Enhancing the Development Process with Wikified Widgets

Gary Gotchel
Regis University

Follow this and additional works at: https://epublications.regis.edu/theses
Part of the Computer Sciences Commons

Recommended Citation
https://epublications.regis.edu/theses/146
Regis University
College for Professional Studies Graduate Programs
Final Project/Thesis

Disclaimer

Use of the materials available in the Regis University Thesis Collection ("Collection") is limited and restricted to those users who agree to comply with the following terms of use. Regis University reserves the right to deny access to the Collection to any person who violates these terms of use or who seeks to or does alter, avoid or supersede the functional conditions, restrictions and limitations of the Collection.

The site may be used only for lawful purposes. The user is solely responsible for knowing and adhering to any and all applicable laws, rules, and regulations relating or pertaining to use of the Collection.

All content in this Collection is owned by and subject to the exclusive control of Regis University and the authors of the materials. It is available only for research purposes and may not be used in violation of copyright laws or for unlawful purposes. The materials may not be downloaded in whole or in part without permission of the copyright holder or as otherwise authorized in the “fair use” standards of the U.S. copyright laws and regulations.
Enhancing the Software Development Process

with Wikified Widgets

by

Gary Gotchel
Gary.Gotchel@web.de

A Thesis/Practicum Report submitted in partial fulfillment of the requirements for the degree of
Master of Science in Software and Information Science

School of Computer and Information Sciences
College for Professional Studies
Regis University
Denver, Colorado

August 29, 2008
Abstract

Wikis are now merging with other applications. Structured wikis, also known as application wikis, have evolved in recent years. This second generation has embedded applications into the wiki, which are used to structure access to the wiki repository's otherwise unstructured content. This paper investigates the other possible approach to merging wikis with applications. It researches the utility of embedding wiki functionality into common applications. Utility will be measured by the benefits to the software development process offered by this approach. I introduce a new term, wikified widget, to describe a GUI component that provides structured access to a wiki repository from within non-wiki applications. Merging wiki and applications in this manner provides immediate and contextual access to a wiki repository from standard applications. There is great potential here, because this builds a new path for communication between all users of an application and its designers. Increased, and constant collaboration between these groups supports rapid iterations of software development, and should improve the dismally low success rate of software projects.
I would like to thank my thesis advisor, Dr. Douglas Hart, for his guidance and advice, and my family, for their support and patience.

This research paper investigates one way that humans collaborate using technology. It relied heavily on software provided by the many contributors of the open-source community, and was inspired by the success of their collaborative genius.
Table of Contents

1. Chapter 1 Introduction......................................................................................................................1
   1.1. Problem Statement.....................................................................................................................1
   1.2. Thesis Statement.......................................................................................................................3
   1.3. Project Relevance......................................................................................................................3
   1.4. Project Scope............................................................................................................................4
   1.5. Project Limitations...................................................................................................................5

2. Chapter 2 – Review of Literature and Research..............................................................................6
   2.1. Definition of Terms..................................................................................................................6
   2.2. Customer Communication.......................................................................................................8
       2.2.1. Previous Studies..............................................................................................................8
       2.2.2. Potential Contributions...................................................................................................9
   2.3. Hindrances to Using a Wiki.....................................................................................................10
       2.3.1. Previous Studies..............................................................................................................10
       2.3.2. Potential Contributions...................................................................................................11
   2.4. Wikis in a Corporate Environment........................................................................................11
       2.4.1. Previous Studies..............................................................................................................12
       2.4.2. Potential Contributions...................................................................................................13
   2.5. Unstructured Nature of Wikis................................................................................................15
       2.5.1. Previous Studies..............................................................................................................15
       2.5.2. Potential Contributions...................................................................................................16

3. Chapter 3 - Methodology..............................................................................................................17
   3.1. Overview................................................................................................................................17
       3.1.1. Choice of Methodology.................................................................................................17
       3.1.2. Questions Derived from Thesis Statement.....................................................................17
   3.2. Exposing Participants to Wikified Widgets...........................................................................18
       3.2.1. The Wikified Context Widget........................................................................................20
           3.2.1.a. Design Considerations.............................................................................................20
           3.2.1.b. Screen Layout.............................................................................................................22
           3.2.1.c. Context Dependent Invocation................................................................................23
           3.2.1.d. Edit Page Function..................................................................................................27
           3.2.1.e. Add Comment Function..........................................................................................29
       3.2.2. The Wikified Welcome Page Widget...............................................................................32
           3.2.2.a. Design Considerations.............................................................................................32
           3.2.2.b. Implementation...........................................................................................................33
       3.2.3. The Wikified All Comments Widget..............................................................................35
           3.2.3.a. Design Considerations.............................................................................................35
           3.2.3.b. Implementation...........................................................................................................36
       3.2.4. The Wikified Keyword Registry Widget..........................................................................38
           3.2.4.a. Design Considerations.............................................................................................38
           3.2.4.b. Implementation...........................................................................................................39
       3.2.5. Help Buttons and Pages.................................................................................................41
       3.2.6. Single Sign-on..................................................................................................................41
   3.3. Observations..............................................................................................................................42
   3.4. Followup Questionnaire..........................................................................................................45
4. Chapter 4 – Analysis and Results............................................................................................ 48
   4.1. Evaluation of Project Goals............................................................................................. 48
   4.2. Role of Project Variables in the Results........................................................................... 51
   4.3. Contributions to the Body of Knowledge...................................................................... 53
5. Chapter 5 – Conclusions and Next Steps............................................................................... 54
   5.1. Conclusions................................................................................................................... 54
   5.2. Next Steps.................................................................................................................... 54
6. References............................................................................................................................ 56
7. Appendix A- Software System Design.................................................................................... 60
   7.1. Architectural Goals......................................................................................................... 60
   7.2. Front-end Architecture................................................................................................. 60
       7.2.1. Front-end Widgets............................................................................................... 61
       7.2.2. Front-end Services............................................................................................. 62
   7.3. Back-end Architecture................................................................................................. 63
       7.3.1. Single Sign-on..................................................................................................... 63
       7.3.2. Back-end Services............................................................................................. 64
           7.3.2.a. Lightweight Remoting................................................................................. 65
8. Appendix B – User Questionnaire........................................................................................... 66
## List of Figures

- Wikified Context Widget overview .......................................................... 22
- Context determined by the currently active tooltip .................................. 24
- Relationship between tooltip text and wiki page title ............................ 25
- Demonstration of the default context of a screen .................................... 26
- Render mode ....................................................................................... 27
- Edit mode .......................................................................................... 28
- Opening page in a new browser window ............................................... 29
- Adding a comment ............................................................................. 30
- How comments are displayed ............................................................. 31
- Manually invoking the Wikified Welcome Page Widget ......................... 33
- Wikified Welcome Page Widget .......................................................... 34
- Invoking the All Comments Wikified Widget ......................................... 36
- The All Comments Wikified Widget ..................................................... 37
- Resorting the comments table ............................................................. 38
- Keyword Registry Button .................................................................. 40
- The Keyword Registry Widget ........................................................... 40
1. Chapter 1 Introduction

1.1. Problem Statement

The software development process has been, and remains to this day, an extraordinarily imperfect science. The purely sequential, waterfall model advocates a division of the software development process into a fixed sequence of project phases. Extensive planning plays a critical role, because after requirements have been specified, they may not significantly change. However, decades of experience has shown that even moderately-sized software projects can involve high degrees of uncertainty, and that agreed-upon requirements were often incomplete, or had been misunderstood. Globalization, and the growth of the Internet, has led to increased competition. Rapid adaptation to change has become more essential to corporate success than ever. The waterfall paradigm lacks the flexibility required by dynamic environments, and has fallen out of fashion.

In response, the IT industry has turned to iterative approaches, in which the sequence of steps in the development process can be revisited. Unlike the waterfall model, where problems and their solutions are stated presumptively, the iterative paradigm supports gradual and continuous discovery. But iteration alone has not guaranteed success, and the search for a more effective software development paradigm is ongoing. One such new paradigm that has been optimized for responsiveness to evolving requirements is agile methodology. The Manifesto for Agile Software Development, published in 2001, declares in few words the convictions of the agile movement. The authors of this manifest believe that responding to change is more important than following a plan. Iteration plays a central role, but so too does the ongoing dialog between between the users of a system and its designers. Customer collaboration, rather than contract negotiation, should be
emphasized (Grisham & Perry, 2005).

It is far from clear if the agile paradigm can become the industry standard for software
development. However, the merit of one of its principle tenets, that communication with users
needs to be emphasized, is hard to dispute. Research has shown that insufficient communication
between the end users of a system and its designers is a major reason for project failure(Ceschi,
Sillitti, Succi & De Panfilis, 2005). Tools that further the communication between users and the
development team, and support their collaboration, therefore contribute significantly to the
advancement of the software development process. Tools that enhance communication will retain
significant utility for the development process, regardless of the paradigm currently in vogue.

