Specifying semantic information on functional requirements

Wuping Yao

University of Tampere
School of Information Sciences
Computer Science
M.Sc. thesis
Supervisors: Zheying Zhang
Jyrki Nummenmaa
May, 2012
Abstract
Requirements engineering is a challenging process in software development projects. Requirements, in general, are documented in natural language. They often have issues related to ambiguity, completeness and consistency. How to improve the quality of requirements documentation remains a classic research topic. This research aims at improving the way of editing and documenting functional requirements. We propose a meta-model to specify the semantic information of functional requirements, and deploy it in the customized wiki platform named Semantic REWiki to support requirements specification. Taken advantage of the collaborative and open feature of a wiki, Semantic REWiki can improve the unambiguousness, completeness and consistency of semantic information specification for functional requirements.

Keywords: Requirement Engineering, Requirements Specification, Functional Requirements, Semantic Information, MediaWiki, Semantic MediaWiki.
Acknowledgements

I wish to express my sincere thanks and appreciations to my supervisors Zheying Zhang and Jyrki Nummenmaa for their patient guidance, valuable comments and constructive advice. I would like to pay special thanks to Zheying Zhang for the supervision and shaping of the thesis.

I wish to extend my thanks to my parents for the essential support in finance and spirit, my deepest thanks to my girlfriend Juan Lv for the encouragement, company and love, and thanks to my brother Wuquan Yao for being there for me.

Finally, I would like to thank Jing Ma for the help in this research, Robert Hollingsworth for the guidance in academic writing. I wish to thank Yanzhao Wen, Liwei Xie and Da Ke for the participation in the experiment and my heartiest thanks to all my friends for the care.

Thank you!

Wuping Yao
Tampere, May 20th, 2012
Contents

1 Introduction .................................................................................................................. 1
2 Requirements Engineering ............................................................................................ 3
  2.1 Requirements ........................................................................................................... 4
  2.2 Functional requirements .......................................................................................... 4
  2.3 Characteristics of good requirements specification ............................................... 7
  2.4 Requirements engineering processes ...................................................................... 8
  2.5 Tools for requirements engineering ....................................................................... 10
    2.5.1 Characteristics of requirements engineering tools ...................................... 10
    2.5.2 A comparison of different tools for requirements engineering ................. 11
    2.5.3 Prospects of requirements engineering tools ............................................. 12
3 Semantic Information Specification in Functional Requirements ............................... 13
  3.1 Case grammar ........................................................................................................ 14
  3.2 A set of adapted semantic cases ............................................................................ 15
  3.3 A meta-model of functional requirements specification ..................................... 17
  3.4 An example of applying the meta-model to specify FRs .................................... 19
4 Wikis and their support in requirements engineering .................................................... 22
  4.1 An overview of wikis ............................................................................................. 22
  4.2 Semantic Web ......................................................................................................... 23
  4.3 Wiki applications .................................................................................................... 25
    4.3.1 MediaWiki ....................................................................................................... 25
    4.3.2 MediaWiki extensions .................................................................................... 26
  4.4 Applying semantic wikis to support requirements development ...................... 27
  4.5 Wiki tools for requirements engineering ............................................................... 28
5 Semantic REWiki ......................................................................................................... 36
  5.1 Semantic model in wiki .......................................................................................... 36
    5.1.1 Feature ........................................................................................................... 37
    5.1.2 FunctionalRequirement ............................................................................... 37
    5.1.3 Entity .............................................................................................................. 38
    5.1.4 Condition ....................................................................................................... 38
  5.2 Using Semantic REWiki to edit and specify FRs .................................................... 38
6 Evaluation and Discussion ............................................................................................. 41
  6.1 Evaluation based on the goals of the three dimensions of RE .............................. 41
  6.2 Evaluation based on IEEE recommended practice .............................................. 43
  6.3 Experiment analysis ............................................................................................... 44
    6.3.1 Experiment background .................................................................................. 44
    6.3.2 Process analysis ............................................................................................. 45
    6.3.3 Feedback analysis .......................................................................................... 47
  6.4 Discussion ............................................................................................................... 48
7 Conclusion ........................................................................................................................................50
References ..........................................................................................................................................51
Appendix 1: Semantic information of online shopping application ...........................................59
Appendix 2: Questionnaire for Semantic REWiki .............................................................................60
1 Introduction

Requirements engineering (RE) is widely believed to be the most important process in software engineering (SE) [Damian et al., 2005]. The outputs of a RE process are requirements specifications, which document the important information on functions, behaviours, and quality factors of the system to be developed. They take an important role to guide the analysis, design, implementation, testing in software development. Therefore, the quality of requirements specification forms an essential issue in software development.

Requirements specification is the basis of requirements analysis, requirements validation and requirements verification [Nummenmaa et al., 2010]. In general, requirements are documented in structured natural language (NL). The format of the documentation follows specific criteria. It is more formal than those which are written without any restriction. However, using NL to specify requirements will cause ambiguity which inherits from the nature of NL. In addition, it cannot ensure the consistency between and/or inside different specifications. How to improve the quality of requirements specification forms the starting point of this research. Concerning that requirements specification is a broad area to explore; the scope is further limited to focus on the representation and documentation of functional requirements (FR). Taken these factors into account, two research questions are proposed:

- What is the essential information on a functional requirement?
- How to specify the semantic information on a functional requirement in a precise and consistent way while remaining the ease of understanding?

Based on Fillmore’s [1968] case grammar theory, a set of adapted semantic cases is proposed to specify the semantic information on FRs. The meta-model of adapted semantic cases is similar to the one proposed by Guo et al. [2010], which focuses on unifying the semantic information of FRs. Our approach aims at identifying different linguistic elements of FRs and documenting the semantic information in classified units in order to facilitate the follow-up requirements analysis and validation.

In order to evaluate the approach of applying adapted semantic cases to specify FRs, a suitable platform is required to construct the implementation of the meta-model. Concerning that the general word processors cannot provide support for collaborative communication and the dedicated software for RE is too complicated to use for non-technical stakeholders, the options are narrowed to collaborative tools. Wiki becomes the most attractive choice since it is a flexible and open platform for asynchronous collaboration. In addition, the wiki community owns a huge extension family which
makes it possible to enhance the functionality and user interface. Starting from this point of view, I decided to proceed the research following the constructive approach. Taken the advantage of Semantic Forms (SF) [Koren et al., 2008] extension, a set of forms and templates can be defined in wiki system to facilitate the process of requirements specification.

It is not a new field to develop wiki-based platform for RE. Many applications or research prototypes such as SOP [Decker et al, 2012], Moki [Rospocher et al., 2009], ReqWiki [Concordia University, 2012], etc. have provided support for requirements development and management. However, these tools document and manage requirements on a coarse-grained level, which is insufficient to support requirements analysis and modelling [Jing et al., 2012]. The expected contribution is to combine the representation of the semantic information of a FR with the functionality of a semantic wiki to specify requirements in a clear and consistent manner.

This thesis is composed of seven chapters. Chapter 2 starts with the basic concepts of requirements, requirements engineering processes and then introduces some criterions to evaluate the quantity of requirements. It ends with the discussion of requirements engineering tools. Chapter 3 analyses Fillmore’s [1968] case grammar theory and present the adapted semantic cases for requirements specification. Chapter 4 starts with an introduction with regard to the concept of wikis, wiki applications and Semantic Web [Palmer, 2001]. After that, it presents five RE-specific wikis and analyses the characteristics of them. Chapter 5 combines the definition of semantic forms and an example to explain the implementation of wiki-based solution for requirements specification. Chapter 6 evaluates the quality of documented requirements and analyses the result of the experiment with regard to Semantic REWiki. Finally, the thesis ends up with the contribution, limitation and future work concluded in Chapter 7.
2 Requirements Engineering

Requirements engineering is a branch of software engineering which focuses on the goals for functions of the system, the constraints on the system and the relationship of these factors. It applies the methodology of engineering to deal with the developing and management process of requirements. [Zave, 1997]

Pohl [1992] summarized the process of RE into three dimensions: specification, representation and agreement. It starts from the initial input which includes opaque specification, informal representation and personal view to the desired output including complete system specification, formal representation and consistent agreement.

![Figure 1. The three dimensions of requirements engineering [Pohl, 1992]](image)

The specification dimension concerns with the degree of requirements understanding at a given time. The specification should be correct, be correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable and traceable in order to achieve the goal of this dimension. The representation dimension focuses on the representation approach of the information regard to the system. At the beginning of the RE process, it tends to apply informal language to represent the requirements since the requirements are opaque and informal language is user-friendly. However, formal specification should be generated in the end. The agreement dimension is related to the degree of agreement the stakeholders reach on a specification. The stakeholders initially have their own view towards the requirements of the system. The aim of this dimension
is that the stakeholders hold the common agreement about the final requirements specification.

In order to assure the success in a proceeding of requirements engineering, it is necessary to comprehend the concept of a requirement and the approaches to improving the quality of the requirements.

2.1 Requirements

“Requirements are...a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system.” [Sommerville and Sawyer, 1997] Requirements reflect the real needs of the stakeholders, and form the basis for the following activities in software engineering.

From the perspective of the scope, requirements can be divided into three levels: user requirements, business requirements and system requirements. User requirements define user goals or tasks that the users must be able to perform with the product in order to achieve business goals. Business requirements describe high-level objectives of the organization. They concern with business needs. System requirements describe the top-level requirements for a product that ensures to satisfy stakeholder’s requirements. [Wiegers, 2003]

Requirements can be classified into two types: functional requirements (FR) and non-functional requirements (NFR). FRs describe functions and services of the system that support user goals, tasks or activities [Malan and Bredemeyer, 2001]. Basically, FRs are composed by verb-noun expressions. For example, “The system should display the balance”. The core content of this FR is to display the balance. NFRs are the constraints upon the behaviour of the system [Andrews, 2009]. NFRs can be extracted from the adverbs or modifying clauses [Rawlins, 1999]. For example, “The system should safely display the balance”. Compared to the former example, it contains an additional NFR: keep the security of the system while displaying the information. From the comparison of FRs and NFRs, it is obvious that the context information of FRs is the environment, constraints and limitations of them; that is to say, the NFRs. In addition, there are some dependencies among the FRs, such as the sequence dependency or the logical relationships. These dependencies, together with NFRs, composed of the context of FRs.

2.2 Functional requirements

FRs define the intended functions that a system is desired to accomplish. It may concern with the definition of the characteristics of a product including the technical details, data manipulation and other functionalities [Wikipedia, 2012; Salustri, 2004]. The functions
of a product are different from its behaviours. Behaviours describe how the product responds a stimulus, whereas the functions describe how that stimulated response serves some purpose in a specific environment [Salustri, 2004]. Hence, a FR is a precise and complete specification of the function of a product, phrased in terms of tasks or activities it shall accomplish during the operation [Salustri, 2004].

