Wiki-based Tool for Requirements Engineering according to the ProjectIT Approach

David de Almeida Ferreira, and Alberto Rodrigues da Silva
INESC-ID / Instituto Superior Técnico
Rua Alves Redol 9, 1000-029 Lisbon, Portugal
david.ferreira@inesc-id.pt, alberto.silva@acm.org

Abstract

Nowadays, the software development process still presents serious flaws, which results in large number of projects completed with failure, or even abandoned. Studies on software's quality emphasize that many of these aspects could be eliminated, or at least mitigated, if given more attention to the early stages of the process, such as those covered by Requirements Engineering. The successive increase of the size and complexity of modern information systems require new socio-technical approaches to prevent errors in the upstream phases of the software's life cycle, thus minimizing the cost and effort impact to correct them.

In this paper we present a Requirements Engineering MDE-based approach to address some of these issues. We also introduce a CSCW Wiki-based platform that we are developing, that facilitates communication, cooperation, and negotiation of the stakeholders during the process of requirements development. Additionally, we discuss that it is possible to combine these benefits with Natural Language Processing techniques, already provided by ProjectIT's CASE tools, to leverage the quality and rigor of requirements engineering workflows.

1. Introduction

Requirements Engineering (RE) is crucial for achieving success on IT projects. The RE process consists of a structured approach to the activities of capturing, organizing, documenting, and maintaining the requirements of a target system. The use of techniques and models throughout the RE process supports the systematic execution of these activities. Wiegers [1] argues that the process of engineering requirements should be separated into two workflows: (1) *development of requirements*, which is devoted to introducing new requirements during the early stages of the software development process; and (2) *requirements management*, intended to monitor changes to all requirements already established and to assess the relevance of its inclusion into the target system. Regarding the requirements development process, the early activities with which it deals are of paramount importance for both project management and the software development process [2]. The target system's requirements are the starting point to define the scope of the problem to be solved. Mistakes made near the beginning of a software system development, such as inadequate, inconsistent, incomplete, or ambiguous requirements, outcome in quality problems and high costs to correct them [3]. Therefore, requirements specifications influence all the subsequent activities and define the basis for assuring software's quality.

Historically, the requirements specification process has consisted of creating a natural language description of what the target system should do or constraints about its behavior [4]. However, this form of specification is both ambiguous and, in many cases, unverifiable because of the lack of a standard machine-executable representation [4]. Apparently, the usage of formal methods could overcome these problems. But it isn't trivial as it seems, because we still need to take care when interpreting the natural language requirements to create a formal specification. Aside of formal methods, textual specifications are still the most suitable, fast, and preferred manner (by non-technical stakeholders) to initiate the requirements specifications of the target system. This is mainly due to natural language's expressive power and because non-technical users are familiar with it in their daily interactions. To address this problem we created an extensible requirements specification language based on linguistic patterns [5] and a CASE tool that supports non-technical stakeholders during the specification activity. However, despite of the results achieved with the desktop-based approach, we realize the importance of social, organizational, and collaborative issues during requirements elicitation, especially when dealing with virtual and distributed teams. To address this high level of uncertainty it is required an evolutionary discovery of requirements [6].

Wikis provide a feasible solution to address the aforementioned problems. They are lightweight authoring systems that support interactive and collaborative work through an online environment. The Wiki philosophy is simple [7] and follows an open model and easy edition workflow: through a standard web-browser, any reader is simultaneously a potential author and reviser. Throughout the process the users not only become aware of what their peers know, but they also benefit from an implicit self-assessment effect. Since Wikis support work group processes, they are used in several contexts to record and refine information, namely they are excellent platforms for supporting project management activities [8]. Wikis are suitable to become the

underlying technology for a Web 2.0 RE tool because they can easily integrate different stakeholders' perspectives and support software development processes.

The remainder of this paper is structured as follows. In the next section we present an overview of the ProjectIT initiative. Thereafter, in the third section, we describe how the prototypical tool that we are developing provides support for the ProjectIT approach. The fourth section is devoted to the discussion of related work. Finally, the fifth section concludes the paper highlighting its main contributions and providing some directions for future work.

