# Wiki Supported Collaborative Requirements Engineering

David Ferreira
INESC-ID, Instituto Superior Técnico
Rua Alves Redol 9
Lisbon, Portugal
david.ferreira@inesc-id.pt

Alberto Rodrigues da Silva
INESC-ID, Instituto Superior Técnico
Rua Alves Redol 9
Lisbon, Portugal
alberto.silva@acm.org

## ABSTRACT

The high number of unsuccessful IT projects, due to the inconsistent and ambiguous requirements specifications, justifies the proposal of new socio-technical approaches to overcome these software quality issues. To address these problems it is required a platform that simultaneously fosters stakeholders' involvement to capture their tacit knowledge, but also to enforce Requirements Engineering best practices. In this paper, we present our approach for enhancing the quality and rigor of requirements specifications by combining emergent Web 2.0 concepts with CASE tools for requirements specification and validation. This approach provides a flexible platform for collaborative Requirements Engineering: non-technical stakeholders are assisted during the elicitation process and requirements engineers benefit from the seamlessly integration with a broader CASE tool.

## Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications—*Elicitation Methods, Tools*; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*CSCW, Web-based interaction*

## General Terms

Design, Human Factors, Languages

## Keywords

Web 2.0, Wiki, CSCW, Requirements Engineering, NLP

## 1. INTRODUCTION

Requirements specification is crucial for achieving success on IT projects. These early activities of Requirements Engineering are crucial for both project management and software development process. They are the starting point to define the scope of the problem to be solved (i.e., the target system). Mistakes made during these early phases of a system development (e.g., inadequate, inconsistent, incomplete, or ambiguous requirements) result in quality problems and large costs to correct them [3]. Therefore, requirements specifications influence all the subsequent activities and define the basis for assuring quality through test specifications.

Despite of the availability of formal methods, the majority of requirements documents are still specified with natural language, mainly because of its expressiveness and familiarity to non-technical users. To address this problem we created a requirements specification language based on linguistic patterns. Additionally, we developed a supporting CASE tool in the scope of the ProjectIT initiative. This tool, the ProjectIT-Studio/Requirements, supports non-technical stakeholders during the specification process by validating their input with Natural Language Processing (NLP) techniques. However, despite of the results achieved with our desktop-based approach, we realize the importance of social, organizational, and collaborative issues during requirements elicitation, specially within contexts regarding to virtual and globally distributed teams. These scenarios require an evolutionary discovery of requirements in order to address the high level of uncertainty [11].

Wikis are lightweight, and social web-based systems that promote collaborative work, namely they provide the foundations for constructivist learning environments. The philosophy behind the Wiki concept is simple [8]. It is based upon an open model and easy edition workflow. In this model, by using a standard web-browser, any reader is simultaneously a potential author and reviser. By comparing and commenting other participants inputs, the user not only becomes aware of what his peers know (the community knowledge), but he also benefits from an unconscious self-assessment effect, hence consolidating his believes. Wikis support work group processes, thus they are used in several contexts to record and refine ideas and information. There are even emergent approaches that use Semantic Web concepts to support enterprise-wide knowledge management, especially tacit knowledge transferred between coworkers, which is highly valuable but also extremely difficult to capture [12]. Besides these advantages, Wikis are excellent platforms for supporting project management activities, since they allow one to share ideas, keep logs of project progress, documenting project plans, and coordinating conferences [12]. Because Wikis can easily integrate different stakeholders' perspectives and manage software development processes, we consider Wikis not only as suitable to become the underlying technology for a Requirements Engineering web-based tool, but also as a prominent Web 2.0 [2] technology


*Wikis4SE'08 Workshop* September 8, 2008, Porto, Portugal

for integrating into ProjectIT-Enterprise, our existing web-based tool for process definition and project management.

This paper presents our proposal of combining our stand alone IDE-like tool, designed for capturing requirements, with the benefits of Web 2.0 enabled Computer Supported Cooperative Work (CSCW) tools. Section 2 introduces the ProjectIT initiative, highlighting its architecture and main tools. Section 3 provides a detailed analysis of our proposal for a Wiki-based Requirements Engineering tool, whose pros and cons are thoroughly analyzed in Sections 4. Still in Section 4, we refer projects and tools related with our vision and goals. Finally, in Section 5, we draw our conclusions, which justify our perception that this proposal has innovative contributions for the community.

