2009 Fourth IEEE International Conference on Global Software Engineering

Requirements Reasoning for Distributed Requirements Analysis using Semantic Wiki

Peng Liang and Paris Avgeriou
Department of Mathematics
and Computing Science
University of Groningen, The Netherlands
{liangp,paris}@cs.rug.nl

Viktor Clerc
Department of Computer Science
VU University Amsterdam
The Netherlands
viktor@cs.vu.nl

Abstract

In large-scale collaborative software projects, thousands of requirements with complex interdependencies and different granularity spreading in different levels are elicited, documented, and evolved during the project lifecycle. Non-technical stakeholders involved in requirements engineering activities rarely apply formal techniques; therefore it is infeasible to automatically detect problems in requirements. This situation becomes even worse in a distributed context when all sites are responsible to maintain their own requirements list using various requirements models and management tools, and the detection of requirements problems across multiple sites is error-prone, and unaffordable if performed manually. This paper proposes an integrated approach of basing distributed requirements analysis on semantic wiki by requirements reasoning. First, the functions concerning reasoning support provided by semantic wiki for requirements analysis are proposed. Second, the underlying requirements rationale model for requirements reasoning is presented with sample reasoning rules. Third, our rationale model is mapped to the WinWin requirements negotiation model which further adds to its credibility.

1 Introduction

Markets globalization has dramatically impacted software development. More projects are run in geographically distributed environments, and Global Software Development (GSD) is becoming a norm in the software industry. This trend makes a great impact in the Requirements Engineering (RE) and the RE practice has been a key challenge in GSD [9]. In collocated software development, synchronous (e.g. face-to-face) communication is the most important way for requirements elicitation, while in a distributed context, to overcome the time zone difference and distance barrier, asynchronous (e.g. text-based) communication is most frequently employed. Experimental research indicates that requirement communication (e.g. negotiations) is more effective when stakeholders conduct asynchronous discussions prior to the synchronous communication in a distributed RE context [5].

Wiki, as a lightweight documentation and distributed collaboration platform, has demonstrated its capability in distributed requirements elicitation [7] and documentation [27]. Wiki mainly addresses two challenges in distributed RE: end users' participation and collaboration, by providing well-suited functions for requirements documentation, and communication with versioning support. One of the challenges in the RE field is the integration of RE activities [4], which logically demands the integration of RE tools (functionalities) in an integrated platform supporting the whole RE process. Wiki, as a distributed requirements documentation platform, is insufficient to perform automatic requirements analysis (*understand the requirements, detect their overlaps and conflicts* [28]) without decent semantic support. The other drawback of wikis is that they document requirements in free-text or templates (e.g. use case description) which cannot ensure the correct understanding of requirements due to the diversity of stakeholders' background (e.g. understanding the use cases) and interests, and the cultural and cognitive differences in a distributed context. The poorly understood requirements result in time-consuming debates, unwanted work, and finally extended lead times and costs.

Semantic wiki, as a semantic extension on plain wiki, fairly addresses above issues by providing semantic support (e.g. semantic annotation and query) [25] and has been employed in software engineering activities, such as architecture design [10], software reuse [26] and RE activities as well. Most of existing semantic wikis focus on the requirements formalization[1] in order to promote the

[1]The "formalization" of requirements in this paper refer to the explicit

978-0-7695-3710-8/09 $25.00 © 2009 IEEE
DOI 10.1109/ICGSE.2009.61

IEEE computer society

Authorized licensed use limited to: University of Groningen. Downloaded on February 8, 2010 at 14:53 from IEEE Xplore. Restrictions apply.

semantic-enabled requirements understanding and communication among stakeholders. Little work has been done on the reasoning support by the formal semantics which is fundamental for the automatic distributed requirements analysis, since requirements analysis by human effort is error-prone and in some cases unaffordable [8]. In this position paper, we propose the envisioned support (use cases) by reasoning in semantic wiki for requirements analysis with an underlying Requirements Rationale Model. We anticipate that this work can contribute to the reasoning applications of semantic wiki in the RE field.