2. ProjectIT Overview

According to our point of view, the emphasis in the software development process should be on the project management, requirements engineering, and design activities. Hence the effort in production activities, like software programming and testing, should be minimized and performed as automatically as possible. To substantiate this vision, the Information Systems Group of INESC-ID started, some years ago, a research initiative that we called ProjectIT [9]. Regarding to the RE and the MDE fields of research, the ProjectIT initiative's main goals are to enhance productivity and rigor of Software Engineering activities. Therefore, we designed the ProjectIT approach [10], which synthesizes the process used to develop software with this initiative. This approach is supported by a set of modeling languages of which, within the context of this paper, we emphasize the ProjectIT-RSL. This language consists of a controlled natural language, based on linguistic patterns, for capturing textual requirements [5]. To assess our research ideas, we began the development of a complete software development workbench, which supports project management, requirements engineering, analysis and design, and also code generation activities.

Currently, the ProjectIT initiative provides two complementary tools: ProjectIT-Enterprise and ProjectIT-Studio. The former is a web-application that supports the management of virtual and distributed teams, and presents a strong emphasis on activities related with project management [11]. ProjectIT-Enterprise supports the definition of tailored software development processes. This feature entails the definition of best practices, both project and artifact templates, and also supports the instantiation of designed process afterwards as concrete projects. The prior existence of a process has a normative influence and accelerates the project bootstrap within organizations by reducing coordination efforts, due to the clear definition of roles, workflows, and artifacts.

On the other hand, the latter is a desktop-based CASE tool whose goal is to provide a set of tools for enhancing productivity of requirements development and management, modeling with domain-specific languages, and automatic code generation through transformations and generative processes. For dealing with RE issues we designed a meta-model for requirements specifications which, by raising their rigor and quality through verification, facilitates the reuse and integration with model-driven development environments, specifically ProjectIT-MDE tools [10]. ProjectIT-Studio/Requirements [12] is the tool that embodies these features, and it is implemented as a plugin for ProjectIT-Studio. It provides a rich text editor that supports ProjectIT-RSL, a controlled natural language designed for requirements specification (for details see [5]). This editor supports the vision of a tool for writing requirements documents, like a word processor with intellisense that detects and gives instant feedback regarding errors violating the requirements language. It uses Natural Language Processing (NLP) techniques to extract the underlying requirements model.

Although ProjectIT-Studio/Requirements addresses most of requirements development activities, it lacks support for requirements management. Moreover, there is no support for communication mechanisms and the collaborative environment is insufficient. Hence, we decided to develop a collaborative environment, according to the Web 2.0 principles [13], which would mimic the ProjectIT-Studio/Requirements features, namely its rich text-editor, by employing AJAX (Asynchronous JavaScript and XML) techniques. Moreover, according to the characteristics mentioned in Section 1, a Wiki appears as a more adequate and feasible approach to the shortcomings aforementioned. Therefore, we decided to develop a Wiki-based system to endow ProjectIT-Enterprise with collaborative Requirements Engineering support.

3. Tool Support

To address the socio-technical issues related with the RE process's nature we decided to shift from a desktop-based CASE tool [12] to a CSCW Wiki-based tool. The RE process is both collaborative, but most often it is simultaneously competitive. Negotiation and the process of reaching consensus are of paramount importance during the project's scope definition. Consensus needs to be achieved during both the requirements development process and throughout the activities of requirements management process. The driving forces are intrinsically social, dealing with several aspects of human characteristics.

The ProjectIT approach processes can be divided in two main moments, thus the RE processes that it entails follow the same structure (see Figure 1).

3.1. Process Level

At *Process Level*, the *Architect* is responsible for defining higher-level issues, namely those regarding to adjustments or extension of both RSL languages and the definition of the requirements development process. The former consists on enriching the ProjectIT-RSL with new linguistic patterns

