# Wiki-Based Stakeholder Participation in Requirements Engineering

**Björn Decker, Eric Ras, Jörg Rech, Pascal Jaubert, and Marco Rieth,**
*Fraunhofer Institute for Experimental Software Engineering*

Wikis offer a flexible platform for asynchronous collaborative support of active stakeholder participation in requirements engineering.

Research has shown that, in projects with high uncertainty, stakeholders' participation improves the quality of requirements products.[1] Particularly in distributed environments, participative requirements engineering needs a platform that can support effective, efficient collaboration among an often large number of diverse stakeholders. The platform must account for the following challenges concerning stakeholders:

- different perspectives on the system under development;
- different backgrounds, which can cause communication problems;
- different objectives, which influence views on the requirements;
- different abilities to express requirements and document them using a technical platform; and
- different involvements—for example, some stakeholders are entitled to make decisions and others aren't.

It's important to address these challenges by fostering communication and decision making. In particular, project participants need a way to detect misunderstandings and conflicts and solve them as early as possible. This communication can be synchronous—through face-to-face meetings and videoconferencing, for example—or asynchronous, which is our focus here.

Requirements elicitation and documentation are complex activities. So, not only the requirements themselves but also the people involved and the means for managing the requirements will evolve during the project. For example, it might be necessary to add RE personnel, to document templates, or to change the requirements classification scheme. In summary, especially in participative RE, the underlying platform as well the RE approach must be flexible.

As the well-known Wikipedia shows, wikis provide a flexible platform for asynchronous collaboration to create content in general.[2] In this article, we investigate how to adapt this approach to support active stakeholder participation in RE. We include a document structure for


0740-7459/07/$25.00 © 2007 IEEE

**Table 1**

**Pros and cons of tools commonly used in requirements engineering**

| | Plain office suite/text documents | General collaborative tools | Dedicated RE tools |
|---|---|---|---|
| Description | ■ Requirements are stored in office software suites and word processors such as Microsoft Office, Open Office, and Framemaker | ■ Requirements are stored in collaborative tools such as Microsoft's Sharepoint server or Google's Writely | ■ Requirements are stored in dedicated RE tools such as Telelogic's Doors or Borland's CaliberRM |
| Pros | ■ Support for structuring requirements in sections<br>■ Low cost and widespread availability (every organization has it by default)<br>■ Also applicable in other development phases, such as testing | ■ Support for collaboration<br>■ Support for grouping requirements into individual documents (for example, for specific releases)<br>■ Support for structuring requirements in sections<br>■ Also applicable in other development phases, such as testing | ■ Support for collaboration<br>■ Support for grouping requirements into individual documents<br>■ Support for structuring requirements in sections<br>■ Support for versioning and baselining of requirements |
| Cons | ■ Collaboration chaos from concurrent changes by different stakeholders or late exchanges via email<br>■ Distribution chaos if requirements are distributed across several documents; high risk that document links will break<br>■ Untyped links (link semantics can't be captured)<br>■ No explicit versioning and baselining of requirements | ■ Distribution chaos if requirements are distributed across several documents; high risk that document links will break<br>■ Untyped links<br>■ No explicit versioning and baselining of requirements | ■ Specialized tools that aren't optimized for nontechnical users<br>■ High cost, especially for small and medium-sized enterprises that might need a license for every stakeholder |

wiki-based RE and discuss potential problems and solutions based on our experience.

## Using wikis in requirements engineering

Several studies confirm that small and medium-sized enterprises use only a few tools in their RE activities,[3–5] mostly general tools such as office suites or Web sites. Table 1 summarizes characteristics of the most frequently used RE tools.

Instead of these tools, we're using wikis as a base technology to create software project requirements with a variety of stakeholders. In general, wikis are a lightweight approach to documentation—more powerful than plain office suites or collaborative tools and easier to use and tailor than proprietary RE tools. Panagiotis Louridas has characterized wikis as both a technical solution for collaborative editing of Web pages and an underlying philosophy of how this collaboration should occur.[2] Both aspects are beneficial in participative RE.