# 2. PROJECTIT INITIATIVE

As a result of the experience gathered from previous practical projects, the Information Systems Group of INESC-ID, started an initiative, called ProjectIT [22]. Its main goals are to enhance productivity and rigor of Software Engineering (SE) activities. In order to achieve our goals, we began the development of a complete software development workbench, which supports project management, requirements engineering, analysis, design and code generation activities.

## 2.1 ProjectIT-Requirements

ProjectIT-Requirements is the conceptual component of the ProjectIT architecture that deals with Requirements Engineering activities. Its main goal is to develop a model for requirements specification which, by raising their rigor and quality through validation, facilitates the reuse and integration with model-driven development environments, specifically ProjectIT-MDE tools. One of the results of this project is a new requirements specification language, denominated ProjectIT-RSL. Its design took into account the format and structure conventions of requirements documents we have elaborated for in-house projects. Throughout the process we identified a set of linguistic patterns and best practices associated with requirements specification. From these patterns we determined the main concepts used in requirements specification, how they are structured, organized, and combined into wider scope blocks. Afterwards we derived a metamodel of the identified concepts, which is also the base of the UML profile common to all our tools. The metamodel main focus is to capture interactions, through *operations*, between target system's *actors* and the *business entities* that it manages. The complete description of ProjectIT-RSL is beyond the scope of this paper (for further detail see [24]).

## 2.2 ProjectIT Tools

In order to test and consolidate our research ideas we have been developing a suite of tools, particularly: (1) ProjectIT-Studio, a desktop-based CASE tool (i.e., with rich-client interface) for high productivity tasks; and (2) ProjectIT-Enterprise, a web-application (i.e., with standard web-client access) focused on collaborative work and project teams' management. These tools are complementary: the Studio version goal is to provide a bundle of tools for enhancing productivity of requirements specification and management, design models, automatic code generation, and software development. On the other hand, the Enterprise version provides collaborative mechanisms to manage virtual and globally distributed teams, and presents a strong emphasis on activities related with project and documents management.

### 2.2.1 *ProjectIT-Studio*

ProjectIT-Studio is an integrated environment that supports important tasks of the software development life-cycle, such as requirements specification, architecture definition, system design and modeling and code generation. In addition, and because it promotes productivity, ProjectIT-Studio provides innovative features, such as requirements-to-models, models-to-models, models-to-code transformation techniques, template managing and UML profile definition. The set of tools that together comprise the ProjectIT-Studio workbench are illustrated in Figure 1.

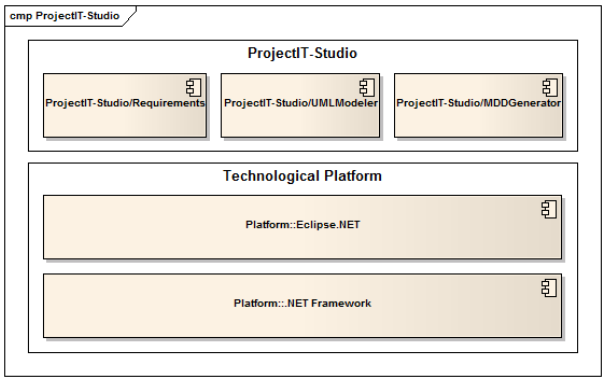


**Figure 1: ProjectIT-Studio components.**

ProjectIT-Studio/Requirements is a ProjectIT-Studio plugin that implements ProjectIT-Requirements. It includes a rich text editor for supporting the vision of a tool for writing requirements documents, like a word processor that detects and gives instant feedback of errors violating the requirements language. This editor uses NLP techniques to capture the underlying requirements model and validate it. Syntax-highlighting, auto-complete, and on-the-fly error checking with textual annotations provides feedback accordingly.

### 2.2.2 *ProjectIT-Enterprise*

ProjectIT-Enterprise is a web-based collaborative system focused on IT project management activities, supporting the definition of tailored software development processes. The process configuration entails the definition of best practices and both project and artifact templates. The designed process can be instantiated afterwards as concrete projects. The a priori 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. This tool's components are depicted in Figure 2.