A FR typically consists of a unique name or number to identify itself, a brief summary to explain the content of the FR and a rationale to state the reason for the FR [Wikipedia, 2012]. The core of a FR is the required behaviour [Wikipedia, 2012], which is basically specified in verb-noun phrases as mentioned before. However, there are some other details needed to be specified, for example, the relevant actors involved in the FR, the environment, etc. The absence of these details will cause problems in FR specification. The following gives two bad FRs to illustrate the issues commonly occurs in FRs specification [Salustri, 2004; Fleck, 2007]:

- Use a grade of steel that maintains toughness.
- The system shall validate and accept credit cards and cashier’s checks.

One issue in the first FR is that it does not explain the subject or actor of the FR. Who will use the steel? Another issue is that the reference to a material is not appropriate [Salustri, 2004]. It lacks of an explanation to the material. The problem in the second FR is that it actually contains more than one FR. Combing several FRs into one statement always causes issues relates to ambiguity and precision. In this example, if the validation of the credit card passes, but the validation of the cashier’s check fails, what shall it react? The result is misunderstanding.

In order to ensure the quality of the FR, the following information shall be specified in a FR: the identification, the required behaviour, the initiator of the behaviour, the inside limitations, the outside constraints and a complement description. The identification is a symbol to distinguish one FR from the others. The required behaviour describes the action and the executing target of it. The initiator explains the source of the action. The inside limitations and the outside constraints compose the whole environment of the FR. Finally, any further detail, instruction or the rationale can be put in a complement description, and then they form a complete specification of a FR.

In general, requirements are specified in structured NL, and the template of IEEE Standard 830-1998 is widely used in companies as the guideline to specify the requirements. It is well-structured and comprehensively-defined so that it is suitable for many different kinds of projects. However, it is difficult to maintain the specification
when the scale of the requirements increases. It always causes consistency problems. In addition, the statements with ambiguity will affect the implementation of the system.

Another approach is to use a graphical notation for requirements specification, for example, UML [Object Management Group, 1997]. UML can address the specification of all the important analysis, design, and implementation decisions which must be made in developing and deploying a software-intensive system [Booch et al., 1998]. Class model, which is the core of the UML specification, together with models including use case model, activity model and collaborations has become a standard language for writing software blueprints [Glinz, 2000; Booch et al., 1998]. Furthermore, use cases are always used to capture the FRs. “The use cases capture who (actor) does what (interaction) with the system, for what purpose (goal), without dealing with system internals” [Malan and Bredemeyer, 2001]. They can be achieved from organizational or business rules, or discovered through elicitation sessions with the stakeholders in the organization [Wikipedia, 2012]. Sometimes, the use cases are generated after gathering and validating a set of FRs [Wikipedia, 2012]. A complete set of use cases can specify all the available ways to use the system, and therefore defines all behaviour required of the system [Malan and Bredemeyer, 2001]. Each use case illustrates the behavioural scenarios through one or more functional requirements [Wikipedia, 2012]. In addition, use cases can generate collaboration diagram and sequence diagram. The collaboration diagrams provide an approach for considering run-time qualities such as performance, security, etc. [Malan and Bredemeyer, 2001]. The sequence diagrams describe the object interactions arranged in time sequence [Wikipedia, 2012]. The use case approach is an efficient and effective technique for collecting essential requirements from customers and for contributing to focusing on their real needs [Wiegers, 1997]. However, Use cases describe system behaviour mainly from an actor's point of view, which omits much detailed information. For example, use cases can explicitly illustrate the interactions in the front end of a website, but it cannot explain the detail of the data processing and transforming in the back end. The developers need many other views to properly design and implement a system [Wiegers, 2003].

Except the informal way and semi-formal way, FRs can be specified in formal specification. “A formal specification is the expression, in some formal language and at some level of abstraction, of a collection of properties some system should satisfy” [Lamsweerde, 2000]. A formal language is a language with a mathematically defined syntax and semantics [Gobbo, 2000]. Generally, it is based on an algebra or calculus [Rob, 2002]. Due to the mathematical definition of the formal language, the ambiguity problem in requirements specification can be avoided and the consistency could be automatically checked [Gobbo, 2000]. However, there are two problems in applying
formal approach to specify FRs. Firstly, it requires additional training of the formal language. The process is expensive and time-consuming. In addition, the formal way may contain the essential components of a FR, but it normally is not equipped with a field to specify the rationale or to make some further explanation in free text. The problems are always related to the communication process.

The existing techniques have their own strengths in FRs specification. However, all of them have issues in the follow-up software analysis or design process. The rest part of this thesis focuses on explaining the solutions for the second research question. The discussion starts with the discussion with regard to the quality of requirements.

2.3 Characteristics of good requirements specification

A software requirements specification (SRS) is an important part of the requirements process of the software life cycle and is used in design, implementation, project monitoring, verification and validation, and training as described in [IEEE Std 1074-1997]. Characteristics of good software requirements specification have been summarized in [IEEE Standard 830-1988]. It is argued that an SRS should be correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable and traceable.

**Correct**

An SRS is supposed to be correct if and only if all the stated requirements reflect the needs which the software really means to have [IEEE, 1998; Rajnish, 2010]. The correctness should be semantically and syntactically checked [Firesmith, 2003].

**Unambiguous**

An SRS is supposed to be unambiguous if and only if all the stated requirements have only one way to interpret [IEEE, 1998]. Generally, SRS is written in NL. However, NL is highly prone to be ambiguous. Fortunately, there are several ways to avoid ambiguity, including conducting formal inspections of the requirements specifications, writing test cases from requirements, creating user scenarios that illustrate the expected behavior of a specific portion of the product and writing the SRS in a particular requirements specification language [IEEE, 1998; Wiegers, 1999].

**Complete**

An SRS is supposed to be complete if and only if it includes all significant requirements, definitions of responses to all input, definitions of all terms, units of measure, full labels and references to all pictures, tables, and diagrams [IEEE, 1998]. However, it is difficult to spot the missing requirements since they are not elicited. An approach to improve
this situation is paying more attention to requirements hierarchy of SRS so that it is more possible for the viewers to discover the missing part [Wiegers, 1999].

**Consistent**
A SRS is supposed to be internally consistent if and only if no subset of individual requirements described in it conflict. There are three kinds of potential conflicts, including conflict between the specified characteristics of real-world objects, logical or temporal conflict between specified actions and conflict in the terminology of the same real-world object [IEEE, 1998].

**Ranked for importance and/or stability**
An SRS is supposed to be ranked for importance and/or stability if all the requirements have an identifier to indicate either the importance or stability of that particular requirement. It is understandable since each requirement has different importance. To rank the importance and stability of all the requirements makes the specification clearer and more explicit [IEEE, 1998].

**Verifiable**
An SRS is supposed to be verifiable if and only if there exists some finite cost-effective process with which a person or machine can check that the software product meets all the requirements in it [IEEE, 1998].

**Modifiable**
An SRS is supposed to be modifiable if and only if any changes to the requirements of it can be made easily, completely, and consistently while retaining the structure and style. The prerequisite for a SRS to be modifiable is to have a coherent and easy-to-use organization, separately express each requirement and avoid redundancy [IEEE, 1998].

**Traceable**
An SRS is supposed to be traceable if each of its requirements is clear and it facilitates the referencing of each requirement in the later development or enhancement documentation process [IEEE, 1998]. Traceable requirements are uniquely labelled and well-structured written instead of using large, narrative paragraphs or bullet lists [Wiegers, 2003].

### 2.4 Requirements engineering processes

The process of engineering requirements can be separated in two workflows: requirements development and requirements management. Compared to requirements management, requirements development focuses on introducing new requirements during the early stages of the software development process. It can be further divided
into requirements elicitation, requirements analysis, requirements specification and requirements validation [Abran et al., 2005]. Requirements management concerns with monitoring changes to all requirements already established and to assess the impact of including them into the target system [Wiegers, 2003].

Software Engineering Institute [1994] defined requirements elicitation as “the process of identifying needs and bridging the disparities among the involved communities for the purpose of defining and distilling requirements to meet the constraints of these communities”. Stakeholders include requirements analyst, customers, developers and end-users are involved in requirements elicitation process [Wood, et al., 1994]. Due to the reason that different stakeholders have distinct ways to store, recognize and express their knowledge regard to the problem domain, appropriate techniques or methods should be applied according to the stakeholders to assist the process of requirements elicitation [Zhang, 2007].

“Requirements analysis deals with large number of requirements information, detects and resolves conflicts, scopes the system and defines interfaces with the environment, translates system requirements into software requirements and provides feedback to the stakeholders” [Felici, 2004]. The goal is to discover problems, incompleteness and inconsistencies in the elicited requirements [Smith et al., 2004].

A specification of the system, at least for the current version of the system, must exist at the end of requirements engineering [Pohl, 1992]. The software requirements specification, which is also called a functional specification, a product specification, a requirements document, or a system specification in different companies, is used to explain the functionalities, capabilities and constrains of a software system. It is necessary to be documented as detailed as possible due to that it is the basis for project planning, designing, coding and testing process [Wiegers, 2003]. Structured NL is the most common way to document the requirements specification.

The purpose of validation is to confirm that the description of the requirements specification document is acceptable for the system which is going to be implemented [Kotonya and Sommerville, 1998; Saqi and Ahmed, 2008]. The validation process checks the content of what the system is going to be is consistent with the needs for the stakeholders and then achieves feedback from the stakeholders in order to inform further iterations of elicitation and/or modelling/specification [Jureta et al., 2009]. In detail, requirements validation process aims at ensuring that all the requirements are correct, complete, and consistent; a model can be created which satisfies the requirements; a real-world solution can be built and tested to prove that it satisfies the
requirements [Bahill and Henderson, 2004]. Generally, the description of requirements is fine when they are in the SRS. [Wiegers, 2003]

Shroff [2001] compared different definitions of requirements and summarized it as that “Requirements Management is a process of eliciting, documenting, analysing the impact of change, tracing, storing, organizing, visualizing the requirements, identifying and managing multiple versions of items, and communicating the changed requirements to all the team members.” The purpose is to establish a common understanding between the stakeholders on customer's requirements that will be addressed by the software project [Paulk et al., 1993]. It concerns with all the activities that maintain the integrity, accuracy and currency of the requirements agreement and includes everything which is related to the requirements in the product lifecycle [Wiegers, 2003 and 2005].

2.5 Tools for requirements engineering

Requirements management tools aim at managing and controlling requirements within a geographically-distributed team whose members come from different field [Lang and Duggan, 2001]. To apply a suitable requirements management tool would efficiently improve the process of requirements development and effectively maintain the stability of the requirements.