Two main technical features of wikis are particularly helpful:

- Easy page linking reduces redundancy by making it easier to link content than to copy a page.
- Page-history capture, which most wiki software offers, provides a foundation for requirements traceability on a per-document basis.

These technical features support the underlying wiki philosophy: because new users can easily learn wikis and because mistakes are easily corrected by retrieving previous document versions, project management can involve new stakeholders when needed, without putting the achieved results at risk. Wikis foster an evolutionary mind-set by quickly achieving a document status that stakeholders can work with and improve as needed. The wiki makes it easy to integrate different stakeholders' views and so supports the learning curve for requirements during a project. Finally, wikis provide only basic features for collaboration and leave the way they're actually used to the people using them. This flexibility supports RE process changes based on the agreement of the participating stakeholders.

### Managing communication

To promote communication, a moderator (the requirements engineer or project manager) must first provide a general structure for requirements topics, templates for contributing

Wikis offer features to detect not only expressed but also unexpressed conflicts.

content, and overviews. On the basis of this structure, the moderator can assign requirements to stakeholders. This approach takes care of stakeholders' differing perspectives on requirements and supports quick access to the requirement documents—that is, pages—particularly for part-time participants in the project. You can structure the information by using the wiki's classification features or by referencing pages that contain links to relevant information.

To further stimulate discussion, the wiki can present undefined and newly added pages on a special page or the wiki's home page. This increases the possibility that stakeholders will create the needed requirements or review those pages.

Additionally, the stakeholders must agree on rules for using the wiki and for reaching decisions about requirements. The wiki must document these rules in a prominent place, such as a document linked from the home page, to help new stakeholders get on board quickly. For example, the stakeholders must agree on how they inform each other of a substantial change to a requirement description.

However, wiki communication can also be overstimulated. So, the moderator must analyze whether the discussions are focused. Most wikis provide overviews to support this task. The overviews are indicators, although a large number of overview entries doesn't necessarily imply an unfocused discussion. The moderator must analyze the discussions to make this determination. For example, a new-pages overview indicates the pace of requirements being defined and any potential overlaps. An undefined-pages overview similarly helps to indicate unfocused or redundant entries by listing pages that are linked from other pages but aren't yet defined. A third indicator of unfocused discussion is insufficient linking of requirements among each other. A low number of links between pages indicates that requirements pages aren't well integrated with each other. Orphan pages—that is, pages that aren't linked to other pages—are a similar indicator of this problem.

Analyzing such indicators helps the moderator determine whether to change the templates, classification scheme, rules, or current work assignments to serve the project better. The moderator must pay particular attention to two things: first, ensuring that stakeholders know about changes to templates, classifications, and work assignments; second, checking whether stakeholders are acting according to these changes.

### Handling conflicts

Managing requirements elicitation effectively requires detecting and handling conflicts and misunderstandings. These problems can arise from the stakeholders' different backgrounds and objectives, which can generate contradicting requirements or conflicts based on limited implementation resources. Wikis don't provide a direct means to handle these problems. Nevertheless, they provide ways at least to indicate misunderstandings and conflicts. For example, when stakeholders review other stakeholders' requirements, they can tag conflicts and misunderstandings and solve them via direct discussion on a separate page. These tags can also become the basis of overviews of current problems, thus further guiding the requirements process.

In addition to expressed conflicts, wikis offer features to detect unexpressed conflicts. For example, frequent changes to pages—so-called edit wars—are an indicator of conflicts. An old-pages overview can also indicate conflicts: pages not edited for a long time might indicate stakeholders silently withdrawing from the project to avoid conflicts. Information on stakeholders withdrawing from a project is also available in activity overviews for stakeholders, if the organization's privacy policy allows this evaluation. Finally, the project manager should inspect the content of critical requirements from time to time.

