Effective Communication of Software Development Knowledge Through Community Portals

Christoph Treude, Margaret-Anne Storey
Dept. of Computer Science, University of Victoria
ctreude@uvic.ca, mstorey@uvic.ca

ABSTRACT

Knowledge management plays an important role in many software organizations. Knowledge can be captured and distributed using a variety of media, including traditional help files and manuals, videos, technical articles, wikis, and blogs. In recent years, web-based community portals have emerged as an important mechanism for combining various communication channels. However, there is little advice on how they can be effectively deployed in a software project.

In this paper, we present a first study of a community portal used by a closed source software project. Using grounded theory, we develop a model that characterizes documentation artifacts along several dimensions, such as content type, intended audience, feedback options, and review mechanisms. Our findings lead to actionable advice for industry by articulating the benefits and possible shortcomings of the various communication channels in a knowledge-sharing portal. We conclude by suggesting future research on the increasing adoption of community portals in software engineering projects.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management

General Terms

Documentation, Human Factors

Keywords

Community Portal, Knowledge, Documentation

1. INTRODUCTION AND MOTIVATION

Software development is knowledge-intensive [25], and the effective management and exchange of knowledge is key in every software organization. Knowledge is distributed through various artifacts and media forms, from formal documentation and technical articles, to blogs and wikis. However, many organizations struggle with the effective exchange and dissemination of knowledge across the community [15, 28].

Although various forms of documentation exist, software projects encounter many knowledge management challenges [9, 24]: How should knowledge be distributed? How should knowledge be kept up to date? How should feedback be solicited? How should knowledge be organized for easy access? There is no roadmap for what kind of information is best presented in a given artifact, and new forms of documentation, such as wikis and blogs, have evolved [20]. Unlike more formal mechanisms, wikis and blogs are easy to create and maintain. However, they do not offer the same authoritativeness that comes with traditional documentation, and they can become outdated and less concise over time [5]. While the informality of a wiki page or blog is sometimes enough, users often expect reviewed technical articles.

One mechanism that has emerged recently and brings various communication channels together is the use of web or community portals. Such portals are not just used in software communities, but are essential to companies, such as Amazon.com, eBay or TripAdvisor, where they enable the development of communities around products. Similarly, many of today's software projects wish to solicit feedback and input from a broader community of users, beta-testers, and stakeholders. While several projects use community portals, such as Microsoft's MSDN[1] or IBM's Jazz[2], the use of portals for software development projects has not been studied yet. It is unclear how portals can play an effective role in software development, and how different communication channels and artifacts should be utilized.

This paper presents a first study of a successful software project – IBM's Rational Team Concert – and its use of a community portal. Our methodology follows a grounded theory approach, using data from 13 semi-structured interviews with developers, ethnographic field notes gathered during observations on-site, and quantitative data on the content of the community portal. Based on our findings, we develop a model of artifacts in a community portal that characterizes the artifacts along different dimensions, such as intended audience, content type, feedback options, and review mechanisms. We provide actionable advice on how a community portal can provide benefits to a software project, and we discuss how the different channels and artifacts available in a portal can be used effectively. We also identify the shortcomings of different artifacts, and suggest improve-

[1] http://msdn.microsoft.com/en-us/
[2] https://jazz.net/pub/index.jsp

ESEC/FSE'11, September 5–9, 2011, Szeged, Hungary.

ments to current tools and processes. With this research, we aim to assist managers and developers in their decisions about using community portals.

The remainder of this paper is structured as follows. Related work is summarized in Section 2. In Section 3, we describe the community portal case study, and in Section 4, we introduce our methodology. In Section 5, we report our findings and describe a knowledge artifact model. We discuss our findings in Section 6, and we report limitations of our work in Section 7. Section 8 concludes the paper and outlines future work.

2. BACKGROUND AND RELATED WORK

We discuss three areas that are key to our research: knowledge management, documentation in software development, and the use of community portals to share information.

2.1 Knowledge in Software Development

Knowledge management has long been recognized as essential to software development [14]. Rowley [26] defines knowledge management as *"concerned with the exploitation and development of the knowledge assets of an organization with a view to furthering the organization's objectives. The knowledge to be managed includes both explicit, documented knowledge, and tacit, subjective knowledge."* How knowledge is managed in an organization depends on the particular style of software development. Plan-based or traditional methods usually rely on the management of explicit knowledge, while Agile methods rely on the management of tacit knowledge [22].

The number of studies on knowledge management in software development projects is limited. An overview of knowledge management in software engineering was conducted by Rus *et al.* [27]. The focus of their review is on motivations for knowledge management, different approaches to knowledge management, and factors that are important when implementing knowledge management strategies in companies.

Dingsøyr and Conradi [8] report on a literature survey of the lessons learned on the actions taken by companies, the effects and benefits of these actions, and descriptions of the strategies for knowledge management. Bjørnson and Dingsøyr [2] also identified the following main finding across several papers: there is a need not to focus exclusively on explicit knowledge but also on tacit knowledge.

Within knowledge management, the transfer of knowledge is particularly challenging. As described by Komi-Sirviö *et al.* [17], sharing knowledge is difficult in the light of short-term goals and companies often fall back to needs-based approaches for knowledge transfer. Knowledge transfer is essential when integrating new developers into a team. In a recent study using grounded theory, Dagenais *et al.* [4] identify early experimentation, internalization of structures and cultures, and progress validation as the main factors for the integration of newcomers. They also found that documentation was often not helpful and outdated.

Our research focuses on the knowledge captured in a community portal and on how to present it effectively.

2.2 Documentation in Software Development

Documentation has long been prominent in the list of recommended software engineering practices [6]. In 1986, Parnas and Clements [23] described a design process in which documentation plays a major role. They argue that many developers regard documentation as a necessary evil, written only as an afterthought to adhere to bureaucratic regulations. Therefore, documentation ends up being incomplete and inaccurate. Parnas and Clements identified four organizational issues that lead to those problems: poor organization of the documents which makes it harder to maintain them, boring prose that leads to inattentive reading, confusing and inconsistent terminology, and myopia caused by authors who know the documented system too well to take a comprehensive point of view.

In a survey of software professionals, Forward and Lethbridge [10] found that content, up-to-dateness, availability, and the use of examples are the most important document attributes. In a related study, Lethbridge *et al.* [19] explored how software engineers use and maintain documentation. They found that most software engineers do not update documentation in a timely manner, with the exception being highly-structured, easily-maintained forms of documentation, such as test cases and in-line comments.

Kajko-Mattsson [15] reports on a study in which she found poor documentation practices, even though her interviewees understood the necessity of documentation. Her participants identified the lack of a detailed documentation model as the main problem. Similar results were found in a study by Visconti and Cook [28]. They discovered a maturity gap between policies and the adherence to those policies.