For the past several years, another important trend has accompanied the increasing emphasis on
iteration in the software development process. This trend is fueled by the recognition that powerful
synergies arise in social collaboration, and it has given us a new set of Internet-based tools known
collectively as Web 2.0. Perhaps the most prominent technology of this new, participative Web is
the wiki. Organizations already use this flexible tool for a wide range of purposes, and surely, many
novel applications are waiting to be discovered.

Wikis have been integrated into the software development process of many corporations, for
purposes ranging from simple brainstorming to project management. A wiki can serve as the
vehicle for communication between end-users and the development team that is considered so
essential to a successful iterative development process. Hence, two global trends, on-line social
collaboration, and increasingly iterative software development, have crossed paths at a class of
software known as wiki.
1.2. Thesis Statement
I investigate the convergence of these two trends in my practical research, and have posed this question:
Can the integration of wiki functionality into applications improve the communication between an application's users and the development team, thereby supporting the iterative model of contemporary software development?

1.3. Project Relevance
I believe that a new array of wiki-based tools can support rapid iterations of the software development life cycle. Such tools can enhance, or even make possible for the first time, communication between all users of an application and its development team. Because these tools are embedded into standard applications used daily, they are omnipresent to the user community. Users can submit feedback, and be queried from the development team, on any day and at any time. I shall refer to GUI components that have been integrated with applications in order to provide structured access to a wiki repository as “wikified widgets”. This is, to my knowledge, a new term, that until now, does not exist in the literature1.

Wikified widgets may be valuable to implementing Extreme Programming (XP), a cornerstone of agile methodology. XP's software development model recommends that the user community have an on-site presence with the development team in order to maintain a high level of communication.

1 A similar term, “wiki widgets”, is currently being used for a set of extensions to a wiki application, the popular MediaWiki open source product. However, that term refers only to widgets that run within the MediaWiki application.
Such a presence can be costly (Grisham & Perry, 2005), and wikified widgets can complement, or even offer an alternative to XP's on-site user presence.

Wikified widgets can provide a number of benefits to a corporation beyond the software development process. They can support knowledge management, since they can increase participation in a corporate wiki, which is often dismally low. And they can enable users to help each other, by sharing tips and experiences about an application through the application itself.

The distinction between wikis and other applications is blurring. Applications have invaded the previously unstructured world of wikis. A second generation of wikis, known as structured, or application wikis, has emerged in recent years. Applications running within such wikis provide structured access to the wiki's content. It has been predicted that soon, the reverse will occur, and that wikis will invade applications, and that the distinction between wikis and applications will disappear (Woods & Thoeny, 2007). My research investigates this latter aspect of wikis merging with applications. I have not found any other studies on this topic, and it may be the first study to investigate this prediction.

1.4. Project Scope
The participants of my research all work for the same corporation. They are the users, developers, and quality assurance personnel of a browser-based web application used by the management and clerical staff of the corporation's warehouses. I investigated this group's interaction with a set of four distinct widgets that provide access to a wiki repository from within this application. The widgets support two of the most essential features of wikis, adding comments and editing pages. Research was conducted during the early testing phase of the software development process.
1.5. Project Limitations

This study was conducted in one specific corporate environment. Other corporate contexts, or the public realm, were not considered. Only one class of software – a browser application – was used to host the widgets to be tested. Not all of the functionality available in traditional wikis was implemented in the widgets developed. For example, the functionality for deleting the pages or comments in a wiki was not implemented. The investigation was carried out during early testing of the application. Other phases, in which, for example, prototyping is conducted, or in which the software has reached maturity, were not investigated.
2. Chapter 2 – Review of Literature and Research

2.1. Definition of Terms

**AJAX.** A set of technologies used to create web applications that rival desktop applications in responsiveness. The technologies used are XML for data interchange, HTML and CSS for the application's interface, the XMLHttpRequest object for asynchronous communication with the server application, and JavaScript for dynamic interaction with the provided data. AJAX enables one to build applications that are part of the Web 2.0 revolution (Chaganti, 2007).

**First-generation wiki.** A term used to refer to wikis based on the original wiki model as created by Ward Cunningham. Simplicity is emphasized in first-generation wikis (Cunningham, 2002), and databases were not used to store the contents of the wiki repository. Access to such a wiki is said to be unstructured. In contrast, the second generation of wikis do provide structured access, and are therefore known as structured wikis.

**Google Web Toolkit.** A Java-based development environment for building AJAX applications using the Java language (Chaganti, 2007).

**Structured wiki.** A hybrid of a traditional wiki and a database. Structured wikis include basic programming mechanisms, which have transformed the wiki into a simple application (Woods & Thoeny, 2007). Structured wikis are also known as application wikis, or second-generation wikis.

**Web 2.0.** A term first coined by O’Reilly Media in 2004, it refers to a new generation of Web
applications that provide for online participation, collaboration and interaction (Pilgrim, 2008). Web
2.0 applications are changing the Web from a read-web to a read-write-web, thereby fulfilling
Berners-Lee's original vision of the Web (Ulrich, Borau, Luo, Tan, Shen & Shen, 2008)

**Widget.** Widgets are the basic building blocks of a Graphical User Interface, or GUI. Some
commonly-used widgets are buttons, scrollbars, and drop-down lists, and menus.

**Wiki namespace.** A feature available in advanced wikis. The concept of namespace is common in
information technology. A wiki namespace is a collection of wiki pages, and permits the same page
name to be used in different contexts.

**Wiki page.** The fundamental storage unit for the content of a wiki. Also referred to as a wiki topic,
or a wiki article.

**Wiki markup syntax.** Along with simple text, wiki pages can include markup. This syntax is not
based on HTML, nor is there currently a universally accepted standard. The syntax used for wiki
markup is specific to the wiki engine used.

**Wiki.** This term was first defined by Ward Cunningham as the simplest online database that
could possibly work. A wiki is a piece of server software that allows users to freely create and edit
Web page content using any Web browser (Cunningham, 2002). The first Wiki was created in 1995
by Cunningham as a method for researching the nature of software development (Gonzalez-
Reinhart, 2005).
2.2. Customer Communication

This section surveys research into the role of customer communication in the success of software development projects.

2.2.1. Previous Studies

A number of studies have shown that most software project failures involve stakeholder problems. One study found that, of the top six reasons for failure, five can be attributed to communication problems between the development team and the customer. Another study claims that the second of 10 most important reasons for success is user involvement. And a Standish Group survey of 8000 projects revealed that only 26 percent of the projects were completed on time, on budget, and with all the originally planned functions. Understanding what customers really want is critical to success, and requires their constant involvement in the project. In order to achieve high levels of involvement, 60% of companies practicing the agile methodology have customers present on-site (Ceschi et al., 2005).

Extreme Programming (XP) is a cornerstone of the agile methodologies. XP advocates on-site customers, yet the cost of such full-time presence measures is high. And, the greater the customer involvement, the greater the potential for customer dissatisfaction. When customer involvement in projects becomes too intimate, new problems can arise. Researchers argue that more analysis and discussion of the impact of XP on the developer-customer relationship is needed. They warn that
customer relations can be endangered using XP. For example, deficiencies of the development staff can become visible to the customer. And placing on-site customers into the middle of the development team, as advocated by XP, can make it difficult for project managers to exercise control due to customer interference (Grisham & Perry, 2005).

The utility of agile methodologies for web environments, where users can offer feedback on-line, has been noted by researchers. Dynamic environments with changing requirements can be seen as moving targets, and giving up control is part of the appropriate design strategy. To enable the evolution of a software service, it is necessary to continually alternate between designing, talking to users and releasing new improved versions of the service (Matheson, 2006).

In the 1990s, numerous proposals in Knowledge Management (KM) were made on how to best utilize employee knowledge, which was perceived to be a company’s most profitable resource (Gonzalez-Reinhart, 2005). However, proprietary KM systems suffer from major limitations, which arise from their static accumulation of dynamic knowledge. A decade later, wikis offer a new, social software solution for the management of corporate knowledge. They are transforming the way corporate America works, since they effectively reduce the need to use conference calls, meetings and emails to resolve issues and understand requirements.

2.2.2. Potential Contributions
I am convinced that Wikis can lead to an improvement in the level of communication between customers and the project development team, and that this form of communication can be evaluated by embedding Wiki functionality directly into everyday applications. Deficits in communication have been shown to be a leading cause of project failure, and this new approach should lead to
greater success rates in software projects. Furthermore, this approach can be used to facilitate an on-going communication between users and development staff that should serve well the continued evolution of completed systems (Grisham & Perry, 2005).

2.3. Hindrances to Using a Wiki

This section surveys research into the technical aspects of wikis that hinder participation.

2.3.1. Previous Studies

Wikis are a form of social software that leads to increased online collaboration, cooperation, and conversation. However, it is difficult for non-technical users to adopt wiki technology, because the interface is non-intuitive, non-graphical, and unattractive. Ward Cunningham, inventor of the Wiki, acknowledged that editing a wiki page is an intelligent test of sorts (Tepper, 2003).