## A document structure for wiki-based RE

An adequate document structure for the stored content improves an RE wiki's usefulness considerably. A clear document structure

- helps avoid uncontrolled content sprawl,
- adds semantics to the wiki, thus alleviating the problems of untyped links and lack of information about the elicitation process's status, and
- makes it easier to detect and solve conflicts and misunderstandings.

We developed the document structure in figure 1 to address these needs. We derived a min-

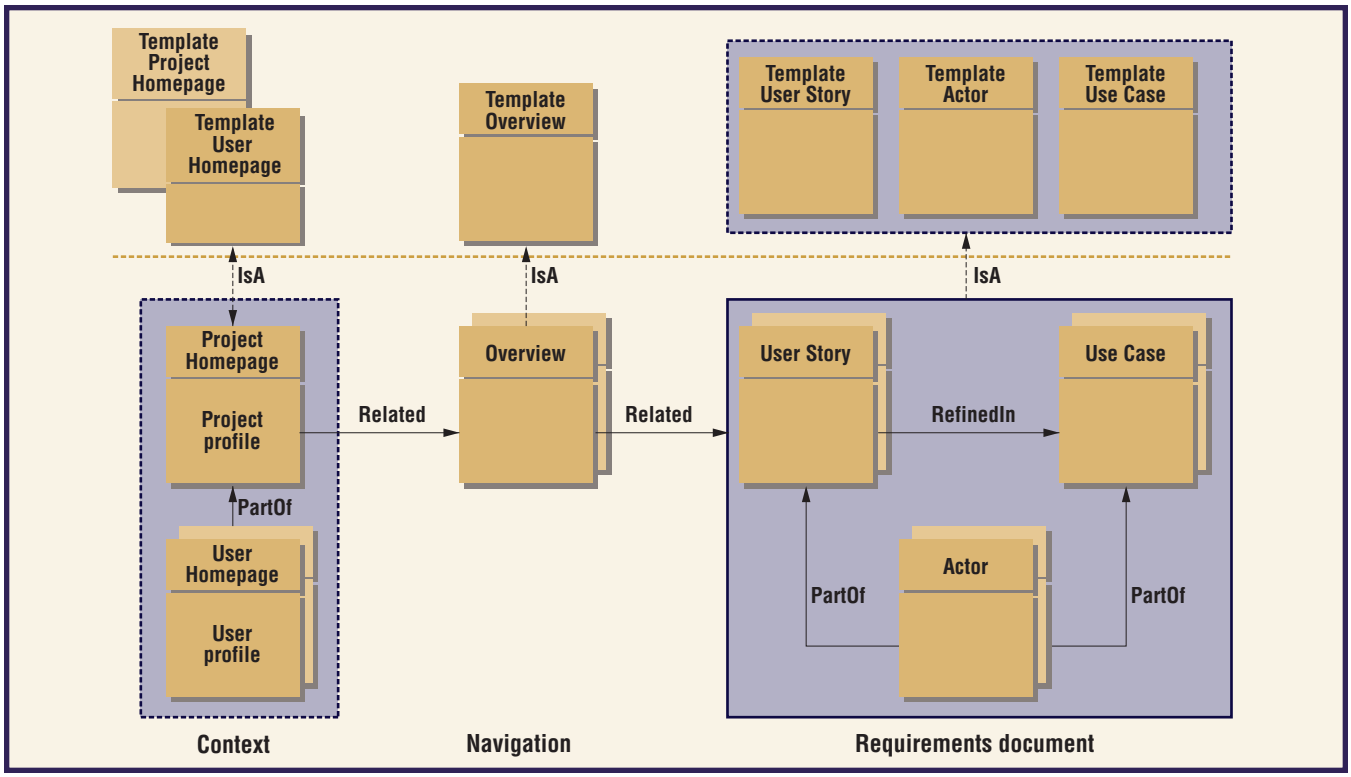



**Figure 1. The basic document structure for wiki-based requirements engineering.**

imum document set from published examples (specifically, Alistair Cockburn's use case template,[6] the Volere document template (www.volere.co.uk), and the ReadySET requirements templates (http://readyset.tigris.org). We evolved the set further in several application projects. Typed links relate the different document types to each other.