Dagenais and Robillard [5] conducted a study to understand how documentation in open source projects is created and maintained. Among other findings, they report that the use of wikis for documentation has several shortcomings, such as a lack of authoritativeness, and that the use of a separate documentation team results in naming inconsistencies and out-of-date documentation.

Recent trends, such as agile documentation [1], suggest that executable products should be preferred over static documentation. From the agile point-of-view, documentation should be just good enough and it should only be updated *"when it hurts"* [1]. The rationale behind this is that the fundamental issue is communication, not documentation, and that comprehensive documentation does not ensure project success. While the developers in our study did not use agile documentation, only some of the documentation was formally organized, whereas other parts of the community portal evolved without strict rules.

2.3 Research on Community Portals

Community portals have become a platform used by online communities for the dissemination of information on the Web [13]. Members of a community can find relevant information on portals and can often contribute their own content. These community portals attract a disproportionately large amount of web traffic and engage visitors with free content [7]. Kondatova and Goldfarb [18] identify three objectives of such portals and the communities surrounding them: to supply content to users, to encourage members to contribute, and to facilitate communication and interaction. Portals aim to meet these objectives by offering various forums, wikis, mailing lists, and other communication mechanisms [18]. As Millen [21] points out, the archive of conversation in an online community is often a valuable information resource.

To date, research on community portals has focused on areas such as education and e-Government. Katz [16] exam-

ines how the use of portals allows educational institutions to integrate information and services used by their communities, and aims to guide institutions in their decisions about using portals. Gant and Gant [12] found the responsibilities of simultaneously providing breadth and depth in online content to be a challenge for governments using portals.

As mentioned earlier, the use of community portals by software development projects has not been studied yet. Several open source projects, such as Eclipse[3], PHP[4] or the Apache HTTP Server[5], offer portals that bring together a variety of artifacts, such as bug trackers, mailing lists, forums, wikis, and events. More recently, commercial products, such as Microsoft's Visual Studio or IBM's Jazz, have started using community portals. For example, the MSDN portal[6] features a library with API references, code samples and tutorials, a learning section, forums, and a blog. Next, we describe the IBM Jazz community portal in more detail.

3. THE COMMUNITY PORTAL

IBM's Jazz is a technology platform for collaborative software delivery and the development of products on top of the platform is made transparent through the community portal jazz.net. The first product developed on the Jazz platform is Rational Team Concert (RTC), a software development team collaboration tool that was first released in 2008. Recent additions to the Jazz platform include products such as Rational Insight and Rational Build Forge. In our study, we focus on RTC as it is the most mature of the Jazz products.

Jazz.net hosts the entire development process including work items, developer mailing lists, the development wiki, forums, the team blog, and a library. The library features technical articles, podcasts, presentations, videos, and official product documentation, with tags organizing all artifacts in the library. All artifacts on jazz.net contain information that indicates their associated product, with some artifacts belonging to more than one product. Figure 1 shows a screenshot of one of the portal pages. All content is accessible to users after a free registration.

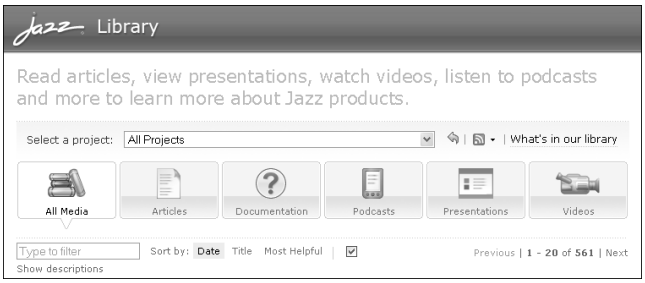

Figure 1: The IBM Jazz community portal

Technical articles are written by developers and product experts, and explore tasks, use cases, solutions, and concepts in depth. Podcasts stem from various sources, including news and updates about Jazz products, and often feature Jazz developers and managers. The presentations have been given by developers and product managers at conferences, or they are simply informational. Videos are produced by Jazz developers or experts, and typically demonstrate the use of Jazz products or introduce enhancements. The official product documentation consists of HTML content that is part of the product releases. The content is produced by a separate documentation team and is also accessible through the product help menus. Table 1 shows the different RTC-related artifacts that were available on jazz.net at the time of our study. Developers started to contribute to the community portal in the summer of 2007.

Table 1: RTC-related artifacts on jazz.net

type	amount
forum	8,413 threads
wiki	3,334 pages
mailing list	3,292 messages
blog	77 posts
official product documentation	4 manuals
library	93 articles
	58 videos
	30 presentations
	7 podcasts

4. RESEARCH METHODOLOGY

Here, we outline the research methodology used to understand how software developer knowledge is communicated through the Jazz portal. We cover the setting of our study, as well as the data collection and analysis methods used.

4.1 Research Setting

Our study took place with the development team of Rational Team Concert at IBM in the summer of 2010. The team consists of approximately 150 contributors in about 30 functional teams, with some teams acting as sub-teams of larger teams, and with many contributors assigned to multiple teams. The team members are located at approximately 15 locations, primarily in North America and Europe. The development follows the *"Eclipse Way"* development process [11]. Created by the Eclipse Development team, this process is an agile, iteration-based process with a focus on consistent, on-time delivery of software through continuous integration, testing, and incremental planning.

4.2 Data Collection

Our methodology followed a mixed methods approach, collecting both quantitative and qualitative data to allow for triangulation. To gather quantitative data on jazz.net, we used web scraping to download the artifacts shown in Table 1, along with their tags. At the time of our data extraction, the RTC team was working on milestones towards the 3.0 release. The quantitative data was collected to gain initial insights into the kinds of artifacts available on the community portal. These insights were used to guide our qualitative data collection that was done through a series of 13 semi-structured interviews. Seven interviews were conducted in person at an IBM location, and the remaining six interviews were conducted by phone. We interviewed five developers, four component leads, the RTC development manager, the RTC project administrator, and two client developers that work on Jazz-related projects inside IBM. For the remainder

[3]http://www.eclipse.org/
[4]http://www.php.net/
[5]http://httpd.apache.org/
[6]http://msdn.microsoft.com/en-us/

of this paper, we use D1 to D5 to refer to the developers, L1 to L4 to refer to the component leads, M1 and M2 to refer to the individuals with project-wide tasks, and C1 and C2 to refer to the client developers.

The initial interview script contained about 40 questions on participants' use of jazz.net[7]. As expected with grounded theory, new questions emerged which led us to refine the interview questions used during the course of the study. We asked our interviewees how they learn about Jazz functionalities, what kind of questions they ask using jazz.net, and how they use the published artifacts. We also investigated whether they had ever contributed content, and if so, who or what had triggered that contribution. We asked if there were perceived gaps in the documentation, and about potential tool and process enhancements. We used follow-up questions for clarifications and additional details, trying to understand the scenarios that had led to the creation and use of different artifacts. In addition, one of our researchers spent three months on-site, frequently engaging in informal discussions with developers regarding their use of jazz.net, and facilitating the member checking of our findings. These observations were recorded using ethnographic field notes and allowed for insights into the internal processes that were not revealed in interviews, in particular the processes that would ultimately lead to the creation of new artifacts.