One study researched the use of wikis at seven sites, each in a different industry. It was discovered that users had greater difficulties using the wiki at non-technical sites than at the technically-oriented sites. Specifically, page creation, and using wiki syntax to overcome the weaknesses of the WYIWyG editor, were difficult for the non-technical users. Non-technical users should not be expected to learn and use wiki syntax (White & Lutters, 2007).

In a study of how Italian teachers used Twiki, a structured wiki, Da Lio, Fraboni, and Leo discovered that even technologically skilled teachers disliked having to learn the Twiki syntax, which they perceived as too complicated. A number of the teachers referred the authoring of their Twiki documents to other teachers. The teachers perceived page creation, and attachments management, as complicated. These factors prevented the active participation of some teachers (Da
Wagner and Majchrzak observed that creating wiki pages, and seeding those pages with starter information, can be helpful in getting users involved in contributing to a wiki (Wagner & Majchrzak, 2007).

Hemphill and Yew reported on how an academic community successfully changed its wiki platform from Twiki to WetPaint. The latter did not require users to learn a Wiki syntax, since it has a WYSIWYG editor. The authors concluded that low barriers to participation, such as not having to learn wiki syntax, are essential for encouraging user participation (Hemphill & Yew, 2007).

2.3.2. Potential Contributions
A clear result of research is that participation in a wiki is often hindered by technical barriers. Wikified widgets can be constructed that simplify participation, and make participation easier than using the wiki's own front-end. Users dislike having to learn special wiki syntaxes. Important contributions can be made using just simple, unformatted text, which does not require any markup. By designing wikified widgets to work with such simple text, the process of participation will be perceived as extremely simple. Wikified widgets that provide an extremely simple mechanism for adding content to a wiki repository support the participation of all users, whether they are technically inclined or not.

2.4. Wikis in a Corporate Environment
This section surveys research into problems with participation that are specific to corporate wikis.
2.4.1. Previous Studies

Buffa found that the content of most intranets was produced by a small number of employees assigned to this task (Buffa, 2006). Certain social conditions in the corporate environment, such as the lack of user anonymity and competition between employees, prevent wikis from achieving the same success they enjoy in the public realm. The philosophy of the original wiki concept, as proposed Ward Cunningham, cannot be transposed to corporate intranets. Nevertheless, participation levels in corporate wikis are often higher than those of public wikis.

Researchers believe that wikis will revolutionize employee interaction with corporate knowledge, leading to a profound change in corporate culture. They compare wikis to traditional knowledge management systems, and describe how wikis place less emphasis upon centralized control, strict discipline, and extensive monitoring. By relinquishing control, the responsibility for knowledge management within a corporation is greatly broadened. Management should treat this shift as a benefit, and not a threat. Using wikis, quality assurance can be done by the most qualified peers. And, group participation is driven when an interdependence among employees exists. In such situations, wikis challenge opponents to build consensus in order to get their work done (Hassan & Pfaff, 2006).

Majchrzak, Wagner and Yates speculate that the motivations for participation in public and corporate wikis are different (Majchrzak, Wagner & Yates, 2006). They surveyed 168 individuals who had participated in corporate wikis. Research on participation in open source sites has revealed that enhancement of personal reputation is an important motivation for participation. The authors' research results do not support this as a significant reason for participation in corporate wikis.
Instead, motivation is work-related, and arises from the need for collaboration and improvement in work processes. Contrary to expectations, they found that relying solely on a central wiki as the only communication channel was not helpful. Rather, corporate collaboration is better served when multiple channels of communication are used, for example combining email with wikis.

Wagner and Majchrzak investigated how wiki technology can be used to facilitate higher levels of constructive customer engagement. They noted that many organizations create touch points, such as focus groups, through which some customers can interact with the organization. However, the many customers outside of these touch points are largely ignored. They also emphasize that the granting of full editing rights to all users is a necessary prerequisite for significant user participation. The predecessor to Wikipedia, called Nupedia, failed because it used a different participatory model, in which the role of users was limited to “feeding” the articles, and editing was performed a small team of core editors. After 18 months, a mere 20 articles had been created. The authors also believe that to stimulate participation in a corporate wiki, tasks should be selected that require the collective wisdom of many, and not the expert knowledge of a few. Finally, seeding pages with starter information can help get users involved. To successfully collect information from customers, one needs collaboration and engagement, not just communication (Wagner & Majchrzak, 2007).

2.4.2. Potential Contributions

Wikified widgets can be used to increase the number of participants in a corporate internet, because they make wiki content highly accessible to all users through standard applications. Feedback on the features and evolution of an application is not restricted to a select few, but open to all users of that application. Researchers have determined that many corporations have not established sufficient numbers of such touch points because they rely largely on a small number of focus groups
Wikified Widgets

(Wagner & Majchrzak, 2007). Wikified widgets provide, from within an application, a multitude of touch points for user interaction.

Researchers believe that the motivation to participate in corporate wikis arises from the need to improve work processes through collaboration (Majchrzak et al., 2006). Wikified widgets are, by definition, part of work processes, because they are embedded in applications that play important roles in work processes. They can be used to document problems in an application, and this information is immediately visible to all other colleagues using the same software. If used by application users as a tool for daily collaboration in this manner, the wiki repository becomes an integral part of work processes, and will, according to the research cited above, increase the motivation of users to participate in the corporate wiki.

Wikified widgets provide application users a medium for reporting on problems in the application. Hence these widgets enable the most qualified peers, namely the end users of an application, to be closely involved in quality assurance, a benefit of using wikis that has been noted by researchers (Hassan & Pfaff, 2006). Those authors are convinced that an organization wil profit by relinquishing absolute control of the development process, and following instead the dynamic, unanticipated needs of the customers. Wikified widgets support this strategy by giving an organization the means to continuously harvest customer knowledge for rethinking development plans.

Wikified widgets can be designed that permit anonymous contributions. This has been identified as a feature that is particularly important to corporate wiki environments (Buffa, 2006).
Wikified widgets can be used by development staff to actively prompt customers for opinions and information, something not possible using traditional wikis. This form of active information harvesting can result in collaboration and engagement, not just communication. Researchers believe that such collaboration and engagement are essential to the success of corporate wikis (Wagner & Majchrzak, 2007).

2.5. Unstructured Nature of Wikis

This section surveys research into problems that arise from the unstructured nature of a wiki repository.

2.5.1. Previous Studies

Buffa observed that large wikis, whether public or corporate, pose distinct problems. Interviews conducted with users of a large university wiki revealed that the number one problem reported was that the open structure of the wiki made navigation, orientation, and searching difficult. Often, persons could not locate information in the wiki, even though they knew it was there (Buffa, 2006).

Köhler and Fuchs-Kittowski believe that wikis are eminently suited for the creation and evolution of knowledge in communities. However, they observe that knowledge generated by communities is often characterized by chaotic structure and rapid growth. This is why they believe that tools that support this knowledge creation, such as wikis, need to provide context-based access. They identify the absence of context-based access in current wikis as a major weakness (Fuchs-Kittowski & Köhler, 2005).
2.5.2. Potential Contributions

I believe that wikified widgets can be used in applications to provide direct, contextual access to the relevant pages of a wiki repository. This approach can help overcome a major weakness of wikis, their absence of context-based access (Köhler & Fuchs-Kittowski, 2005). And wikified widgets can be designed to present the application user with a short-list of wiki pages relevant to that application. Hence, pages can be more easily located, sparing users the frustration that has been observed (Buffa, 2006) of searching for relevant wiki content, and failing to find it. Researchers have determined that not every user community succeeds in finding meaning for a wiki (Da Lio et al., 2005). But the development staff can assign meaning to the corporate wiki by seeding it with pages, and providing contextual access points to those pages by using wikified widgets.
3. Chapter 3 - Methodology

3.1. Overview

3.1.1. Choice of Methodology
My investigation was conducted on-site at my place of employment, a German retailer. I am a Java developer, and was responsible for the rewrite of a legacy mainframe application for the Web platform. I used this application in my study to host the wikified widgets that were to be tested. This application is only accessible to a small group of users.

I concluded that a qualitative approach was appropriate for my study, because the small number of participants would have been unfavorable for a quantitative study. Further, I believe that, as the developer of the application, my experiences as a member of the development team are useful and should be included in the research. Hence my research utilized the methods of a qualitative case study, namely, conversations and informal interviews, participant observation, field notes, and a questionnaire.

3.1.2. Questions Derived from Thesis Statement
The following is a list of questions derived from my thesis hypothesis. They influenced design decisions made when creating the wikified widgets. They also provided a framework for the interviews I conducted with the participants, and formed the basis for the follow-up questionnaire.

1) Research has shown that users often reject wiki participation because they perceive it as too complicated (Hemphill & Yew, 2007). Will the use of wikified widgets to access the
contents of a wiki repository be perceived by application users as simple?