ProjectIT-Enterprise is built upon WebComfort platform, which is a Content Management System (CMS) based on ASP.NET 2.0 technology. WebComfort supports the operation and integrated management of web-applications by providing tools and mechanisms for management of structured and unstructured content through standard web-browsers. This platform presents several advantages when compared with the ad-hoc and manual process of developing web-applications, namely: (1) provides platform extensibility,

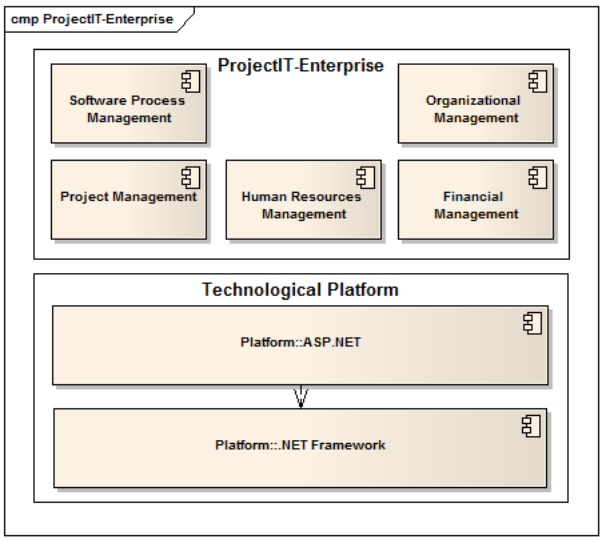


**Figure 2: ProjectIT-Enterprise components.**

modularity, and reuse; (2) offers consistent and usable interfaces, with internationalization and localization features; (3) avoids content duplication and redundancy; (4) separation of concerns between presentation (e.g., visual themes) and contents (both static and dynamic); (5) supplies an integrated portal administration, with an easy installation process based on module toolkits; and (6) user management features and fine-grain access control mechanisms.

# 3. PROJECTIT-WIKI/REQUIREMENTS

The main motivation behind our proposal to develop a Wiki-based platform for dealing with Requirements Engineering issues arises from the paramount importance that social, organizational, and communication aspects acquire during the process. Moreover, albeit currently commercial suites do support distributed teams, allowing a broader stakeholder's participation [25], the majority of these tools are mostly desktop-based. Therefore, collaborative Requirements Engineering tools are needed, in order to address and facilitate the negotiation among technical and non-technical stakeholders during the requirements elicitation process.

## 3.1 Vision

At the present there are no integrated web-based environments that cover the entire software development life-cycle [25]. We intend to contribute with a platform with innovative features that overturns the current state of the art of Requirements Engineering tools. We are developing a set of WebComfort modules, using Wiki as a base technology, for integrating both our desktop-based ProjectIT-Studio/Requirements tool with the project management features already provided by ProjectIT-Enterprise. With this approach we can capture, in an evolutionary manner, the state-of-mind shared by stakeholders and, simultaneously, reduce conflicts inherent to the coordination of large teams. These conflicts can occur since the requirements elicitation process is cooperative, but also competitive. Moreover, by fostering participation, this approach can leverage unique skills and background of each team member. Our goal is to improve quality of IT projects focusing on requirements specifications rigor allied with an iterative and incremental approach. Furthermore, we seek to achieve a greater productivity (more interactive and responsive) by endowing the proposed web-based tool with Web 2.0 technology.

### *3.1.1 Wiki Content's Language*

The preferred medium of communication between human agents is still unstructured and informal natural language acts, such as conversations and meetings. To address this fact, commercial tools for requirements management, such as IBM Rational RequisitePro, Telelogic DOORS, and Borland CaliberRM, provide the use of natural language to specify requirements. Additionally, these tools have a strong emphasis on traceability and change impact analysis. However, their natural language approach deals with requirements as black boxes, i.e., without extracting their meaning; thus the validation process is limited to the analysis of handmade metadata or the structure of relationships between "requirements boxes". Moreover, due to the usual tension among language expressiveness and simplicity, the commercial tools' approaches do not eradicate natural language usage drawbacks, namely ambiguity and inconsistency. The storage of informal information opens the door for inconsistent, incomplete, and generally low-quality requirements.

Although ProjectIT-Requirements approach also seeks for the expressiveness and familiarity of natural language to capture requirements, it addresses these undesirable aspects with a controlled natural language and a pipeline of NLP techniques, such as free-form text normalization (format, typographic, and morphologic transformations), full-text index generation, part-of-speech (POS) tagging, and heuristic fuzzy match parsing. With this technology we ensure rigor while reducing the conceptual gap between the non-technical stakeholder's mental model and the requirements engineer conceptual perception of the target system. Moreover, the usage of NLP techniques mentioned above can be used to capture semantic information required to connect and formalize requirements, as stated by Witte et al. [27].