4.3 Data Analysis

Our data analysis followed the grounded theory approach as described by Corbin and Strauss [3]. Grounded theory implies that data collection and analysis are interrelated processes, and that concepts are the basic units of analysis. These concepts are obtained using *"open coding"* in which the collected data is conceptualized line by line and concepts are only created when they are present repeatedly. We applied open coding to the transcripts of our interviews, to the data downloaded from jazz.net, and to the field notes collected on site. Based on the concepts, more abstract categories are developed and related. Each category is developed in terms of its properties, dimensions, conditions, and consequences. In the next step, called *"axial coding"*, data is put together in new ways, thus making explicit connections between categories and sub-categories. Sampling in grounded theory is done on theoretical grounds where incidents and events are sampled rather than subjects or data sources. In the final step of *"selective coding"*, the core category is identified and systematically related to other categories. Since all findings in a grounded theory study are linked to specific evidence, we are able to attribute all the findings we reported to interview quotes, field notes, or specific content on jazz.net. We considered all quantitative and qualitative data collected in our study during the analysis.

5. MODEL OF KNOWLEDGE ARTIFACTS

The *"core category"* identified in our grounded theory study is a set of key characteristics of different artifacts in a community portal. We can distinguish the artifacts along eight different dimensions that emerged from the axial and selective coding as part of our data analysis. The dimensions underline the particular role of each artifact.

[7]The set of questions used in the interviews is available online at http://tinyurl.com/jazz-net.

Content: the type of content typically presented in the artifact.
Audience: the audience for which the artifact is intended.
Trigger: the motivation that triggers the creation of a new artifact.
Collaboration: the extent of collaboration during the creation of a new artifact.
Review: the extent to which new artifacts are reviewed before publication.
Feedback: the extent to which readers can give feedback.
Fanfare: the amount of fanfare with which a new artifact is released.
Time Sensitivity: the time sensitivity of information in the artifact.

For the official product documentation, technical articles, blog posts, and the developer wiki, we analyzed these dimensions in detail. Figures 2 and 3 summarize the findings and show an example for each kind of artifact. We focus on these particular kinds of artifacts because they are also common in other community portals and in software development in general. The findings can be traced back to quotes from the interviews we conducted, and the interviewees are noted in the figures and text using subscript. Where relevant, we also discuss the artifacts and the ethnographic field notes that we analyzed related to the dimensions. The following sections report on each of the dimensions in detail. In addition, Section 5.9 highlights several explicit and implicit connections between the artifacts.

5.1 What content is communicated?

Content is the first dimension along which the different artifacts of the community portal can be distinguished. The official product documentation is based on **features** and does not cover scenarios: *"I would try to cover all the corner cases, and I found those were often omitted in the help doc because the effort to get into those corner cases were lengthy things. You would have to have a whole section of how to put yourself into a corner, and then a whole other section of how to get yourself out of this corner."*$_{D5}$ However, documenting scenarios is important: *"It's always good to document a widget, but it's more important in many cases to document a process and being able to follow the process of how you do an upgrade or how you do a plan and all the widgets that you touch during that. [...] It's the context of how you use the widget that's much more important."*$_{C2}$

This need is addressed by technical articles that feature **scenarios** and offer more depth than the official documentation. D4 described the reason to write an article: *"Because I wanted to put in a lot of screenshots, and also explain what's happening. [...] You had to write some text around it. Why you're doing something and why is that needed."* Technical articles also differ in their style of writing: *"Articles spice it up a little. There's more pictures and more personal I guess. People have their own style."*$_{D1}$

Tags also shed light on the content of different kinds of artifacts. The tags are assigned by a single person$_{L3}$. Table 2 shows the most-used tags per type. Videos are typically used for high-level overviews or demos$_{L3,D2}$, whereas presentations are related to conferences.

Blogs add a **personal** note to the artifacts on the community portal: *"a personal view on something and not really documentation. [...] You want to make people aware of something or tell them about something, but more like a*

