JOURNAL OF ORGANIZATIONAL COMPUTING
AND ELECTRONIC COMMERCE *16*(3&4), 343–363 (2006)

Virtual Community Use for Packaged Software Maintenance

Helena Holmström
Viktoria Institute

Brian Fitzgerald
University of Limerick

In this study, we investigated the use of virtual communities for involving distributed customers in the maintenance of packaged software. On the basis of an empirical study, we suggest that virtual communities can be usefully leveraged for corrective, adaptive, and perfective software maintenance. Specifically, the virtual community allowed for quick discovery of bugs and a rich interaction between developers and customers in the categories of corrective and adaptive software maintenance. However, although contributing also to the perfective category of software maintenance, this was the category in which several customer suggestions for modification were actually ignored by the developers. This implies that community use is indeed beneficial for maintenance related to coding and design errors as well as for maintenance of an adaptive character. However, it has limitations when associated with major changes such as software functionality addition or modification as those experienced in the category of perfective maintenance.

packaged software, software maintenance, virtual communities

1. INTRODUCTION

Despite research that has suggested that software maintenance consumes between 40% to 75% of the total resources in software development (see, e.g., [1–4]), maintenance appears not to be highly regarded by software programmers or their managers [5]. One of the pioneers in the software field, Ed Yourdon, captured this well in the contention that for many programmers, maintenance was a fate worse than death, a view reinforced by Schneidewind [6] who suggested that to be identified as working as a maintenance programmer was equivalent to being perceived as having bad breath. This results in the paradoxical situation that although maintenance of existing software is arguably more intellectually challenging than development of new software, the most junior and inexperienced programmers are frequently charged with the task. Thus, any initiative that can facilitate the maintenance task deserves to be examined closely.

Correspondence and requests for reprints should be sent to Helena Holmström, Viktoria Institute, Gothenburg, Sweden. E-mail: helena.holmstrom@viktoria.se

Much research has been conducted on the maintenance topic, and a tripartite typology of corrective (repairing faults after delivery), adaptive (adapting the software to new operating environments), and perfective (adding to, or modifying, software functionality) maintenance has been widely adopted (e.g., [1, 3, 7]). This research has identified a number of factors that could contribute to easing the maintenance task including assessment of software maintainability [8], factors associated with software repair [3], and various maintenance tools [9]. However, this research has not typically focused on possible expansion of the role of the customer, a notable exception being the study of Hirt and Swanson [10] in which they investigated the expanded role of customers in the maintenance of ERP systems. Likewise, in the open source software (OSS) area, Schmidt and Porter [11] investigated the role that users could play in debugging, documentation, mentoring, and technical support. Given the suggestion that 60% of the time spent on a program modification request is consumed in locating the lines to change [9], any help that the customer community can provide in elaborating the nature of the problem and thus helping to identify the section of the program to be changed could be very beneficial. Although the difficulties of software maintenance have been the subject of much research to date, an important development in more recent times has been that maintenance increasingly takes place in the context of packaged software development. In this mode, customers form a diverse group who are often far removed from developers. Thus, all the traditional difficulties manifest in the maintenance process are further exacerbated. For example, a critical problem in software maintenance is the elicitation of changing customer needs and requirements [12, 13]. This is particularly true in relation to packaged software that is sold to pan-globally located customers. Recent research on innovation and distributed development suggested that innovation is stimulated by the diversity that can naturally arise by leveraging the expertise and diversity of geographically distributed customer groups [14]. This is particularly important in the case of perfective maintenance, as the identification of new functionality requires both innovation and creativity.

Recognizing these problems, our primary research objective in this study was to explore how virtual communities, as platforms for interaction, could be used to elicit changing customer needs and requirements in the maintenance process of packaged software and hence involve distributed customers in the software maintenance process. Although there is considerable research on virtual communities as beneficial to software development (see, e.g., the literature on OSS development by Raymond [15]; Feller & Fitzgerald [16]), in this article, we explore the specific deployment of virtual communities in the process of packaged software maintenance. Based on an empirical study of a computer game community, we illustrate how virtual communities can support the software maintenance process for corrective, adaptive, and perfective maintenance.

2. BACKGROUND

2.1 Software Maintenance

Pressman [17] stressed the importance of distinguishing between the maintenance process and the software configuration management process. According to Pressman [17], software maintenance is a set of software engineering activities that occur after software has been delivered to the customer. Changes are made in response to changed requirements, but the fundamental structure of the software remains stable. Software configuration management, on the other hand, refers to the set of tracking and control activities that are initiated when a software project begins and terminate only when the software is taken out of operation. Hence, software maintenance can be regarded as a subset of the software configuration management process, and in this article, we focus on the corrective, adaptive, and perfective maintenance processes.