2) Can the communication between application users and the development team be increased using wikified widgets?

3) Can wikified widgets be used by the development team for the gathering and clarification of requirements?

4) Can wikified widgets be used to obtain feedback from users that might otherwise be ignored?

5) Research has shown that, to increase participation in a wiki in a corporate wiki environment, the wiki should be made part of everyday work activities. Can wikified widgets become part of everyday work processes, thereby facilitating wiki participation?

6) Communication between application users and developers commonly include email, telephone conversations, and meetings. Will users and developers perceive wikified widgets as an alternative to these channels of communication?

7) Research suggests that the lack of anonymity reduces participation in corporate wikis (Buffa, 2006). Will a wikified widget that records comments anonymously be perceived as useful to the user community, thereby improving participation and communication in the software development process?

3.2. Exposing Participants to Wikified Widgets

The application used in this study to host the wikified widgets is used by a small group of warehouse managers and office staff to review statistics on warehouse performance. Five users of the this application worked for four weeks with wikified widgets while they were testing the application. During this period, weekly meetings were held to discuss the progress of the
application with five of its users, a member of the quality assurance staff, and members of
management. I used these meetings as an opportunity to demonstrate the features of the wikified
widgets, and to observe this small community's reactions to them. A discussion of these
observations is provided in Section 3.3 below.

Reasons for the technical design decisions made in creating the software is provided in Appendix A.
The rest of section 3.2 describes each of the four wikified widgets used in this study. For each
widget, I provide a discussion of the rationale for its design. This is followed by one or more
sections that describe how the widget's implementation of that design.

The screen-shots used in this chapter demonstrate how participants of the study interacted with
these widgets. To facilitate an understanding of this interaction, I added comments and symbols to
the screen-shots. Translations of German text were inserted, and I edited out the names of
participants and places. To ensure the anonymity of the participants, I followed the convention of
using the color green for all alterations. The color green was selected because it was not used by the
the wikified widgets or the hosting application, which allows the reader to easily recognize all green
elements of the screen-shots as alterations.
3.2.1. The Wikified Context Widget

3.2.1.a. Design Considerations

Research has found that people do not contribute to wikis, because they perceive the adding and editing of pages as too complex (Da Lio et al., 2007). Therefore simplicity of use was an important consideration for designing all of the wikified widgets.

This widget provides two functions that are essential to every wiki: editing the contents of a wiki page, and adding comments to that page. However, it cannot be used to add new pages, a feature that is also essential to a wiki. This should not be viewed as a deficit, but rather, as a design feature. There is evidence that, for many users, participation in a wiki is facilitated if pages are created for them and seeded with content (Wagner & Majchrzak, 2007). The Wikified Context Widget was designed to support these findings, since users can only edit pages that already exist. Before research with the participants began, I created, and seeded the content of, the set of pages these widgets can access.

Research has concluded that users should not have to learn wiki syntax in order to participate in a wiki (Hemphill & Yew, 2007). Advanced wiki applications, such as the one used in this research, Xwiki, offer WYSIWYG editors that hide the peculiarities of wiki markup syntax. But users with little technical background experience difficulties even using such editors (White & Lutters, 2007). This widget was therefore designed to support the simplest possible, albeit rudimentary, editing of wiki pages.
This strategy cannot, however, prevent users from being exposed to wiki markup. If they choose to edit a page that contains markup, the markup becomes visible in the editing panel. But with some experience, they will learn to recognize the markup in a page, and how to edit around it so as to leave it intact.

Of course, users may want to take advantage of wiki markup. To do so, they have two options. First, if they know the required syntax, they can use this widget to add the appropriate markup using this editing panel. A help page that lists syntax details can be displayed by pressing the help button of this widget, as described in section 3.2.4. Or, the user can use invoke the WYSIYWG page editor of the external wiki application by pressing the link titled “open in new window”, as demonstrated in section 3.2.1.b.

Research has shown that the open structure inherent to wikis make navigation, orientation and searching difficult (Buffa, 2006). Knowledge generated collaboratively by communities is often characterized by chaotic structure and rapid growth, and tools that support this form of knowledge creation should provide context-based access. A major weakness of wikis is that they do not provide context-based access (Fuchs-Kittowski & Köhler, 2005). Therefore, it was essential to design wikified widgets that provides such access.

This is why I named the first widget I designed the Wikified Context Widget. The wiki page it displays is determined by the context of the hosting application at the time the widget is invoked.
3.2.1.b. Screen Layout

Figure 3.1, shown below, provides an overview of the Wikifed Context Widget.

![Wikified Context Widget overview](image)

The title of a wiki page, “Extras Tab”, is displayed at the top of the widget. The contents of the page are displayed in a panel I call the wiki page render area. In the example above, two lines are shown using list bullets. That's because the contents of this wiki page include special wiki markup for such bullets. The widget renders the HTML representation of the page content in the browser. The HTML was generated by the wiki application's built-in rendering engine.

Directly below the render area is a row of three buttons and a link. Clicking on the link, titled “Open in new window”, will open a new browser window. The new window presents the same page
displayed in the external wiki application used for this project, Xwiki. XWiki includes a WYSIWYG editor, and users can easily select that tool from within the newly opened window. Figure 3.4 provides an example of how Xwiki renders a page after being opened with the “Open in new window” link.

The Change button is used to edit the content of the wiki page, and its use is described in Section 3.2.1c. The Keyword Registry button invokes the Wikified Keyword Registry Widget, which is described in Section 3.23. The Welcome Page button invokes the Wikified Welcome Page Widget, described section 3.2.1.f.

Below this row is a white panel, the comment input area, which is where the text for comments on the wiki page are entered. The Add Comment button and Anonymous Comment button below the white panel are used to commit the comment text to the wiki repository, and are described in section 3.2.1d. The All Comments button invokes the Wikified All Comments Widget, and is discussed in section 3.2.2. Finally, the Help button at the bottom of the widget is discussed in section 3.2.4.

3.2.1.c. Context Dependent Invocation

The widget is invoked by pressing the F4 key. The wiki page to be displayed is determined by the current context of the hosting application. The current context is defined by a simple rule. If a tooltip is currently being displayed, the wiki page associated with that tool tip will be displayed. Otherwise, the default wiki page for the currently active screen will be displayed. Figure 3.7, below, demonstrates how tooltips are used to determine the current context.
Figure 3.2 Context determined by the currently active tooltip.

The tooltip “Auswahll des Sortiments” appears when the user moves the mouse over the label titled “Sortiment”, which is part of the hosting application. The information necessary to retrieve that page from the wiki repository – the name of the page and its namespace – is associated with the label “Sortiment”. If the user now presses the F4 button, the Wikified Context Widget is invoked and it will display the page that has the title “Auswahl des Sortiments”. This mechanism extends access to the wiki repository to all widgets of the hosting application that can display a tooltip. It transforms the standard widgets of a hosting application into wikified widgets.

Figure 3.8, below, visually clarifies the relationship between the text of a tooltip and the wiki page that is retrieved by pressing the F4 key.
The wiki application used for this study, Xwiki, distinguishes between a page's identification and its title. A page is identified by the unique combination of namespace and page name. The page title is a separate attribute of a page. The tooltip shown above stores a specific page id and namespace, which are used to retrieve the page's title. If the title of the page is changed in the external wiki, the text displayed by the tooltip changes as well.

As noted previously, if no tooltip is active, then the current context defaults to a page that has been associated with the currently active screen. The next figure, Figure 3.9, demonstrates one such default context. The figure shows what happens when the F4 key is pressed when no tooltip is
active.

![Figure 3.4 Demonstration of the default context of a screen](image)

In the figure above, the widget displays the page titled “Extras Tab”, because that page serves as the default context for the “Extras” tabbed panel, the currently active panel. Members of the software development team can edit the content of the “Extras Tab” page in the wiki, and immediately change the contents for this page of this online help system. Using the Wikified Context Widget, members of the user community can modify that content, perhaps to improve it, or to share useful tips about the application with each other. Editing page content is described in section 3.2.1.d. Or, application users can use this widget to easily add comments to the page. These comments are immediately visible to the development team and the rest of the user community. This feature is
described in section 3.2.1.e.

### 3.2.1.d. Edit Page Function

To begin editing a wiki page, the change button must be pressed, as demonstrated below.

![Figure 3.5: Render mode](image)

The change button is marked in Figure 3.2. Notice that the panel above the button, the render area, cannot be edited until after the change button has been pushed. After pressing the change button, the render panel is replaced by an editable text area, as shown in Figure 3.3.
The Wikified Context Widget is now in edit mode, and the page can be edited. In the example above, I appended a translation of the German text to the wiki page. During edit mode, the text for the Change button changes to “Update”. Once editing is finished, the user needs to press this button to commit changes to the wiki repository.