The rest of this paper is organized as follows. In section 2, related work on using semantic wiki in RE, the reasoning support and rationale models in RE is reviewed and discussed. The problem to be addressed in this paper is further discussed in section 3. The use cases concerning reasoning support for requirements analysis are proposed in section 4, and the concept model of requirements rationale that support requirements reasoning are presented in section 5. In section 6, a proof of concept is provided to demonstrate the applicability of the proposed requirements rationale model. The paper concludes with next steps in section 7.

2 Related Work

Many practitioners and researchers already found that semantic wiki technology can be beneficial in distributed RE, and methods and tools have been proposed and developed with various focus. The SoftWiki project aims to provide an agile methodology for requirements elicitation and management in distributed software development [18]. SoftWiki focuses on semantic annotation and sharing of requirements artifacts using the underlying conceptual model - SoftWiki Ontology. The shortage of SoftWiki is that the underlying model cannot be changed, and it does not provide requirements reasoning support. The RISE (Reuse in Software Engineering) project develops the SOP-Wiki (Software Organization Platform) to manage requirements elicitation and documentation by distributed stakeholders [7]. SOP-Wiki employs the document ontology based on a use case approach, applies the semantic annotation to wiki pages (e.g. annotate a wiki page as a *User Story* or *Actor*), and defines typed links between wiki pages (e.g. *Actor* is `PartOf` *User Story*). SOP-Wiki offers a case-based reasoning support for the retrieval of similar documents (requirements), but it does not allow users to add semantics to requirement artifacts within a wiki page, which makes it incapable to perform automatic requirements analysis. López *et al.* propose the NDR (Non-functional requirements and Design Rationale) ontology to assist the understanding, and facilitate the sharing and reusing of NDR knowledge across organizations [19]. Their work mainly targets the conceptual support to knowledge management without much attention to the reasoning support by formal ontology.

Requirements are essentially the knowledge collected from all the stakeholders. Reasoning over formalized requirements (knowledge) to support requirements analysis is not new in formal RE methods. There has been a long history of using formal representation to perform automatic reasoning for requirements analysis [32][16]. The major problem is that formal methods in RE are definitely helpful but rarely employed in practice due to the added cost and learning curve for non-technical stakeholders who are the majority in the RE process. Semantic wiki, as a lightweight knowledge management tool and methodology, may partially address this problem by providing transparent reasoning support.

Design rationale was originally proposed in the context of software design as means of presenting the "why" of a design. Rationale in RE process, which is also a design process in a broad sense [20], plays an important role in RE as well [3]. In RE context, the rationale is about "why" a particular requirement is selected out of the others or prioritized. Most of rationale models employed in RE process are based on or original from IBIS (Issue-Based Information System) [14]. For example, Ramesh *et al.* proposed REMAP model, which includes the IBIS model, to record the deliberations in RE process [23]. Rooksby *et al.* proposed a hybrid approach to the upstream requirements negotiation by combining IBIS and cognitive mapping [24]. Thurimella *et al.* abstracted the rationale concepts (*issue* and *option*) from IBIS, DRL [15] and QOC [21], and proposed the issue-based variability modeling for the selection of candidate requirements based on tradeoffs in software product line [29]. The shortage of existing rationale models (e.g. IBIS, DRL, QOC) is that they primarily focus on the representation and documentation of deliberation and argumentation process for rationale understanding and decision-making without automatic reasoning support.

3 Problem Statement

According to the analysis of related work on semantic wiki for RE activities, we focus on the reasoning support which is not fairly addressed by existing semantic wikis. The problem can be detailed as following:

Envisioned **use cases** of requirements analysis system: What tasks of requirements analysis can be supported by requirements reasoning? What functions should be provided to support the reasoning itself, such as producing the formalized requirements?

Underlying **conceptual model** for requirements reasoning: What concepts should be included in the conceptual model to support the reasoning functions, such as in [29].

requirements representation codified using a conceptual model or ontology of which the exact semantics are defined, e.g. in [30].

Authorized licensed use limited to: University of Groningen. Downloaded on February 8, 2010 at 14:53 from IEEE Xplore. Restrictions apply.

The model should be extensible for accommodating new concepts that may arise and change in distributed RE.