2.5.1 Characteristics of requirements engineering tools

A summary of the characteristics of RE tools is shown below [Wiegers, 2003; Hoffmann et al., 2004; Lang and Duggan, 2001]:

RE-specific tools facilitate the process of communication in two aspects. One is the contribution to the collaboration between stakeholders. RE-specific tools tend to be built based on the web, which means the stakeholders would be able to work concurrently even though they are geographically distributed. Another one is the improvement to the understanding between the stakeholders and the requirements. The requirements which are written in NL with the help of rich media like pictures and videos are more understandable for non-technique stakeholders. However, the technique stakeholders prefer to describe the requirements in formal language or model-based description in order to ensure the conciseness, unambiguity and accuracy of requirements. A sophisticated RE-specific tool can balance the technical and non-technical notation of the requirements to meet the needs of different stakeholders.

RE-specific tools can improve the process of change control. First of all, it can provide access control to the change of the requirements. Change requests can be proposed by stakeholders with low-level authority, and then authenticated and approved by stakeholders with high-level authority. It contributes to the stability of the changes to
Moreover, when a change is conducted, the status of the change including the date, the person and the content of the change should be recorded to facilitate the tracing process later on. The impact of a change can be handled by RE-specific tools through maintaining the links between the requirements. The links can facilitate the impact analysis by tracing the affected requirements of the specific change.

RE-specific tools normally are able to maintain the documents of the requirements in four aspects. First of all, they support to document the acquainted requirements with a unique and identifiable description assigned to them. Secondly, they would be able to classify the requirements into well-defined groups which would be convenient to navigate. Last but not least, they can generate reports which apply the template of specific standard.

2.5.2 A comparison of different tools for requirements engineering

Currently, numerous RE-specific tools exist. They can be classified as three types: plain text documentation tools, general collaborative tools and dedicated RE tools. A comparison of the pros and cons of the tools is listed in Figure 2.

![Pros and cons of different tools for RE](Decker et al, 2007)

The text documentation tools store the requirements in an office suite. It is an economic way to manage requirements and it is easy to use since the knowledge of office suite is widespread among companies and organization. However, it sacrifices the integrity of the RE-specific tools due to its simplicity. Features include status tracing and change control cannot be satisfied. Normally, text documentation tools are suitable for small projects which have lower complexity. In addition, it can be regarded as an auxiliary tool to help the requirements management process when applying general collaborative tools or dedicated RE tools.
The general collaborative tools store the requirements on the web to improve the concurrent collaboration. They meet the balanced point of complexity in operation. The general collaborative tools normally can be accessed through the browser, which improves the flexibility. However, some of the collaborative tools still lack of some features like versioning and tracing.

The dedicated RE tools has a specific database or distribute-databases to store the requirements and the status of them in a well-organized structure. They can provide very professional requirement management functions throughout the whole RE process. However, the disadvantages of these tools are also very impressive. One issue is that the licence is very expensive, so normally only large companies are able to use them. Another issue is the complexity of the tools. It needs a training process to learn to use them since they are too professional. The non-technique stakeholders cannot take advantage of this kind of tools, which decreases the collaboration to some extent.

2.5.3 Prospects of requirements engineering tools

“Most of the advances in software development tools and techniques over the last three decades have focused on productivity rather than quality, and have been primarily concerned with the work of individuals” [Booch, 1998; Lang and Duggan, 2001]. Improving the productivity and collaboration is then become the issue needed to be concerned when developing new RE-specific tools.

The key characteristics of RE-specific tools in the future are low-cost, geographical-distribution, domain-specialization and mobile-integration [Finkelstein and Emmerich, 2000; Alexander, 2012]. Since the tools appear to the market become increasingly cheaper, the price of RE-specific tools tends to cost less, or even totally free. In order to improve the collaboration issue of some of the existing requirements management tools, the tools should be distributed instead of centralized. So they tend to integrate with web to eliminate the geographical communication problem. With the rapid development of software industry, software products have covered most of the domain in the world. The general RE tools will not be applicable for all the applications from different fields. Domain-specification is an inevitable trend for RE tools. Domain-specific knowledge will be taken into account when development the tools to provide domain-oriented features. Nowadays, mobile device-based applications play a more and more important role. It will also be a trend for RE tools to be development based on mobile devices. The stakeholders then would be able to trace the status of changes at any time.
3 Semantic Information Specification in Functional Requirements

Requirements specification is the basis of requirements analysis and modelling. Well-defined and correct requirements have traditionally been seen as a critical factor behind software project success [Pohl, 1992; IEEE, 1998]. There are several criteria to evaluate the quality of requirements specification, such as the three dimensions of RE [Pohl, 1992] and the IEEE recommended practice [IEEE, 1998], which have already been introduced in Chapter 2. These standards or guidelines focus on the higher level of FRs specification and propose the main features that a functional requirements specification should have. However, they pay less attention to the basic elements that an individual FRs statement should have.

The approach to represent requirements can be categorized into three types: formal, semi-formal and informal specification. The desired output of a RE process is a complete system specification expressed using a formal language on which all people involved agree [Pohl, 1992]. In order to represent the semantic meaning of a FR in an explicit manner, formalized representation is needed. The formal representation has precise syntax and rich semantics and thus, provides a better basis for reasoning and verification. However, it is difficult for non-technical users to understand. The semi-formal representations are based on graphical modelling of the system, which provides a clear and more understandable view of the system. It can also reduce the ambiguous semantics, but it provides limited support in reasoning and verification. The informal representation which is normally written in NL is easy to understand, but it often inherits ambiguity from NL. Since FRs are commonly written in structured NL in industry, there is hence a need to develop or adapt a suitable approach to represent FR without ambiguity while providing a better understanding for both technical and non-technical users.

Even if a FR was specified correctly and precisely, the action and its associated information might not be equally obvious for heterogeneous groups of stakeholders to achieve a common and correct understanding [Nummenmaa et al., 2010; Lamsweerde, 2000; Young, 2002; Pohl, 1993]. Many researchers have proposed approaches to achieving the basic information about the functions of a system and have contributed to capturing and analysing FRs. However, they still miss some detailed information about the function, such as conditions, triggers and constraints of the function. In order to obtain a clear and common understanding of FRs, the FRs need to be further specified based on individual statements. Functional requirements specifications are typically expressed in NL and each statement can be regarded as a common sentence including nouns, verbs, and adjectives among others. Therefore, analysing the semantic cases for
FRs is the prerequisite for FRs modelling and reasoning, and can serve as a basis for functional requirements specification.

3.1 Case grammar

According to Fillmore [1968], a simple sentence consists of a verb and a set of noun phrases. Each noun phrase holds a relationship of a particular semantic type with the sentence. These types linguistically correspond to different cases. Therefore, an essential set of cases should exist to fit in the case system of every known language. Fillmore [1968] incorporated these cases into a system of linguistic analysis called case grammar, which focuses on the link between a verb and the grammatical context it requires [Fillmore, 1968):

- **Agentive (A)**: the instigator of an action. It can be identified by recognizing the source of the action and normally it is an animate entity.
- **Dative (D)**: the acceptor of an action. It can be identified by recognizing the destination of the action and normally it is an animate entity.
- **Factive (F)**: the acceptor of an action. It can be identified by recognizing the destination of the action or the result of executing an action.
- **Objective (O)**: the object related to or affected by an action. It can be identified by recognizing the nouns which are not agentive, dative and factitive.
- **Instrumental (I)**: the approach to conducting an action.
- **Locative (L)**: the location or environment in which an action happens.

Each case in the case grammar addresses a particular semantic concern of the verb in a sentence and represents a potential semantic slot that may or must be related to the verb. The verb is the core of a sentence. It will cause a case frame which contains a set of semantic slots to describe the state or certain circumstance. For example, the verb “give” is obviously not only connected to an objective slot such as "what gives?" or "what is given?" but also related to an agentive slot to answer “who gives?” In addition, a dative slot, “give to whom?”, and an instrumental slot, “open with what?” can also be filled for further explanation. [Fillmore, 1968; Nummenmaa et al., 2010; Niu and Easterbrook, 2008; Liaskos et al., 2006]

Case grammar can be used to formalize requirements specification and to derive different system analysis models. Nummenmaa et al. [2010] take advantage of grammatical conversions to produce action-based executable specifications from the FRs. Since there is no deliberate separation of an action and its associated information in the requirements specification written in NL, they create an executable software system specification from requirements to stimulate and explore the dynamic behaviour...
of the required system so that different stakeholders can have a common understanding of it. [Nummenmaa et al., 2010]

Mu et al. [2009] have adapted case grammar to customize a general set of dimensions for conceptualizing the variation structure of FRs profile. They propose an extended FRs framework which consists of ten cases and then analysed the textual documents based on it. The framework can contribute to extracting detailed and integrated information with regard to the variability of FRs and to generating a functional variability model from text-based requirements with less effort.

Liaskos et al. [2006] also introduce a general set of semantic cases for variability concerns based on the universal set of frame elements in case grammar. Their approach is to enhance variability in goal decomposition and to aim at supporting requirements identification for highly customizable software.

Moreover, many recent researches on domain engineering [Moon et al. 2005][Niu and Easterbrook, 2008][Guo et al., 2010] [Guo et al., 2012] adapted semantic cases for analysis and documentation of requirements variability in a software product line. They achieved a prominent effect in promoting the quantity of requirements specification and further prove that the analysis of the details in NL expressed requirements is a promising research field.

3.2 A set of adapted semantic cases

Even though the case grammar theory helps to analyze semantic role of elements associated with an action in a FR, it is difficult to precisely identify various semantic cases expressed in a textual requirement statement. If the requirements can be documented into a set of semantic cases, the achieved information of FR can be directly used in the later RE process. An adapted set of semantic cases is proposed in order to adjust to the needs of requirements specification. In this rule, the semantic cases are categorized into entities and constraints on action.

An entity is a thing with a distinct and independent existence. It can be a person, an object, a place or an event. A FR may concern more than one entity. An entity may have a dynamic role in FRs: it can be the one which causes an action and can also become the one which is affected by another action. Therefore, an entity can be categorized as agentive, dative and objective according to its semantic role taken in a FR.

- **Agentive (Ag)** is the initiator of an action; it is the one who causes or dominates the action. An agentive might be a system or its components, an external actor or a group of actors, e.g. “The \{system\}$_{Ag}$ displays the menu to the client”.

• Dative (Da) is the beneficiary of an action; it is the one who is indirectly affected or dominated by the action. A dative might be an external actor or a group of actors in an interaction, e.g. “The system displays the menu to the {client}Da”.

• Objective (Ob) is also the beneficiary of an action; it is the one who is directly affected or dominated by the action. A dative might be the system or a group of data and it should be inanimate, e.g. “The system displays the {menu}Ob to the client”.

Constraints are the prerequisites for or restrictions on an action. Generally, they are conditions under which an action can be achieved or the means through which an action can be executed. From this point of view, the constraints can be further divided into two sets. One of them focuses on the state constraints which should be satisfied when an action is activated or completed. This extracted semantic set can be defined as conditions. The following is a detailed explanation of three of these conditions.

• Pre-condition (Pr) is a condition or predicate that must be satisfied prior to the execution of an action. It is used to represent the state of entities before the trigger is activated, e.g. “{the client should login}Pr before he/she starts to browse the information”.