In the corrective maintenance phase, coding errors, design errors, and requirements errors are handled. Although coding errors are relatively cheap to correct, design errors are more expensive, as they may involve the rewriting of several program components. Most expensive, however, are requirements errors because they might require extensive system redesign [18]. In contrast to a common belief, repairing system faults is not the most expensive maintenance activity. Studies have shown that only 17% of maintenance is concerned with correcting faults [3]. Rather, evolving the system to cope with new environments and to new or changed customer requirements consumes most maintenance effort.

Adaptive maintenance is required when there is a need to adapt the software to a different operating environment, for example, if some aspect of the system's environment such as the hardware, the platform operating system, or other support software changes or if other environmental changes require the adaptation of the software. In a study by Lientz and Swanson [3], it was discovered that about 18% of the maintenance work was concerned with software adaptation.

Finally, perfective maintenance is concerned with software functionality addition or modification. This type of maintenance is necessary in response to changes in customer requirements. According to Lientz and Swanson [3], 65% of the maintenance effort is distributed on functionality addition or modification due to changes in customer requirements. Hence, of critical concern to this process is the elicitation of changing customer needs and requirements. However, to elicit these is a complex process. Goals such as identifying system boundaries, identifying stakeholders, and identifying different customer groups are important but inherently difficult to accomplish. In addition to this, it is often the case that customers find it difficult to articulate their needs and requirements in an early stage of the process [13]. To help in this process, there are several elicitation techniques. Besides observations, questionnaires, interviews, and analysis of existing documentation, there are group elicitation techniques, prototyping, model-driven techniques, cognitive techniques, and contextual techniques [19]. Also, different techniques have been categorized as either "informal" or "formal" in which the informal approaches consist of face-to-face conversations between customers and developers, whereas the formal approaches consist of structured documents that are produced and signed off by the customers [18]. Often, changes to the software are implemented iteratively, and customers can be directly involved in testing new versions of the software. Requirements analysis, which is conducted on the basis of actual experience with a real system artifact, is a much richer experience and is more likely to yield more insightful and accurate requirements than the conventional model that typically requires customers to express their notional requirements in the absence of any detailed interaction with a real system artifact.

However, despite these techniques, the process of requirement elicitation and the ability to adjust to changing customer needs is still difficult. Furthermore, there has been a recent shift in software development processes and software products. To a large extent, software development is now performed by software vendors, and contrary to custom information systems in which made-to-order systems are built for specific customers, many software products of today are sold as packaged software, that is, tradable products intended for mass use [20]. As recognized by Sawyer [20], this will alter way we think about software development, and certainly this will have implications also for the process of software maintenance. Although the same categories of maintenance still have to be accomplished, this has to be achieved in cooperation with a distributed customer group that never interacts physically with the software developers. This implies that the process of software maintenance is different—and perhaps even more complex—from that described in traditional software engineering literature. Given the considerable effort and cost of software maintenance, it is worthwhile exploring alternative approaches for interacting also with distributed customers. In this article, we discuss how such interaction was achieved by using a virtual software community.

2.2 Software Communities

With interaction media such as e-mail, chat, and conferencing systems, computer networks of today allow for people to create a wide range of new social spaces. On the Internet, people engage in topic-specific discussions groups, play games, entertain one another, and even work on complex collective projects [21] such as software development [22].

In this article, we focus on software communities, that is, communities in which people interested in particular software products meet to collectively discuss these products and in some communities, also participate in developing these. Primarily, these communities can be found in relation to software such as Web infrastructure applications and computer games, and collective action is related to the performance of different development tasks such as debugging, modification, and improvement of that particular software. For example, in OSS communities, world-wide communities of software developers engage in developing software that can be freely shared for review, reuse, and modification. According to Sharma et al. [23], the OSS model is a fundamentally new way to develop software and one that provides unique opportunities in terms of developer base and user input. In the Apache HTTP Project, there are developers from Canada, Germany, Italy, the United States, and the United Kingdom [24], and recent studies on the Linux kernel development community show activity in more than 28 countries [25]. Based on the Linux case, OSS proponents argue that the model makes it possible for quality software to be produced in a short period of time, with little cost, and by some of the best programmers in the profession [23].