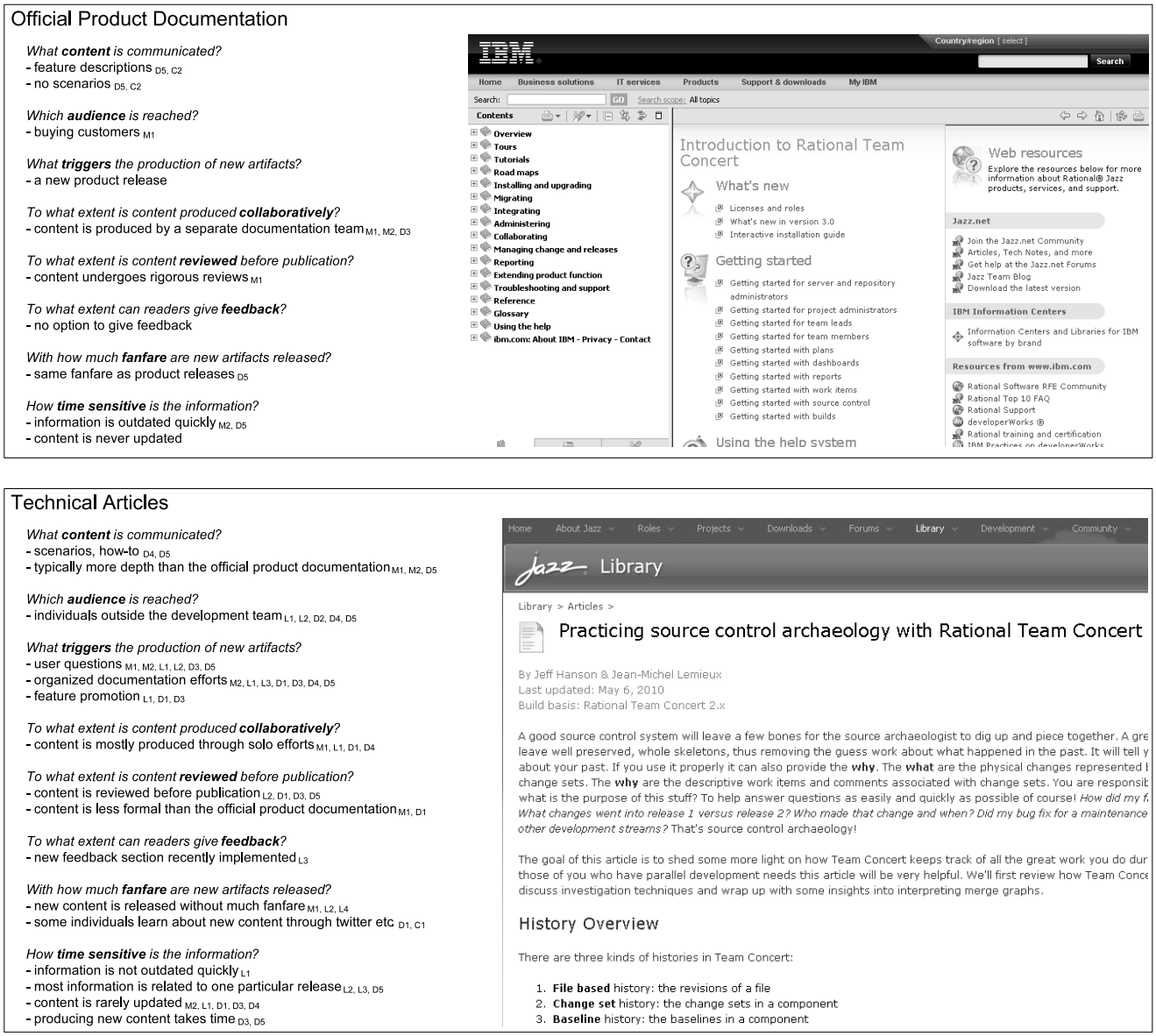

Figure 2: Model of Artifacts in a Community Portal—Findings for Product Documentation and Articles

Table 2: Most-used tags in jazz.net library by type

type	top two tags
article	agile (15), getting started (15)
video	how-to (12), introduction (7)
presentation	RSC 2009 (9), agile (5)
podcast	agile (3), scrum (2)
official doc	help (4)

teaser, you can go somewhere else to find the actual documentation. It's more like you write about something new and cool and what you think about it."$_{L1}$ Therefore, the blog also plays a role in **marketing**$_{L4,D3,C1}$. The most commonly-used tags on blog posts are *rational-team-concert* (26 posts), *video* (13 posts), *self-hosting* (12 posts), and *conference* (11 posts). Most blog posts are related to RTC and provide a conference report or give a view on the Jazz team's self-hosting experience.

The developer wiki plays many roles in the internal development processes, ranging from **planning** to descriptions of **scenarios**, detailed **instructions** or lists of **references** to other resources. Occasionally, wiki content goes beyond supporting development processes: *"He's written some really good wiki topics. The material there is really going beyond what we typically have in the wiki and really should be as help or as articles."*$_{L2}$

5.2 Which audience is reached?

A community portal caters to a diverse audience: *"A good percentage of our community are internal IBMers, business partners, students, academics as well as customers."*$_{L3}$ From a documentation point of view, the distinction of **end users**

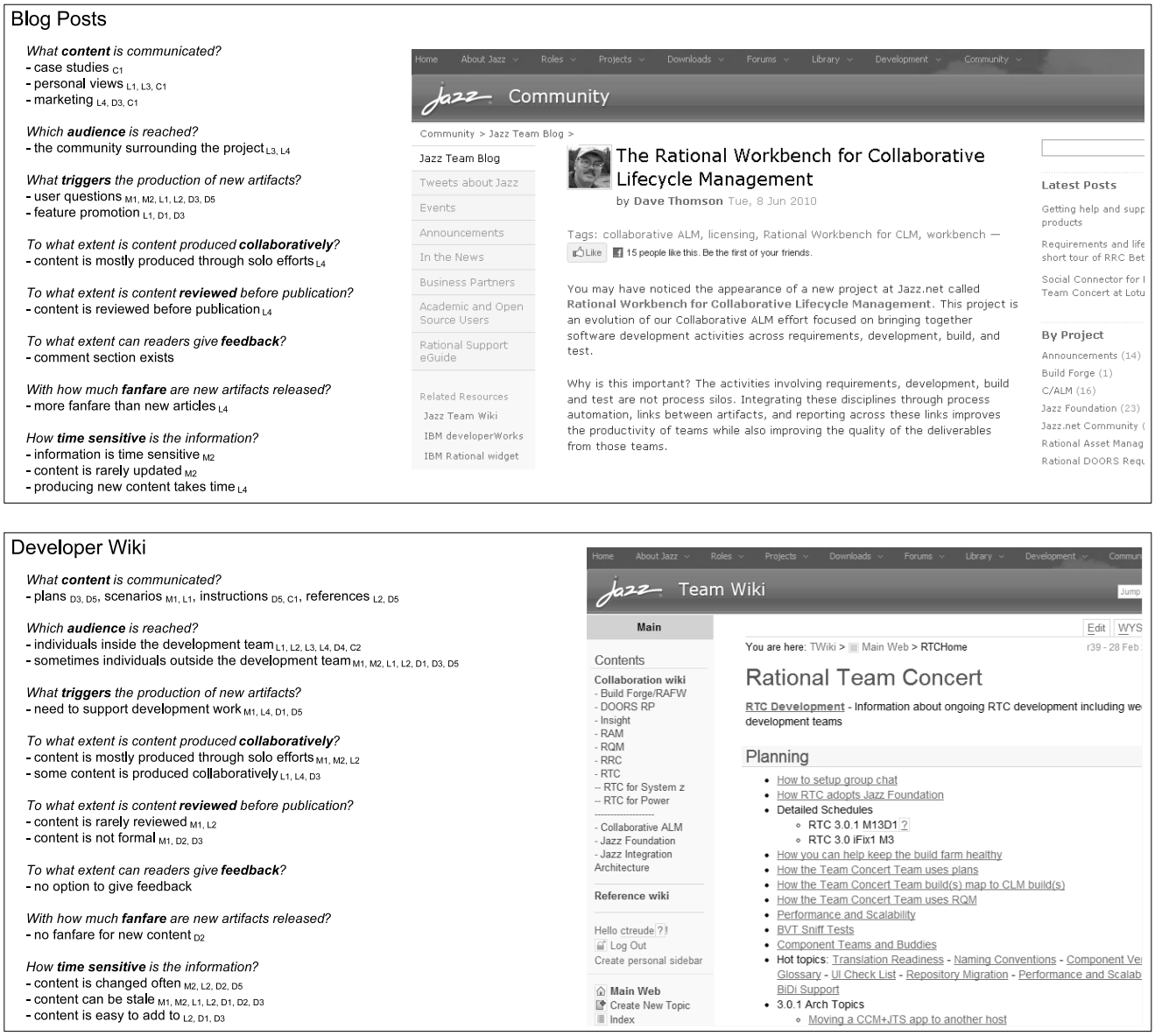

Figure 3: Model of Artifacts in a Community Portal—Findings for Blog Posts and Developer Wiki

and **client developers** is particularly interesting: *"There's two kinds of customers. We have customers who are using the product and trying to do something, and then we have third party developers. And third party developers are – for them to see the wiki makes sense, because we have instructions on how to make stuff."*$_{D3}$ In addition, some communication with customers is done outside of the community portal for confidentiality reasons$_{L2}$.