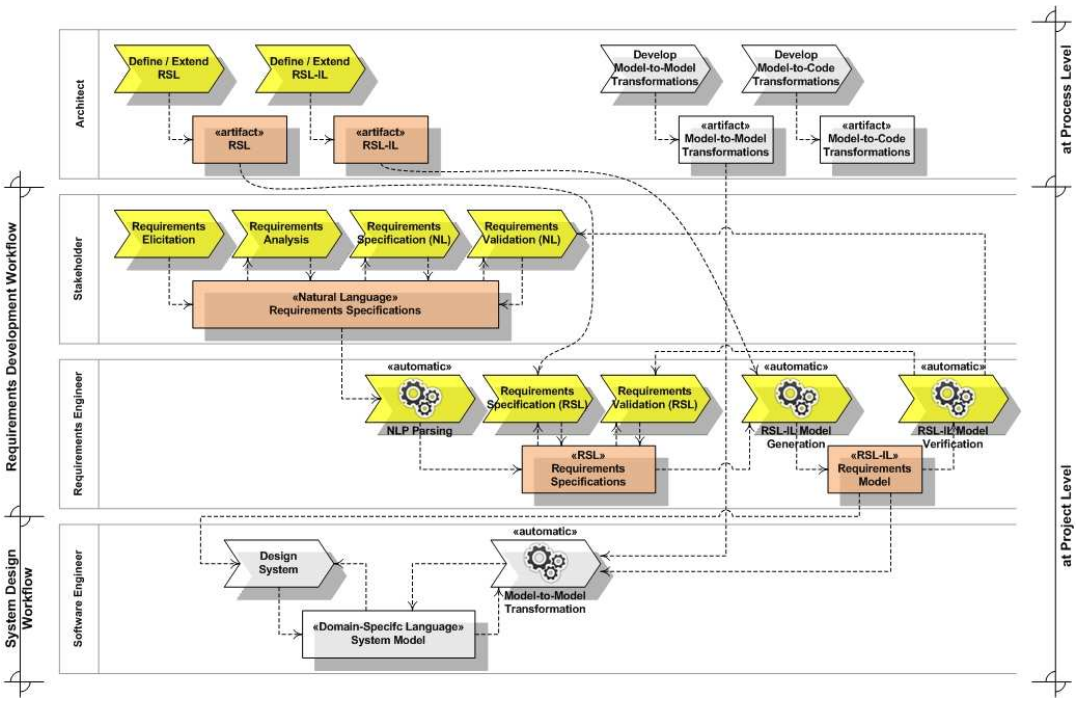

Figure 1. Requirements Development Workflow.

related to the applicational / business domain and, additionally, to enhance the intermediate language RSL-IL with the proper adjustments to support new lower-level concepts that may eventually arise from the extensions added to ProjectIT-RSL. These tasks are performed out of the Wiki-based tool, since they affect the underlying parsing mechanisms. These mechanisms are later used by the NLP algorithms that automatically extract concepts and relations from the domain model of the target system. The Wiki-based tool provides a configuration page for uploading these definitions and extensions of the RSL languages. The latter, related with the definition of the requirements development process, consists of embedding the RE's best practices into the Wiki-based tool's content, artifacts, training pages, and tutorials, in the form of templates. The prior existence of templates accelerates the process of bootstrapping the project and provides guidance to stakeholders, thus helping them to rapidly tackle the problem at hand, and to achieve effectiveness through a methodological approach. The process of capturing and reusing the RE's best practices entails the usage of a dynamic attribute-based mechanism based upon a generic Wiki metamodel for content-related structural elements [14]. However, the technical details of this mechanism are beyond the scope of this paper. This mechanism provides support for managing dynamic metadata of the Wiki-based tool's concepts responsible to supply structure to the Wiki's content. The creation of a specific attribute set for each one of these concepts permits a fine-grain selection of reusable items to extract and incorporate into new or existing templates. This mechanism is more flexible than the traditional process of defining and setting beforehand templates that cannot be changed. Although we can still support this conventional process, which consists of the creation of a new Wiki and populating it accordingly to the RE's best practices that we want to enforce during the requirements development process, this mechanism supports a project *post mortem* process to learn from past mistakes and good practices. Still at *Process Level*, the *Architect* may also require to add new model-to-model or model-to-code transformations to address new requirements models' refinement features or to support early prototypical interaction, respectively. The aforementioned processes are illustrated in the first partition of the diagram depicted in Figure 1.

3.2. Project Level

After these more abstract definitions at *Process Level*, we can apply them to a concrete scenario, which we called *Project Level* (the remaining partitions of Figure 1).

3.2.1. The Bootstrap Activities. At this level, the *Requirements Engineer* can instantiate a new Wiki project, and specify which of the existing templates should be applied. Beforehand, the *Requirements Engineer* must identify the major stakeholders' groups, namely among the customer's personnel and the different final users. The criteria is project-dependent, hence there isn't a single valid solution. There

are several business-related, or even personal, characteristics that might influence this clustering task. However, a good heuristic consists of following a departmental or enterprise functional division. Another alternative is to define user communities according to the target system's functionality they will use. Additionally, the Wiki-based tool provides a mechanism to anonymously vote for members of each of the identified cluster in order to elect the group's champion. This representative member plays an important role since it makes decisions on behalf of the rest of the group. However, the tool also ensures the champion to be notified of any changes regarding to features in which the group has interests. Additionally, the tool provides means to monitor the champion's participation in order to notify the project manager that the person assigned to this task is playing its role correctly.