This has also encouraged for-profit organizations to try to build business models around the open source paradigm, and now companies such as Hewlett Packard, Intel, IBM, and so forth are helping create an open source development laboratory to promote OSS collaboration and growth. Typically, OSS communities develop Internet and Web infrastructure applications such as, for example, the Apache Web server and the Mozilla Web browser. The development process is iterative [23] and characterized by parallel development, prompt feedback to user and developer contributions, and the use of extremely rapid release schedules [16]. Furthermore, OSS community members value altruism, reciprocity, and gift giving; and although financial reward is the main motivation in conventional software development, this does not seem to be that significant for OSS community members. Instead, the personal benefit of using an improved software product is the driving force in OSS communities [23].

Also, software communities are found in relation to computer game development. As recognized by Scaachi [26], the release of DOOM onto the Web in open source form in the middle of the 1990s began what is recognized as the landmark event that started the development and redistribution of open software game variants, so-called PC mods. *Mods* are game variants that are created by small numbers of users who want to modify games instead of using them as they are provided. Today, the scope of mods has expanded to include entire new game types, game character models and skins (surface textures), levels (game play arenas), and artificial intelligence (AI) game bots (in-game opponents). As in OSS communities, game community members value trust and reputation, and to be generous with one’s time, expertise, and source code are valued traits of community participants [27].

In looking at these two examples, there is little doubt that software communities offer interesting opportunities in terms of involving customers in the software development process. Whereas open source communities allow for users to access the source code and modify the software, other software communities allow for electronic discussion forums in which software users provide each other and the software developers with important feedback on the particular software. In such communities, users do not modify the software themselves but contribute to the development process in terms of knowledge they acquired when using the software. As recognized within the field of packaged software development, customer involvement is not common [20], and when present, often in the form of intermediaries or customer surrogates [28]. Hence, software communities can be seen as an interesting approach for involving distributed customers in the development and maintenance of software. Also, preceding the examples indicate an expansion of the traditional role of customers. In OSS development, for example, software users are often also the software developers; and although there are indeed hierarchies within open source communities, there is a high user dependency and hence high user impact. In this article, we focus on the maintenance process of packaged software. Although there has been considerable research on software maintenance and how this task can be facilitated, this research has not typically focused on possible expansion of the role of the customer. Therefore, we took a closer look at the opportunity to use software communities to elicit changing customer needs and requirements and hence expand the role of customers to more active participants in the software maintenance process.

3. EMPIRICAL SETTING AND RESEARCH PROCESS

3.1 Daydream Software

Daydream Software is a Swedish computer game developer with its foundations in Sombrero AB, a company focusing on software systems and hardware sales. Daydream Software was founded in 1994 and is currently focused on producing interactive entertainment. During the period of this study, January 2000 to October 2002, the company consisted of employees ranging from managers, administrative personnel, and marketing people to game developers, graphical designers, and Web designers. With successful products such as "Safecracker" and "Traitors Gate," Daydream has a large international customer base and well-established customer communities around its products. In developing Safecracker and Traitors Gate, all software was developed in house and then sold as packaged software in which distributors and publishers were important actors. As common in packaged software development [28], customer polls and market research reports helped the developers in getting information about customers' needs and requirements. Also, beta testing was performed by parts of the customer group to facilitate the development process and bring a complete product to the market. However, when released, both Safecracker and Traitors Gate were static in the sense that customers could no longer influence the products. This was recognized by one of the developers at Daydream:

> Both Safecracker and Traitors Gate were static products without vivid customer communities. During development we got user feedback in terms of beta-testing, but customers were not directly involved in any further modification of the games.

Following on the success of these products, Daydream introduced Clusterball, a multiplayer computer game in which players fly around ships trying to collect balls and steal them from other players. In contrast to Safecracker and Traitors Gate, which were both commissioned work, Clusterball was the result of an in-house vision—the idea of an online sport accessible also via the Internet. An important point of departure for Daydream, and also for the inspiration for this study, was the explicit intention from the outset to utilize customer knowledge in the maintenance and improvement of the game. To this end, a virtual community was established that would cater for customer–developer and customer–customer interaction not only during beta testing but also during the maintenance process. For Daydream, the community would allow for the continuous elicitation of customer needs and requirements, something that had been difficult to achieve when developing the two previous products. Also, the community would provide a mechanism for the customers to influence the maintenance process in terms of fault repair, software adaptation, and software modification. This is a well-known yet difficult challenge in traditional software maintenance, and the attempt by Daydream to use a virtual community represents a quite novel way to address it.