The link to the far right of the Update button can be used to open a new browser window, in which the wiki page is rendered from within the external wiki application. Figure 3.4 shows how the changes made above appear when rendered by the wiki application.
The wiki I used for my research, Xwiki allows both comments and attachments to be added to pages. The figure above displays how the a page is rendered by the Xwiki application. Comments and attachments, if present, are displayed after the main body of the page. As shown above, Xwiki indicates that zero comments and zero attachments have been added to this page. The following section describes how the Wikified Context Widget can be used to add comments.

### 3.2.1.e. Add Comment Function

To add a comment, the comment text must be entered into the input area for comments, after which the Add Comment button must be pressed, as shown in Figure 3.5.
After the add comments button is pressed, the wiki page is updated in the wiki repository with the comment, the page is fetched anew from the repository, reloaded and rendered. The new comment now appears below the page content, as shown in Figure 3.6.
Figure 3.9: How comments are displayed

The figure above demonstrates how comments are displayed by the Wikified Context Widget. If comments exist for a page, a comment table is added to the widget, directly below the render panel. If there are numerous comments, a scroll bar will appear so that all of the comments can be viewed. The table has three columns. The first is for the name of the person that contributed the comment. The second shows the time and date the comment was added. And the last column displays the comment itself. If multiple comments have been added to the page, the comments table can be sorted by clicking on the blue title area of any of the columns. By default, the table sorts on the
date/time column, so that the most recent comment is shown first.

3.2.2. The Wikified Welcome Page Widget

3.2.2.a. Design Considerations

Research has revealed that a high percentage of software projects fail. Often, the reasons for failure can be attributed to communication problems between the development team and the customer (Ceschi et al., 2005). And to enable the evolution of existing software services, it is necessary to continually alternate between designing, talking to users and releasing new, improved versions of the service (Matheson, 2006). The development team needs to fully understand customer requirements, and wikis are already being used for this purpose (Gonzalez-Reinhart, 2005). The Wikified Welcome Page Widget was designed specifically for the communication needs of the development team. Hence it supports the communication between the development team and customers that researchers believe is necessary for project success.

This widget is named the Wikified Welcome Page Widget because it always displays the same wiki page, namely, the Welcome Page specific to the hosting application. Each hosting application can have an individual welcome page, where the development team can place any new information regarding the application that they need to communicate to the user community. A design goal for this widget was to provide the application users quick and easy access to a page they will recognize as a meeting place, where they can find the newest information on their application from the development team. A discussion on the various uses that were made of this tool during the study is provided in Section 3.3.
3.2.2.b. Implementation

This widget is a special form of the previously described Wikified Context Widget. It appears automatically in the hosting application at startup. Its context can be defined as the start of the hosting application. At any time following startup, the application user can revisit this widget by pressing the “Welcome Page” button that is part of the Wikifed Context Widget. Users can quickly revisit the widget in two steps. First, the F4 button is pressed from anywhere in the hosting application. That displays the Wikifed Context Widget. The second step requires that the user then press the “Welcome Page” button of that widget, as demonstrated in Figure 3.10.

![Figure 3.10 Manually invoking the Wikified Welcome Page Widget](image)

The next figure, Figure 3.11, displays the state of this widget on one day of the 4 week study. The content of the article is in German, and contains a number of links. One link is to the protocol of a recent meeting, one is to a new page just contributed by a member of the user community, and there are four links to previous versions of the welcome page.
Note that the title of the wiki page displayed is “BOB Frontend Welcome Page”. BOB is the name of the hosting application used for this study. As described in section 3.2.2.a, each hosting application has its own individual Welcome Page in the wiki repository, so that the information provided is specific to that application.

Figure 3.11 Wikified Welcome Page Widget

It is clear from looking at the figure above that this widget appears to be nearly identical with other instances of the Wikified Context Widget. It differs from that widget in the following ways:
It does not have a button titled “Welcome Page”, because it is the Welcome Page.

It always appears at the start of the hosting application.

It can be invoked at any time by pressing the Welcome Page button that is found on all other instances of the Wikified Context Widget.

In all other aspects, it is identical in form and functionality to that widget.

3.2.3. The Wikified All Comments Widget

3.2.3.a. Design Considerations

Researchers have discovered that people have difficulty in finding information stored in a wiki (Buffa, 2006). The comments that have been entered into a wiki repository also contain important information, yet traditional wikis offer no mechanism to efficiently browse through. The design goal for wikified widgets is not only to make an application user's interaction with a wiki repository immediate and contextual, but to make that interaction as simple as possible. The Wikified All Comments Widget was designed to give users a simple tool with which they can quickly access all of the comments that have been contributed to all of the wiki pages associated with a given application. Simplicity of use has been identified as an important factor for increasing participation in a wiki (Da Lio et al., 2005).

In corporate wikis, the motivation to participate arises from the need to improve work processes through collaboration (Majchrzak et al., 2006). The Wikified All Comments Widget permits the users of an application to quickly find comments made by colleagues who use the same software.
and are part of the same work process. It can be used to quickly document problems in an application, and these comments are immediately visible to all other colleagues using the same software. If used by application users as a tool for daily collaboration in this manner, the wiki repository becomes an integral part of work processes. This could, according to the research cited above, result in users becoming motivated to make additional contributions to the wiki repository.

3.2.3.b. Implementation

The Wikified All Comments Widget displays, in tabular form, all comments that have been contributed to the set of wiki pages associated with the hosting application. Section 3.2.4b explains how this set of pages is defined.

The widget is invoked by pushing the “All Comments” button of the Wikified Context Widget, shown below in figure 3.12.

![Figure 3.12 Invoking the All Comments Wikified Widget](image)

After the button is pressed, the widget appears, as shown below in figure 3.13.
The table of comments displayed by this widget has four columns. They are name of contributor, time-stamp of comment submission, page that the comment belongs to, and the text of the comment. By default the table is sorted on the time-stamp column, so that the most recent comments appear at the top of the table. However, the table can be resorted on any of the four columns by clicking on the blue title of the desired column. This is demonstrated in figure 3.14.

---

**Figure 3.13 The All Comments Wikified Widget**

The table can be sorted using any of the 4 columns.
Figure 3.14 Resorting the comments table

In the figure above, the blue title area labeled “Artikel” was clicked, and the table resorted based on the title of the page to which the comment belongs.

### 3.2.4. The Wikified Keyword Registry Widget

#### 3.2.4.a. Design Considerations

Researchers know that people experience significant difficulties in finding information stored in a wiki. In one study, the number one problem reported by users of a wiki was its intrinsically open structure. Navigation, orientation and searching this structure were perceived as difficult. Persons
were not able to locate information in the wiki, even though they knew it was there (Buffa, 2006). Hence, a key goal in designing wikified widgets is to provide structured paths of access so that users easily find content they need to read or edit. Wikified widgets will only be useful to the software development process if they are designed as tools that provide efficient access to the content of a wiki repository. Application users shy away from features they perceive are difficult to use (da Lio et al., 2005).

The four wikified widgets I have designed provide different kinds of structured access to the content of a wiki repository. The Wikified Context Widget offers context-dependent access, the Wikified Welcome Page accesses one specific page, and the Wikified All Comments Widget accesses all of the comments that have been made on a set of pages. This Wikified Keyword Registry Widget provides access to a pre-defined set of pages.

3.2.4.b. Implementation

This widget displays, in tabular form, a set of pre-defined wiki pages. For this study, I defined this set to be the union of two sets of pages. The first set includes all of the wiki pages available via contextual access from within the hosted application. This number of pages in this set is static. The second set includes all of the wiki pages that belong to the "KeywordRegistry" namespace of the wiki repository (contextual access was explained in section 3.2.1.c). The number of pages from this set is dynamic, because a wiki namespace can hold any number of pages, and can change over time.

Figure 3.2.5 demonstrates below that this widget, like the Wikified All Comments Widget, is invoked from a button located in the Wikified Context Widget.
The following figure, figure 3.16, shows the Wikified Keyword Registry Widget after the button shown above was clicked. The title bar indicates that this list contains the articles for 34 keywords.

Clicking on the title of a page in the list will open that page in the Wikified Context Widget. Note
that the page I have marked to open is titled “Meeting BOB von 13. August 2008”. This page contains the protocol of a meeting held to discuss the progress of BOB, the application that hosted the wikified widgets for this study. The protocol of a meeting cannot really be considered as a keyword. I address this issue in Section 3.3.

3.2.5. Help Buttons and Pages

A help button is located in the middle of the last line of each wikified widget. When pressed, the help button invokes the Wikified Context Widget, which displays a page of the wiki that contains useful information specific to that widget. The help page for the Wikified Context Widget, for example, includes a link to another help page provided by Xwiki that describes the markup syntax used by the Xwiki engine. That information would be useful to those users who wish to learn how to add wiki markup to the wiki pages.