• Post-condition (Po) is a condition or predicate that must be satisfied after the execution of an action. It is used to represent the state of entities resulting from the action, e.g. “after the client clicks ‘submit’ button, {the form should be saved}Po”.

• Trigger (Tr) is the action exerted on the system or an attribute of entities that activates the action of the FR, e.g. “after {the client clicks ‘submit’ button}Tr, the form should be saved”.

While the other cases are more associated with the means through which the action is implemented, they can be understood as adverbial modifiers in a NL sentence. Those cases can then be regarded as constraints which include temporal, locative and instrumental:

• Temporal (Te) defines the frequency or the duration of an action, e.g. “the system should restart {every week}Te to install updates”.

• Locative (Lo) defines the location(s) where the action is supposed to take place, e.g. “when the client chooses an item, it should be put into the {shopping cart}Lo”.

• Instrumental (In) defines the approach such as the tool(s) or method(s) used by the agentive to perform the action, e.g. “the client should pay the bill through {net bank}In”.
With the definition of entities, conditions and constraints above, the semantic information on a FR can be explicitly specified. However, the entities and conditions may have dependencies on each other. It is necessary to design a model which is able to not only specify the semantic information on a FR, but also describe the relationships inside/between entities and/or conditions on it.

3.3 A meta-model of functional requirements specification

On the basis of the semantic cases adapted to specify semantic roles of segments of information in a FR, a model to specify FRs is proposed to provide semantic support for FRs representation. The following is an entity-relation description for the model. All the elements analysed in section 3.2 are taken into account and then are abstracted as four main entities: “FunctionalRequirement”, “Entity”, “Condition” and “Constraint”.

A FunctionalRequirement consists of an id, an action, a segment of description and several entities, conditions and constraints. The id attribute is used to identify and track a piece of requirements. Action is the sign to confirm a FR, so a FR can only contain one and only one action. In addition, the Entity can be used to describe the one who causes the action and the one who is affected by it. Condition describes the prerequisite for the FR and Constraint explains the limitations of it. As mentioned in section 2.1, requirements can be divided into three levels: business requirements, user requirements
and system requirements [Wiegers, 2003]. This relationship can be reflected by an aggregation relation from the FunctionalRequirement to itself. Furthermore, even the same level requirements might have a sequential dependency on each other. The dotted line is used to demonstrate the order between the same level requirements and the activation dependency among cross level requirements.

All the instances included in the FRs are abstracted as an Entity class. An entity has a name to identify itself and an attribute to clarify its constraint or property. Moreover, an entity always belongs to a type: an actor, a kind of data, a set of device or the whole system. These types are grouped into an enumeration named Entity Type. To distinguish the entities which cause the action and the ones which are affected by it, three subclasses are derived from it. Agentive is used to represent the agent of an action. A Dative is the directly affected entity while an Objective is the one indirectly affected. Furthermore, the entities can be categorized into different groups, so there is a composition relationship from class Entity to itself.

A condition has a name and a part of description to explain it in NL. Since conditions are used to express the prerequisite for a FR and since it should reflect the effect happening/which happens to the entities, so there is a relation to Entity class. In addition, a group of conditions have a very complex relationship with each other. Depending on the occasion when a condition occurs, the conditions can be further abstracted as three subclasses. A complicated condition can be further divided into detailed conditions, so there is a composition relation to it. In addition, a logical “AND” and “OR” relationship might happen between conditions. The dotted dependency is used to express this kind of logic relevance.

In addition to a name attribute and a segment of description, a Constraint also has a pair of attributes to clarify the limitation: type and content. As discussed in section 3.1, the constraints may belong to different types: instrument, temporal and locative. They are grouped into ConstraintType, which is an enumeration collection. An additional attribute, content, is attached to describe the detailed information for a certain type of constraint.

The meta-mode explicitly defines semantic elements within an action of a FR. Following the definition of the meta-model, we can easily identify concepts (i.e. entities) that are used in the application domain, analyse the properties possessed by entities (i.e. attributes of an entity), and analyse the possible sequence of operation by checking the conditions of an action. It is intuitive to further generate data structure models and behaviour models for software analysis and design.
3.4 An example of applying the meta-model to specify FRs

In order to further demonstrate the performance of the meta-model, the information of FRs for the online shopping application can be specified as an example. In this example, only FRs are taking into account. Table 1 lists the contained FRs in a complete feature of the online shopping application called “Place an order”. One key point which needs to mention is that each single cell in Table 1 does not mean an individual FR. Since the proposed semantic cases in this research can be realized as entities and constraints on actions, the core point to elicit a FR is to locate the verb to fit the action slot in adapted semantic slots.

Based on the meta-model, the table in Appendix 1 presents the semantic information of the FRs listed in Table 1. The four columns respectively specify the four FRs including order creation, order modification, order display and order confirmation. Due to the reason that some of the requirements contain more than one action, for example, add/remove items in the requirement of order modification, the last three FRs are further divided into several sub-requirements to obey the principle that a FR is elicited based on an action. The analysis in these tables gives the users a detailed understanding with regard to the roles of different ingredients in semantic forms. They form the basis of the following requirements documentation process, or even be the basis for requirements analysis and modelling.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order.Place</td>
<td>The system shall show the item information to the client.</td>
</tr>
<tr>
<td>Order.Place. Create</td>
<td>The client can select one or more items, define the quantity of the selected items, and create a new order. The new order will be the current order.</td>
</tr>
<tr>
<td>Order.Place. Modify</td>
<td>The client can choose a quantity of items for the current order. The client can modify (i.e. add/remove) items into the current order. The client can modify the quantities of items for the current order. If there are any changes in the order, the system shall prompt the client for the change confirmation.</td>
</tr>
<tr>
<td>Order.Place. Display</td>
<td>The system shall display the confirmed order information. The system shall display the updated order information, whenever the client accepts/rejects the changes.</td>
</tr>
<tr>
<td>Order.Place. Confirm</td>
<td>If an order is initially accepted, the system shall prompt the client to confirm the acceptance of the order. The client can confirm and accept the order. The system shall display the order information.</td>
</tr>
</tbody>
</table>
If the client confirms the order, the system shall prompt the client to select the shipping methods.  

Shipping methods (variation point): self-pickup (valid time period, place), post (post methods and the cost rate), home delivery (home address, time, cost).

The system shall prompt the client to select the payment methods.  

Payment methods (variation point): cash (limited to self-pickup), credit card, debit card, PayPal.

The system shall display the confirmed order information.

Table 1. FRs of the feature Place an order

Taken the FR of “Create an order” in the above table as an example, its requirements model instantiated from the meta-model is presented in Figure 4. It not only specifies the semantic information in the FR: two related entities (i.e. Agentive and Objective), two conditions, a trigger and a constraint; but also explains the relationships and dependencies between them. For example, the two conditions have an “AND” logical connective to each other.

The example explicitly reflects the semantic information associated with the FR. The action occurs when the submit button is clicked. The online shopping system performs the action, and creates an order on the basis of the user’s selection of the items and their quantity. The information is specified into semantic roles. For example, the system is the agentive and the order to be created is the objective. The selection of the item and its
quantity forms the pre-conditions of the FR. The trigger slot is related to the click operation to confirm the creation of the order. The constraint specifies the happening location of this action is on catalogue list page.

In order to evaluate the performance of it in practice, the adapted semantic cases can be implemented in a RE-specific tool. The following chapters will start to discuss a promising platform to construct RE-specific application: wiki.
4 Wikis and their support in requirements engineering

4.1 An overview of wikis

The term wiki comes from a Hawaiian word “wikiwiki” which means “quick”. It was first introduced in 1995 by Ward Cunningham who was developing scripts aiming at creating, editing, linking, deleting, or renaming web pages without using HTML, software packages, or file transfer [Withers, 2005]. According to Leuf and Cunningham [2001], “A wiki is a freely expandable collection of interlinked Web ‘pages’, a hypertext system for storing and modifying information - a database, where each page is editable by any user with a forms-capable Web browser client”. Wiki combines advantages of conversational knowledge management technologies while avoiding many of their disadvantages and then become fashionable for a time since it was born [Wagner, 2004]. Compared to a blog whose content is controlled by one owner, wiki allows multiple content providers [Bean and Hott, 2005]. The design principles include simple, open, incremental, organic, mundane, universal, overt, unified, precise, tolerant, observable and convergent [Cunningham, 2006]. The Wiki philosophy is simple and follows an open model and easy edition workflow: through a standard web-browser, any reader is simultaneously a potential author and reviser [Ferreira and Silva, 2009].

Wiki is famous for its key outstanding characteristics: [Wagner, 2004; Hoenderboom and Liang, 2009; Grace, 2009]

- Page editing: Wiki applies a very simple mark-up language to edit web pages. It is famous for its openness since anyone is able to operate on it without any knowledge about programming. [Schaffert et al, 2008] With the help of some extensions, the editing process can be conducted in ‘What You See Is What You Get’ (WYSIWYG) mode, just like working under word processing system.
- Page linking: Wiki applies hyperlinks to refer to any pages inside or outside of the wiki system. A new page can be created by making a hyperlink which refers to a non-exist page.
- Access control: Wiki has a built-in authority control mechanism. Users can only operate specific set of pages with corresponding authority.
- Version control: Wiki saves all the modifications to its content and it is possible to roll backed to any history version according to the requirement.
- Data query: Wiki has built-in query function which can facilitate the process of information achievement.

One of the most outstanding benefits of wiki is the ease of use. Since wiki uses mark-ups for editing instead of HTML or other languages, it is not difficult to grasp the skills...
of usage and it will not cost much time for training. In addition, wiki is accessed through a browser; therefore the users do not need to install a specific software package on their machine. This feature extends its use place since even a mobile phone is able to operate on it. The built-in version control mechanism of wiki maintains the modifications to all the pages. The editing history not only records the changes to the content of wiki, but also marks the author who conducts the operation. It is a useful feature when a specific version of content cause problems and the reviser is required to identify. In addition, the editing history becomes significant when the content is distorted to spread malicious information. It can save the situation by immediately reverting to an old version. Another benefit is the contribution in collaborative communication. The knowledge constructing process of wiki is asynchronous and all the users can work on the specific page at the same time. Wiki can ensure the content presented to the users is updated and the feedbacks of the content can be quickly collected.

The main challenge of using wikis is the way to organizing the structure of information. The New York Times Digital tried to apply wiki at first, but then they realized that they had to abandon it since maintaining the structure and organization was a serious problem and it was almost impossible to manage [Buffa, 2006]. This phenomenon is mainly caused by two reasons. One is the feature of freedom operation. Another one is that the classification made by one might be affected by his or her subjective consciousness while other users may have conflict in their opinion. In fact, wiki is a thought more than a technique, the content in the wiki need to be carefully irrigated and cultivated, as the president of Wikipedia foundation explained at the Wikisym conference in 2005: [Buffa, 2006]

“A wiki is like a garden: users (or at least the ones with a gardener’s spirit) must take care of it. Start with some seeds and watch it grow, and the wiki will become moderated by its users’ community, respect and trust the users, leave them anonymous in order to avoid ego problems, …good things happen when you trust people more than you have reason to, let everybody express his opinion, no censorship, consensus must be reached, …the wiki is adapted to a dynamic social structure because of its refactoring features. Do not impose a rigid structure, users will refactor and structure the wiki as it grows.”