3.2 The Clusterball Community

Here we use Hamman's [29] four criteria to characterize virtual communities: namely, group of people, shared social interaction, common ties, and shared social area. These are used to describe the characteristics of the Clusterball community.

3.2.1 A Group of People. The Clusterball community is a game community consisting of members from northern Europe and the United States in the main. Depending on previous game scores, each member is categorized according to the official Clusterball ranking ranging from "newbie," "ballboy," and "trainee" to "master," "grand master," and "cluster king." In total, there are 20 different ranking categories, and members with the highest rankings are well-known and celebrated members in the community. Together, they engage in discussions concerning Clusterball; and on a regular basis, they arrange tournaments and team play as well as tutorials and training sessions for all Clusterball beginners.

To stimulate the interaction between customers and developers, a "community manager" was appointed in August 2000. This person was responsible for responding to—and implementing—suggestions put forward by the customer community. This helped to ensure that the community was nurtured and that valuable feedback was not lost.

Not surprisingly, many of the developers at Daydream are active community members. This is a feature of many open source software projects also in that invariably the software developers are themselves also actual users of the software, something that facilitates in the process of building a community [16].

3.2.2 Shared Social Interaction. With approximately 17,000 postings distributed among two different forum tracks over a 3-year period, the Clusterball community provides an active discussion forum for the development and the modification of the game. As recognized by Baym [30], the communicative style of participants in such communities are often oriented around common interests and practices even before they enter the computer-mediated world, and often the members adhere to certain norms of rational discourse. In this case, the technology becomes an enabler of already established physical communities—a description that is very apt for the Clusterball community.

However, clusterball.com is not the only place where the Clusterball community meets. Besides this forum, there are *fan* Web sites (Web sites developed by community members themselves) that offer forums and chat rooms for community members and *team* Web sites where different teams meet and sign up for tournaments. One of the most impressive fan Web sites is Ballsnatchers.com, which was originally developed exclusively for Clusterball by two of the players that is now maintained and further developed by a team of Clusterball players from all over the world. Here, players have their own "hall of fame" (player/team victory announcements), a "haiku corner" (player poems), and a "player gallery" (player portraits).

3.2.3 Common Ties. The common interest in the Clusterball community is computer games in general and Clusterball in particular. In the different forums, community members discuss configuration and installation problems as well as tournaments, team play, and how to improve the game. At Clusterball.com, there is the "technical" and the "general" forum, and at Ballsnatchers.com, there is a specific forum for beginners called "young wings" in which new players can post any questions they might have to the rest of the community. Also, there is a "chat-and-gossip" forum in which players discuss anything that comes to mind.

The devotion and motivation among community members can also be seen in the activities they organize. For example, there are several "Clusterball Schools" for beginners (see, e.g., Ootpek's Clusterschool, Kronix Tips, and Lava-Lava's Clusterball Tips at www.clusterball.com), a "Skin Tutorial" in relation to a skin site on which players upload their individually designed skins so that other players can download and use them, and a testimonial site where Clusterball players share experiences from their initial contact with Clusterball.

3.2.4 Shared Area. To communicate, the Clusterball community members send postings to electronic fora consisting of several different tracks. In these, headings are shown for all topics, and all postings are presented as threaded lists. Also, there is a chat so that people can meet before the game, as well as after, to discuss issues concerning that particular gaming session. In addition to this, there are the fan Web sites where several other fora and chats can be found and where many of the Clusterball players spend time on a regular basis.

3.3 Research Methodology

3.3.1 Research Design. The research we outline in this article was part of a longitudinal interpretative case study [31] conducted at Daydream Software between January 2000 and May 2001. This study consisted of an exploratory study in which we sought an initial understanding of the company, an in-depth study involving participant observation at the research site, and a complementary data collection phase in which we carried out qualitative interviews and a Web survey. In addition to this, a follow-up study was conducted between June and October 2002. In this, we held additional interviews, and we sought to deepen our understanding of the particular context of Daydream Software and the way in which a virtual community was used for improving customer–developer interaction in the software maintenance process.

3.3.2 Data Sources and Applicability of Results. In this particular study, our objective was to investigate to what extent virtual communities can be used for involving distributed customers in the maintenance process of packaged software and hence how they might address the problematic issues as we identified earlier in the article. To do this, we extracted four categories of different empirical data from the Daydream study (Table 1).