The main goal of ProjectIT-Wiki/Requirements is to provide an AJAX-enabled text editor with rich user interfaces and able of dealing with heavy interaction to support WYSIWYG features, following ProjectIT-Studio/Requirements design principles. Despite of this fact, the choice of a Wiki tags' notation (for navigation and formatting purposes) is still an important issue. Since we are developing a Wiki engine from scratch, and also aiming at standards compliance and content portability, we decided to follow a Wiki Creole 1.0 [15] native implementation [26] approach, probably with some Creole additions in the future.

### *3.1.2 Wiki Technology's Adequacy*

One of the most important tasks of both Project Management and Requirements Engineering is the scope definition, by clearly specifying the target system's requirements (i.e., what it should do). The collaboration at this level of upfront negotiation has profound impact on software's quality; thus it can compromise its success. Wikis appear as a feasible solution to address these early issues, since they foster an evolutionary mind-set of all participants throughout the elicitation process. During the process each participant must agree with the contribution of the others stakeholders or, alternatively, negotiate through communication mechanisms until a consensus is achieved. Wikis can perform the

role of project documentation repositories, thus supporting this iterative and incremental approach [1]. Furthermore, through web-based applications one can achieve easier cross-organization collaboration, namely asynchronously gathering stakeholders' feedback about requirements. Requirements elicitation is a dynamic and social process that must reflect the community consensus [13].

Nowadays, the most cited quality assessment for requirements specifications is still the straightforward comparison with domain experts' opinion [16]. This evaluation criterion is in line with the collaborative features of Wikis, namely inspection and reviews. During these tasks several participants are brought together, hence problems and errors are easily identified through the conjunction of multiple perspectives. However, in current practice, customers are only initially consulted about requirement needs, which are then integrated into a final set of requirements that drive the whole development of the target system [25]. This approximation does not follow Requirement Engineering best practices since it only supports a static perspective of requirements, where customers are only engaged during the requirements elicitation. The assumptions of this approach are unrealistic because they do not support change management, namely requirements' change requests. Within Requirements Engineering it is fundamental working with customers to ensure that the artifacts accuracy reflect their real needs, not only their initial vision of the target system.

Regarding to the integration of our tools with Wiki technology (i.e., a mixture of web- and desktop-base environments), this approach will combine both the benefits of ProjectIT-Studio and ProjectIT-Enterprise. The latter's project templates will reduce the amount of coordination required to initiate an IT project; hence participants can quickly tackle the project at hand. Requirements elicitation activity can be significantly enhanced if there is a background methodology encompassing artifact templates with a predefined structure, clear identification of the participant roles, and steps to be performed. Following our approach [23] each type of artifact has its own semantics, ranging from free-form controlled natural language to formality of programming languages, passing by semi-formal semantics of UML diagrams and UML profiles.

Finally, the Wiki's collaborative nature encourages negotiation and discussion to achieve consensus and concepts' consolidation through successive refinements. To interact with others, participants must formulate their own opinions about the subject at hand, thus avoiding absence in the participation. This leads to awareness of personal opinions and learning by self explanation effect.

## 3.2 Wiki Design

According to Cunningham's "Wiki Design Principles" [8], a Wiki engine should be incremental and organic, supporting an evolutionary approach for content creation. It should be natural and universal, offering intuitive and identical interfaces for both edition and management. Furthermore, a Wiki engine should discourage content duplication (i.e., redundancy) and, simultaneously, enforce content convergence. Finally, it should offer support for rudimentary social network analysis with built-in observation features, in order to provide awareness of community activity.

Bearing in mind these design principles and also the criteria proposed by Murugesan et al. [20] for choosing a Wiki engine, we decided to develop a new Wiki system because our needs for collaborative Requirements Engineering is not entirely compatible with what we found in existing Wikis [5, 6]. WebComfort, the technological platform of ProjectIT-Enterprise, appears to be a sound framework for developing from scratch a Wiki system tailored for Requirements Engineering tasks, offering deep integration with project management features provided by ProjectIT-Enterprise.