4.2 Semantic Web

In the current Web, the information is organized in well-defined structure and towards particular domain. The Semantic Web can be regarded as an extension to the current web which applies ontologies to describe the semantics of the data and aims at defining
and linking the information throughout the web [Hendler et al., 2002; Doan et al., 2003]. It is a vision of a new architecture for the World Wide Web instead of a technique [Payne and Lassila, 2004].

The concept of semantic web can be comprehended through two dimensions [Marshall and Shipman, 2003], as shown in Figure 5. The horizontal dimension describes the ability to represent information. It evolves from traditional idea which aims at publishing information for a specific application, to the idea of semantic web which focuses on adjusting information in universal purpose. The vertical dimension explains the intended object of the web. It ranges from the initial idea which only concerns the communication among humans, to the one in a semantic web which enables it to be accessible for machines.

Figure 5. The two dimensions of semantic web [Marshall and Shipman, 2003]

In order to standardize the approach to defining and exchanging information through the Web, Resource Description Framework (RDF) is introduced by W3C to depict the web resources include the title, author and copyright and so on [Passin, 2004]. It aims at enable the resources understandable for the machine instead of displaying to the users. In a traditional web, the information is stored in several web servers and users have to access the specific servers to achieve the desired information. One of the benefits of RDF is the possibility to spread information in a decentralized model [Palmer, 2001]. That is to say, the users can obtain the resources by accessing the servers which hold the reference of the information in RDF format to specific servers. It expands the scope of
information spreading and increases the utilization rate of the resources in the web. RDF extracts and represents the information as binary relationships between objects [Krötzsch et al., 2007]. It applies specific Universal Resource Identifier (URI) to represent a certain field [Shadbolt et al., 2006].

Furthermore, based on RDF, RDF Schema is invented to specify external semantics to the resources and format the description of the information [Fensel, 1997]. RDF Schema is a framework to help categorize the objects into groups and represent them into property-value pairs [Krötzsch et al., 2007]. For example, there are two objects Monkey and Animal, and it is possible to express the relationship that Monkey belongs to Animal.

However, the description is still limited even though with the help of RDF Schema. It is not able to express the composed relationships. To break the limitation and represent the information in a more specific way, the Web Ontology Language (OWL) was introduced. Ontologies are a set of meta-data schemas to define the vocabulary of concepts in the format of an explicitly defined and machine understandable semantics [Maedche and Staab, 2001]. OWL is a more powerful language which is able to deal with complicated information like inversions and conjunctions [Krötzsch et al., 2007]. For example, there are two classes Plant and Animal. OWL is able to point out that there is no conjunction between Plant and Animal. An additional example is that for the class book and its attribute author, OWL can indicate that class author is the inverse attribute of class book.

4.3 Wiki applications

There are numerous software packages based on the idea of wiki. They are written in many different languages, such as Java, JavaScript, PHP, Python and so on. One of the famous wiki engines is MediaWiki [WikiMedia Foundation, 2002].

4.3.1 MediaWiki

MediaWiki is a kind of wiki implementation which is written in PHP. It is developed by WikiMedia Foundation in 2002 to run their products include Wikipedia, Wiktionary and Wikinews [Wikipedia, 2012]. MediaWiki provides support to manage categories of the data. It organizes the inheritance and subordinate relationships of the wiki pages by defining them under the namespace Category [Hendry et al., 2005]. MediaWiki is designed to be a scalable platform and extensions can be integrated for specific purpose. Since there are too many extensions in the gallery that it becomes very difficult to pick out the desired one. Koren et al. [2008] came up with the idea to bind the extensions include Semantic MediaWiki (SMW), SF etc. together and then named them as Semantic Bundle to construct semantic environment for MediaWiki.
4.3.2 MediaWiki extensions

The SMW extension enables the content in the wiki to be annotated with explicit and machine-readable information and then publish the information into Semantic Web [Krötzsch, 2006]. It is implemented by annotating the links with specific symbol to express the meaning in order to follow the vision and high-level functionality of the Semantic Web [Schaffert, 2006; Rutledge and Oostenrijk, 2011]. The purpose of SMW is to seamless integrating semantic technologies into MediaWiki system. Since the initial idea to develop SMW is to support Wikipedia, it pays more attention to the concerns of scalability and backwards compatibility [Schaffert et al, 2008]. SMW is available in many languages and is designed to provide support for further localisation [Krötzsch, 2007].

SF is an extension for Semantic Media-Wiki that facilitates the operation of data by enabling users to add, edit and query data through forms without any prerequisite
knowledge about programming [Koren et al., 2008]. SF introduces a new namespace ‘Form’ to hold a list of Template in order to customize the pattern of a wiki page. A Template is composed by a group of properties which come from the data type of SMW to specify the rule to accept input data in the user interface [Rutledge and Oostenrijk, 2011]. SF takes advantage of form-based interface to gather the data and presents them in well-organized tables [Rutledge and Oostenrijk, 2011]. In order to avoid the duplication of attribute name, SF develops the auto-complementation function to prompt the existed values in a specific field when users input the data [Koren et al., 2008].

Semantic MediaWiki Plus (SMW+) is a well-known enterprise-oriented extension package which is distributed by the Germany Company Ontoprise. It addresses the needs of corporate environments include usability, reliability in daily operation, scalability, expressivity, interoperability and professional services [Erdmann and Hansch, 2011]. SMW+ is mainly composed by Halo extension and part of Semantic bundle such as SMW. Halo extension is based on SMW and features for wiki browsing enhancement, knowledge authoring improvement and knowledge retrieval simplification. SMW+ is designed to be good at creating, sharing, publishing and reusing knowledge in a wiki system [Wikipedia, 2012].

4.4 Applying semantic wikis to support requirements development

As discussed in the previous sections, wikis provide a flexible and open environment for asynchronous collaboration. They are helpful for organizing, tracking, and publishing the work in the geographically-distributed software projects [Ebersbach et al., 2005]. In addition, it is able to facilitate relevant knowledge achievement and make structured data more accessible for users by applying RDF for semantic annotations of wiki pages [Oren, 2006; Kousetti et al., 2008]. In this chapter we will analyse several semantic wiki-based applications that support the RE process.

The well-known wiki features such as easy page-linking, convinent access approach, detailed history recording, etc have proven that wiki platforms are good at knowledge sharing and collaborative communication. “Wikis foster an evolutionary mind-set by quickly achieving a document status that stakeholders can work with and improve as needed” [Decker et al., 2007].

Based on the discussion from Decker et al. [2007], the difference from general wikis is that RE-specific wikis cover the shortages of general wikis, such as the difficulty in exporting page content, in order to achieve more professional support for RE. The advantages of using general wiki and RE-specific wiki in RE are almost the same. These benefits, including collaboration support, pages grouping and version managing
etc. are all inherited from wiki environment by following wiki’s philosophy. However, since the philosophy is designed towards the whole domain of knowledge management, it lacks support for requirements management. In order to fulfill the needs of requirements development and management, RE-specific wikis make some improvements based on the pure wiki environment, such as tailoring the history module of wiki to support detailed version control of requirements, providing export support for requirements documentaation, defining the template and classification for achieved requirements data, and so on.

RE-specific wiki contains more professional approach for requirements developing and management than general word processors and still keep the ease of use from them, while saves more money and decreases difficulty for non-technique users to operate in contrast of other RE-specific application. Wiki platforms are suitable to be used as the basis for Web 2.0 RE-specific application since they can easily integrate different stakeholders’ perspectives and manage software development processes [Ferreira and Silva, 2009]. The charm of wiki is now attracting more and more researchers and developers in exploring the possible approaches to support RE.

4.5 Wiki tools for requirements engineering

Taking into account the benefits from the semantic annotation in the context of software development, numerous researchers have tried to develop applications for creating, maintaining and managing software artefacts, especially at the requirements stage. Based on the SMW extension, many researchers and organizations developed applications to provide a technical platform with semantic support for requirements development and management. The following is a discussion regarding to five famous semantic wiki-based platforms for RE including Software Organization Platform (SOP) [Decker et al, 2012], Moki [Rospocher et al., 2009], ReqWiki [Concordia University, 2012], SoftWiki [Softwiki Project, 2012] and WikiReq [Korolev, 2009].

SOP [Decker et al., 2012] is a semantic wiki-based extension for RE which is developed in the research project RISE (Reuse in Software Engineering). It is based on ReqMan framework [Adam et al., 2006], which is a reference model of wiki-based support throughout the whole processes in RE. SOP provides support in eliciting requirements, developing initial architecture of the software and assign development tasks based on these requirements [Decker et al., 2007]. In addition, SOP contains a Flex-based RIA (Rich User Application) front-end that aims to increase the usability and acceptance for inexperienced wiki users [Uenalan et al., 2008]. It defines a set of requirements document templates such as use case template, system requirements template, etc. and related page links to facilitate the requirements collecting and specifying process. In
Figure 7, the list of titles in the green frame is derived from a defined template. In addition, the links with regard to this template is shown in the blue frame.