Shared **understanding** on requirements reasoning. Due to the diversity caused by distribution, distributed teams and stakeholders may have different understanding (mental models) about the requirements reasoning, so how to reconcile their conflicting views?

Implementation of specific features on semantic wiki for requirements reasoning. Although wikis are well suited for collaborative tasks, and semantic wikis provide various semantic support in certain degree. They are not originally built to analyze and reason about requirements. How to implement the reasoning support based on the existing (semantic) wikis in order to save development cost and facilitate the use of existing wikis.

Cost and benefit of using formal representation for requirements reasoning. Formal representation of requirements is beneficial and also a prerequisite for automatic requirements analysis. Producing formalized requirements in semantic wikis is not a trivial task, which contributes added cost and has unclear perceived benefits [11]. It is a key issue for practitioners to predict the cost and benefit before applying this method.

In this paper, we target the first two issues (use cases model and conceptual model). The remaining issues, which are also important, will be further investigated as next steps.

4 The Use Case Model

Reasoning is the process of deriving conclusions from formal representations and knowledge, which is in RE context constituted by the formalized requirements of a semantic wiki. The "conclusions" of reasoning can be of many different types according to different reasoning context and purposes. Krötzsch *et al.* identified four functions, with which reasoning support could provide actual benefit to wiki users [13]. The use cases (UCs) shown in Figure 1 comprise four refined functions for and of reasoning in semantic wikis [13] (UC1˜UC4) for general RE activities, and the specific use cases supported by reasoning for requirements analysis (UC4a˜UC4c). For example, UC3 (Query Requirements) is a refined function of Querying the Knowledge in [13]. The use cases descriptions are presented below, and the term "requirements specifications" in the use case description refers to the composition of the Requirements Knowledge Entity (RKE, discussed in section 5) formalized and stored in a semantic wiki.

UC1 Annotate Requirements: users can select a piece of text in a wiki page, and annotate the selected text with a concept (e.g. *Motivator*). The annotated text is called a RKE, which can be subject to semantically querying and reasoning.

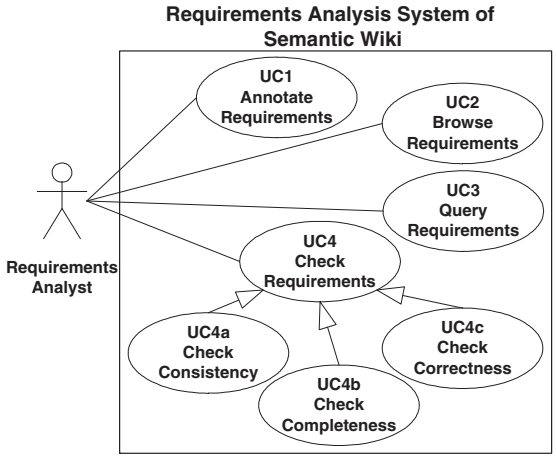

Figure 1. The Use Cases for Requirements Analysis concerning Reasoning.

UC2 Browse Requirements: users can browse the annotated requirements specifications in a semantically sound way, showing related requirements, tradeoffs and stakeholders, and filtering and grouping the requirements. For example, requirements for radio telescope data processing system has thousands of requirements with different relationships to other requirements, and stakeholders and motivations. Filtering, grouping and ordering requirements is needed to display such semantic information (e.g. some requirements is `causedBy` other requirements).

UC3 Query Requirements: users can query the RKEs using semantic query language (e.g. SPARQL [22]) by asking complicated queries, for example, query all the *Candidate Requirements* `causedBy` a *Motivator* and without any `conflictWith` a *Obstacle* (see Requirements Rationale Model in section 5).

UC4 Check Requirements: users can check the problems in requirements by reasoning over the annotated RKEs using the constraints defined in Requirements Rationale Model (i.e. if the requirements specification does indeed adhere to the constraints). This use case can be further detailed as three use cases (UC4a˜UC4c).

UC4a Check Consistency: users can check the inconsistencies in the requirements specifications by reasoning over the annotated RKEs. In RE context, inconsistent requirements are those requirements which conflict with other requirements. An inconsistent requirement can be changed, removed or tolerated for later consideration. For example, the requirement R1 *"the system should have multiple levels of security check."* and R2 *"the system should have easy access."* are conflicting requirements with each other.

