Is part of	Works on
Hilde	Hildevg	product owner		Impl LeSS
Jeroen	Jeroen	product owner		Impl LeSS
Jos	Jos	scrum master	DevelopmentTeam1; DevelopmentTeam2	Impl LeSS
Koen	Koen	product owner		Impl LeSS

Figure 47: Firing rules on TestUser page

The rendering of the output is as flexible as wiki-texts are. In our work we changed the output to a rather Ampersand style by defining templates.

8.3 Conclusion

Based on the research of Bao (Bao et al. 2009) the handling of business rules by Ampersand (Joosten et al. 2010) and with the help of the created ontology in Semantic MediaWiki, we could easily implement some business rules. The work is straightforward. By the use of a template to fire the business rule, the template has to be set on the pages where the check of the rule is needed.

However, some questions remained without an answer. Which effect will unknown information have on the way business rules are implemented in our work? Another question is the ability of Fresnel Forms to render text output, needed for the use of business rules.

Due to the former question above, we were not able to extend the comparison of relational algebra and SWRL (Bos 2013) with Semantic MediaWiki.

9 Results and conclusions

9.1 Introduction

In this work we researched the use of business rules within Semantic MediaWiki by a proof of concept focused on the roles in Large Scale Scrum (LeSS) to get an answer to the question '*How and to what extent can business rules be implemented with Semantic Web reasoning processed with a semantic wiki interface support of the scale agile development process?*'. To answer the question, we first searched a scale agile development methodology for which an ontology exists and for which business rules are defined. This search led us to Large Scale Scrum (LeSS). After creating a new ontology for LeSS, linked with K-CRIO, we developed three proofs of concept. The first one created a role-dependent view on the sprint page, the second role-dependent forms to manage the information of an item and for the third we implemented two LeSS business rules on roles. Based on these proofs of concept and our study we concluded that Semantic MediaWiki is a platform that could be used to help introduce large scale scrum methodologies. The question to what extent is not totally answered because all business rules implemented were based on a counting, so we did not have the open world assumption of unknown data (Bos 2013).

This chapter repeats the sub questions and gathers the answers to each of them (9.2). These are then combined to answer our main question: implement Semantic Web reasoning to help support the scale agile development process (9.3). A summary of our conclusions can be found under 9.4.

9.2 Sub questions

9.2.1 Which scale agile development framework can we implement in our case study?

The first decision to take in our work was the selection of the scale agile development process. The study by Van Leeuwen (van Leeuwen 2015) gave a clear overview of SAFe, DAD and LeSS with the latter following the scrum methodology. We found an ontology made for the scrum methodology, named K-CRIO, designed by Lin (Lin et al. 2011). Furthermore, the LeSS Company formulated business rules for LeSS (The LeSS Company B.V. 2016c). Based on these findings, the conclusion to use LeSS in our work was straightforward.

Thus, the sub question *c) Which scale agile development framework can we implement in our case study?* is answered. We used the LeSS framework because this framework is based on scrum, for which an ontology already exists and business rules are written.

9.2.2 How and to what extent can we use an existing ontology for agile methodology?

Lin created the K-CRIO ontology in the context of an organization with sub organizations (Lin et al. 2011). The ontology uses only a few predicates which makes semantic reasoning difficult (5.2). Besides, the terms for roles in scrum and LeSS have not always the same definition. Some of the differences are found in the cardinality (5.2). To solve both of the mentioned problems, we decided to create a new ontology to support the LeSS terminology. Keeping in mind the explosion of standalone prototype ontologies (Fitsilis et al. 2014), we linked our ontology to K-CRIO (5.3). The choice to the focus on roles, gave us the opportunity to link the created ontology with the FoaF ontology (Fitsilis et al. 2014), by the Person class or category respectively for Semantic Web and Semantic MediaWiki.