With the information gathered during the research, we defined a Wiki metamodel (Figure 3) to capture the most common concepts that our solution should support. Although extremely simple, this metamodel has some peculiarities. The main concept is named *WikiPage* and corresponds to a generic page of the Wiki system. The *WikiPage* can contain any type of content. Besides the conventional textual content, *WikiPage*s can contain coarse-grain, block-like content, which we named *WikiPageItem*. Typically, a *WikiPageItem* maps to a text chunk or other more elaborated textual structure. Additionally, a *WikiPage* can contain links to another *WikiPage*s, enabling straightforward navigation between them. Despite its content-related flexibility, every *WikiPage* must obey to the constraints (e.g., structure, style, active operations, etc.) specified by the *WikiPageTemplate*. This last concept is extremely useful to enforce Wiki-related best practices. Finally, a *WikiPage* can be enriched with *WikiPageView*s (automatically generated views concerning their content or relations among them).

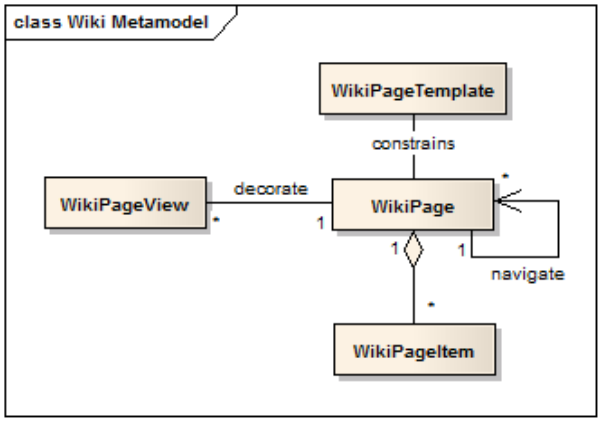


**Figure 3: Wiki metamodel (simple view).**

After the definition of the Wiki metamodel, we specified the ProjectIT-Wiki/Requirements model (Figure 4), which applies the stereotypes (i.e., archetypes) of the former to concepts related with Requirements Engineering. The main concept of this model is *RequirementsDocument* which is derived from *WikiPage* and clearly maps to the homonymous artifact. Each *RequirementsDocument* can be traced by a *System*. The artifact to which the *System* concept refers to is not stored or managed itself by the ProjectIT-Wiki/Requirements system. However, it is important to include the *System* concept in the model to support impact analysis by using traceability relations.

*System*s can be decomposed into smaller units of behavior. This composite relation between *System*s is represented with a reflexive aggregation. To reflect this relation between *System*s, the *RequirementsDocument* concept presents a similar reflexive aggregation, meaning that a *RequirementsDocu-*

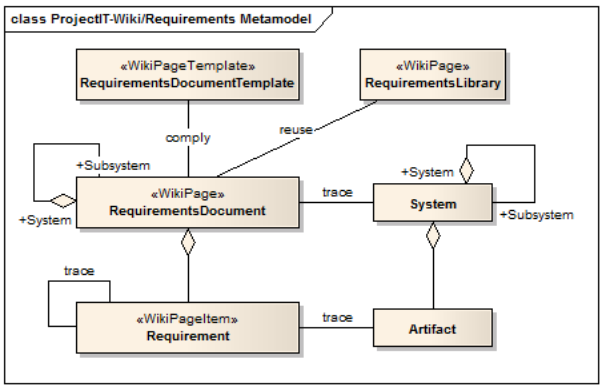


**Figure 4: PIT-Wiki/Requirements model.**

*ment* can encompass several other *RequirementsDocument*s in a hierarchical manner. This relation ensures that the hierarchy between *RequirementsDocument*s is aligned with the *Systems*' composition. An innovative characteristic of the Wiki metamodel is that it defines a fine-grain level of the *WikiPage*'s content, which we designated *WikiPageItem*. The latter, is naturally refined into the *Requirement* concept, the unit block of our requirements document model. *Requirement*s should be related among them for traceability concerns. A *RequirementsDocument* can aggregate several *Requirement*s. Following the same rationale that supports the relation between *System* and *RequirementsDocument*, the *Requirement* should be traced by an *Artifact* (e.g., diagram, source code, document, etc) of the target *System* or Sub*system*. Another important specificity of this model is that it contemplates reuse, through a *RequirementsLibrary* concept. Finally, the best practices of requirements specification should be enforced with the relation between a *RequirementsDocument* and a *RequirementsDocumentTemplate*, which derives from the *WikiPageTemplate* stereotype.