The intended audience of the official product documentation, technical articles, and the blog is **outside** of the development team. The intended audience for wiki pages is more difficult to determine, and some of the developers have contradicting views on the role of the wiki: *"For the wiki it's developers [...] that I actually have on my [chat] list or where I can put a name to a face."*$_{D5}$ However, customers are occasionally pointed to wiki pages: *"We also point customers to wiki pages. But wiki pages are typically [more] interesting for development teams. It's not as polished usually as articles and probably also not as maintained."*$_{L1}$ For customers, it is important to distinguish between official and informal wiki content: *"I'm not sure if customers should be taking that [wiki page] and making plans, planning their day on that."*$_{D5}$ However, outdated wiki content is of concern: *"Once or twice I've had customers say I'm trying to do stuff from the design document. And it's structured as instructions which is dangerous because they're not correct."*$_{D3}$

5.3 What triggers the production of new artifacts?

There are no formal conventions as to what knowledge should be externalized in technical articles, blog posts, or wiki pages: *"We haven't gotten to the point where we've had to dictate how often certain articles are developed. The content has flown fairly actively, especially with videos. People*

are getting in the habit of delivering release videos at each release with demos and how-to articles. So so far it's been able to happen organically."$_{L3}$ Articles are not written systematically but *ad hoc*$_{D3,D4}$, and without a formal process to trigger articles on certain topics, choosing the right things to document is subjective: *"The features we choose to work on, they're chosen in an ad hoc manner and we do the same thing with our documentation."*$_{D3}$ Initially, contributions to the blog were organized: *"When we first started doing it, a [...] bunch of us were pulled together and were asked to contribute."*$_{L4}$ However, blog posts now occur in an *ad hoc* manner as well, and over the last three years, there have been 43 different blog authors.

In the following paragraphs, we discuss common motivations for producing content that emerged from our study: user questions, organized efforts, self-promotion, and support of development work.

Knowledge is externalized in the form of articles and blog posts when users repeatedly ask the same **question**$_{M1,M2,L1,L2,D3,D5}$, or when developers expect many questions on a subject: *"Because we have a feeling that there will be so many users that have the same questions that it will be more beneficial to sit down, write the article once."*$_{D5}$ Questions arise from the forum$_{M2}$, they are reported back from customer representatives$_{L3}$, they come from feedback to other parts of the documentation$_{D4}$, or they come by email: *"I try and take every email conversation that I end up having and put it into an article."*$_{M2}$ The motivation for creating wiki pages is similar: *"I wanted not to be the sole source of this information. I wanted them to have a place that they could go and look and refer back to."*$_{D2}$

Some of the articles are the result of **organized efforts** among developers: *"There are efforts internally where groups of people get together and say, what's missing? [...] What are the customers asking for? We need this article or that article. [...] Some of the articles come up that way, and then other articles just happen organically as the team decides [...] they want to write an article on a particular topic."*$_{L3}$ Several of our interviewees reported that they had prompted articles from other individuals$_{M2,L3}$ or that articles had been delegated to them$_{L1,D1,D3,D4,D5}$.

Technical articles and blog posts are also written by developers who want to **promote** a feature they authored$_{L1,D3}$ or who want to promote themselves: *"Part of my thought is also, when I put my name on something, if and when I ever want to be interviewed for something, I want it to be on my resume."*$_{D3}$ D1 had a similar motivation for his article: *"'Cause I wrote it I want to get exposure."*

The externalization of knowledge to the wiki mainly happens to **support development work**, either for an initial *"brain dump"*$_{D5}$, as a central entity for resources$_{D1}$, to facilitate a discussion$_{M1,D5}$, or to *"stage something that needs to be formalized"*$_{D5}$. L4 explains: *"It becomes kind of an open email if you will, to evolve an idea."*

5.4 To what extent is content produced collaboratively?

Artifacts on a community portal also differ in the amount of collaboration that happens around them. The official product documentation is written by a separate **documentation team**, a process which involves a certain amount of interaction with the core developers$_{D5}$. Authoring technical articles, blog posts, and wiki pages is usually a **solo** effort, and only involves other individuals for reviews. For selected wiki pages such as descriptions of parts of the system architecture, the amount of **collaboration** can be higher: *"There's one wiki page that we have on how to set up the development environment [...] and this is something where anybody can add information, and actually also does."*$_{L1}$

The author information on articles is another indicator of collaborative content. Out of 93 technical articles, 35 have 1 author associated with them, 9 articles have 2 authors, 2 articles have 3 authors, and 1 article has 5 authors. An additional 34 articles have at least 1 team name as author, and 12 articles do not have any author information.

5.5 To what extent is content reviewed before publication?

Since the official product documentation is part of the product release, it has to undergo a **formal** review process in which content is reviewed by technical writers as well as by the product teams of the documented components. *"They go through a pretty rigorous process of producing that content working with the various teams that provide the features. [...] So the quality of it is very good."*$_{M1}$ The official product documentation has commitment from the developers and it is also the only artifact that is translated into other languages. Such a process is not transferable to all kinds of artifacts as it requires a lot of resources and therefore only happens once for every major release.

Technical articles are **less formal**. While there are internal reviews before content is posted online, the style is more personal: *"The articles tend to be an easy read, they're more down to earth and personal and not like you learn at school how they have to be formal and very professional and impersonal."*$_{D1}$ Blog posts have to undergo a similar internal review process where one individual on the team proof-reads content before it is posted online. Initially, there was some guidance on blogging: *"When we first started writing blogs for jazz.net, someone who had been a blogger [...] before that, he basically gave us guidance in the Jazz sphere. He said read these guidelines before you write blogs."*$_{L4}$

Wikis are the **least formal** of the artifacts and are rarely reviewed. However, contributors are aware that the wiki is open to the public and some of the wiki content, such as details of new features, has to be coordinated with management$_{M1}$. The lack of review often leads to inconsistencies on the wiki: *"There's pages that I think they're just sitting there, there's just certain people who know about that – they're just often in their own little world."*$_{D2}$

5.6 To what extent can readers give feedback?

Feedback is another dimension that distinguishes the different kinds of artifacts on a community portal. While the official documentation does not allow for feedback, and the developer wiki is read-only for everybody outside the core developer team, a feedback section has recently been implemented for technical articles, podcasts, and other items in the library. 40 of the 58 videos on jazz.net are hosted on YouTube, which provides view counts (considered as implicit feedback) and allows comments. At the time of our study, the videos had a median of 1,074 views, ranging from 188 to 17,157[8]. Only 7 videos had comments, with a maximum of 2 comments per video.