Figure 7. A screenshot of SOP [Decker et al, 2007]

Moki [DKM Research Unit and the Know-Center, 2012] supports enterprise modelling through well-structured wiki pages. It presents enterprise models in an informal, semi-formal and formal format [Rospocher et al., 2009]. It focuses on the construction of domain ontology models and process models and provides a graphical interface to browse and edit the models. It develops a multi-mode access mechanism to the page content to facilitate the domain experts and knowledge engineers to use [Ghidini et al., 2010]. Figure 8 shows the interfaces of Moki. The first page is the navigation interface which represents the structure of the information. The next page shows an example model named "Write a paper". It describes the procedures to write a paper. The last page shows the editing interface of Moki.
ReqWiki [Concordia University, 2012] is a semantic system which provides support for collaborative communication in RE. It is developed by Semantic Software Lab in Concordia University. ReqWiki includes NL processing assistants (NLP) to support the creation of requirements and the derivation of traceability matrices. Taking advantage of the SF extension, it defines a set of semantic forms and templates to help users in eliciting and gathering requirements. After collecting data from the clients, it creates semantic meta-data for corresponding information and populates underlying ontology improve the tractability of the requirements [Concordia University, 2012]. Figure 9 shows the vision interface of ReqWiki. When the user clicks the page link in the navigation bar in the left part of this page, a defined template of the desired page will display on the right side. In addition, ReqWiki defines some semantic forms in order to document specified information. For example, “Create a stakeholder” and “Create a problem” are two links to respective semantic forms. Moreover, there are some semantic query sentences inserted in a template. The information of a page is able to dynamically change according to the semantic queries, as it is shown in the right-bottom corner of Figure 9.
SoftWiki [Softwiki Project, 2012] is a plug-in for a semantic data wiki named Ontowiki. It enables large and distributed stakeholders to collaboratively collect, specify, query and rearrange requirements. The Softwiki Project spent much effort in studying SoftWiki Ontology for RE (SWORE) [Riechert et al., 2007] and Softwiki exploits the interrelation defined in it to develop and manage requirements [Riechert, 2009]. In addition, it takes advantage of Semantic Web technologies to interlink and exchange the shared conceptual knowledge foundation of the requirements [Lohmann et al., 2008]. Figure 10 is a screenshot of the prototype of SoftWiki. It classified wiki pages into different groups, as it is shown in the “Topics” box in the left side of Figure 10. On the upper half part of the right side, it specified the content and author of a requirement. The bottom half part of it is the comments and reviews module, which is used to facilitate the communication of the requirements.
WikiReq exploits the collaboration feature of wiki to develop an argumentation system for both synchronous and asynchronous discussions among stakeholders and create a common knowledge base regard to the organization, the goals and the processes of a company [Abeti et al., 2009]. An outstanding feature of WikiReq is that it enables the semantic wiki-based requirements developing and managing environment to cooperate with an Integrated Development Environment (IDE) platform such as Eclipse [Eclipse Foundation, 2012a]. It takes advantage of a PHP script to manage semantic annotations in the database of wiki, and it can transform the semantically annotated requirements to an Eclipse Modelling Framework (EMF) [Eclipse Foundation, 2012b] instance which would then be able to present in graphical interface in Eclipse [Abeti et al., 2009]. Figure 11 displays the editing interface of the actor “Sales Manager” in WikiReq. It applies SF to specify the information of a requirement. As we can see from Figure 11, most information is documented in description field in NL, except the goal and role information of the actor.
### Edit Actor: Sales Manager

<table>
<thead>
<tr>
<th>Wiki Requirements</th>
<th>View</th>
<th>Discussion</th>
<th>Edit wiki</th>
<th>Log history</th>
<th>Chat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>About WikiReq</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Manual</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Help</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Warning:** You are not logged in. Your IP address will be recorded in this page's edit history.

**Actor Viewpoint**

**Infrole:**

**Goal:**

**Description:** A sales manager have to maintain accurate offers, in terms of content, price and margins. It manages all related to sales chain, for instance if a product change/upgrade can be sold to existing customers, too.

**Comment:**

**Free text:**

---

**Figure 11. A screenshot of WikiReq [Korolev, 2009]**

All of the following RE-specific wikis benefit from the navigation function of wiki platform. It contributes to organizing and classifying the information in the wiki, and then presents a clear structure of the content to the users. Based on the research from Hoenderboom and Liang [2009], a comparison regarding to the RE activities covered by these five applications is listed in Table 2. The activities to be evaluated are derived from Hoenderboom and Liang’s [2009] research, but the number of the target platforms is extended to five. In addition to SOP, WikiReq and SoftWiki which have already been compared in their research, Moki and ReqWiki are also taken into account.

The comparison contains core activities throughout different processes of RE which includes requirements elicitation, modelling, analysis, communication, agreement and evolvement. According to the comparison in Table 2, SOP is a requirements management oriented application since it almost supports all activities in requirements evolvement. Moki, as its name has already explained, is good at requirements modelling and analysis. ReqWiki, SoftWiki and WikiReq support many common activities since they all pay more attention to improve the collaborative cooperation for the stakeholders and concern more regard to exploit the resource of semantic wiki environment. There are no absolute criterions to evaluate the quality of different wiki-based application for RE. Most of the applications are still under construction and they will become more robust and integrated. The purpose is the most important factor to be taken into account when choosing the wiki-based application for requirements development and management.
<table>
<thead>
<tr>
<th>Core Activity</th>
<th>Sub-Activity</th>
<th>SOP</th>
<th>Moki Wiki</th>
<th>Req Wiki</th>
<th>Soft Wiki</th>
<th>Wiki Req</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eliciting Requirements</td>
<td>Identifying System Boundaries</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Identifying Stakeholders</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Eliciting Goals</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Applying Use Cases</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modelling and Analysing</td>
<td>Modelling Goals</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements</td>
<td>Automated Reasoning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Checking Consistency</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Validation and Verification</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communicating Requirements</td>
<td>Requirements Management</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Requirements Traceability</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Agreeing Requirements</td>
<td>Requirements Validation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Requirements Negotiation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Evolving Requirements</td>
<td>Configuration Management</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Requirements Version Management</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Fixing errors in Requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Managing Change Impact</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 2. RE activities covered by wiki-based applications
(Extended from the one in Hoenderboom and Liang’s [2009] research)

The semantic annotation in the above applications improves the process of navigating and searching semantic information on FRs. However, the support for requirements representation and documentation is still limited. Currently, the existing wiki-based RE-specific tools can only collect the whole information involved in a requirement into one text field. An approach of precisely analysing and representing the semantic knowledge
inside a single FR can be developed. The syntax elements of a FR can be divided according to their roles and functions in the FRs.

According to the meta-model proposed in Chapter 3, the semantic annotation can be further refined to specify entities, conditions and constraints on the action of a requirement. Based on this idea, a semantic wiki-based RE-specific tool named Semantic REWiki is proposed in order to fine-grained representing and documenting requirements. Concerning that the grammar cases in a NFRs are too complicated to analysis in the initial stage of the research, it focuses on analysing FRs. Semantic REWiki is based on MediaWiki. With the help of SMW extension, it is able to profit from Semantic Web. In addition, it applies SF extension to enable users to have forms for adding, editing and querying data on the wiki without any programming [Koren et al., 2008]. It takes advantage of SMW Plus extension to provide user-friendly cooperative interface.
5 Semantic REWiki

Semantic REWiki is deployed on an Apache server in the Linux system. It uses MySQL to deal with the data in the wiki. Semantic REWiki features for its defined forms. The construction of these forms is mainly based on SF extension. SMW organizes the structure of data into category, form, template and property.

The basic data unit is property. According to the characteristics of different data, a property can store them into a list of specific data types including Boolean, String, Text, Page, etc. A template can be created to hold a list of properties and set the display of the semantic data on a page. It is the concept to group the related properties for reusing in different Forms. The schema of a wiki page can be defined by applying a form. Each wiki page holds one and only one form. A form contains a (set of) template(s) to construct the content of a page and every template can be used more than once as required.

5.1 Semantic model in wiki

According to the meta-model defined in Chapter 3, four forms including Feature, FunctionalRequirement, Entity and Condition are defined to support requirements specification process, as shown in Figure 12.

Every form contains a template Description to describe the contents of the form in a free text field. The Description template in form Feature and FunctionalRequirement tends to briefly explain the information of a feature or a FR while in the form Entity and Condition it tends to further complement the information depicted in the tables. As it is shown in Figure 12, there are some links between forms and templates. The links indicate the value of a property which is stored in a Page type linked to another page in the wiki and the page name is the value of the property. The links which is drawn in dark black represent the composition relation between two forms. E.g. A feature contains a list of FunctionalRequirements. In contrast, the links drawn in light black represent self-loop relation. E.g. A feature is composed by a set of small features.
5.1.1 Feature

A Feature contains a list of FRs, as shown in the template Requirement in which the property requirement name is a Page type and links to another page which has the default form FunctionalRequirement. It can be regarded as a use case which describes a function in a coarse-grain level. When the FunctionalRequirements inside one feature are defined, the order of them can be adjusted by dragging the FunctionalRequirements up and down in the feature form in order to express the sequential dependency among the FRs. As a requirement name property can hold more than one links, it is suggested to put the FRs which are in the same level into one field and separated them by comma. Generally, the hierarchy of FRs in a feature can be well organized by these two mechanisms.

5.1.2 FunctionalRequirement

FunctionalRequirement is the most complicated form in this meta-model. A FunctionalRequirement contains a set of templates specifying the action implied by the given requirements and its related information, i.e. Action, Entity, Condition and Constraint. Since a more detailed description of the Entity, Condition and Feature is specified in the corresponding specific wiki page, there are several links in form FunctionalRequirement. The core element of a FR is action, which is represented in the property action name in the Action template. The template Entity, Condition and Constraint are duplicable as required. In addition to the page type attributes, there is an entity role property in Entity template to specify the role of the entity in current FR. The property entity attribute is used to explain the status or the nature of the entity, e.g. new, accepted, etc. In Condition template, condition name has a Page type but it can only hold one Condition in order to match a corresponding condition type to current condition. The property condition type specifies the type of a condition, as described in
the meta-model of Chapter 3. The constraints of a FR is represented in a list of Constraint templates, within which has a constraint type property to point out whether the constraint is related to location, frequency or instrument and constraint content property to describe the detail of the constraint in free text.

5.1.3 Entity
An Entity has property entity type, which has four default values: actor, system, data and device. The properties parent entity and child entity have a Page type linking to another Entity page. The link shows the composition relationship represented in the meta-mode. More than one page links can be put in a single parent entity property or child entity property and it helps to establish the relationship structure among all the entities.

5.1.4 Condition
A Condition contains a Logical Connective template and one or more State Description template. The State Description template specifies the state of the system that shall be satisfied. It consists of an entity (i.e. entity name) and its state (i.e. entity attribute) that shall be satisfied in a given FR. Normally, an adjective is used to represent the state of an entity. If a condition is a trigger and it expresses the action upon the system, it shall be interpreted as a state of an entity. This interpretation can be conducted by transforming the verb into its passive form. In addition, the Boolean property negation state indicates the unary negation connective. If its value is true, it means that the corresponding entity shall not be in the state defined in the property entity attribute. A Logical Connective template specifies the combination relationship of conditions, i.e. binary connectives such as the AND relation and the OR relation. The usage of the relations follows two rules: (1) if a condition A comprises of a binary connective, i.e. B \( \land \) C or B \( \lor \) C, then there will be two links respectively connect to condition B and C in the corresponding Logical Connective property field; (2) if a condition A appears in a compound condition, e.g. (B \( \land \) C) \( \lor \) D, then an auxiliary condition page D’ should be used to replace (B \( \land \) C). After that, apply rule (2).

5.2 Using Semantic REWiki to edit and specify FRs
As an example, the FRs of online shopping application introduced in Chapter 3 can be specified in the defined forms of Semantic REWiki. The elicited FRs can be listed in a single requirement name property in the Requirement template with commas to separate each of them. This mechanism supplements the shortage that the feature form misses the approach to distinguish the FRs which have a sequential dependency from those which are in the same stage. Figure 13 explains the hierarchical relation of the features and FRs in Semantic REWiki. As it is shown, there are several projects in the wiki and
one of them is the online shopping system. It contains a list of features and each feature is subdivided into a set of FRs, i.e. Place an order.