Pertaining to user roles, we identified four main actors that together cover most of, if not all, the possible interactions with ProjectIT-Wiki/Requirements: (1) the *reader*, a project team member with access authorization; (2) *contributor*, same as the previous one but with the ability of adding/revising content (e.g., a domain expert of the organizational unit); (3) *moderator*, same as the previous one but responsible for the whole requirements document (e.g., a department director); and (4) *administrator*, an orthogonal role required to administrate and moderate the Wiki (e.g., requirements engineer or project manager). A broader participation through the adoption of the Wiki philosophy reduces the risk that the final software product does not meet customers' needs. However, it does not necessarily mean that stakeholders would be given access to all artifacts. The main idea is to define confined namespaces that match with subsystems of the target system, which are then assigned by the requirements engineer to the person within the organization that is responsible (e.g., department director) for that business area. Afterwards, the same responsible delegates the requirements specification of the subsystem to coworkers, typically subordinates. With a strong emphasis on the enforcement of confined namespaces we can achieve modular and reusable specifications. This approach eases traceability and conflict detection/resolution.

## 3.3 Architecture

The architecture that we propose, depicted in Figure 5, tries to address some common problems of standard Wiki systems [9]. Since we have already developed some of the mechanisms required by the ProjectIT-Wiki/Requirements system, we started by refactoring this functionality into services to make them reusable. We designed a solution based on a clear separation between Wiki engine and the underlying model for managing requirements with NLP techniques, i.e., we created a Service Oriented Architecture (SOA) where ProjectIT-Requirements provides services to both our tools.

Regarding to the collaborative environment's usability, namely the requirements specifications editor, it should be AJAX-enabled to support WYSIWYG features. It should be highly interactive and the user shouldn't need to explicitly refresh the web page or wait for a batch process to validate the current specification and populate the multi-view controls that enrich the current page. Instead, after a configurable timeout, the content is submitted and the page is refreshed automatically, taking advantage of the full potential of the requirements underlying model and the NLP components. With a NLP framework [21, 7], our Wiki system can support context-aware auto-completion, where related terms are displayed in order to reduce ambiguity. Another important feature is the support of import/export operations from/to an office suite, namely Microsoft Office.

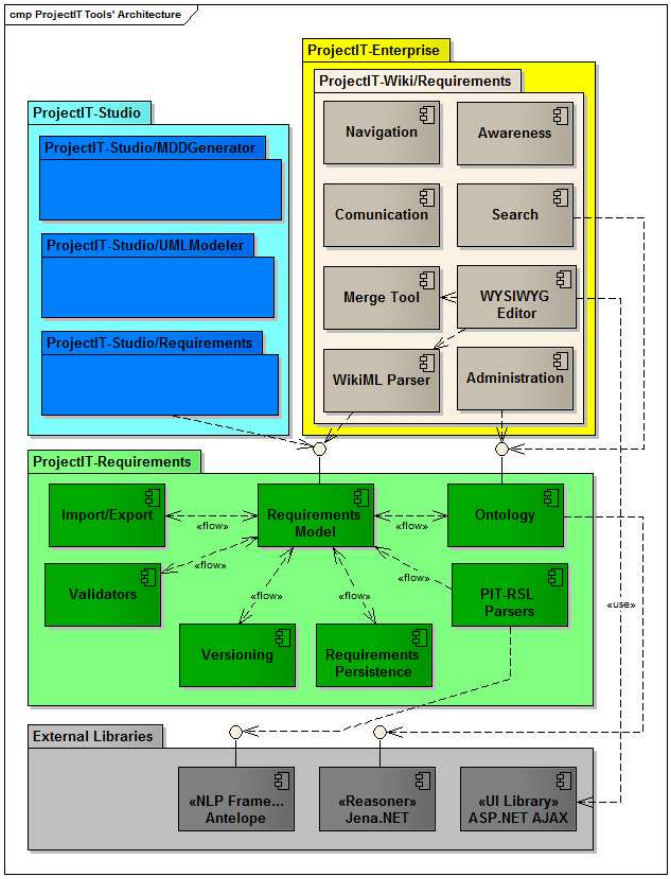


**Figure 5: ProjectIT tools' architecture.**

## 4. RELATED WORK

There are academic and industrial projects that emphasize the benefits of using a Wiki-based approach to support an active stakeholders' participation during Requirements Engineering [9] tasks. Wikis have the potential to leverage its participative component: wikis are a lightweight approach to documentation, and easier to use and tailor than proprietary requirements management tools [9].