3.2.6. Single Sign-on

This widget, as well as the other wikified widgets developed, access the wiki repository with a user name and password. Although it is possible to login to a wiki anonymously, it is extremely useful to be able to identify the authors of comments and page changes. For example, in section 3.2.3.a I discussed how users of an application can add and read comments using wikified widgets to help support their work processes. In this scenario, the name of the comment's author may be needed for contacting the author. Research has indicated that relying solely on a central wiki as the only communication channel is not helpful, and that collaboration is better served when multiple channels of communication are used, for example, by combining email with wikis (Majchrzak et al.,
I therefore decided that the ability to login to the wiki repository using name and password was an important design consideration.

Wikified widgets are embedded into a hosting application, and users typically need to log in to such applications for purposes of authentication and authorization. My implementation of wikified widgets accesses the remote wiki repository using services that are separate from the services utilized by the hosting application. Hence two logins are required, the first to start the hosting application, and the second to use the services that provide access to the wiki repository.

But an overriding principle of the original wiki design was simplicity (Cunningham, 2002), and the research cited in the sections above indicate that simplicity of use is essential to participation in a wiki. This is why I designed the wikifed widgets in such a way that the application user is unaware of the second login. This feature is commonly known as Single Sign-on (SSO).

A further discussion of these services is offered in Appendix A.

### 3.3. Observations

As the developer of the application that hosted the wikified widgets, I was in a position to observe the other participants of the study in weekly meetings, and to conduct telephone conversation with them. In the context of my daily workplace, I discussed the issues that arose during the study with other members of the software development and quality assurance staff.
In an early conversation, a colleague from quality assurance remarked that many application users were adverse to using bug tracking software, because they perceived it has having to learn a separate application. Later, at one of the weekly meetings, another member of quality assurance expressed his concern that the wikified widgets were causing a problem. Due to the simplicity of use, application users had been using the new tool to enter their bug reports, and had made no entries using the established bug tracking system. It became clear then that we would need to consider how this software should be differentiated from the tracking system already in place.

In a later conversation with colleagues from the development staff, we discussed this conflict of the tool with the bug tracking system. One colleague noted that many bug reports are not worthy of entry into the tracking system, because they report on trivial problems, or are perhaps a request for a change in requirements. Further, during the prototyping phase, bug reports need not be generated, and this tool would provide a useful and informal mechanism for reporting bugs during that stage. Another colleague suggested that the comments submitted by application users not be processed immediately, but in regular intervals, perhaps every two weeks. Development staff could meet with members of the functional department to evaluate which of the comments in the wiki repository are worthy of entry into the formal bug tracking system.

During the meetings, I demonstrated improvements made to the hosting application during the previous week, and in the course of the demonstration, we agreed upon further changes to the software. At the first meeting, I made notes for a protocol by writing them down. By the second meeting, however, I began entering protocol notes on the appropriate wiki pages using the Wikified Context Widgets. After the meeting, I created a new wiki page for each meeting, and entered the notes there. The next day I edited the page displayed by the Wikified Welcome Page Widget to
include a link to the new protocol page. In my experience, these tools are very useful for recording points of protocol to the appropriate context in a wiki, and can be used to make this information highly visible to all application users that login the next day. This is particularly valuable for those users who were not present at the meeting.

In the course of evaluating the various comments that the users had entered, I realized that adding comments to a wiki, and then reading them, does not fulfill the question-response paradigm needed to process those comments. Telephone conversations and email were used here to follow-up my processing of the comments. It also became clear to me that, for applications with large numbers of users, some level of mediation would be required to process the feedback given in this manner.

At one meeting a question I had posed on the welcome page was addressed by a participant who had read the request. But he had elected to relate the information to me at a meeting, and not using the wikified widgets, suggesting that he preferred that form of communication to explain difficult issues. In another case, a participant who was using the application from a remote site, and was not able to participate at the meetings, responded to a request for information that I had posted on the welcome page. He elected to use email to explain the issue I had inquired about.

I regularly changed the contents of the page displayed by the Wikified Welcome Page widget, so that users would find useful information there. Soon, I was creating links on the welcome page to older versions of the page, so as to preserve the information contained in them. This was not a good solution, so I started placing old versions of the welcome page in the wiki namespace used by the Wikified Keyword Registry Widget. That's because all pages found in that namespace are automatically display by that widget and do not require links to be created to access them. This led
me to realize that future versions of the wikified widgets should utilize several namespaces, so as to allow related pages to be stored and displayed as a group. I recognized the need for one additional namespace for previous versions of the welcome page, and one for pages that contain the protocols of meetings. Additional buttons in the wikified widgets so that the Wikified Keyword Registry Widget could be invoked to display the pages contained in these new namespaces.

3.4. Followup Questionnaire

A questionnaire was distributed to the five participants of the study that will be using the hosting application regularly, and as a part of their daily work processes. In the introduction of the questionnaire, I guaranteed the respondents anonymity. The questions posed, and the participants responses to them, are available in Appendix B. They were formulated to help answer the list of questions developed for this methodology, which are described earlier in this chapter in section 3.1.2.

The questionnaire consisted of 12 questions. Eleven were multiple-choice, and a prompt for any additional remarks was inserted after each question. The twelfth question prompted for any additional comments the participant might wish to make. One of the participants responded to this final prompt. And one of the respondents provided extended feedback to question 11. Otherwise, no extended feedback was given.

Here is a summary of their responses:

1. The first question asked the respondents to rate the degree of difficulty of using the tested tools to add comments. Three found usage simple, two found it be normal (i.e. Neither simple nor difficult).
2. The next question addressed the degree of difficulty for using the widgets to edit the content of a page. Three respondents answered that they had not used this tool, one replied that it was simple to use, and one that it was normal.

3. Question 3 asked if it made sense for the application users and development team to communicate with each other using these tools. Four responded positively and one negatively.

4. Question 4 asked if it made sense to use the welcome page to inform the users about changes in the application. The five respondents agreed unanimously that this was a good idea.

5. Question 5 asked if the feedback tool should be deactivated once the test phase of the hosting application's development was concluded. Three responded that the tool should be left activated, one felt that it should be activated, and one answered with maybe.

6. The next question asked whether the tool should be made accessible only to a restricted group of users, in the event that it remained activated for production use. Three responded that access should be restricted, one was undecided, and one responded that the tool should be accessible to all users.

7. Question 7 asked whether the feature of adding comments anonymously would be of value to application users. Two responded that this feature had value, and three responded that it might have value for the users.

8. The next question asked whether they would read the welcome page displayed by the tool every day. Two responded positively, and three responded with a no.

9. The next question asked if it made sense to use the tool as a replacement for email. Two responded with yes, one with maybe, and two with no.

10. Question 10 asked whether it made sense to use the tool as a replacement for telephone
conversations. Three responded positively, one with maybe, and one negatively.

11. The next question asked if it made sense to use the tool as a replacement for meetings. Two responded with maybe. The three other respondents answered negatively. One of the respondents added with additional text that meetings were preferrable to this tool, because at a meeting one can explain a problem in more detail.

12. The final question was free-form, and gave the respondents the opportunity to give any other feedback they wished to offer. One respondent answered. The response included the following points. The tool was uncomplicated and simple to use. Bugs could be corrected more quickly using this tool than using a bug reporting system. The tool can provide users with answers to questions about the software, which reduces the number of questions that are asked of the developers.

An analysis of the participants’ responses can be found in Chapter 4.
4. Chapter 4 – Analysis and Results

4.1. Evaluation of Project Goals

In Section 3.1.2 I stated seven questions my research needed to address. I now revisit those questions and evaluate to what degree that I have been able to answer them in the course of this research. In the discussion that follows, I refer to the follow-up questionnaire described in section 3.3 of the previous chapter.

1) Will the use of wikified widgets to access the contents of a wiki repository be perceived by application users as simple?

I am able to answer this question with a qualified yes. Results of the survey indicate that the users experienced the wikified widgets as simple to use. In my conversations with the participants, that was the only opinion expressed. However, the questionnaire revealed that only two of the respondents used the editing feature to change the content of a wiki page. One of the two said that usage was simple, and the that usage was normal. It was disappointing that the remaining three respondents had not used the tool and formed an opinion on its ease of use, because the editing function might play an important part if these tools were used regularly.

The two respondent that had tested the page editing feature had made changes only for testing purposes. None of the participants used the edit feature to contribute meaningful content to any of the seeded pages. All pages had been previously created by me and seeded with minimal content. A page was created for each of the standard widgets of the hosting application that had pop-up tool-tips, one page was created for each of the tabbed panels, and several other pages were created that
could be accessed using the Wikified Keyword Registry Widget.

2) Can the communication between application users and the development team be increased using wikified widgets?

I was able to answer this question with a yes. Four of the five applications users that took part in the study contributed comments using the tool. And four of the five questionnaire respondents answered that it makes sense for users and the development team to communicate using these tools. All five respondents answered that it made sense to use the welcome page widget to inform users of changes that have been made to the application. At a meeting at the end of the four weeks I offered to remove the wikified widgets from the their application. They were unanimous in their opinion that the tool be left active. I was convinced by this reaction, and the unanimously positive response to the question on the usefulness of the welcome page widget, that application users are very interested in receiving information about their application from the development team in this manner.