![Diagram of Online Shopping System and Project 2]

Figure 13. Requirements documentation in the wiki

Figure 14 shows the screenshots of FRs documented for the online shopping application. Page 1 is a form-based editing interface which enables users to document FRs in wiki pages. The duplicable template is equipped with an “Add another” button and a “Remove” button in order to customize a quantity of a certain template as required. When the user typed in the entire information and click “Save page” button, the SF extension will generate presentations of this form-entered data for each wiki page in tables, e.g. Page 3 presents the FR “Create an order” that is edited in page 1. The semantic elements of the requirement are represented in tables, with the captions shown in the left columns and the content shown in the right ones. The data in duplicated templates are represented in continuous but separated tables, i.e. entity System and entity Order in page 3. Page 2 presents the view of the feature “place an order” which contains a list of requirements such as “Create an order”. The link of requirements leads to the corresponding requirement representation wiki page. Page 4 shows the AND relation of two pre-conditions in the requirement “Create an order”, i.e. items are selected and quantity is defined. Details of the conditions are further presented in Condition pages 5 and 6, which have the default form Condition.
Figure 14. Screenshots of “Place an order” in Semantic REWiki
6 Evaluation and Discussion
This chapter evaluates the achievement of Semantic REWiki in requirements specification. Firstly, I discuss how Semantic REWiki supports the fulfilment of the goals of RE, and how it enhances the preciseness and consistency of semantic information specification on FRs. Secondly, we study how Semantic REWiki can be used in practice to specify FRs. An experiment is conducted to collect the feedback and comments.

6.1 Evaluation based on the goals of the three dimensions of RE
Taken the goals of the three dimensions of RE into account, Semantic REWiki has a good performance in the specification dimension and representation dimension. It also provides support in agreement dimension, but the degree of the support cannot be confirmed without applying it in a real project.

Semantic REWiki documents the FRs by dividing the semantic cases of each FR into defined forms so that it is able to eliminate the ambiguity which is inherited from the nature of NL. The ambiguity can be caused by different ways of combining the semantic elements. When these elements are mapped into the slots of defined semantic roles, there is only one way to explain the meaning of the statement. For example, the sentence “He gives her dog food” is ambiguous. However, taking account of the adapted semantic cases, if “her dog” is put into a dative slot and “food” is put into an objective slot, it means he gives some food to her dog; if “her” is put into a dative slot and “dog food” is put into an objective slot, it means he gives some dog food to her. From this point of view, the preciseness and the degree of understanding of the requirements can be improved. In addition, the defined semantic forms can guide the users to document specific information of the FRs and avoid omitting the details of them. In case some information is missed, other users still have a chance to supplement the forms. Combining these two aspects, the quality of the requirements develop toward the goal of the specification dimension, which is to achieve clear and complete requirements.

In the representation dimension, the table-based requirements specification has a higher level of formality than textual requirements specification. The Description field in each form is a user-oriented design. It reserves the advantage of the simplicity of NL from informal specification. In contrast to the paragraphs and chapters in structured requirements specification, the information in tables provides a systematic and consistent view for the readers. In addition, the division of semantic elements in a requirement can efficiently eliminate the language preference of different users. It omits the decoration of a statement while retaining the cores content to specify its meaning.
Moreover, the details of documented requirements are resolved and classified in specific semantic slots and they can be used for further analysis and modelling. Hence, the performance of Semantic REWiki in representation dimension is excellent. It derives the essence of informal representation and semi-formal representation, and the formality is promising to catch up with formal representation.

In the agreement dimension, Semantic REWiki enables multiple users to access and edit requirements. It can also distinguish the revisers so that the stakeholders know whose opinion it is from. In addition, the stakeholders can also write the comments regarding to a FR in the free text box of semantic forms. Additional requirements can be elicited and the conflicts can be detected based on the modifications and the comments. Hence, the combination of version control and discussing by comments is helpful for improving the common understanding of the requirements. Moreover, the browser-based accessing approach facilitates the process of communication. With more chances to discuss, it will be faster for the stakeholders to achieve the common understanding.

The initial goal of Semantic REWiki is to support requirements specification. Based on the well-documented requirements, the goal will extend to support requirements analysis and requirements management. A comparison of covered RE activities with the 5 RE-specific wikis mentioned in Chapter 5 is presented in Table 3. Currently, Semantic REWiki is able to check the consistency of FRs, manage their versions and improve the process of negotiation. It is still under construction. More support in RE activities will be developed in the future.

<table>
<thead>
<tr>
<th>Core Activity</th>
<th>Sub-Activity</th>
<th>SOP</th>
<th>Moki Wiki</th>
<th>Req Wiki</th>
<th>Soft Wiki</th>
<th>Wiki Req</th>
<th>RE Wiki</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eliciting Requirements</td>
<td>Identifying System Boundaries</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Identifying Stakeholders</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Eliciting Goals</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Applying Use Cases</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modelling and Analysing Requirements</td>
<td>Modelling Goals</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Automated Reasoning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Checking Consistency</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Table 3. RE activities covered by wiki-based applications
(Extended from the one in Hoenderboom and Liang’s [2009] research)

<table>
<thead>
<tr>
<th></th>
<th>Validation and Verification</th>
<th>Communicating Requirements</th>
<th>Agreeing Requirements</th>
<th>Evolving Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Requirements Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements Traceability</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Requirements Validation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements Negotiation</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Configuration Management</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Requirements Version Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixing errors in Requirements</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Managing Change Impact</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

6.2 Evaluation based on IEEE recommended practice

Taken IEEE recommended practice for software requirements specification into account, FRs can be consistently, unambiguously and completely specified and the documented FRs are verifiable, modifiable and traceable in Semantic REWiki. The result of evaluation is shown in Table 4.

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Supportability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct</td>
<td>-</td>
</tr>
<tr>
<td>Unambiguous</td>
<td>○</td>
</tr>
<tr>
<td>Complete</td>
<td>○</td>
</tr>
<tr>
<td>Consistent</td>
<td>○</td>
</tr>
<tr>
<td>Ranked for importance and/or stability</td>
<td>-</td>
</tr>
<tr>
<td>Verifiable</td>
<td>○</td>
</tr>
<tr>
<td>Modifiable</td>
<td>○</td>
</tr>
<tr>
<td>Traceable</td>
<td>○</td>
</tr>
</tbody>
</table>

Table 4. Supportability of Semantic REWiki in IEEE recommended practice
According to the example of ambiguity processing in section 6.1, Semantic REWiki is able to generate the FRs which have only one interpretation. The ability to eliminate the ambiguity is one of the major features of Semantic REWiki. Another attracting feature is its consistency. The relevant information can be achieved through the links on the wiki page and every wiki page is assigned a unique name. Since each FR is put into an individual wiki page, the information of the requirements can be efficiently separated and the redundancy of information can be avoided, which ensures the consistency of the data. The evaluation of completeness is challenging since it is unknown if some information is omitted. Nevertheless, the defined forms in Semantic REWiki provide enough space for complementing, for example, the description field and the free text box. Moreover, the FRs are modifiable in Semantic REWiki. Stakeholders can easily modify the content in editing mode of a wiki page. The structure of requirements will not be affected with the benefit from the forms defined by SF extension.

In current version, Semantic REWiki limits its supporting scope in specifying FRs, which decreases the complexity of the evaluation in verifiability. With the help of the history module in the wiki, the FRs in Semantic REWiki are traceable. Due to requirements management issues are not concerned in the initial version of Semantic REWiki, the defined forms does not contain the attributes to rank the importance of the requirements. In addition, the correctness cannot be confirmed without applying in a real project.

6.3 Experiment analysis

The subjects of the experiment are three students from school of computer science in University of Tampere and they are team members in a mobile game project of Demola named “Double Hunting”.

6.3.1 Experiment background

The experiment is executed in one of the subject’s house and it takes approximate 3 hours. It is executed according to the following stages:

1. Briefly describe the main idea of the project and write down at least five FRs in NL.
2. Follow the instructions provided by Semantic REWiki to document the collected requirements in the wiki system.
3. Report feedbacks on the questionnaire for Semantic REWiki. (see Appendix 2)
4. Evaluate and analysis the documented FRs and the questionnaire.

In the game “Double Hunting”, the players act as the role of wolf man to protect the animals in the forest from attacking by the hunters. The hunters always disguise as the animal to follow the animals returning to the forest and players have to recognize the
tricky hunters and prevent them from stealing the animals away. In the user interface of the game, the animals (which may mix with some hunters) come from forest (the top of the screen) to their home (the bottom of the screen). They all hide behind a piece of grass and they will not remove the grass until they almost reach their home. The players have to judge whether the “animal” should pass or not. If yes, nothing needs to be done; if no, they need to prevent them by dragging them aside. The requirements are shown in Table 5.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game.Score.Show</td>
<td>The game shall show the score of the player.</td>
</tr>
<tr>
<td>Game.Element.Control</td>
<td>The player can control the item when it reaches the bottom of the screen.</td>
</tr>
<tr>
<td>Game.Element.Create</td>
<td>The items are created at the top of the screen.</td>
</tr>
<tr>
<td>Game.Element.Exposal</td>
<td>When the item reaches the region which is located in the middle of the road, the cover on the item will be moved away.</td>
</tr>
<tr>
<td>Game.End</td>
<td>If the player let one hunter pass the game will be over.</td>
</tr>
<tr>
<td>Game.Stop</td>
<td>The game will not stop until the player fails or exits the game.</td>
</tr>
<tr>
<td>Game.Mode.Set</td>
<td>The game has easy made and hard mode.</td>
</tr>
<tr>
<td>Game.Speed.Change</td>
<td>As the game time goes by, the speed of the items will keep increasing.</td>
</tr>
</tbody>
</table>

Table 5. FRs of the game *Double Hunting*

6.3.2 Process analysis

With a brief introduction to Semantic REWiki, the subjects have a common understanding about background of the system. According to the usage background survey from the questionnaire, the subjects have experience in the usage of word processors and some collaborative tools such as Google Docs and wikis. The ease of use to operate a wiki leaves a deep impression in their minds and they all agree that wiki is a promising platform for requirements management, even though their knowledge regard to wiki is still limited in the general wiki. Nevertheless, they lack the awareness to deal with the requirements by these tools. They generally analyse the requirements in face-to-face meetings, write done the requirements on the paper and then documented the requirements in structured requirements specification. It is the first time for them to use RE-specific tool to develop and manage requirements. The shortage of background information in RE-specific tools will have some unnecessary negative opinion in the evaluation. The problems caused by this factor will then not be taken into account. Main issues elicited in the course of the experiment are listed below:
From the subjects’ point of view, the prompt information is not enough.

Some fields in defined forms are confusing, e.g. the differences between dative and object.

The meaning of some fields is considered to be vague, e.g. the “attribute” field of an entity.

The defined tables cannot match the requirements written in NL.

The documentation process of requirements is time-consuming.

According to the functionality survey from the questionnaire, the subjects agree that it is a creative feature to fine-grained unit such as entities, conditions, etc. In addition, they deem that specifying the requirements into a series defined forms and then presenting the FRs into table-based pages is a considerable formal way for requirements specification.