UC4b Check Completeness: users can check the incompleteness in the requirements specifications by reason-

Authorized licensed use limited to: University of Groningen. Downloaded on February 8, 2010 at 14:53 from IEEE Xplore. Restrictions apply.

ing over the annotated RKEs. In RE context, incomplete requirements are those requirements whose related and indispensable elements are not addressed. An incomplete requirement should be improved by adding related elements. For example, if a requirement R1 does not have any stakeholders who propose it, then this requirement is incomplete.

UC4c Check Correctness: users can check the incorrectness in the requirements specifications by reasoning over the annotated RKEs. The correctness is normally defined by and in line with certain formal semantics, e.g. relationships in a conceptual model. In RE context, incorrect requirements are those requirements which violate domain assumptions. An incorrect requirement should be changed or removed. For example, if there is a domain assumption of the system that *"the users are accustomed to work with command interface."*, then any candidate requirements about graphical user interface are incorrect requirements.

The sample reasoning rules for consistency, completeness, and correctness checking (UC4a˜UC4c) are presented in section 5 based on the Requirements Rationale Model.

5 Requirements Rationale Model

The general objective of requirements analysis is to understand the requirements, and detect their overlaps and conflicts [28]. Requirements analysis can be regarded as a high level design activity in problem space, which is comparable to the design activity in solution space (e.g. architecture design), performed by a requirements analyst, who needs to "design" the requirements by making tradeoff and compromise. A prerequisite of this activity is to understand the rationale underneath the requirements (i.e., why particular requirement is selected out of the others or prioritized, what is the business and technical motivations for achieving them). The rationale contains the arguments for and against each alternative requirement, including the functional (FR) and non-functional requirement (NFR) [3]. When this information is captured in the rationale for a requirement, it provides reasoning facilities for detecting the conflicts between requirements (between two FRs, two NFRs or FR and NFR) on the software system.

The Requirements Rationale Model (RRM) is heavily based on our previous work in Griffin project [1] on the rationale model for architecture design [30], which is similar to DRL [15] with added reasoning semantics. The initial result of RRM, i.e. the derived concepts and their relationships are presented in UML as shown in Figure 2, which is comprised of the following concepts:

Stakeholder: anyone who has direct or indirect interest to the system. *Stakeholders*, who can `propose` or `objectTo` any *Candidate Requirement*, are the original source of requirements.

Candidate Requirement: is any requirement proposed

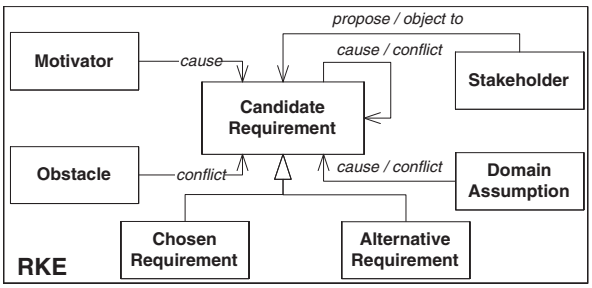

Figure 2. Requirements Rationale Model for Requirements Reasoning.

by *Stakeholder*, and it is also a generalized requirement of *Chosen Requirement* and *Alternative Requirement*. It relates with other requirements rationale elements and itself by `conflictWith` and `causedBy` relationships.

Chosen Requirement: for a *Motivator* or *Obstacle*, there are sometimes multiple *Alternative Requirements* are suitable, but only one of them is chosen (or some of them are prioritized) to address the described motivator or obstacle. The *Chosen Requirement* is the requirement being selected or prioritized. It is a subclass of *Candidate Requirement*.

Alternative Requirement: to address the *Motivator* or *Obstacle*, besides the *Chosen Requirement*, one or more potential *Alternative Requirements* can partially address the motivator or obstacle. These *Alternative Requirements* will be reconsidered when the system context changes. It is also a subclass of *Candidate Requirement*.

Motivator: is an incentive to a *Candidate Requirement*, and it has positive impact for a requirement to be selected or prioritized. Motivation of *Chosen Requirement* is `subClass` of *Motivator*.