First, we analyzed postings to the technical forum at clusterball.com with regard to discussion theme and result in terms of modifications to the software. The postings included those sent to the forum between the release date July 17, 2000, and

Table 1
The Different Categories of Empirical Data That Were Extracted From the Overall Daydream Study for the Purpose of this Article

Empirical Data	*Description of Empirical Data*
Postings	Written messages revealing customer needs and requirements as well as suggestions for software improvements
Patch specifications	Technical specifications including modifications and new features that were implemented in each of the new software versions that were released as responses to customer needs and requirements
Web survey	A Web-based questionnaire including questions on community use and to what extent customers felt that they could influence the software maintenance process
Interviews	Qualitative interviews revealing the developers' apprehension of the development process of Clusterball and to what extent the community allowed for customers to participate in this process

May 2001 when the complementary data collection phase was finished. During this period, 1,116 messages were posted to the technical forum, of which the major part were sent between July and December 2000 when the game was still new and when there were a lot of technical issues to handle. In reading the postings, we took special concern to those reflecting customer needs and requirements and whether these were implemented in the coming patches.

Second, we studied patch specifications to learn about the changes that were implemented in new versions of the software and whether these could be related to the postings in the forum. To do this, we analyzed specifications of six different patches. The patches included in this study were released on July 18, August 25, October 19, and December 20, 2000; and February 22 and April 29, 2001. Although we found parts of these specifications at www.clusterball.com, we obtained other parts directly from the developers at Daydream.

Third, we sent a Web-based survey out to 200 Clusterball community members ranging from newbies (not very experienced gamers) to "Ring Kings" (very experienced gamers). We sent the survey out as part of the complementary data collection phase in October 2000 and consisted of questions regarding the use of the community and the way in which community members felt that they could influence the maintenance work of Clusterball. With a response rate of 52%, the survey helped us in exploring community use and community influence in the software maintenance process.

As a complement, we conducted qualitative interviews with the lead programmer and one of the graphical designers. We conducted these interviews during the follow-up study, and in these, we asked the interviewees to look back on the maintenance process of Clusterball and evaluate how, and in what situations, customers in the virtual community contributed to the different categories of maintenance. Each interview lasted for about 1.5 hr, and we both recorded and transcribed them.

In terms of generalizability, case study research has often been criticized for being nonrepresentative [31] and therefore of limited use outside its specific context. However, from an interpretive position, representativeness in a statistical sense is not a key goal but instead the plausibility and cogency of the reasoning used in describing the results from the case and in drawing conclusions from these results [31]. In our study, we explored the use of virtual communities for involving distributed customers in the maintenance process of packaged software. In this, our goal was not to present generally applicable results. Undoubtedly, not all software communities are like the computer game community we presented here, and not all software has characteristics similar to those in a computer game. The extraordinary motivation level of Clusterball community members and hence the benefit of involving them as active participants in the maintenance process of Clusterball may not be applicable in other communities or in relation to other software products. Still, the Clusterball case constitutes an interesting example that illustrates the potential use of virtual communities for software maintenance and the extended customer role that is associated with this. Although it might be difficult to translate all its aspects to the maintenance process of other software products, we provide a detailed account involving specific implications in this particular domain of action [31]. In doing this, the study adds to the understanding of virtual community use and for what particular categories of software maintenance such an approach might prove useful.

4. OVERVIEW OF CLUSTERBALL PATCHES

The Clusterball game uses the 3DGM graphical engine. It is programmed in C++, modeled in Java, and the Sourcesafe product was used for version control for the project. To implement changes in customer needs and requirements, Daydream released six patches to the game. As early as July 18, 2000, only 1 day after the official release, the first patch to Clusterball could be downloaded from the Internet. In addition, the second patch was released on August 25. Together, these patches solved many of the initial installation problems and start-up errors as well as host errors that were recognized by customers.

On October 19, 2000, the third patch was released. This patch included several adjustments and modifications as suggested by the customers. For example, the patch included

- Replay and recording.
- Did not finish (DNF) feature (players could still get points even if they did not finish the game).
- Capability to lock a server on a minimum and maximum ranking basis (to avoid for newcomers to play against too highly skilled players or for highly skilled players to play against too novice players).
- Pregame chat (chat to other players while waiting to join a game).
- Ranking for team play.
- Longer chat lines in the in-game chat.

On December 20, 2000, the fourth patch was released. This was an A2D Driver patch that was released to solve a problem related to customers using the ATI Rage Pro LT graphics card. The A2D Driver patch installed all the necessary A2D drivers for the graphics card and thus provided support for customers using this particular graphics card.