They can document most of their FRs into Semantic REWiki, but they also have difficulties in specifying certain FR. In fact, the difficulty is not caused by Semantic REWiki, the actual reason is that the subjects do not have a clear concept with regard to the difference between FRs and NFRs. They tried to document a NFR in Semantic REWiki. However, the support for NFR specification is not implemented. Without an obvious word to put in the action slot in semantic forms, they generated an incomplete requirement which missed the action. The same issue occurs to the choice of dative and objective. These problems can be solved by training the real users with the background information of RE and Semantic REWiki.

Since the same requirements can be represented in different ways according to the emphasis, some of the subjects find it is difficult to map the complex requirements which contain clauses or more than one verb. One potential reason for this problem is that the subjects write unclear requirements and there are more than one ways to interpret the requirements. Actually, the ability of specification of Semantic REWiki is more powerful than the understanding from the subjects. The listed FRs are not complicated enough so that some features such as AND condition, OR condition, etc. have not been experienced. Semantic REWiki is able to deal with FRs which have complicated dependency among their conditions.

The most frequently reflected issue by the subjects is that the entity attribute property in Entity template tends to cause confusion. The entity attribute property is necessary in some cases, but mostly it can be omitted. What need to be filled in this field is considered to be not exactly defined and they think it is easy to make the users get stuck in it. However, the charm of entity attribute property is that it is able to specify the
attribute of an entity in a more detailed approach. For example, if “card number” is considered to be an entity, then “card” can be put in the entity name field and “number” can be put in its attribute field. The specification implies that the action is related to the number of card instead of the card. From this point of view, the degree of accuracy is increased. Hence, the problem is not the definition of entity attribute property; it is the degree of the understanding to it. The same situation occurs to the last point in the former problem list. The subjects also complain about the efficiency to document the requirements in Semantic REWiki. They think it spends too much time to recognize the semantic cases in every FR and conduct the mapping logic. It can be solved by improving the understanding of the requirements and of the usage of Semantic REWiki.

According to the user experience survey from the questionnaire, the experience of Semantic REWiki is very poor. First of all, the prompt information is still insufficient. The subjects agree that users cannot understand the meanings of all the fields without domain knowledge in RE. Secondly, the existing help information in the form is not so clear. They might roughly guess out what they should put in some field, such as the condition name property field, but they all consider that it is a burdensome job to assign a name for a short condition. They further point out that the requirements editing process cannot be completed without a given example and the tutorial. Last but not least, they made some unexpected typos, but the auto-completion function will also remember and prompt for them. The subjects appreciate the auto-completion function and they think it would be improved by fixing this issue. As the system is developed in the initial stage, it really remains flaws in its user interface. The usability-related issues will be improved in the later mature version.

6.3.3 Feedback analysis

After the experiment, the subjects also proposed their suggestions with regard to the functionality and usability of Semantic REWiki:

- Improve the meta-model of the semantic forms.
- Develop the support for requirements management.
- Improve the user interface for users to edit and browse requirements.

According to the feedback survey from the questionnaire, the approach to document FRs is regarded as an interesting and useful idea, even though one of the subject argues that the defined from would limit his mind to specify requirements. They suggest improving the meta-model of the semantic forms since there are indeed some requirements which cannot be properly documented into the current version of Semantic REWiki. For example, the requirement “Game.Mode.Set”: the game has easy mode and hard mode. Some of them regard “game” as the adjective, “has” as the action and
“mode” as the objective. Others put “mode” in the agentive slot and put “easy/hard” in its “attribute” field. Neither of them is wrong, but none of them explicitly specified the requirements. The problem of the first approach is that it takes “has” as the action. The word “has” is a verb, but it cannot reflect the happening of an action. The second way does not even have an action, which is considered to be the core of a FR in our theory. Another suggestion is related to requirements management, e.g. change control can be taken into consideration. When a FR is changed, the related information should also be updated. An additional field can be inserted into the FunctionalRequirements form and Feature form to identify the rank of importance of the FRs. Last but not least, they proposed suggestions with regard to user experience. It is suggested to auto-organize the links of the completed wiki pages.

With the help of the experiment, the performance of Semantic REWiki is initially tested. It is really a helpful platform for RE, and the subjects provides their feedback of the experiment. The following section will have a discussion with regard to the strengths and weaknesses of Semantic REWiki.

6.4 Discussion

Semantic REWiki is a promising approach to support requirements representation and documentation. First and foremost, the defined forms facilitating the users to specify the requirements and decreases the possibility of omitting some detail of the requirements since the forms can function as a certain reminder. Secondly, when the content of a wiki page is changed, the related pages will not be affected. The organization of wiki page links increases the consistency of the FRs. Last but not least, the semantic wiki platform can be regarded as a powerful platform that improves the collaboration of the stakeholders for managing the documented FRs and increases the reusability of them.

However, the development of Semantic REWiki is still on the initial stage and there are plenty of aspects need to be improved. Firstly, the definition of meta-model of requirements is not completed. The approach to organize the relationship of the conditions and entities are not intuitive. A series of links need to be executed to comprehend the hierarchy of the entity and condition family. Moreover, the definition of constraints is very poor now. Constraint is a complex case and more attention need to be paid. In addition, the attribute property in entity template need to be refined since it is difficult for the users to understand and use. It is necessary to develop a new mechanism to deal with the expression of the nature of entities. Secondly, the requirements is simply documented in the repository of wiki platform and cannot contribute to other processes of RE. The dependency among the requirements is required to be refined and further development should be taken into consideration to take advantage of identified semantic cases information. Thirdly, the usability should be improved. For the editing
process, more detailed instructions or examples need to be provided to guide the users in documentation process. For the presenting process, the information of requirements could be reorganized to be easier to understand instead of applying the table-based form. In order to achieve this goal, a SF-based extension needs to be developed. Last but not least, a challenging improvement which needs long-term effort is to transform the form-based auto-completion mode of requirements documentation into a more dynamic and directed approach.
7 Conclusion

In this research, a SMW-based solution named Semantic REWiki is proposed for the representation and documentation of FRs. The design of defined forms in Semantic REWiki is based on the meta-model of adapted semantic cases. The contextual information can be divided into four groups: entity, action, condition and constraint. Therefore, Semantic REWiki analyses the contextual information of FRs in a fine-grained approach by categorizing the semantic cases into entity and constraints on action. These defined forms are able to precisely, consistently and completely specify the contextual information of the FRs. The table-based way to present detailed information increases the ease of understanding for the users.

My contribution in this research can be divided into two parts. On the one hand, I attend the research to adapt Fillmore’s [1968] case grammar in order to specify the contextual information of a FR. On the other hand, I focus on defining and implementing semantic forms based on the proposed meta-model in the semantic wiki platform. I compare and analysis the potential support of different wiki software packages for RE and concentrate on improving the functionality and user experience in Semantic REWiki. Currently, the initial stage of construction and deployment of this platform is completed and it can run smoothly.

According to the discussion in Chapter 6, there are still some limitations in this research. The adapted meta-model is applicable, but it is required to be improved to increase its robustness. The adapted semantic wiki platform eliminates most of the drawbacks compared to the general wikis, but specific extensions for this research needed to be developed in order to provide better support for the usage and understanding. From the long-term perspective, the specified and documented contextual information of FRs should be utilized for activities and processes of RE, including requirements modelling, requirements validation, requirements verification and requirements management. Semantic REWiki is a promising approach to provide all-round support for RE.
References


[Jing et al., 2012] Jing Ma, Wuping Yao, Zheying Zhang, Jyrki Nummenmaa, Using a semantic wiki to improve the consistency and analyzability of functional requirements. Submit to the 18th international conference on information and software technologies, 2012.


[Liaskos et al., 2006] Sotirios Liaskos, Alexei Lapouchian, Yijun Yu, Eric Yu and John Mylopoulos, On Goal-based Variability Acquisition and Analysis. In:


[DKM Research Unit and the Know-Center, 2012] DKM Research Unit and the Know-Center, MoKi: the Enterprise Modelling WiKi. Available at: https://moki.fbk.eu/website/.


## Appendix 1: Semantic information of online shopping application

<table>
<thead>
<tr>
<th>Name</th>
<th>Order.Place.Create</th>
<th>Order.Place.Modify</th>
<th>Order.Place.Display</th>
<th>Order.Place.Confirm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agentive</td>
<td>System</td>
<td>Client</td>
<td>Client</td>
<td>System</td>
</tr>
<tr>
<td>Objective</td>
<td>Order</td>
<td>Item</td>
<td>Item</td>
<td>Order</td>
</tr>
<tr>
<td>Dative</td>
<td>-</td>
<td>-</td>
<td>Item</td>
<td>Client</td>
</tr>
<tr>
<td>Action</td>
<td>Create</td>
<td>Add</td>
<td>Remove</td>
<td>Client</td>
</tr>
<tr>
<td>Pre-condition</td>
<td>Items are selected</td>
<td>Order is initially accepted</td>
<td>Order is initially accepted</td>
<td>Order is initially updated</td>
</tr>
<tr>
<td>Post-condition</td>
<td>Order is initially accepted</td>
<td>Order is initially modified</td>
<td>Order is initially modified</td>
<td>Order is updated</td>
</tr>
<tr>
<td>Trigger</td>
<td>Customer click 'Submit' button</td>
<td>Customer click 'Modify' button</td>
<td>Customer click 'Modify' button</td>
<td>Customer click 'Confirm' button</td>
</tr>
<tr>
<td>Locative</td>
<td>Product catalogue list page</td>
<td>Shopping cart page</td>
<td>Shopping cart page</td>
<td>Shopping cart page</td>
</tr>
<tr>
<td>Instrument</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Frequency</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Appendix 2: Questionnaire for Semantic REWiki

Usage background survey
1. Have you ever used any requirements-related software to develop and/or manage requirements? If so, please explain the use experience of it; if not, explain why?
2. Have you ever tried wikis? How much do you know about wikis?

Functionality survey
1. How do you think about the collaborative co-operation of Semantic REWiki?
2. Can you understand every field in the form? Which field is hard to understand? Which field is hard to specify?
3. Is it able to completely specify the functional requirements with Semantic REWiki?
4. Do you think the table-based representation is a formal enough to specify requirements?
5. Do you think the description field is useful? Why and why not?
6. Are there any functional requirements you cannot precisely specified in Semantic REWiki? If yes, please explain it.
7. How do you think about the efficiency to specify requirements with Semantic REWiki? Please explain your answer.

User experience survey
1. Is the prompt information clear and comprehensive to understand? Is there any missing information for help?
2. How difficult is it for you to learn to use Semantic REWiki?
3. How is the ease of use when specifying the requirements?

Feedback survey
1. What are the difficulties when you developing functional requirements?
2. Do you like the idea to document a FR by specifying its semantic information in a form? Why and Why not?
3. Do you have any suggestions to improve Semantic REWiki?