[8]An introductory video for RTC, available online at http://www.youtube.com/watch?v=ILvsGQQqAF0.

Blog posts invite interaction with a **commenting** feature. At the time of our study, there was a total of 183 comments on 77 blog posts. 28 blog posts had no comments, 15 blog posts had 1 comment, and 7 posts had 2 comments. 27 posts had 3 comments or more, with a maximum of 12 comments per post. Blog posts with a lot of comments are typically product announcements or discussions of new functionality.

Community members can also **vote** on blog posts. At the time of our study, 446 votes had been cast. The most voted-for post had 22 votes, and only 4 out of the 77 posts had not received any votes. On average, articles received 4.6 stars out of 5, and only 6 articles received less than 4 stars. A recent addition is the option to *"like"* posts using Facebook Connect, and 4 out of the 10 newest posts had already received more than 10 likes at the time of our study. While the web traffic[9] for artifacts, such as technical articles in the library, is monitored as well, pure numbers do not allow conclusions about the usefulness of the content$_{L3}$.

Feedback is often generic: *"I got a thumbs up, like good thing to do, but not any specific critique on my writing. More like, that's a good thing to do, keep doing that."*$_{D2}$ Some developers even go out of their way to search for feedback: *"There's no feedback and I don't really know what's happened, I don't know if people read [the article]. [...] Then by searching just to see how exposed it was, I found a few forum discussions on jazz.net [about the article]."*$_{D1}$

While the forum and the article comment feature were designed for users to give feedback$_{L3,D4}$, feedback comes through different channels$_{L1,L4,D3}$. Questions about content on jazz.net are mostly asked through direct communication channels, such as chat$_{L1,D5}$ or email$_{M2,L2}$. The customers often work within IBM and therefore have access to the internal instant messaging system$_{L1,C1}$.

Developers are aware that putting their name on artifacts can lead to many questions$_{D3}$ or spam$_{D4}$. *"Usually once you have documented something that people are using, then they come back to you directly with questions."*$_{L1}$ During our ethnographic observations, we also encountered an instance of a developer deciding not to write an article because they anticipated a high workload due to questions and feedback.

5.7 With how much fanfare are new artifacts released?

Artifacts also differ in the fanfare with which they are released and in the number of people that are aware of a new artifact after its creation. While the official product documentation is released alongside the products once a year, blog posts have a higher impact than articles. *"Articles are a little bit quieter."*$_{L4}$ This is confirmed by developers who are not aware of new articles$_{M1}$ or only read articles when prompted$_{L2}$. Other developers follow the JazzDotNet twitter account to learn about new articles$_{C1}$, or they check the feed on the front page of the portal$_{D1}$. New or changed wiki pages have the lowest amount of fanfare as the wiki implementation used by this group does not offer notifications.

5.8 How time sensitive is the information?

Some artifacts are more time sensitive than others. The official product documentation gets **outdated** quickly, mainly because it is only updated once a year (see Table 1). Technical articles are less time sensitive due to the nature of the topics: *"I think the topics are usually chosen so that they stay valid for a longer time. [...] An article that describes for example customization of a work item, these are longer lived features that we do not break as easily."*$_{L1}$ For articles, knowing which release they belong to is important: *"We try to mark every article as relevant to a particular release."*$_{L3}$

For articles and blog posts, the **writing process** takes a few days$_{L4,D3,D5}$, and there is more flexibility on when they are published: *"Sometimes we write articles after the fact. So we didn't get time, we delivered the feature late or something, or we didn't get to document it in time for the release, so then we say OK, we're going to write an article."*$_{M1}$ Developers also choose to write articles after the code freeze point at the end of a release cycle$_{D3}$. However, the review process can lead to publishing delays: *"I've had experiences in the past where I want to get something out in timely way but because of the review it doesn't necessarily get there."*$_{L4}$

Technical articles, blog posts, and wiki pages contain tools to **update** content, but most authors in our study have never updated their articles$_{L1,D1,D3}$. Since the website is hosted by a specific team, developers were unsure what the article change process would be after the initial publication$_{D1}$. In other cases, developers passed on the ownership of an article before they switched teams to make sure updates would happen: *"For the second release of the product, one of my team members here updated the article. [...] He took ownership."*$_{D4}$ Blog posts are more time sensitive than articles, but only one of the developers in our study had made updates to a blog post$_{M2}$. This is because blogs express personal views and usually do not feature technical details$_{L1,L3,C1}$.

In contrast, wiki pages are quick and easy to create and modify: *"I think the wiki is good 'cause it can be sort of fast and loose for getting content out there and the articles tend to be more reviewed, and they're also harder to change after the fact."*$_{L2}$ There is no formal process outlining when wiki pages should change, and some developers only update their wiki pages if they are reminded by somebody else$_{M2,D3}$ or if they happen to see that something is wrong$_{D1}$. Updates are usually done to describe new features$_{M2}$, to answer questions$_{L2}$, or to reorganize a few pages$_{L2}$. Wiki pages aimed at customers are kept current$_{L2}$ and the index is also kept up to date$_{M2}$.

Readers cannot expect all wiki pages to be up to date: *"If it's incorrect I would say, it's a wiki, you can't expect that to be correct."*$_{D5}$ A wiki page implicitly conveys the **uncertainty** of its content: *"It helps us to communicate to customers that it's actually not done yet."*$_{M1}$ As observed by many of our interviewees, wiki pages can have stale content: *"We write a topic on something and then we walk away from it, we just ignore it."*$_{D3}$

5.9 Artifact Connections

Artifacts on a community portal, with all their characteristics as shown in Figures 2 and 3, cannot be treated separately. There are several explicit and implicit connections between them. For example, content or structure can be reused from different artifacts. The structure for an article can come from a wiki page$_{D5}$, a blog post can be distilled from wiki pages$_{L4}$, wiki pages can draw content from forum posts$_{L2}$, and forum posts can be referenced in work items$_{D1}$.

Several wiki pages have the potential to be turned into technical articles or blog posts, but that does not happen often: *"If we had something that we were formulating on*

[9]Data on web traffic was not made available to us.

the wiki and we're getting feedback and so forth and we're altering it over time – once we have decided that it's final, we could basically copy and paste all the stuff and put it in a better looking web page. An article possibly, depending on what it is, or some official document. I can't actually think of specific cases where it has [happened]. [...] We tend to leave a lot of the stuff [on the wiki]."$_{M1}$ Another example is an article created by L4: *"[I added] an article that basically has no content just a description and a link to our wiki [...] using all the keywords that make sense to find it. And that way when somebody goes in looking for this information, [...] we point them into the wiki, which isn't ideal but it does surface the content."*

Relationships between artifacts also exist from a client developer's perspective. C2 describes using the wiki and technical articles as his first place to go for questions, and posting to the forum if he cannot locate an answer: *"That's the fallback. And if I don't get a response in the forum I actually will post a defect or an enhancement."*

6. DISCUSSION

One of the goals of our research is to provide advice to managers and developers on how to effectively use a community portal. In this section, we present advice on the findings from our case study of a successful software project that leverages a community portal.