3) Can wikified widgets be used from the development team for the gathering and clarification of requirements?

Here, too, my answer is yes. As developer of the application that was being tested by the participants, I used the welcome page widget to pose two questions to the user community. These were questions concerning the functional domain that I was unable to answer. In one case, I received a detailed answer from one of the participants the same day. In the other case, one of the participants addressed the issue I had address at the next meeting.
4) Can wikified widgets be used to obtain feedback from users that might otherwise be ignored?

Four of the five application users who participated in the study worked on-site, and I met with them on a weekly basis to discuss the progress of the application that hosted the wikified widgets. One, however, worked at a remote site hours away, and did not take part in any of these meetings. This was the participant mentioned above that responded to my inquiry on the same day. The question I had raised was extremely important for a warehouse he worked at, but of little concern to the other participants, who worked in the central warehouse. As developer of the application being developed, I had not been aware just how important this issue was for a successful design of the software. If I hadn't used the wikified widgets to pose that question, I would not have received the input from the participant from the remote site at that time.

5) Can wikified widgets become part of everyday work processes, thereby facilitating wiki participation?

What is clear is that applications users welcome the opportunity to use the add comment function to submit bug reports. On several occasions, and in the responses entered in the questionnaire, they expressed that it was much easier to use this feature of the widgets to submit a bug report than it was to use the bug tracking software they used for other applications. This finding is supported by my conversation with a member of quality assurance, who explained to me that application users perceive using bug tracking software as having to learn a separate application. Wikified widgets offer a key advantage here, because they are not separate from the user application, but a part of it. Wikified widgets can become a part of daily work processes. However, my research could not determine if this form of participating in a wiki leads to an increase in participating in a wiki in other ways, such as contributing content to a wiki page.
users were adverse to using bug tracking software, because they perceived it has having to learn a separate application.

6) Will users and developers perceive wikified widgets as an alternative to other forms of communication, such as email, telephone conversations, and meetings?

My research indicates that such widgets can be used as a complement to email and telephone conversations, but that they are not useful as a substitute for meetings. Three respondents replied that these tools could not be used as an alternative to meetings, and two replied with maybe.

7) Will a wikified widget that records comments anonymously be perceived as useful to the user community?

My research suggests that yes, the option of submitting comments anonymously is important to application users. All of the questionnaire respondents answered that anonymity was either useful, or that maybe it would be useful. None denied its usefulness. And in a conversation with a member of quality assurance, I learned that many users do not enter bug reports, because they fear that they might be mistaken, and would be embarrassed if they submitted an unfounded bug report. This, too, supports my finding that this feature is useful, and can increase participation in the corporate wiki.

4.2. Role of Project Variables in the Results

My research on how users interact with wikified widgets lasted only four weeks. A longer period of research might have led to other results.
The number of study participants was small, just five application users, several members of management and quality assurance staff, and myself as member of development staff. This study was qualitative in nature, and did not therefore require a larger, statistically relevant number of participants. Still, a larger group may have brought a richer variety of personal backgrounds to the study; resulting in a greater yield of participant responses. Only one of the five questionnaire respondents provided feedback to the final question, which asked them to enter any additional opinions or experiences.

Of the five participants of the study that were application users, four met in weekly meetings to discuss the progress of the application being developed. In many scenarios, the users of an application have no contact with each other. The results of the study may have been different if these users did not have weekly, on-site contact with each other.

The hosting application used to test the wikified widgets was in the early testing phase of the development life cycle. The application was still unstable, and the data that it displayed and calculated was not yet of sufficient quality to have justified a significant amount of testing effort on their part. Consequently, the participants were not using the application nearly as intensively as they would have later in the development cycle, or after it had gone into production. Results may have been different if the research had been conducted during another phase of development, for example during prototyping, or after having reached maturity.
4.3. Contributions to the Body of Knowledge

My research confirms that the level of communication between the users of an application and the development team can be increased by embedding access to wiki repository within that application.

In section 2.3 I cited studies that emphasize the importance of simplicity for participation in wikis. My research confirms that, given a simple means to contribute, the users of an application will readily participate in a wiki that stores information about that application. This result, however, has only been shown to apply to a restricted form of wiki participation, namely, the contribution of comments to existing pages.

In this study, the restricted form of participation mentioned was used largely by the application users to communicate bug reports. This result suggests that, by embedding wiki access into applications, a new generation of bug-tracking systems can be developed.

My research shows that placing access to the wiki repository within an application leads to exchanges of information between the development team and the users of an application that otherwise would not occur. It has been shown that tools based on this approach can be successfully used by a development team to request, and harvest, functional knowledge of a system from the user community of the application hosting the tools.

This study supports the findings of other researchers that, in a corporate environment, the option to contribute anonymously to a wiki is an important one.
Lastly, this study has provided demonstrations of some of the ways that structured access to a wiki repository can be embedded into applications. In particular, it demonstrates how the strategy of embedding provides context, the application itself, for structured access. These demonstrations might be useful as examples for other researchers.

5. Chapter 5 – Conclusions and Next Steps

5.1. Conclusions

This research project studied, for a four-week period, the interaction of a small community of application users, developers and quality assurance staff with a set of tools that provide access to a wiki repository from within an application. The goal of the study was to evaluate the usefulness of such tools for the software development process. Participants of the study found wikified widgets easy to use, and used them to communicate problems in the hosting application to development staff. Applications users also welcomed the use of wikified widgets to receive information about changes in the application from the development staff. These tools also aided the development staff gather information on functional requirements from the application users. Hence, the original expectation of the study, that such widgets can enhance the software development process, was met.

5.2. Next Steps

The results of this study need to be substantiated by further research. The participants of this study
had no previous experience contributing to wiki. The tools introduced in this research can be used for a comparative study of participants that have had previous experience contributing to a wiki. It would also be useful to apply these tools to study a group of participants that use a common hosting software, but which do not meet weekly, or have other means of communicating with each other. The users of a portal would be appropriate subjects for this kind of study. And, a comparative study needs to be conducted on the usefulness of wikified widgets during different phases of the software development life-cycle.

Quantitative research can be pursued by using more participants over a longer period of time. Such research could, for example, take advantage of the logging function common to many servers to record the usage patterns of the different widgets.

This research has revealed that application users embrace this tool for reporting software bugs. Further research needs to determine what ways, if any, this technology can be used as a replacement for, or in conjunction with, traditional bug tracking systems.
6. References

(Source: http://www-sop.inria.fr/acacia/WORKSHOPS/IntraWebs2006/Buffa_Intrawebs2006.pdf)

(Source: IEEE Digital Library http://doi.ieeecomputersociety.org/10.1109/MS.2005.75)


(Source: ACM Digital Library http://doi.acm.org/10.1145/1104973.1104982)


(Source: www.uhisrc.com/FTB/Wiki/wiki_way_brief%5B1%5D-Jennifer%2005.pdf)


(Source: ACM Digital Library http://doi.acm.org/10.1145/1083106.1083113)


(Source: ACM Digital Library http://doi.acm.org/10.1145/1228175.1228250)


(Source: ACM Digital Library http://doi.acm.org/10.1145/1316624.1316665)


(Source: ACM Digital Library  http://doi.acm.org/10.1145/1149453.1149472)


(Source: ACM Digital Library http://doi.acm.org/10.1145/1182475.1182549)


(Source: ACM Digital Library:  http://doi.acm.org/10.1145/1379092.1379144)


(Source: ACM Digital Library http://doi.acm.org/10.1145/940830.940831)


(Source: ACM Digital Library http://doi.acm.org/10.1145/1104973.1104979)


(Source: ACM Digital Library:  http://doi.acm.org/10.1145/1367497.1367593)

*Journal of Management Inform. Systems* 23(3) 17–43.

(Source: http://mesharpe.metapress.com/index/J72X514QM0V55544.pdf)


*Proceedings of the 2007 symposium on Computer human interaction for the management of information technology, Article No. 5.*

(Source: http://mesharpe.metapress.com/index/J72X514QM0V55544.pdf)

7. Appendix A- Software System Design

7.1. Architectural Goals
I have researched the value of embedding wiki access into applications as a means of supporting the software development process. This research could have been conducted, and can be continued, using numerous application hosts running on different platforms. An important architectural goal was therefore to design the wikified widget system to be reusable in a wide range of technical contexts.

The tools I used were all based on the Java language. For the front-end, I developed wikified widgets using the Google Web Toolkit (GWT). This toolkit compiles Java source code into Javascript that will run on any of the major browsers. That’s because it compiles to a set of Javascript files, where each file is specialized for one browser. GWT automatically detects the browser type and delivers the appropriate Javascript file at runtime.