In addition to this, the fifth patch was released on February 22, 2001. This was the GL Setup patch that detected what kind of graphic card the user had and then downloaded and installed the latest drivers for that particular card.

Finally, the sixth patch was made available on April 29, 2002. This patch included bug fixes, software adaptations, and functionality additions. Among other things, these new features were included:

- Improved support for joysticks including "twist handle" functions.
- LAN play without restriction of Internet access for host.
- Improved AI in the training (offline) mode.
- Also, the following bugs were addressed:
- Crash bug in the pregame chat.
- Freeze bug when viewing replays.
- Throttle bug on joysticks.
- DNF bug in match history.
- Sound volume bug.
- Font problem in chat.

In studying the content of the Clusterball patches, it is evident that many of the improvements that were made to the software originated in customer suggestions as reflected in the community postings. In the following, we discuss this process in more detail and illustrate the community contribution to the various categories of software maintenance.

5. COMMUNITY CONTRIBUTION TO CLUSTERBALL MAINTENANCE

Clusterball was released on July 17, 2000, and made accessible to customers all over the world. At this point, improvement in terms of software maintenance began. In the following discussion, we analyze community postings to illustrate the use of the virtual community in the maintenance process of Clusterball.

5.1 Corrective Maintenance

Corrective maintenance is concerned with software fault repair, coding errors, design errors, and requirements errors. As recognized by Sommerville [18], these types of errors are not the most expensive to correct. However, they need to be attended to on a continuous basis. In this process, Daydream got significant help from the community. Consider these community postings—all regarding different error messages:

> A few times now I've had the game just hang. It's always right after a game when it says "time limit reached" or on the loading screen before a game starts. It will just stay at those screens forever and nothing will happen. I was curious if this was related to Win2K or something else. Machine is P2–450, 348 ram, Voodoo3 3000, Win2K Pro. I've installed the host error patch as well though this happens when I'm joining a game not hosting.

> Right when the loading screen appears I get an illegal operation and I have to close it. This happens every single time. I have a diamond ViperV550, running 1280x1024 32bit,

> and win 98. I have had some problems with other games not switching to direct 3d mode but nothing like this. Please help, I'm very annoyed.
>
> Hi, My crashes end with: CLUSTERBALL caused an invalid page fault in module CLUSTERBALL.EXE at 015f:0054e7e4. It crashed mid-game. I have a 450 Athlon, 256mb, GeeForce256. Any suggestions? Thanks.

These errors were handled in the first and second patches. Most installation and start-up problems were solved in the first patch, and in the second patch, released on August 25, host errors were solved.

Additional software faults that were recognized by customers concerned font problems, sound volume problems, and a crash bug that appeared when too many customers joined the pregame chat:

> I like the new patch, but damn, I can't read anything when joining the chat. I have a 19" monitor but I still can't read that crappy font. Also the sound is still a problem in XP with SB Live sound card.

This posting got a quick response from one of the developers at Daydream:

> The soundcard thing is out of our reach, let's just hope that Creative will update their crap drivers for XP soon. I've got this sound volume problem with lots of other games in XP as well … see what I can do.

Regarding the crash bug in the pregame chat, this was mentioned by one of the customers:

> When there are too many "activities" going on in the pre-game chat room, Clusterball has a tendency to crash. I didn't note the error message though.

The font problem, the sound volume problem, and the crash bug in the pregame chat were solved in the sixth patch that was released in April 2002. In this patch, fixes for these bugs were included together with several additional features as requested by the customer community.

5.2 Adaptive Maintenance

Adaptive software maintenance is required to adapt the software to different operating environments, for example, if some aspect of the system's environment such as the hardware, the platform operating system, or other support software changes or if other environmental changes require the adaptation of the software [18]. Consider the following community postings, all concerning software adaptation:

> I wish the standard "control setting" was better. At present, getting a good control setting is too much a case of trial and error. But if people have problems with this they don't play.
>
> It would be better if there was support for more video cards.
>
> I would like to see the Mac version of the game.

> It would be helpful, if it was possible, to run Clusterball in Windows Mode, or at least if you could minimize it.
>
> I think you should integrate an IRC [DEFINE IRC] client into the software so that you could access the Clusterball channel from inside the client.

A common theme across these postings was the desire for adaptation of the software to other operating environments or to be able to play the game using other configurations.