Decker et al. [9] present a proposal for a basic document structure for Wiki-based Requirements Engineering based on well-known and adopted use case templates. They introduce Wikis as general-purpose and flexible documentation platforms, with the ability to be extended in order to capture additional information. Wikis can be adapted to cover most of the organization-specific needs of an IT project.

Aguiar et al. [1] present Wiki systems as effective documentation platforms for storing and integrating heterogeneous artifacts (from multiple sources) associated with a software development project. Documentation plays a central role in software development activities: all artifacts produced during the software life-cycle require continuous review and modifications to preserve their semantic consistency throughout the process. Moreover, documentation should preserve the rational behind the decision-making process during the conceptualization phases.

Witte et al. [27] present 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 since, by applying NLP techniques, one can automatically obtain semantic metadata, which can be further used to bootstrap a Semantic Wiki's ontology by populating it.

Buffa et al. [4] present Wikis as social web sites. While growing they begin to present some problems. Most of these issues can be solved with the addition of semantic information. SweetWiki assumptions stress that the social aspects are important and cannot be neglected, since they are crucial to increase content sharing. However, Semantic Wikis, such as SweetWiki, have a drawback: the semantic information is embedded within the pages themselves.

Kuhn' project [17, 18] appears as the fusion of the previously mentioned research projects, i.e., 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, ACE is neither domain-specific, nor presents rule-based extensibility features like ProjectIT-RSL. The major drawback of this project is the lack of focus on semantic validation.

Xiao et al. [28] developed a Wiki-based software development environment for both project coordination activities and also source code and documentation repository. Additionally, it extends Wiki standard platform with development functionalities (e.g., compiling, executing, and debugging), which are only found in traditional desktop-based IDEs. Their vision is supported by the assumption that writing a Wiki page is the same as writing a source code file and its documentation. This project's goal is not to replace existent desktop-based IDEs but, instead, to reduce the barriers of peripheral open source developers.

Our vision appears as a natural sequence of our previous work and its feasibility is corroborated by the orchestration of several other research projects, namely natural language integration with a Wiki system [27], a IDE-like development environment [28], and a Semantic Wiki with controlled natural language [17, 18]. We tried to be aligned with Wiki design principles [10] and address most of Wiki problems [20]. As Whitehead [25] points out, the future directions for collaboration in Software Engineering include 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.

Regarding content structure, we use the *WikiPageTemplate* concept to enforce the requirements document structure. Our approach is aligned with the work presented by Iorio et al. [14], where they introduce the concept of light constraint to directly encode community best practices and peculiar issues of domain-specific content. We agree with the justification that, although the concept *per se* seems to contradict the Wiki philosophy, it is useful to enforce well-formedness conditions on specific page contents and should be employed in order to improve quality of the requirements specifications. On the contrary, we think that, despite being the most suitable approach for ontology engineering, Kuhn's approach [17, 18] is not adequate for Requirements Engineering, namely because it doesn't make sense to force a non-technical user to explicitly differentiate between formal and informal statements during the specification process. We advocate that the system should adapt itself to the user, and not the opposite. To ensure requirements' reuse and documentation the user must obey only to the predefined template structure. However, the user should be able to insert free-text input and see if the system understands these inputs or at least partially captures the intended meaning of the requirements specifications that the user wrote.

Finally, concerning semantics, Kuhn's work [17, 18] is noteworthy regarding this paper's context. Albeit having different goals, it seeks to minimize the learning phase of non-technical users by using a controlled natural language: it fosters communication with natural language's higher familiarity and expressiveness. The goal is to delegate on final users and business people the responsibility of directly specifying what they want, because the overall quality or value of information is judged by themselves (the domain experts). 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 is that Kuhn's work only ensures syntactically and grammatically correctness, but does not addresses semantic validation.

## 5. CONCLUSIONS

This paper presents the design of a tailored Wiki engine for supporting Requirements Engineering activities, particularly for requirements specification with a "controlled natural language" and automatic validation with NLP techniques. We present and argue the evidence of feasibility of our proposal, not only based on our previous work, but also

supported by related work. Furthermore, we conduct an analysis of the state of the art of Wiki systems and their extensions, namely Semantic Wikis and Wikis embedded with NLP mechanisms. They all support our vision of creating a Wiki-based tool for dealing with Requirements Engineering activities. However, in the near future, we intend to conduct some case studies to prove our ideas and validate the respective tools.