Obstacle: is a disincentive to a *Candidate Requirement*, and it has negative impact for a requirement to be selected or prioritized.

Domain Assumption: is a kind of requirement statement in the "indicative" mood describing the environment as it is in the absence of the machine or regardless of the actions of the machine; these statements are also called *Domain Knowledge*.

RKE: since all annotated text is a kind of *RKE* produced by UC1, the above concepts are all subclasses of the *RKE* concept.

It is worthy to discuss the simplicity of this rationale model which excludes the rationale concept (the reason why a requirement is selected or prioritized) deliberately, e.g. *TradeOff*, *Argument* or *Rationale*. Our argument is that the objective of this model is to support the automatic reasoning for requirements checking. The rationale concept

Authorized licensed use limited to: University of Groningen. Downloaded on February 8, 2010 at 14:53 from IEEE Xplore. Restrictions apply.

Table 1. Sample Reasoning Rules for the Implementation of Use Cases based on RRM

Rule	Description	UC
RR1	If a *Candidate Requirement* R1 is `causedBy` a *Motivator* T, which is also an *Obstacle* `conflictWith` *Candidate Requirement* R2, then R1 and R2 `conflictWith` each other.	UC4a
RR2	Two *Chosen Requirements* can not `conflictWith` each other.	UC4a
RR3	Deducted from RR1 and the `subClass` relationship between *Chosen Requirements* and *Candidate Requirements*. If the *Motivator* and *Obstacle* of two *Chosen Requirements* are the `same` *RKE*, then these two *Chosen Requirements* `conflictWith` each other.	UC4a
RR4	Each *Chosen Requirement* should be `causedBy` at least one *Motivator*.	UC4b
RR5	Each *Candidate Requirement* should be `proposedBy` at least one *Stakeholder*.	UC4b
RR6	Any *Candidate Requirement* can not `conflictWith` any *Domain Assumptions*.	UC4c
RR7	Deducted from RR6 and the `subClass` relationship between *Chosen Requirements* and *Candidate Requirements*, any *Chosen Requirement* can not `conflictWith` any *Domain Assumptions*.	UC4c

is well suited to record the reason description for human understanding, but not suitable for machine processing. It is also a wise solution to extend the RRM model with rationale concepts for documenting the rationale knowledge, and provide reliable information for requirements analyst to change and evolve the system requirements.

Sample reasoning rules based on the RRM model are presented in Table 1 described in natural language for easy understanding. All these rules can be formally expressed by description logic, which is the logic foundation of OWL [6]. Both of them provide the formal representation mechanism underlying most of semantic wikis [25]. As mentioned above, the RRM is not fixed, but can be extended and evolved according to concrete applications, which is natural in a distributed RE context when distributed teams or stakeholders employ various requirements models. For example, *Goal* concept in [31] can be extended as a `subClass` of *Motivator* concept, and *Risk* concept as a `subClass` of *Obstacle* concept. For the tool supporting the changeability and evolvability of RRM model, we have implemented a tool suite for architectural knowledge management using different underlying models [17]. This provides the technical foundation for the tool implementation in RE context.

6. From RRM to WinWin Model

To demonstrate the applicability of RRM model, we map this model into the WinWin requirements negotiation model, which is used for capturing requirements rationale knowledge during requirements negotiation, and has been successfully used in more than 100 real-world projects in various domains [2]. The WinWin negotiation model has four main conceptual artifacts, and the detailed mapping is described and discussed below:

Win Condition: *capturing the desired objectives and constraints of the stakeholder* . This concept can be directly mapped to the *Candidate Requirement* and its related *Motivator* (objectives) and *Obstacles* (constraints).

Issue: *capturing the conflict between win conditions and their associated risks.* This concept can be represented by the conflicting *Candidate Requirements* (Win Conditions) and the `conflictWith` relationship between them. The associated risks can be represented as a `subClass` of *Obstacle* associated with *Candidate Requirements*.

Option: *capturing a decision choice for resolving an issue.* This concept represents the selected requirement which satisfies two conflicting Win Conditions, and can be perfectly mapped to the *Chosen Requirement*.