In contradiction with former studies at the Open University of the Netherlands (Rutledge et al. 2016), we did not use Protégé to create the ontology. The ontology is created directly in the wiki by the use of the MediaWiki categories and Semantic MediaWiki properties. The link with the FoaF ontology (Horrocks 2008) is created by using the property `imported from`, available once the FoaF ontology is imported in the wiki. This property will be translated by `same as` when the ontology is exported by the use of the extension RDFIO (Lampa et al. 2017). For the K-CRIO ontology (Lin et al. 2011) we could not use this technique because K-CRIO is not published on the web. For this reason we used the Semantic MediaWiki special property `equivalent URI`, which marks the page as having a meaning in an external URI, thus beyond the wiki and exported the same way as the `imported from` property.

Further research has to prove that the exported ontology has all the features known for ontologies designed as OWL-files, e.g. created by Protégé. The import on the other side has been proven by former research in the Netherlands (Rutledge et al. 2016).

Summarizing above, existing ontologies can be used by the property `imported from` for published ones and `equivalent URI` for not published ones. A real reuse of an existing ontology was not possible because of the difference in terms and the lack of available properties in K-CRIO. We needed more detail and created a new ontology, directly in Semantic MediaWiki, to capture the definitions of LeSS, and created links with already existing ontologies, such as FoaF and K-CRIO. For this result we needed to extend Semantic MediaWiki with the extension RDFIO.

Hereby, the sub question for the ontology is answered: The use nor reuse of an existing ontology seemed to be a good choice. We could avoid to create a standalone ontology by linking this new one to K-CRIO and FoaF.

9.2.3 Which interface can we use in our case study to allow the interaction with the user?

Based on the analysis of Kleiner (Kleiner 2015), the choice for Semantic MediaWiki as platform was obvious. By the use of a wiki all information is available for all users having access to the wiki. So, the requirement that all users have to have access to the information was straightforward. But managing the information gave us some problems.

The extension Page Forms (Koren et al. 2017) allows to structure the input of the user, included basic controls as types and allows value if they are set. As the name suggests, the limit of Page Forms is that the form controls the whole page and access by the use of another form is not possible. To avoid this limitation, we used sub pages, which are technically pages. MediaWiki provides magic words to select the name of the current page or parts of its name but this latter did not seem to work correctly. Thus, we used hard coded references to the sub pages used in the proof of concept. Further study is needed to detect the source of this problem.

Furthermore, Page Forms are used in former research to manage the data in info-boxes defined by Fresnel Forms (Rutledge et al. 2016). Fresnel is Semantic Web-based, thus its lenses use SPARQL for his selector, where Semantic MediaWiki uses inline queries to select the information that has to be rendered.

So, the chosen interface and the answer to the interface sub question, being Semantic MediaWiki, allowed us to provide all needed interactions with the user, included input of information by the use of forms and to compare the results with Fresnel Forms.

9.2.4 How and to what extent can Semantic MediaWiki technology provide an interface to display the progress of product development?

Our research on displaying the progress of the product development is done by a proof of concept implementing the view on the current sprint, with the status of each item of the sprint backlog. A stakeholder does not need the same view as a product owner or a team member. We limited the implementation by only creating an interface for the product owner and the stakeholder, where the former needs much more information than the latter.

As Bao (Bao et al. 2009) suggested, we used inline queries. The needed parameter, being the user, is captured by an existing parser function UserGetName (Ejcaputo 2010). For the implemented descriptive rules (Kardasis & Loucopoulos 2004), we did not need parser functions for Event-Condition-Actions, nor loop functions (Bao et al. 2009). We only needed to extend Semantic MediaWiki by the extension UserGetName (Ejcaputo 2010) to obtain the user's role, used in the inline queries. The parser functions if and UserGetName, combined with inline queries on the item and sprint backlog, created the possibility to display the progress of the product development dependent on the role of the user who logged in in the wiki. The selection of the data from the item and sprint backlog would also be possible by the use of Fresnel lenses, but the question remains how the user's role can be used as parameter to adapt these lenses.