Project Homepages and User Homepages help improve collaboration and the system's ease of use. They give users information about how to use the wiki and who the stakeholders are.

- A Project Homepage is the entry page to the requirements. It contains information about the project such as its mission and links to requirements overviews. Furthermore, it includes the initial information for people joining the project, helping new stakeholders become quickly involved.
- A User Homepage contains information about the people involved in the project. Each User Homepage contains the person's project role and contact information. By linking to stakeholders' user homepages, the moderator or other stakeholders can define the stakeholders responsible for a certain requirement. Furthermore, other project team members can contact stakeholders.

Overview pages increase understanding of the overall requirements-elicitation process. They guide stakeholders to the requirements and through the requirements-elicitation process. They include a list of the requirements documents according to their document type. For example, they list both completed and unfinished requirements. They can point to conflicts and misunderstandings that stakeholders must address. Therefore, they establish landmarks for navigation and for opening discussions between different stakeholders.

User Story, Actor, and Use Case pages help the user elicit and document requirements according to appropriate templates. The use of these templates ensures consistency between documents of the same type and in the relationships among different types.

- A User Story is a short prose specification of system interactions. This style lets stakeholders specify requirements in a natural way. It should include at least the actors involved and the responsible author.
- An Actor defines the roles involved in a use case or a user story. Stakeholders can review the documents of the actor groups they belong to. The *backlink-feature*—an overview that lists all pages referencing

**Extensions of the wiki document structure can capture information such as GUI designs and business rules.**

the current page—then lets stakeholders identify further relevant documents.

- A Use Case refines a User Story in a structured representation. The actual structure depends on the project's needs. Each use case contains a sequence of actions undertaken by the actors involved in this use case. On the basis of this sequence, stakeholders can then check whether the derived use case reflects the underlying user story.

These documents are intended to be a starting point for requirements elicitation. The moderator can extend the structure to the project's actual needs. Projects need this extension capability; in Cockburn's "Hub and Spoke" model,[6] for instance, use cases cover only about one-third of the requirements documentation. Because wikis are a general-purpose documentation platform, extensions of the document structure can capture additional information, such as GUI designs, business rules, and nonfunctional requirements. In general, the extensions are of two types: those that add sections to an existing template and those that integrate new templates into the document structure. For example, stakeholders can state performance issues directly in a use case subsection, or they can collect general performance information on a page of the performance issues type, which a use case document can then reference.

## Application experience

We've used our wiki approach in the context of industrial and academic projects. The initial project, in which we developed and applied most of the findings we've presented here, was a German collaborative research project called RISE (Reuse in Software Engineering, http://rise-it.info). This project ran from 2003 to 2005 and involved 12 people with different roles (researchers, implementers, and managers) from five organizations. The project used the wiki to elicit requirements, develop an initial architecture, and assign development tasks based on these requirements. The requirements phase produced descriptions of 20 high-level use cases and five actors. As testimony to the wiki's usefulness, the project's two industrial partners initiated software engineering wikis in their organizations that are still in use (as of November 2006).

We've elaborated these ideas in another project, where we're developing a wiki system called SOP (Software Organization Platform). The project itself is using this wiki for its development. The SOP-Wiki implementation is based on mediawiki (www.mediawiki.org), a free software wiki package originally written for Wikipedia. The SOP development project currently has about nine subprojects, each developing between five and 10 use cases, with a staff of three project managers and eight developers (students). Most developers join the SOP project only during their subproject's runtime. The use cases provide a substantial communication medium between the developers and the managing team. Furthermore, SOP uses the wiki for other software engineering artifacts, such as project management and installation issues.