Another problem recognized by customers in the community was the joystick problem:

> I tried playing again today after a week … and sometimes the stick works okay. But the last few games I was only trying to stay on course (slamming the platforms and bumping into the equipment houses. This takes so much time, and other pilots start taking over my route, and then when I fly normally again, I arrive late everywhere. It occurs suddenly, and then it stops. Because of these problems other players have a lot of chances and I can't avoid them shooting because I can't fly properly. Does anyone have these problems with their joystick or am I the only one?????? Can someone help me out here????

Furthermore, throttle problems with the joystick were discovered:

> I've just bought the Saitek Cyborg 3D Joystick which has solved the jerky controls of my previous low budget version. The problem is the throttle doesn't seem to give full speed and the response to change direction etc is very slow. I use standard PRO settings. The joystick seems to be calibrated and profiled correctly but I can't get round these problems. Any ideas anyone?

The joystick problems were all solved in the sixth patch, which also handled the DNF issue that was flagged by one of the customers:

> My experience of DNF is that I send data, but don't receive anything, and when enough time has passed, my computer evaluates this as the server having gone down (not closed, that is a different message!), and ends the session. Is there a way for us DNF-targeted to ignore "Bad connection" for a longer period, maybe set this in the configuration file?

On December 20, 2000, the fourth patch was released. This was an A2D Driver patch that was released in relation to graphic card problems that resulted in strange coloring of the balls in the game. This problem was identified by one of the customers:

> I got an ATI Rage 128 graphics card. I'm not sure what you mean about the "environment map." But when I play Clusterball, all the ships and balls are black. I need some help. I can still play but it is very annoying. Well thanks for the help.

In responding to this, one of the developers elaborated further on the nature of the problem:

> I have actually seen this "black ball" phenomenon happen during development. I think it was with very old drivers on a Riva TNT [SPELL OUT TNT] card. Since you want FULL support I'll just start asking my (huge) line of questions:
>
> 1. Have you installed the latest OpenGL drivers for your ATI Rage 128?
>
> 2. Could you check in the Control Panel-> Display-> Adapter that OpenGL is chosen the DEFAULT renderer (important!)
>
> 2.2 Also, while in the Display settings, what level of 3D acceleration is set, FULL, 75%, 50%, 25%, or what?
>
> 3. There is a possibility that the game tries to run software renderer instead of OpenGL … is your graphics "grainy" like software rendering?
>
> 4. Do you have DirectX 7.0 installed?
>
> 5. What is your computer name, processor speed, etc?
>
> 6. If you open the config.cfg file in Wordpad, what value does it say after renderer?

Other community members, using other graphics cards, were also involved in the discussion:

> Dear Clusterball Support: Is CD supposed to work with the Voodoo5 FSAA? Whenever I have it enabled, starting a match freezes my system 80% of the time online, and about 20% when offline training. FSAA works 100% fine on ALL of my other games, online and offline.
>
> Clusterball is not working so well with the new driver from NVIDIA. I'm using a TNT Ultra 2 and have tested the new drivers. I installed it and I'm using the old one from January 2000.

To solve the graphics cards problems, the A2D Driver patch installed all the necessary A2D drivers and hence provided support for users using an ATI Rage Pro LT graphics card. In addition, the fifth patch was released on February 22, 2001. This was the GL Setup patch that detected what kind of graphic card the customer had and consequently downloaded and installed all the latest drivers for that particular card.

5.3 Perfective Maintenance

Perfective maintenance requires functionality addition or modification in response to changes in customer needs and requirements. In the Clusterball community, changes in customer needs and requirements were reflected in postings concerning, for example, the ranking system, the need for a comprehensive chat feature, and the desire for a "player search" function. The following postings all exemplify customers' suggestions for improving the ranking system:

> I would like to see tournaments for middle class rankings. There are tournaments for new people and for high class players, but the middle men are left out.
>
> I would like to improve the match making—to allow the possibility to find other players closer to my skill level.
>
> I would like to be able to set a minimum and maximum player ranking when I host a game. In this way, a Newbie game will really be for Newbies, experts won't come along and thrash everyone. Similarly, a group of experts won't have to worry about a raw "what do I do with these balls?" beginner unbalancing team play.

To some extent, these problems were catered for in the third patch in which the functionality to lock a server on a minimal and maximal ranking basis was implemented, and hence, unbalanced matchmaking could be avoided.

Also, the need for elaborate chat features was expressed:

> It would be great to have a chance to chat with the experts. The "Ring Kings" could participate and give the "Newbies" some hints live.
>
> I would like to have the possibility to talk to other players while waiting for a game.
>
> There needs to be a better chat function in the game. The one that is there in this version is really bad—nobody sees it.