Agreement: *capturing the agreed upon set of win conditions which satisfy stakeholder win conditions and/or capturing the agreed options for resolving issues.* This is essentially a set of *Chosen Requirements* which constitute the agreement for WinWin conditions and agreed requirements for the design phase.

7. Conclusions and Next Steps

In this paper, we present our initial ideas and propositions about using semantic wiki for automatic distributed requirements analysis by requirements reasoning. Based on the analysis of existing work of using semantic wiki in RE field, we noticed that the reasoning support has not been fully explored. The major contributions of this paper are the following: (1) a use case model about what basic functionality semantic wikis should provide for requirements analysis in a RE perspective; and (2) an initial rationale model (RRM) to support the requirements reasoning.

We outline our next steps and research agenda in the following points: (1) implement the proposed semantic wiki for requirements reasoning based on the features survey of existing semantic wikis and wikis for RE activities; (2) perform the verification & validation of the semantic wiki approach for requirements analysis in controlled experiments (especially compared with the traditional RE tools, e.g. DOORS); (3) perform tradeoff analysis between cost and benefit of the proposed approach to understand its applicability and efficiency as a lightweight approach (e.g. in extreme programming); (4) extend and mature the RRM model by introducing concepts that may arise in a distributed RE context (e.g. cognitive and cultural concepts, speech act modality in communication, the partial satisfaction of requirements, and context of motivation etc.); (5) promote the requirements rationale understanding among different organizations who employ diverse mental models (e.g. [12]) by knowledge translation (e.g. conceptual model mapping); (6) how to transform the tacit (personalized) RK into formalized RK for reasoning is a challenging issue.

Authorized licensed use limited to: University of Groningen. Downloaded on February 8, 2010 at 14:53 from IEEE Xplore. Restrictions apply.

References

[1] GRIFFIN: a GRId For inFormatIoN about architectural knowledge. http://griffin.cs.vu.nl/.

[2] B. Boehm and H. Kitapci. The WinWin Approach: Using a Requirements Negotiation Tool for Rationale Capture and Use. *Rationale Management in Software Engineering*, pages 173–190, 2006.

[3] J. Burge, J. Carroll, R. McCall, and I. Mistrik. Rationale and Requirements Engineering. *Rationale-Based Software Engineering*, pages 139–153, 2008.

[4] B. Cheng and J. Atlee. Research Directions in Requirements Engineering. In *Proceedings of the 29th International Conference on Software Engineering (ICSE)*, pages 285–303, 2007.

[5] D. Damian, F. Lanubile, and T. Mallardo. On the Need for Mixed Media in Distributed Requirements Negotiations. *IEEE Transactions on Software Engineering*, 34(1):116–132, 2008.

[6] M. Dean, G. Schreiber, S. Bechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, et al. OWL Web Ontology Language Reference. *W3C recommendation*, 10, 2004.

[7] B. Decker, E. Ras, J. Rech, P. Jaubert, and M. Rieth. Wiki-Based Stakeholder Participation in Requirements Engineering. *IEEE Software*, 24(2):28–35, 2007.

[8] S. Easterbrook and B. Nuseibeh. Managing Inconsistencies in an Evolving Specification. In *Proceedings of the 2nd IEEE Symposium on Requirements Engineering (RE)*, pages 48–55, 1995.

[9] T. Gorschek, S. Fricker, R. Felt, C. Wohlin, and M. Mattsson. 1st International Global Requirements Engineering Workshop - GREW'07. *ACM SIGSOFT Software Engineering Notes*, 33(2):29–32, 2008.

[10] H. Happel and S. Seedorf. Ontobrowse: A Semantic Wiki for Sharing Knowledge about Software Architectures. In *Proceedings of the 19th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pages 506–512, 2007.

[11] M. Heindl and S. Biffl. Risk Management with Enhanced Tracing of Requirements Rationale in Highly Distributed Projects. In *Proceedings of the 1st International Workshop on Global Software Development for the Practitioner (GSD)*, pages 20–26. ACM New York, NY, USA, 2006.

[12] I. Jureta, J. Mylopoulos, and S. Faulkner. Revisiting the Core Ontology and Problem in Requirements Engineering. In *Proceedings of the 16th IEEE International Requirements Engineering Conference (RE)*, pages 71–80, 2008.