The SOP-Wiki is currently being used in BelAmi (Bilateral German-Hungarian Collaboration Project on Ambient Intelligent Systems, www.belami-project.org), a project aimed at developing innovative technologies in the area of ambient intelligence. This project has a staff of about 20 people from six subprojects from four different organizations. The project uses the SOP-Wiki to document the use cases, interfaces, and technical implementations of the underlying service-oriented architecture.

Finally, 14 Technical University of Kaiserslautern students recently used the SOP-Wiki in a capstone project. The evaluation performed in this practical course showed that the students gained insights they wouldn't have obtained using verbal communication. Furthermore, the students, with one exception, stated that the SOP-Wiki supported development activities. Nine students stated that RE benefited from the SOP-Wiki. However, only six stated that it actually saved RE effort. This rating mainly reflected the missing document export, which was not yet available during this project.

The experience gained in these projects supports our view that wikis are indeed useful for stakeholder collaboration, as well as for grouping and structuring requirements.

## Wiki problems—and how to solve them

While applying RE wikis in these projects, we experienced several challenges. The SOP-Wiki system reflects our solutions to them.

### Remembering page names

To establish links, a user must remember the

name of the page to be linked, most often using an overview of the existing pages. The search for the page's title can interrupt editing. Using naming conventions partially addresses this problem by making it easier to remember links.

In the SOP-Wiki, we've also developed a linking support feature. As figure 2 shows, we present links to other documents on the basis of a document's template. This lets a stakeholder easily select relevant pages.

## Versioning across several pages

Because wikis provide versioning only on the basis of a single page, defining releases based on a certain status of several pages (for example, a baseline) can become laborious. Most wikis provide a feature called *permalink*, which lets you establish a link to a certain version of a page. However, a release contains links to multiple pages, so you have to collect these permalinks manually.

To overcome these disadvantages, we created two features that support cross-page versioning. The first feature combines two functions: *link harvester* and *freeze*. The link harvester provides a list of all links on a certain page (in particular, from category overviews). We then place these links on a page representing a release and freeze them—that is, the feature transforms all links into permalinks.

The second feature, *version tagging*, provides an overview that lets the user select wiki pages to assign a version tag. The version tag indicates that a page's current version is official or accepted. The version-tagging feature supports the selection of subsets of wiki pages by presenting the pages grouped according to the categories they belong to. When this official version changes, the wiki displays a notice to the user. Furthermore, the overview of pages belonging to a certain version tag links both the current versions of the pages and the versions when the version tag was assigned. To support multiple baselines, a user can assign a version of a page to several version tags.

## Exporting the content of wiki pages

A third extension lets the user export wiki content to self-contained documents. More precisely, a user can define pages containing links that the wiki then transforms into an Open Office document. For example, the project manager can use this extension to create a contract document containing the created requirements.