On the server-side, the system was designed to be service-oriented. To this end, I made use of a feature of the Spring Framework named Lightweight Remoting, which allows the services I developed to be consumed using a variety of protocols, including WSDL/SOAP. Hence these services can be used by applications not written in Java. For example, a so-called fat client written in C could invoke the services using SOAP.

7.2. Front-end Architecture
Design of the system’s front-end was dictated by my decision to use the Google Web Toolkit to implement the wikified widgets. GWT gives Java developers the means to create AJAX web
applications. AJAX applications rival desktop applications in responsiveness, and are playing an important role in the Web 2.0 revolution (Chaganti, 2007). Wikis, too, are an important part of the Web 2.0 landscape, and I believe that the use of AJAX technology was therefore appropriate for my study of wikified widgets.

GWT provides separate client and server components that communicate using service calls. The Model-View-Controller (MVC) paradigm is commonly used for a front-end architecture. With traditional web application frameworks, such as Struts or Java Server Faces, model and controller components are implemented on the server. With GWT, however, the controller runs in the browser client, and the model is transported from server to client via asynchronous remote procedure calls (RPC). This front-end architecture more closely resembles a traditional client-server application than a web application. After the remote procedure call has completed, the GUI logic residing in the client browser has in-process access to the model data. This explains why GWT and AJAX applications are so highly responsive

7.2.1. Front-end Widgets
For my study, the wikified widgets I developed with GWT were used in an application host that was also written using GWT. Ideally, the wikified widgets I developed should be reusable in other, non-GWT browser applications. Fortunately, GWT components can be embedded in traditional web pages. However, I concluded that an investigation of this possibility is beyond the scope of this study.

I designed and tested four fidgets in the hosting application, named Wikified Context Widget, Wikified Welcome Page Widget, Wikified Comments Widget, and Wikified Keyword Registry Widget. Chapter 2, Methodology, describes the purpose of these widgets, and contains screen-shots
7.2.2. Front-end Services

As noted above, GWT utilizes both client and server components. GWT automatically generates proxy classes to handle all of the RPC plumbing for invoking server-side code and for converting data back and forth between the client and server. Hence the Java developer only needs to define a service interface, and to provide its server-side implementation.

The services I designed transport Data Transfer Objects (DTO) between client and server. The DTOs are designed as simple Java classes that have only getter() and setter() methods. Such beans are commonly known as POJOs, or Plain Old Java Objects.

There are four main DTO classes. The WikiPageSummaryDTO class holds summary information about a wiki page, including page id, title, and url. The WikiPageDTO class contains the detail attributes, and raw text content, of a wiki page. The WikiPageAggregateDTO class aggregates an instance of the WikiPageDTO class with its rendered content. Rendered content is the HTML representation for that wiki page. Finally, the WikiCommentDTO class contains the attributes for a wiki comment. Wiki comments are always associated with a wiki page, and there can be zero, one, or multiple comments for any one page.

These classes define the domain for the front-end model. They, and the service interface used to deliver them, are standard Java classes and are not technically coupled to GWT’s RPC service in any way. Hence, the front-end interface, and the beans they transport, can be reused, as is, by other, non-GWT service implementations. One such service implementation is provided by the Spring
7.3. Back-end Architecture

The back-end system could be run on the same server container as the front-end. But for reasons of security and future scalability, I chose to deploy the back-end services on a separate server. For my study, I used separate Tomcat servers for the front-end and back-end.

The back-end provides services that read, and update, the contents of a wiki repository. For my study, I used XWiki, an open-source product. I chose XWiki primarily because it ships with an XML-RPC interface that allows queries and updates to the XWiki repository. Better yet, XWiki also provides a Java client-side proxy that eliminates most of the plumbing needed to use XML-RPC directly. This client is accessed using Swizzle Confluence, which has an API specialized for accessing wiki repositories. This form of access is limited to those wikis that support the Confluence API. However, other, more universal approaches to programmatically accessing any wiki implementation, which would help overcome the fragmentation of the wiki developer community, have been suggested (Shanks, 2005).

The backend supports a service-oriented architecture, since the services I developed for consumption by the wikified widgets are reusable in multiple contexts using different protocols. Using the Spring framework, this design goal was easily accomplished.

7.3.1. Single Sign-on

Most wikis permit users to login using an id, or anonymously. For my study, both forms of authentication were required, since it is possible that users perceive the need to contribute to a wiki
in both ways. Further, for the sake of user convenience, I decided that some form of Single Sign-On (SSO) was required. Users access the back-end wiki services from an application that hosts the wikified widgets. One goal of my design was to create a system that is perceived by the user as simple to use. Therefore it was important that the user not have to conduct a separate login to use wikified widgets.

In my implementation, the name and password used to login to the back-end Confluence session are taken from the front-end application that hosts the wikified widgets. Hence users of the hosting application need to have an account on the wiki server that uses the identical user name and password. Far more elegant forms of Single Sign-On (SSO) are available, and it is my understanding that the Xwiki will offer SSO in the future.

7.3.2. Back-end Services
The services I created access the wiki repository using the Swizzle Confluence API. They offer functionality similar to the methods of that API, but differ in two important ways.

First, all method signatures of the backend services include user-name and password as passed parameters. These parameters are needed to sign-on to the wiki repository and create a Confluence session, which is then used to invoke the Swizzle Confluence API interface of the wiki repository.

The second distinction between the backend-services and the methods of the Swizzle Confluence API are the objects passed using them. One backend-service method is designed to return a list of all comments that have been entered for a given list of wiki pages. The Swizzle Confluence API did not offer such a method, so I needed to create it myself. I also found that I could simplify the
backend-service API by aggregating the information contained in Swizzle Confluence API beans into new beans. This strategy allows the user of the backend-service API to invoke just one method in order to receive data from the wiki repository that would otherwise have required the chained invocation of multiple methods using the Swizzle Confluence API.

### 7.3.2.a. Lightweight Remoting

A goal was to provide backend-services that could be reused in multiple contexts via different protocols. For example, it might be useful to access the data stored in the wiki repository from a PDA, or to transform the backend-services into SOAP-based Web services. The Spring framework offers an elegant solution called Lightweight Remoting that can be used for both client-server remoting as well as for intra-server remoting. The latter is used for communication between various processes within the same server system. (Johnson, Hoeller, Arendsen, Risberg & Sampaleanu, 2005). Using Lightweight Remoting, the services developed can be accessed using the most common protocols, including RMI, WSDL/SOAP through JAX-RPC, Hessian, Burlap, and HTTP invoker, a protocol unique to Spring.
8. Appendix B – User Questionnaire

The following questions were included in the questionnaire. There were five respondents. The number to the right of each answer indicates the number of respondents that selected this answer.

1) How do you rate the degree of difficulty of using this tool to add comments?
   
   A) I did not use this feature. 0
   B) Difficult 0
   C) Neither difficult nor simple - normal 2
   D) Simple 3

2) How do you rate the degree of difficulty of using this tool to add change the content of an article?

   A) I did not use this feature. 3
   B) Difficult 0
   C) Neither difficult nor simple - normal 1
   D) Simple 1

3) Do you believe that it makes sense for the application users and development team to communicate with each other using this tool?

   A) I have not yet formed an opinion on this matter. 0
   B) No. 1
   C) Maybe. 0
4) A welcome page from the feedback tool always appears when the BOB application starts. Does it make sense to use the welcome page to inform the users about changes in the application?

A) I have not yet formed an opinion on this matter. 0
B) No. 0
C) Maybe. 0
D) Yes. 5

5) Let's assume that our company decides to use the feedback tool. Should this tool only be visible during the testing phase, and be deactivated when that phase has been concluded?

A) I have not yet formed an opinion on this matter. 0
B) No. 3
C) Maybe. 1
D) Yes. 1

6) Let's assume that our decides to use this tool, and that it remains activated after the testing phase has been completed. Should access to this feedback tool then be restricted to just some application users?

A) I have not yet formed an opinion on this matter. 0
B) No. 1
C) Maybe. 1
D) Yes. 3
7) Do you believe that the feature of entering comments anonymously will be used by application users?
   A) I have not yet formed an opinion on this matter. 0
   B) No. 0
   C) Maybe. 3
   D) Yes. 2

8) Let's assume that our company decides to use this tool, and that is used with an application that you use every day. Do you believe that you would regularly read the contents of the tool's welcome page?
   A) I have not yet formed an opinion on this matter. 0
   B) No. 3
   C) Maybe. 0
   D) Yes. 2

9) Does it make sense to use this tool as an alternative to email?
   A) I have not yet formed an opinion on this matter. 0
   B) No. 2
   C) Maybe. 1
   D) Yes. 2

10) Does it make sense to use this tool as an alternative to telephone conversations?
    A) I have not yet formed an opinion on this matter. 0
11) Does it make sense to use this tool as an alternative to telephone meetings?

A) I have not yet formed an opinion on this matter. 0
B) No. 3
C) Maybe. 2
D) Yes. 0