Hence, the answer for how to display the progress is given by Bao and the only added extent we needed to capture the user role to make the interface role-dependent, is given by the UserGetName extension. Due to the need of the role-dependency of the interface, we could not conclude that the same functionality could be implemented by using Fresnel.

9.2.5 How and to what extent can business rules guide the team through the scale agile development process?

As our work focused on user roles, we implemented following rules: (1) The product has just one product owner and (2) the product has minimum one scrum master. These descriptive rules (Kardasis & Loucopoulos 2004) do not need any parameter input, all the information is available in the wiki within the category `HvgLeSS:User`. Hence, the Plan-Do-Check-Act principle (Joosten & Joosten 2005) can check for violation and allows the user to resolve the found violation. To execute this principle we needed inline queries to check the rule (Bao et al. 2009), The result can be given on the page(s) the user wants by calling the template which checks the rule.

The above described way of handling violations does not contradict the agile principle 'Individuals and interactions over process and tools' (Beck 2001) because the violation is solved by the user without driving him in any direction. In the same way other rules can be implemented. Some of these rules are::

- an item needs story points before it can be planned,
- each item needs acceptance criteria
- items, not yet set in a sprint backlog and having the highest priority, have to be popped up during the refinement meeting.

But these checks do not cover enough to answer the question to what extent business rules can guide the team through the scale agile development process. We need to study Event-Condition-Actions, which were considered out of scope in this work, to formulate a substantiated answer on this sub question.

Furthermore, all implemented rules are based on the validation of a number, which exclude the open world result 'unknown' (Bos 2013).

We could conclude that business rules can surely help guide a team through the scale agile development process. But to what extent is not completely answered in our work, due to the chosen focus and the open world assumption of the Semantic Web. Further research is needed to complete this answer.

9.3 Summary of our results

We found the following answers to the sub questions:

- Our case study implements the scale agile development framework Large Scale Scrum (LeSS).
- Use or reuse of an existing ontology seemed not possible, but linking this ontology with existing ones did.
- Semantic MediaWiki could be used to allow interaction with the user by the help of the Page Forms extension which allows only the management of the data for which the user is responsible.
- The progress of the sprint development shows a role-dependent interface which helps the user to focus on the information he needs in his role.
- The business rule implementation focused on roles by informing the user of the violation if the number does not meet the wanted range. This view, and thus the result, is too limited to formulate an answer on the use of business rules within Semantic MediaWiki.

Hence, Semantic MediaWiki supports business rules. To what extent is not proven in our work, but considering the study of Bao (Bao et al. 2009), claiming that Event-Condition-Actions are to be implemented by using parser functions, we might extend our conclusion. Thus, Semantic MediaWiki supports business rules.

9.4 Conclusion

Literature study and some proof of concepts taught us that Semantic MediaWiki has all the abilities to implement business rules. To achieve this result, we needed to install some ready to use extensions for MediaWiki or Semantic MediaWiki (see 4.3 and 4.4). By setting up a wiki with the MediaWiki platform, we created a platform that allows semantic reasoning and creates an interface to support the persons to execute their role in the scale agile development process, being Large Scale Scrum (LeSS).

Extending an existing ontology did not seem possible due to the organization oriented upset for the scrum ontology K-CRIO. Hence, we developed our own ontology and linked this ontology with K-CRIO and FoaF to fulfill the instructions of Fitsilis (Fitsilis et al. 2014). By this choice we hook our ontology in the already available data and help to extend the web of linked data (Bizer et al. 2009).

The proof of concepts to make the interface and form role-dependent, and to implement business rules checking the number of product owners and scrum masters for one product within Semantic MediaWiki taught us that Semantic MediaWiki Platform could support the scale agile development process by semantic reasoning. Especially the first two proofs had a similarity with Fresnel, by selecting the data, format them and generate an output. Semantic MediaWiki makes the selection by the use of inline queries, format the result by skins, style-sheets and the Semantic Result Format extension and generates an output in the wiki. The implementation of the role-dependency, used and implemented in our work within Semantic MediaWiki is not clear and requires further research.