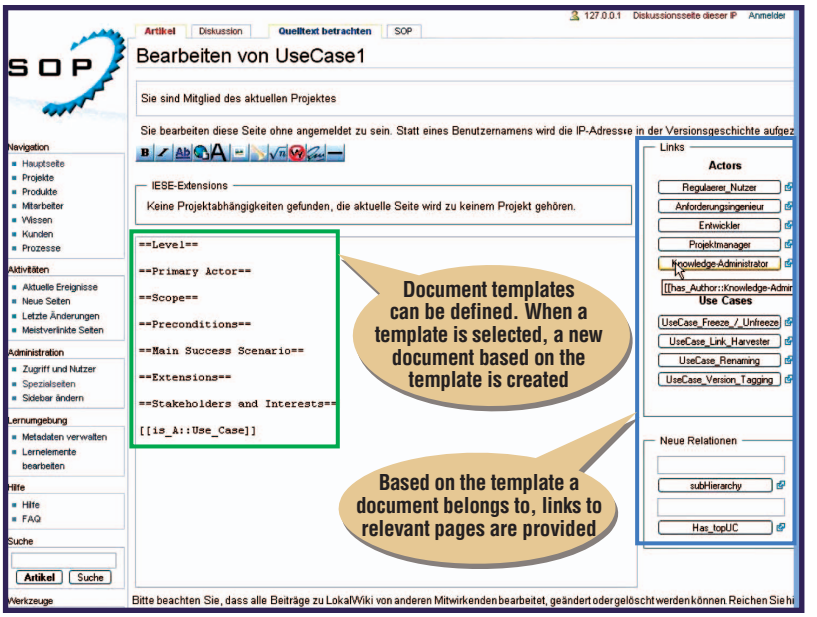



**Figure 2. Linking support. The buttons on the right provide appropriate links to the template instance (document) shown on the left side of the screen.**

This extension motivates stakeholders to use wikis by letting them derive work products from its content.

## Missing features for restructuring content

Most wikis support page structuring only by categories or by prefixes such as namespaces. In other words, they can't express metadata or relational semantics. However, a user needs this additional information to fully understand the relations between requirements. *Semantic wikis* let users add this semantic information to the pages. Within the SOP-Wiki, we use the Semantic MediaWiki extension for MediaWiki (http://ontoworld.org/wiki/Semantic_MediaWiki). This extension lets users add properties to pages and define typed links between pages. We use this feature to express relations between templates and documents ("isA") or between documents, as in the "refined" relationship between an informal and a formal use case in figure 1.

Users get two direct benefits from adding metadata. First, they can create overviews based on the metadata; second, the linking support feature uses this data to facilitate link creation. Furthermore, this information forms the foundation of consistency checks: if a document is changed, a user can check other linked documents to see whether they're affected.

## Table 2

### Pros and cons of wikis used in requirements engineering

| | General wikis | RE-specific wiki (for example, Software Operations Platform wiki) |
|---|---|---|
| Description | ■ Requirements are stored in general-purpose wiki systems | ■ Requirements are stored in RE-specific semantic wikis such as SOP |
| Pros | ■ Support for collaboration<br>■ Support for grouping requirements into individual documents (for example, for specific releases)<br>■ Support for requirements versioning<br>■ Low cost (mostly open source software)<br>■ No tool licenses required<br>■ Also applicable in other development phases, such as testing | ■ Support for collaboration<br>■ Support for grouping requirements into individual documents<br>■ Support for requirements versioning and baselining<br>■ Low cost (mostly open source software)<br>■ No tool licenses required<br>■ Also applicable in other development phases, such as testing |
| Cons | ■ Untyped links (because regular wikis can't capture the nature of a link between documents)<br>■ No explicit baselining of requirements; versioning content across several pages is difficult<br>■ Users must remember page names to be linked<br>■ Difficult to export page content<br>■ Missing features for restructuring content, particularly for classifying and reclassifying pages<br>■ Missing replication of wiki content | ■ Increased complexity of wiki syntax to denominate metadata<br>■ Missing replication of wiki content |

### Multiple page (re)classification

A fifth problem when using wikis is classification and reclassification of multiple pages. Each classification must reflect changes in the requirements classification scheme that might occur in a project. For example, to create an overview for a specific stakeholder requires editing each page in this category.

We developed two extensions for classification and reclassification tasks. Text-Cast lets users add text—for example, that the page belongs to a certain category—to several pages at once. RegExp Cast offers a similar functionality that supports searching and replacing texts on several pages at once.

### Missing replication of wiki content

We experienced one problem that we can't yet solve: the missing replication of wiki content might lead to problems in settings where project members need to work offline from time to time. FlexWiki (www.flexwiki.org) has a replication feature, but we experienced practical problems when we wanted to use this offline feature—for example, insufficient support for linking.