3.2.2. The Stakeholders' Activities. Only from this moment on we can consider that the traditional requirements development process [1] truly begins. This process is performed by both *Requirements Engineer*s and non-technical business people. When considering both roles, and for the sake of simplicity, we plainly designate this generic role as stakeholder at the second partition of Figure 1).

The first activity consists on eliciting the target system's requirements. During this activity the domain experts are invited to contribute with their tacit knowledge. Moreover, their input should address the clear definition of the project's invariants, namely the project's vision and scope. Although change is a constant in all projects, we should consider that there are some guidelines that should be stable between releases. This information must be documented within a specific WikiPage, which must be always accessible from the main navigation panel. This information should give to all stakeholders a common ground of understanding, namely regarding to the purpose and rationale of the system's development, and also a clear separation between what should be and what should not be included in the target system.

Despite of the automation provided by the Wiki-based tool, there are still some tasks that require the active participation of the *Requirements Engineer*s, namely those related with ethnographic techniques, where the *Requirements Engineer*s observe the final users performing their business tasks to clearly identify their responsibilities and to capture the underlying business processes. Another task that strongly depends on the human factor is examining logs, reports, and documentation in general. Although it is not supported yet by the tool, an important feature is to use the same NLP techniques for the automatic extraction of information within these documents after they are uploaded and appended to the respective WikiPages [14]. These last two techniques deal with very useful and rich sources of information during the activity of eliciting requirements.

3.2.3. The Requirements Engineers' Activities. According to the *Stakeholder*'s partition represented in Figure 1, apart from graphical information such as diagrams and explanatory images, all these activities consume and produce natural language content, which we called *Requirements Specifications*. However, in parallel, the *Requirements Engineer*s can take advantage of the tool support. The Wiki-based tool provides more advanced features to the *Requirements Engineer* role, namely it seeks to enhance productivity by automatically performing repetitive and error-prone tasks related to the extraction of concepts and relations of the target system's applicational / business domain. This automatic process makes use of NLP techniques and provides the first drafts of the problem's domain. From this point on the *Requirements Engineer*s benefit from usage of requirements specifications written with a controlled natural language. Since we are considering users with technical background they can easily deal with these more rigorous specifications. Therefore, the Wiki-based tool provides assistance with an AJAX-enabled rich text editor, with features similar to those provided by the ProjectIT-Studio/Requirements [12]. Thus, *Requirements Engineer*s can contribute by specifying directly requirements with this controlled natural language. Following this more technical approach of specifying requirements enhances productivity and, simultaneously, reduces ambiguity and inconsistency. The supporting tool plays an important role throughout the process, since it provides a multiple-view approach, following the dashboard metaphor. Therefore, the *Requirements Engineer*s are assisted during the processes of: (1) revising the automatically extracted requirements specifications from natural language sources; (2) adding or editing existing requirements at different abstraction levels; and (3) validating the specifications of the system under development, through multiple perspectives over their facets. One particular case in which it is desirable to use this more technical specification approach refers to the target system's behavior. Since it addresses complex computations and/or interactions, we advocate that it should be directly specified with the ProjectIT-RSL extensions.

In background the Wiki-based tool automatically extracts RSL-IL models from the ProjectIT-RSL specifications that are being produced in the meanwhile. These models provide machine-computable specifications that can be used in more formal verification processes. Additionally, within the context of ProjectIT approach, the model-to-model and model-to-code transformations, defined in advance by the *Architect* can be applied to produce more detailed requirements specifications in RSL-IL, or event to produce source code for low-fidelity prototyping. The refinement of requirements models produces new RSL-IL statements that can be translated back to ProjectIT-RSL, thus providing feedback for both business people and *Requirements Engineer*s.

3.2.4. Subsequent Activities. The last partition of Figure 1, regarding to the activities performed by the *Software Engineer*, depicts the information flow from requirements models to the next stage of the ProjectIT approach, concerning to the system design workflow. The requirements models produced can be directly used by the ProjectIT-Studio/MDD modeling tools [10] or they can be transformed into a particular domain-specific language related with the applicational / business domain of the software system under development. From this moment on, the *Software Engineer* begins the elaboration of the actual target system's solution following a MDE paradigm. The ProjectIT-Studio provides the necessary support to the employment of MDE techniques to achieve the successive refinement of the models until they reach a proper level of detail to be consumed by the final stage of the automatic generative process.