10 Further work

In our work we designed an ontology in Semantic MediaWiki (5.3) and all literature suggests that this can be exported as RDF triples. Hence, reasoning on these triples by the use of Semantic Web technologies as SPARQL must be possible. Proving this assumption could help to promote the Semantic MediaWiki platform.

Once the data is available on the Semantic Web, Fresnel could be used to display the information. The inline queries on items and backlogs could be transformed into Fresnel lenses. Fresnel formats could define the format and several outputs are possible. But without an answer to the question how role-dependent data can be used with Fresnel, the transformation does not have the same functionalities as our implementation.

Our work did not include Event-Condition-Actions rules, for which further research is needed. We assumed that this further work will prove that these rules can be implemented by the use of parser functions as described by Bao (Bao et al. 2009).

All the rules implemented in our work are based on count functions, which avoid the unknown open world assumption (Bos 2013). How Semantic MediaWiki handles the open world assumption has still to be examined.

11 Bibliography

11.1 Scientific Literature

Azizyan, G., Magarian, M.K. & Kajko-Mattson, M., 2011. Survey of agile tool usage and needs. *Proceedings - 2011 Agile Conference, Agile 2011*, pp.29–38.

Bao, J. et al., 2009. Rule Modeling Using Semantic MediaWiki. *Language*, pp.3–4. Available at: http://eprints.ecs.soton.ac.uk/17714/.

Bizer, C., Heath, T. & Berners-Lee, T., 2009. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3), p.1�22.

Bos, P., 2013. *Bedrijfsregels in verschillende vormen*. Open University of the Netherlands.

Chen, H., Finin, T. & Joshi, A., 2005. The SOUPA Ontology for Pervasive Computing. *Computing Systems*, pp.233–258. Available at: http://www.springerlink.com/index/k127108k44351226.pdf.

Deemer, P. et al., 2012. The Scrum Primer. *InfoQ*, pp.1–20. Available at: http://www.infoq.com/minibooks/Scrum_Primer Translations.

Fitsilis, P., Gerogiannis, V. & Anthopoulos, L., 2014. Ontologies for Software Project Management : A Review. , (December), pp.1096–1110.

Gailly, F. & Geerts, G.L., 2013. Ontology-Driven Business Rule Specification. *Journal of Information Systems*, 27(1), pp.79–104. Available at: http://aaajournals.org/doi/abs/10.2308/isys-50428%5Cnhttp://www.scopus.com/inward/record.url?eid=2-s2.0-84879759316&partnerID=tZOtx3y1.

Horrocks, I., 2008. Ontologies and the semantic web. *Communications of the ACM*, 51(2008), pp.58–67. Available at: http://www.scopus.com/inward/record.url?eid=2-s2.0-59449090600&partnerID=40&md5=c7221d4385c3dbd3297b1756c76285ff.

Jakobsen, D., 2016. Investigating Worldviews with Protégé Investigating Worldviews with Protégé. , (August).

Joosten, S., 2010. Bedrijfsregels. *informatie*, januari-fe(August), pp.44–49.

Joosten, S., Wedemeijer, L. & Michels, G., 2010. Rule Based Design. , (February), p.183.

Joosten, S.M.M. & Joosten, R., 2005. Specifying Business Processes by Means of Rules. , (February).

Kardasis, P. & Loucopoulos, P., 2004. Expressing and organising business rules. *Information and Software Technology*, 46(11), pp.701–718.

Kasauli, R. et al., 2017. Adding value every sprint: A case study on large-scale continuous requirements engineering. *CEUR Workshop Proceedings*, 1796(August).

Kleiner, F., 2015. *A Semantic Wiki-based Platform for IT Service Management*. Karlsruhe: FZI Forschungszentrum Informatik. Available at: https://www.researchgate.net/publication/272183598.