Table 2 extends table 1 by listing the pros and cons of wiki-based systems. On the basis of our positive applications' experience, we plan to elaborate our wiki solution for participative RE. One direction is to investigate acceptance in more detail. We also want to compare the quality of the requirements created with wikis with conventionally created requirements—for example, with regard to specification completeness. To support this research, the SOP project's next steps will deal with identifying inconsistencies among the requirements—for example, an actor that's not used in any of the requirements documents. This is similar to the simple consistency checks—say, for orphaned pages—already available in wikis. In the long term, we plan to use semantic information to gradually connect and formalize requirements. We also plan to make our solution available to a larger group of users.


### Acknowledgments

We thank Michael Eisenbarth for sharing his requirements engineering views. Furthermore, we thank the reviewers and editors of this special issue for providing us valuable suggestions for improving this article.

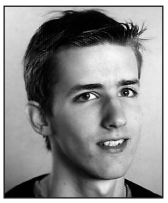
## About the Authors

**Björn Decker** is a project manager and scientist at the Fraunhofer Institute for Experimental Software Engineering in Kaiserslautern. His research interests are in experience management, particularly the collaborative maintenance of online repositories and the use of social software in software engineering. He received the MS (Diplom) in computer science from the University of Kaiserslautern. Contact him at Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany; bjoern.decker@iese.fraunhofer.de.

**Eric Ras** is a senior scientist at the Fraunhofer Institute for Experimental Software Engineering in Kaiserslautern and scientific coordinator of the University of Kaiserslautern's international distance learning program, Software Engineering for Engineers. His research interests are in learning-material production, vocational training methods, software patterns, and experience management. He received the MS (Diplom) in computer science from the University of Kaiserslautern. Contact him at Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany; eric.ras@iese.fraunhofer.de.

**Jörg Rech** is a project manager and senior scientist at the Fraunhofer Institute for Experimental Software Engineering in Kaiserslautern. His research mainly concerns software antipatterns, software patterns, defect discovery, software mining, software retrieval, automated software reuse, software analysis, and knowledge management. He received the MS (Diplom) in computer science with a minor in electrical science from the University of Kaiserslautern. He is also the speaker of the Gesellschaft für Informatik working group on architectural and design patterns. Contact him at Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany; joerg.rech@iese.fraunhofer.de.

**Pascal Jaubert** is a student employee at the Fraunhofer Institute for Experimental Engineering and a student of digital media at the Kaiserslautern University of Applied Sciences. His research interests include modifying wikis to support software engineering documentation, customizing platforms based on J2EE, and visualizing social networks. Contact him at Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany; pascal.jaubert@iese.fraunhofer.de.



**Marco Rieth** is a student at the University of Applied Science in Zweibrücken, Germany, where he studies applied computer science. He just completed an internship at Fraunhofer Institute for Experimental Software Engineering in Kaiserslautern, where his research focused on using wikis to support semantic-based software engineering documentation. Contact him at Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany; marco.rieth@iese.fraunhofer.de.

## References


1. K. El Emam, S. Quintin, and N.H. Madhavji, "User Participation in the Requirements Engineering Process: An Empirical Study," *Requirements Eng.*, vol. 1, no. 1, 1996, pp. 4–26.
2. P. Louridas, "Using Wikis in Software Development," *IEEE Software*, vol. 23, no. 2, 2006, pp. 88–91.
3. U. Nikula, J. Sajaniemi, and H. Kälviäinen, *A State-of-the-Practice Survey on Requirements Engineering in Small- and Medium-Sized Enterprises*, research report, Telecom Business Research Center Lappeenranta, 2000.
4. H.F. Hofmann and F. Lehner, "Requirements Engineering as a Success Factor in Software Projects," *IEEE Software*, vol. 18, no. 4, 2001, pp. 58–66.
5. J. Rech, B. Decker, and E. Ras, *Intelligente Assistenz in der Softwareentwicklung 2006: Zusammenfassung der Ergebnisse* [Intelligent Assistance in Software Development 2006: Summary of Results], Kaiserslautern IESE tech. report IESE-045.06/D, 2006 (in German).
6. A. Cockburn, *Writing Effective Use Cases*, 1st ed., Addison-Wesley Longman, 2000.