4. Related Work

There are academic and industrial projects that stress the benefits of using a Wiki-based approach to support the active stakeholders' participation during Requirements Engineering processes [15]. Wikis have the potential to leverage stakeholders' participation: their lightweight edition workflow and potential to become central documentation repositories, make them easier to use and tailor than proprietary requirements management tools [15].

Regarding to the benefits of using NLP techniques, Witte et al. [16] introduce an innovative vision of a "self-aware" Wiki, capable of reading, understanding, transforming, and writing its own content. Their goal is to go beyond purely syntactic approaches by bringing the full potential of current natural language technologies to the Wiki platform. Although a completely automated understanding of natural language is still not feasible, there exist already a number of robust language processing techniques that can strongly improve the user's experience. Witte's work consists on a complementary approach to Semantic Web: automatically obtaining semantic metadata with NLP techniques.

Still regarding to NLP techniques, Kuhn' project [17], [18] is a Semantic Wiki enhanced with NLP techniques. Kuhn uses Attempto Controlled English (ACE), a controlled natural language, for representing content. The goal is to achieve a shallow learning curve to minimize the integration phase of new users and spare the need of experts' mediation. However, albeit ACE also supports a wide range of natural language constructs it is neither domain-specific, nor presents rule-based extensibility features like ProjectIT-RSL. Another important issue regarding Kuhn's work is that it only ensures syntactically and grammatically correctness, but does not addresses semantic validation.

Our proposal appears as a natural sequence of our previous work and its feasibility is corroborated by the projects aforementioned. Moreover, the proposed enhanced Wiki system design seeks to comply with Wiki design principles [19] and address most of Wiki problems [13]. As Whitehead [20] points out, the future directions for collaboration in Software Engineering include the tight integration between web- and desktop-based development environments, fostering the participation of not only domain experts, but also engaging customers and end users during the entire software development process. The ultimate goal is to delegate on final users and business people the responsibility of directly specifying what they need or want, because the overall quality or value of information is judged by themselves.

5. Conclusions

This paper presents an innovative tool in the area of RE. Its main purpose is to support the RE aspects of the ProjectIT approach, whose endeavor is to seamlessly integrate RE with the MDE processes. Initially, we provide an overview of the ProjectIT initiative's context, namely its approach, languages, and tools. Despite the results achieved, there are still some aspects that can be improved, which justifies our effort to extend our current workbench by implementing part of the functionality already offered by the ProjectIT tools within a CSCW Wiki-based platform. Thus, this new tool seeks to incorporate features of several different fields of expertise and technologies, to address the socio-technical challenges of the RE process's nature. Our vision is aligned with the current trend of moving applications to the web [20]: by creating web-based environments composed of a mix of web-based and desktop systems, we can improve project collaboration trough a broader engagement possible. Regarding CSCW features, it supports and fosters communication-intensive tasks of RE. The new channels of communication that this tool incorporates, provided by the emerging and high-responsive Web 2.0 technologies, avoids confusion and reduces misunderstandings with the purpose of minimizing rework at later stages. Moreover, this Wiki-based tool provides a central repository for all project-related information, such as objectives of the software system being created, and a project glossary that defines specialized terms from the application domain. Furthermore, this Wiki-based solution provides a sound approach for documentation [21], to capture best practices through templates, and to provide training documentation to help stakeholders understanding the project's activities and terminology. The ultimate purpose is to provide an enhanced interactivity and breaking the natural boundaries between stakeholders by accommodating different backgrounds and personal skills; thus fostering team-building for distributed virtual teams: all stakeholders should level up regarding the concepts and terminology of the application domain.

Despite not being addressed by this paper, the described Wiki-based tool supports the requirements management process, besides the requirements development process; thus

covering all RE activities. An aspect of paramount important is to explore traceability mechanisms, no only at the traditional document level, but also at a finer-grain of detail, namely among elements of requirements models and between them and external artifacts, such as diagrams, images, documentation, and source code.

As future work we plan to address the requirements management support both in terms of the ProjectIT approach and the respective supporting tool. Moreover, there are several social and collaboration issues in the context of Software Engineering that we intend to explore, namely the impact of the organizational culture [22], [23] regarding the project's success. Moreover, it also important to analyze to which extend does the up-to-dated requirements information can benefit the stakeholders' awareness of the project status to support their decisions [2]. Still, Wikis should offer support for rudimentary social network analysis with built-in observation features, in order to provide awareness of community activity [7]; it would be interesting to implement introspection mechanisms to monitor working methods and communication channels.