Finally, there are still open issues that are not addressed by our current work, namely concerning social issues or intelligent analysis of requirements documents. A successful implementation of a Wiki relies on several social conditions, which sometimes are not met by the target organizational culture [4, 19]. In the future, we plan to analyze what are the criteria that ensure that a Wiki fits into the culture of the project or organization, and also define extensions to support social network analysis through the introspection of working methods and communication channels. Another feature to explore in the near future is the automatic summarizing of requirements documents, providing outlines of their content [27].

# 6. REFERENCES


[1] A. Aguiar and G. David. Wikiwiki weaving heterogeneous software artifacts. In *WikiSym '05: Proceedings of the 2005 international symposium on Wikis*, pages 67–74, New York, NY, USA, 2005. ACM.

[2] A. Ankolekar, M. Krötzsch, T. Tran, and D. Vrandecic. The two cultures: mashing up web 2.0 and the semantic web. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 825–834, New York, NY, USA, 2007. ACM.

[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*, pages 61–68. IEEE Computer Society Press, 1976.

[4] M. Buffa, F. Gandon, G. Ereteo, P. Sander, and C. Faron. Sweetwiki: A semantic wiki. *Web Semantics*, 6(1):84–97, 2008.

[5] CosmoCode. Wikimatrix: compare them all. Retrieved from `http://www.wikimatrix.org/`.

[6] csharp source.net. Open source wiki engines in c#. Retrieved from `http://csharp-source.net/open-source/wiki-engines/`.

[7] H. Cunningham, K. Bontcheva, V. Tablan, and D. Maynard. Gate, a general architecture for text engineering. Retrieved from `http://gate.ac.uk/`.

[8] W. Cunningham. Wiki design principles, March 2008. Retrieved from `http://c2.com/cgi/wiki?WikiDesignPrinciples`.

[9] 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.

[10] A. V. Deursen and E. Visser. The reengineering wiki. In *CSMR '02: Proceedings of the Sixth European Conference on Software Maintenance and Reengineering*, page 217, Washington, DC, USA, 2002. IEEE Computer Society.

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

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

[13] M. Hepp, D. Bachlechner, and K. Siorpaes. Ontowiki: community-driven ontology engineering and ontology usage based on wikis. In *WikiSym '06: Proceedings of the 2006 international symposium on Wikis*, pages 143–144, New York, NY, USA, 2006. ACM.

[14] A. D. Iorio and S. Zacchiroli. Constrained wiki: an oxymoron? In *WikiSym '06: Proceedings of the 2006 international symposium on Wikis*, pages 89–98, New York, NY, USA, 2006. ACM.

[15] M. Junghans, D. Riehle, R. Gurram, M. Kaiser, M. Lopes, and U. Yalcinalp. An ebnf grammar for wiki creole 1.0. *SIGWEB Newsletter*, 2007(Winter), 2007.

[16] A. Kittur, B. Suh, B. A. Pendleton, and E. H. Chi. He says, she says: conflict and coordination in wikipedia. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 453–462, New York, NY, USA, 2007. ACM.

[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] T. Kuhn. Acewiki: Collaborative ontology management in controlled natural language. In *3rd Semantic Wiki Workshop*. CEUR Workshop Proceedings, 2008.

[19] P. Louridas. Using wikis in software development. *IEEE Software*, 23(2):88–91, 2006.

[20] S. Murugesan. Understanding web 2.0. *IT Professional*, 9(4):34–41, 2007.

[21] Proxem. Antelope, advanced natural language object-oriented processing environment. Retrieved from `http://www.proxem.com/`.

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

[23] 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*, 1(6):217–314, December 2007.

[24] C. Videira, D. Ferreira, and A. Silva. A Linguistic Patterns Approach for Requirements Specification. In *EUROMICRO '06: Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 302–309, Washington, DC, USA, 2006. IEEE Computer Society.

[25] J. Whitehead. Collaboration in software engineering: A roadmap. In *FOSE '07: 2007 Future of Software Engineering*, pages 214–225, Washington, DC, USA, 2007. IEEE Computer Society.

[26] WikiCreole. NativeCreole. Retrieved from `http://www.wikicreole.org/wiki/NativeCreole/`.

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

[28] W. Xiao, C. Chi, and M. Yang. On-line collaborative software development via wiki. In *WikiSym '07: Proceedings of the 2007 international symposium on Wikis*, pages 177–183, New York, NY, USA, 2007. ACM.