[13] M. Krötzsch, S. Schaffert, and D. Vrandečić. Reasoning in Semantic Wikis. In *Proceedings of the 3rd Reasoning Web Summer School*, pages 310–329, 2007.

[14] W. Kunz and H. Rittel. *Issues as Elements of Information Systems*. Institute of Urban and Regional Development, University of California, 1970.

[15] J. Lee and K. Lai. What's in Design Rationale? *Human-Computer Interaction*, 6(3):251–280, 1991.

[16] E. Letier and A. van Lamsweerde. Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering. In *Proceedings of the 12th International Symposium on Foundations of Software Engineering (FSE)*, pages 53–62, 2004.

[17] P. Liang, A. Jansen, and P. Avgeriou. Knowledge Architect: A Tool Suite for Capturing and Managing Software Architecture Knowledge. Technical Report RUG-SEARCH-09-L01, University of Groningen, 2009, http://www.cs.rug.nl/~liangp/download/liang2009kat.pdf.

[18] S. Lohmann, T. Riechert, and S. Auer. Collaborative Development of Knowledge Bases in Distributed Requirements Elicitation. In *Proceedings of the Software Engineering Workshops (SE)*, pages 22–28, 2008.

[19] C. López, L. M. Cysneiros, and H. Astudillo. NDR Ontology: Sharing and Reusing NFR and Design Rationale Knowledge. In *Proceedings of the 1st International Workshop on Managing Requirements Knowledge (MaRK)*, pages 1–10, 2008.

[20] K. J. Lyytinen, P. Loucopoulos, J. Mylopoulos, and W. R. (Eds.). *Design Requirements Engineering: A Ten-Year Perspective*. Springer, 2009.

[21] A. MacLean, R. Young, V. Bellotti, and T. Moran. Questions, Options, and Criteria: Elements of Design Space Analysis. *Human-Computer Interaction*, 6(3):201–250, 1991.

[22] E. Prudhommeaux and A. Seaborne. SPARQL Query Language for RDF. *W3C Working Draft*, 20, 2006.

[23] B. Ramesh and V. Dhar. Supporting Systems Development by Capturing Deliberations During Requirements Engineering. *IEEE Transactions on Software Engineering*, 18(6):498–510, 1992.

[24] J. Rooksby, I. Sommerville, and M. Pidd. A Hybrid Approach to Upstream Requirements: IBIS and Cognitive Mapping. *Rationale Management in Software Engineering*, pages 137–154, 2006.

[25] S. Schaffert, F. Bry, J. Baumeister, and M. Kiesel. Semantic Wikis. *IEEE Software*, 25(4):8–11, 2008.

[26] S. Shiva and L. Shala. Using Semantic Wikis to Support Software Reuse. *Journal of Software*, 3(4):1–8, 2008.

[27] C. Silveira, J. Faria, A. Aguiar, and R. Vidal. Wiki Based Requirements Documentation of Generic Software Products. In *Proceedings of the 10th Australian Workshop on Requirements Engineering (AWRE)*, pages 42–51, 2005.

[28] I. Sommerville. Integrated Requirements Engineering: a Tutorial. *IEEE Software*, 22(1):16–23, 2005.

[29] A. Thurimella, B. Bruegge, and O. Creighton. Identifying and Exploiting the Similarities between Rationale Management and Variability Management. In *Proceedings of the 12th International Software Product Line Conference (SPLC)*, pages 99–108, 2008.

[30] J. van der Ven, A. Jansen, J. Nijhuis, and J. Bosch. Design Decisions: The Bridge between Rationale and Architecture. In *Rationale Management in Software Engineering, A.H. Dutoit, et al., (Eds.)*, pages 329–346. Springer, 2006.

[31] A. van Lamsweerde. Goal-Oriented Requirements Engineering: A Guided Tour. In *Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE)*, pages 249–262, 2001.

[32] E. Yu. Towards Modelling and Reasoning Support for Early-phase Requirements Engineering. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE)*, pages 226–235, 1997.

Authorized licensed use limited to: University of Groningen. Downloaded on February 8, 2010 at 14:53 from IEEE Xplore. Restrictions apply.