References

[1] K. Wiegers, *Software Requirements*, 2nd ed. Microsoft Press, March 2003, ISBN: 978-0735618794.

[2] C. Hood, S. Wiedemann, S. Fichtinger, and U. Pautz, *Requirements Management: The Interface Between Requirements Development and All Other Systems Engineering Processes*, 1st ed. Springer, December 2007, ISBN: 978-3540476894.

[3] T. Bell and T. Thayer, "Software requirements: Are they really a problem?" in *ICSE '76: Proceedings of the 2nd international conference on Software engineering*. IEEE Computer Society Press, 1976, pp. 61–68.

[4] H. Foster, A. Krolnik, and D. Lacey, *Assertion-based Design*. Springer, 2004, ch. 8 - Specifying Correct Behavior.

[5] C. Videira, D. Ferreira, and A. Silva, "A Linguistic Patterns Approach for Requirements Specification," in *Pro. of the 32nd EUROMICRO Conf. on Soft. Eng. and Advanced Applications*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 302–309, ISBN: 0-7695-2594-6.

[6] D. L. Duarte and N. T. Snyder, *Mastering Virtual Teams*. Jossey-Bass Inc., Publishers, 2000.

[7] W. Cunningham, "Wiki Design Principles," March 2008, Retrieved Wednesday 25th March, 2009 from http://c2.com/cgi/wiki?WikiDesignPrinciples.

[8] A. Ebersbach, M. Glaser, and R. Heigl, *Wiki : Web Collaboration*. Springer, November 2005.

[9] A. Silva, "O Programa de Investigação ProjectIT (whitepaper)," October 2004.

[10] A. Silva, J. Saraiva, D. Ferreira, R. Silva, and C. Videira, "Integration of RE and MDE Paradigms: The ProjectIT Approach and Tools," *IET Software Journal*, vol. 1, no. 6, pp. 217–314, December 2007.

[11] P. Martins, "ProPAM - a Software Process Improvement Approach based on Process and Project Alignment," Ph.D. dissertation, Instituto Superior Técnico (IST/UTL), Lisbon, Portugal, September 2008.

[12] D. Ferreira and A. R. da Silva, "A requirements specification case study with ProjectIT-studio/requirements," in *SAC '08: Proc. of the 2008 ACM symposium on Applied computing*. New York, NY, USA: ACM, March 2008, pp. 656–657.

[13] S. Murugesan, "Understanding Web 2.0," *IT Professional*, vol. 9, no. 4, pp. 34–41, 2007.

[14] D. Ferreira and A. R. da Silva, "An Enhanced Wiki for Requirements Engineering," in *35th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, August 2009, (to appear).

[15] B. Decker, E. Ras, J. Rech, P. Jaubert, and M. Rieth, "Wiki-Based Stakeholder Participation in Requirements Engineering," *IEEE Software*, vol. 24, no. 2, pp. 28–35, 2007.

[16] R. Witte and T. Gitzinger, "Connecting wikis and natural language processing systems," in *WikiSym '07: Proceedings of the 2007 international symposium on Wikis*. New York, NY, USA: ACM, 2007, pp. 165–176.

[17] T. Kuhn, "AceWiki: A Natural and Expressive Semantic Wiki," in *Semantic Web User Interaction at CHI 2008: Exploring HCI Challenges*. CEUR Workshop Proceedings, 2008.

[18] ——, "AceWiki: Collaborative Ontology Management in Controlled Natural Language," in *3rd Semantic Wiki Workshop*. CEUR Workshop Proceedings, 2008.

[19] A. V. Deursen and E. Visser, "The Reengineering Wiki," in *CSMR '02: Proceedings of the Sixth European Conference on Software Maintenance and Reengineering*. Washington, DC, USA: IEEE Computer Society, 2002, p. 217.

[20] J. Whitehead, "Collaboration in Software Engineering: A Roadmap," in *FOSE '07: 2007 Future of Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 214–225.

[21] A. Aguiar and G. David, "WikiWiki weaving heterogeneous software artifacts," in *WikiSym '05: Proceedings of the 2005 international symposium on Wikis*. New York, NY, USA: ACM, 2005, pp. 67–74.

[22] M. Buffa, F. Gandon, G. Ereteo, P. Sander, and C. Faron, "SweetWiki: A semantic wiki," *Web Semantics*, vol. 6, no. 1, pp. 84–97, 2008.

[23] P. Louridas, "Using Wikis in Software Development," *IEEE Software*, vol. 23, no. 2, pp. 88–91, 2006.