Krötzsch, M. et al., 2007. Semantic MediaWiki. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5.4, pp.251–261.

Krötzsch, M. et al., 2007. Semantic Wikipedia. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5.4, pp.251–261. Available at: http://delivery.acm.org.ezproxy.hro.nl/10.1145/1140000/1135863/p585-volkel.pdf?ip=145.24.129.33&acc=ACTIVE SERVICE&CFID=46025479&CFTOKEN=81920587&__acm__=1317632119_a2d262fdb07129061213b85c010ce7ea.

Lassila, T Berners-Lee, J Hendler, O., 2001. The semantic web. *Scientific American*, 21.

van Leeuwen, M., 2015. Agile Scaling @ TOPICUS: Scaling Scrum with help of Agile Scaling frameworks at Topicus Finance.

Lin, Y. et al., 2011. Towards an ontological approach for the description of design processes: the Scrum example. *First International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA)*, (January).

Ruiz-Bertol, F.J., Rodríguez, D. & Dolado, J., 2011. Applying rules to an ontology for project management. *Actas de las 16th Jornadas de Ingenieria del Software y Bases de Datos, JISBD 2011*, pp.257–262. Available at: http://www.scopus.com/inward/record.url?eid=2-s2.0-84873889703&partnerID=tZOtx3y1.

Rutledge, L. et al., 2016. From Ontology to Semantic Wiki – Designing Annotation and Browse Interfaces for Given Ontologies. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9507, pp.53–72.

Slootweg, P.C., 2016. *De implementatie van Hohfeldian legal concepts , ambiguïteit en traceerbaarheid met Semantic Web technologieën*. Open University of the Netherlands.

Soundararajan, S., Chigani, A. & Arthur, J.D., 2012. Understanding the tenets of agile software engineering. *Proceedings of the 43rd ACM technical symposium on Computer Science Education - SIGCSE '12*, p.313. Available at: http://www.scopus.com/inward/record.url?eid=2-s2.0-84858984657&partnerID=tZOtx3y1.

Spreeuwenberg, S. & Gerrits, R., 2006. Business Rules in the Semantic Web, are there any or are they different? In *Lecture Notes in Computer Science*. Amsterdam, Netherlands: LibRT. Available at: https://www.researchgate.net/publication/221510742_Business_Rules_in_the_Semantic_Web_Are_There_Any_or_Are_They_Different.

Taherdoost, H. & Keshavarzsaleh, A., 2015. A Theoretical Review on IT Project Success / Failure Factors and Evaluating the Associated Risks A Theoretical Review on IT Project Success / Failure Factors and.

11.2 Websites

Brickley, D. & Miller, L., 2014. FOAF Vocabulary Specification. Available at: http://xmlns.com/foaf/spec/ [Accessed September 9, 2017].

De Dauw, J. et al., 2017. Extension:Semantic Result Formats - MediaWiki. Available at: https://www.mediawiki.org/wiki/Extension:Semantic_Result_Formats [Accessed August 23, 2017].

Ejcaputo, 2010. Extension:GetUserName - MediaWiki. Available at: https://www.mediawiki.org/wiki/Extension:GetUserName [Accessed September 14, 2017].

Koren, Y., Gamble, S. & others, 2017. Extension:Page Forms - MediaWiki. Available at: https://www.mediawiki.org/wiki/Extension:Page_Forms [Accessed July 29, 2017].

Lampa, S., King, A. & Vrandecic, D., 2017. Extension:RDFIO. Available at: https://www.mediawiki.org/wiki/Extension:RDFIO [Accessed August 20, 2017].

SAFe, 2015. SAFe. Available at: http://www.scaledagileframework.com/ [Accessed January 11, 2016].

The LeSS Company B.V., 2016a. LeSS. *LeSS*. Available at: https://less.works/ [Accessed February 10, 2016].

The LeSS Company B.V., 2016b. LeSS Framework - Large Scale Scrum (LeSS). Available at: https://less.works/less/framework/index.html.