Make content available, but clearly distinguish different media forms.

One major advantage of a community portal is the accessibility of all artifacts: *"Because it's actually just as likely that an answer for a question is going to be in a forum or in a mailing list as is it going to be in a wiki."*$_{C2}$ Even though not all content is produced with the intent to contribute to documentation, it can help developers inside and outside of the team to understand a system.

However, wiki pages that are created to support development work and that are often not as well structured as other documentation sources on the community portal should be flagged as draft material. This distinction is currently not always clear: *"So people do have confusion about what's real and what's – what's been approved versus just draft."*$_{M2}$ Individuals from outside the core development team find it difficult to navigate the wiki$_{C2}$ and they encounter stale content$_{C1}$. The lack of structure is also a concern for developers on the team$_{D2,D3}$.

Clearly distinguishing different media forms is also a challenge for search interfaces: *"If you do a search, you're confronted with a blend of all the media and different types of it."*$_{C1}$ Jazz.net offers a main search interface that searches the entire content of the community portal, as well as specific searches for the wiki, the library, and the forum. Most of our interviewees found the current search to be insufficient$_{L1,L2,L3,D2,D3,D4,C1,C2}$. As an alternative to search, the wiki index page is kept up to date$_{L2}$ and tags are used on artifacts in the library.

Make it easier to move content into more formal media formats.

Different media forms in community portals are often disconnected. Content is rarely transferred from informal wiki pages into more formal articles or into the official product documentation. Instead, developers create workarounds, such as articles, that point to a wiki page$_{L4}$. Often, good content is never published beyond wiki pages because developers lack the time to push for a new article or blog post$_{L2}$. Resources are wasted by replicating information that already exists in other parts of the community portal. Part of the problem is the use of a separate documentation team for producing the official product documentation. With the plethora of information on a community portal, the first step when writing a new piece of documentation should be to check if related content already exists.

Be aware of the implications of different media artifacts and channels.

As we have shown with the model of knowledge artifacts in a community portal, different media artifacts and channels have different implications (see Figures 2 and 3). To ensure the effective exchange and dissemination of knowledge, developers and managers need to be aware of these implications:

- Content in **wiki** pages is often **stale**. Therefore, readers will not look at the wiki for reliable information, but rather use it as a backup option if information is not available elsewhere. To communicate important information to the community, articles and blog posts are better suited.
- The **official product documentation** is **reviewed** rigorously. With that in mind, it can serve as the most reliable way to communicate knowledge in a community portal.
- When content is produced by a separate documentation team, **updates** may not be feasible. In such a case, information may become outdated quickly and can only be fixed with a new product release. Consequently, an update process for articles may be needed.
- In this project, new **blog** posts created more "buzz" or **fanfare** than articles or wiki pages. Thus, if there is a need to make a project's community aware of something, a blog post may be the best-suited medium.
- Writing can be **time consuming**, in particular for technical articles and blog posts. In addition, those media forms may need to undergo a review process. To get content out **quickly**, the **wiki** may be the best solution. However, readers may only find wiki pages if they are pointed to them explicitly.
- To solicit **feedback** from readers, articles and blog posts typically offer more **comment** functionality than the official product documentation or the wiki.

Offer developers a medium with a low entry barrier for fast externalization of knowledge.

Developers often prefer to externalize their knowledge in the form of wiki pages because wikis are easy to add to$_{L2,D1,D3}$. To encourage documentation, it is important that developers have such a medium at hand where they can externalize knowledge without having to worry about the correctness of content or whether customers will understand the entire context. While it makes sense to have official documentation artifacts, such as articles or blog posts,

undergo a review cycle, developers also need a platform for producing content with a lower entry barrier.

Involve the community in a project's documentation.

Unlike other forms of documentation, community portals allow for community involvement. While the number of documentation contributors outside the development team is still limited$_{M1,D1}$, there have been successful instances: *"I got someone [...] who had been asking me a lot of questions and I said, well, you write an article. I had a draft ready, but that was still a skeleton and I said, here's a draft, go write your own style to it. It took a while, [...] but he ended up taking all the screenshots, taking my text and doing stuff and I just reviewed it."*$_{M2}$ In particular, articles on topics such as best practices could be written by client developers and end users$_{M1}$. Such contributions could even be encouraged with incentives as they provide considerable value to the community portal$_{M1}$. Review processes would have to be in place to ensure quality. However, giving write access to community members is not realistic for all content, and in particular, not for the wiki. Other ways to increase community involvement could involve more options for marking up and commenting on content. In general, the entry barrier for community members to contribute to the documentation could be lowered$_{D3}$ by making the process more obvious and by offering easily-accessed web forms for contributions. Recent additions to jazz.net, such as commenting and voting features for library content, are steps in the right direction$_{L3}$, but more could be done to increase community involvement.

Provide readers with an option to give feedback.

Feedback on documentation often comes through direct communication channels. Readers try to contact the content author via email or chat rather than ask questions on the forum. While this was not intended, it underlines the importance of the social aspect of documentation. Readers want to know who authored certain content and they want to be able to contact those individuals. However, with different systems for blog, technical articles, official product documentation, and the wiki, it is difficult to locate all contributions of a single individual.

Our study has also shown that readers take advantage of lightweight ways to give feedback, such as rating of content or *"liking"* artifacts using Facebook Connect. These mechanisms have the potential to involve more readers and gather feedback on the quality and usefulness of content. In addition, implicit feedback, such as view counts, should be attended to.

7. LIMITATIONS

As with any research methodology, there are limitations with our choice of research methods. The first limitation of our study lies in the number of interviewees. However, we triangulated data from the interviews with artifact data from the community portal and with ethnographic field notes gathered during the three months spent on site with RTC developers. Also, the interviewees had different backgrounds, from managers to relatively new team members. We focused on the authors instead of the readers in this study to understand their use of the portal. Investigating the reader's perspective will be future work.

While the documentation landscape for RTC was fairly stable at the time of our study, new projects, such as Rational Insight, had been added to jazz.net shortly before our study. We focused our study on RTC as the mature part of jazz.net, but we cannot rule out that other projects might have influenced our findings.

Our conclusions are limited to the jazz.net portal community. However, Jazz is one of the first platforms that opened up its development process to the community without being open source, and therefore provides a unique case between open source and traditional closed source projects. Also, we focus on blogs, wikis, official product documentation, and technical articles in our study. All of these artifacts are widely used in different contexts to document software. Other community portals should be studied to gain further insights into their role in software development.