The LeSS Company B.V., 2016c. LeSS Rules. Available at: less.works/less/rules/index [Accessed June 30, 2016].

Wikimedia Foundation, W., 2017. Manual:Developing extensions - MediaWiki. *MediaWiki*. Available at: https://www.mediawiki.org/wiki/Manual:Developing_extensions [Accessed August 10, 2017].

11.3 Articles

Beck, K. et al., 2001. Manifesto for Agile Software Development. *Agile Alliance*. Available at: http://agilemanifesto.org/.

Dubakov, M. & Stevens, P., 2008. Agile Tools . The Good , the Bad and the Ugly . Available at: http://dimsboiv.uqac.ca/8INF851/tp/paper/agiletools.pdf.

Van Leemputten, P., 2016. Wat loopt er mis met overheidsprojecten? *datanews*. Available at: http://datanews.knack.be/ict/nieuws/wat-loopt-er-mis-met-overheidsprojecten/article-longread-652333.html [Accessed June 30, 2016].

12 Annexes

12.1 Annex A: File K-Crio_final.owl

We received the final K-CRIO file from Lin used in her research (Lin et al. 2011). The content of this file is given below.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Ontology"/>
  <owl:Class rdf:ID="DesignObject"/>
  <owl:Class rdf:ID="Organization">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >0</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="isSubOrganizationOf"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="includes"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="isThePlaceOf"/>
        </owl:onProperty>
        <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:minCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Capacity"/>
  <owl:Class rdf:ID="Role"/>
  <owl:Class rdf:ID="OntologyElement"/>
  <owl:Class rdf:ID="Prediate">
    <rdfs:subClassOf rdf:resource="#OntologyElement"/>
  </owl:Class>
  <owl:Class rdf:ID="Interaction">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >2</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasParticipants"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="FormalizedInteraction">
    <rdfs:subClassOf rdf:resource="#Interaction"/>
  </owl:Class>
  <owl:Class rdf:ID="CasualInteraction">
    <rdfs:subClassOf rdf:resource="#Interaction"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="hasContext">
    <rdfs:domain rdf:resource="#Organization"/>
    <rdfs:range rdf:resource="#Ontology"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#includes">
    <rdfs:domain rdf:resource="#Organization"/>
    <rdfs:range rdf:resource="#Role"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="provided">
    <rdfs:range rdf:resource="#Capacity"/>
    <rdfs:domain rdf:resource="#Organization"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="produces">
    <rdfs:range rdf:resource="#DesignObject"/>
    <rdfs:domain rdf:resource="#FormalizedInteraction"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ensures">
    <rdfs:range rdf:resource="#Prediate"/>
    <rdfs:domain rdf:resource="#Capacity"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="required">
    <rdfs:range rdf:resource="#Capacity"/>
    <rdfs:domain rdf:resource="#Role"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="output">
    <rdfs:domain rdf:resource="#Capacity"/>
    <rdfs:range rdf:resource="#OntologyElement"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="isComposedOf">
    <rdfs:range rdf:resource="#OntologyElement"/>
    <rdfs:domain rdf:resource="#Ontology"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#isThePlaceOf">
    <rdfs:domain rdf:resource="#Organization"/>
    <rdfs:range rdf:resource="#Interaction"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="input">
    <rdfs:range rdf:resource="#OntologyElement"/>
    <rdfs:domain rdf:resource="#Capacity"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#isSubOrganizationOf">
    <rdfs:domain rdf:resource="#Organization"/>
    <rdfs:range rdf:resource="#Organization"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#hasParticipants">
    <rdfs:range rdf:resource="#Role"/>
    <rdfs:domain rdf:resource="#Interaction"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="requires">
    <rdfs:range rdf:resource="#Prediate"/>
    <rdfs:domain rdf:resource="#Capacity"/>
  </owl:ObjectProperty>
```

```
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 2.1, Build 284)
http://protege.stanford.edu -->
```