8. CONCLUSIONS AND FUTURE WORK

Community portals for software projects facilitate the exchange and dissemination of knowledge through many different kinds of artifacts. In this first case study of a software project community portal, we found that RTC developers need different channels, such as blog posts, wiki pages, technical articles and the official product documentation, to externalize different kinds of knowledge. Each of these artifacts play a role in capturing knowledge that can help users, client developers, and team members understand a system. Developers often externalize knowledge without being prompted by formal processes. They respond to questions, promote their part of the system or themselves, support development processes, or contribute through organized efforts.

Unlike traditional forms of documentation, community portals have the potential to involve a community composed of client developers and end users in the process of externalizing developer knowledge. Our study of jazz.net has shown first evidence of such involvement, and it has also highlighted areas where tools and processes could be improved to encourage more participation.

Recent feature additions, such as mechanisms to *"like"* articles, to vote on blog posts and articles, and to comment on parts of the documentation, are further steps towards involving external individuals in the process of documenting software. Further studies will have to be conducted to understand the effects of such mechanisms. In addition, we need to consider the customer's perspective on the role of community portals, and we need to examine other community portals to fully understand how the burden of documentation can be lessened by sharing it among a larger group of contributors with different roles and responsibilities.

We expect the role of community portals to increase in both open source and closed source projects. The findings from our study and from future studies will help improve how they are used to support communication and collaboration in software engineering projects.

Acknowledgments

We wish to thank the team that granted us access to their repositories and conducted interviews with us. This research is supported by a fellowship from IBM. We also appreciate suggestions from Gargi Bougie and Nancy Songtaweesin as well as the logistical support and editing by Cassandra Petrachenko.

9. REFERENCES

[1] S. W. Ambler. Agile/lean documentation: Strategies for agile software development. http://www.agilemodeling.com/essays/agileDocumentation.htm, accessed in March 2011.

[2] F. Bjørnson and T. Dingsøyr. Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. *Information and Software Technology*, 50(11):1055–1068, 2008.

[3] J. M. Corbin and A. Strauss. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, 13(1):3–21, 1998.

[4] B. Dagenais, H. Ossher, R. K. E. Bellamy, M. P. Robillard, and J. P. de Vries. Moving into a new software project landscape. In *ICSE '10: Proc. of the 32nd Intl. Conf. on Software Engineering*, pages 275–284, New York, NY, USA, 2010. ACM.

[5] B. Dagenais and M. P. Robillard. Creating and evolving developer documentation: understanding the decisions of open source contributors. In *FSE '10: Proc. of the 18th Intl. Symp. on Foundations of software engineering*, pages 127–136, New York, NY, USA, 2010. ACM.

[6] S. C. B. de Souza, N. Anquetil, and K. M. de Oliveira. A study of the documentation essential to software maintenance. In *SIGDOC '05: Proc. of the 23rd Intl. Conf. on Design of communication*, pages 68–75, New York, NY, USA, 2005. ACM.

[7] R. M. Dewan, M. L. Freimer, A. Seidmann, and J. Zhang. Web portals: Evidence and analysis of media concentration. *J. Manage. Inf. Syst.*, 21:181–199, October 2004.

[8] T. Dingsøyr and R. Conradi. A survey of case studies of the use of knowledge management in software engineering. *Intl. Journal of Software Engineering and Knowledge Engineering*, 12(4):391–414, 2002.

[9] B. Fluri, M. Würsch, E. Giger, and H. C. Gall. Analyzing the co-evolution of comments and source code. *Software Quality Control*, 17(4):367–394, 2009.

[10] A. Forward and T. C. Lethbridge. The relevance of software documentation, tools and technologies: a survey. In *DocEng '02: Proc. of the Symp. on Document engineering*, pages 26–33, New York, NY, USA, 2002. ACM.

[11] R. Frost. Jazz and the eclipse way of collaboration. *IEEE Softw.*, 24(6):114–117, 2007.

[12] J. Gant and D. Gant. Web portal functionality and state government e-service. In *Proc. of the 35th Annual Hawaii Intl. Conf. on System Sciences*, pages 1627–1636, 2002.

[13] S. Grzonkowski, A. Gzella, S. R. Kruk, J. G. Breslin, T. Woroniecki, and J. Dobrzanski. Sharing information across community portals with foafrealm. *Int. J. Web Based Communities*, 5:351–370, 2009.

[14] J. D. Herbsleb and D. Moitra. Guest editors' introduction: Global software development. *IEEE Softw.*, 18(2):16–20, 2001.

[15] M. Kajko-Mattsson. A survey of documentation practice within corrective maintenance. *Empirical Softw. Eng.*, 10(1):31–55, 2005.

[16] R. N. Katz. *Promise and Peril of Portal Technologies in Higher Education*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[17] S. Komi-Sirviö, A. Mäntyniemi, and V. Seppänen. Toward a practical solution for capturing knowledge for software projects. *IEEE Softw.*, 19(3):60–62, 2002.

[18] I. Kondratova and I. Goldfarb. Design concepts for virtual research and collaborative environments. In *Proc. of the 10th ISPE Intl. Conf. on Concurrent Engineering*, pages 797–803, The Netherlands, 2003. A. A. Balkema Publishers.

[19] T. C. Lethbridge, J. Singer, and A. Forward. How software engineers use documentation: The state of the practice. *IEEE Softw.*, 20(6):35–39, 2003.

[20] B. Leuf and W. Cunningham. *The Wiki way: quick collaboration on the Web*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[21] D. R. Millen. Community portals and collective goods: Conversation archives as an information resource. In *Proc. of the 33rd Hawaii Intl. Conf. on System Sciences - Volume 3*, page 3030, Washington, DC, USA, 2000. IEEE.

[22] S. Nerur and V. Balijepally. Theoretical reflections on agile development methodologies. *Commun. ACM*, 50(3):79–83, 2007.

[23] D. L. Parnas and P. C. Clements. A rational design process: How and why to fake it. *IEEE Trans. Softw. Eng.*, 12(2):251–257, 1986.

[24] A. L. Powell, J. C. French, and J. C. Knight. A systematic approach to creating and maintaining software documentation. In *SAC '96: Proc. of the Symp. on Applied Computing*, pages 201–208, New York, NY, USA, 1996. ACM.

[25] P. N. Robillard. The role of knowledge in software development. *Commun. ACM*, 42(1):87–92, 1999.

[26] J. Rowley. What is knowledge management. *Library Management*, 20(8):416–419, 1999.

[27] I. Rus and M. Lindvall. Guest editors' introduction: Knowledge management in software engineering. *IEEE Softw.*, 19(3):26–38, 2002.

[28] M. Visconti and C. R. Cook. An overview of industrial software documentation practice. In *SCCC '02: Proc. of the XII Intl. Conf. of the Chilean Computer Science Society*, page 179, Washington, DC, USA, 2002. IEEE.