In response to these postings, a pregame chat was implemented in patch number three. Using this, players could talk to each other while waiting to join a game, and they could exchange experiences from previous gaming sessions. Also, longer chat lines in the in-game chat were implemented to improve the overall chat function. Finally, the need for a player search was flagged by the customers:

> A "player search" would be helpful. That way one could find a friend who is somewhere else in the ranking system.
>
> What I miss is some sort of "player search" where you could find out more about a specific player, like for example e-mail, ranking, score, games played, where he/she lives and so on.

Contrary to most other suggestions, the player search was not implemented in any of the patches. The reason for this could not be found in any of the developers' postings to the community. Hence, postings regarding the player search can be seen as suggestions for future software improvement, something that was also the case for the following postings:

> Make more then just the ship playable, most other games have more than one model to choose from. It doesn't have to be that different, but still another model to choose from. Maybe there could be a model editor where players could make their own ships.
>
> Could there be a "viewer system" so that my friends could watch other people play before they participate themselves? The game would be more like a real sport if it was viewable on TV or the Internet.

> Make the venues change weather sometimes. The sun isn't always shining. Perhaps a change in wind could make the venue Egypt more difficult.

> I would like to see a password for the server when hosting a match. Setting the number of games or how long time the dedicated server should run. It would also be great if it was possible to send messages to the players when running a dedicated server.

Interestingly, a similar phenomenon was reported in the Mockus et al. [32] case study of the open source Apache Web server. In their study, Mockus et al. [32] found that 75% of suggestions for software modifications are ignored. This is quite common in open source projects, and it has been suggested that a meritocracy exists whereby the privileged few at the core control almost exclusively the ongoing development of the projects. Although the Clusterball case is not an open source project, there seem to be parallels with the phenomenon as reported on in the Mockus et al. [32] study.

Despite the fact that suggestions, as those we presented previously, were never implemented during the period of this study, they can still be seen as important for Daydream in the maintenance work of Clusterball. Besides revealing suggestions for future software improvements, the postings reveal community engagement and community interest in the software that was produced.

6. DISCUSSION

Based on the empirical findings in the Clusterball case, we suggest that virtual communities, as platforms for interaction, are beneficial to the maintenance process of packaged software in all three of the maintenance categories.

First, in the corrective maintenance process, concrete descriptions on specific software faults were obtained from the users. In the postings, detailed error messages and full descriptions of hardware configurations were included to facilitate the bug tracing activities conducted by the developers. This process of software fault repair can be compared with the different automatic fault reporting systems that are included in many software products. In a similar fashion, customer suggestions were registered and attended to, and without any significant developer–customer interaction, the results could be found in additional software patches. However, although the community could be used as any ordinary bug reporting system, there was also the opportunity for developers to either give personally customized answers to each contributor or to post responses to the community forum so that anybody interested could learn from the answer. From the customers' point of view, there was also the opportunity to have other customers commenting on the particular software fault and in what different situations it appeared. As can be seen in the Clusterball case, this allowed for an open discussion between customers and developers—and between customers—in which there was the possibility of finding not only the origin of a particular software fault but also the different use circumstances in which this fault was evident. Furthermore, our discussion in Section 5.1 revealed how triangulation occurred in that different customer reports helped to refine the location and cause of errors. As has been indicated in literature on requirement acquisition [19], interactive dialogue is often required to establish the precise nature of a software fault, and the triangulation process we described previously helps to ensure such dialogue takes place. When one considers that estimates suggest that 60% of the time spent on a modification task is consumed in finding the location of the lines to be changed [9], this detailed troubleshooting triangulation could be very beneficial indeed.

Second, in the process of adaptive maintenance, community postings revealed different user configurations, different hardware and software equipment that customers used, and how the software could be adjusted to suit the different operating environments represented. In similar fashion as with postings regarding software fault repair, postings on software adaptation included hardware specifications and configuration details. In addition to this, however, software adaptation postings also included suggestions for future improvements of the game and adjustments that would be necessary to meet the requirements presented by other software and hardware equipment. In this, community postings revealed not only information important for the maintenance of Clusterball but also information about the dynamic relation between Clusterball and other software and hardware configurations. As recognized by Norvig and Cohn [33], however, the word *maintenance* can be misleading when referring to adaptive changes such as those we reported on here. According to Norvig and Cohn [33], the term *maintenance* can give the impression that the software has somehow degraded and needs to be refurbished to its original condition. This is misleading because software programs, such as a computer game, do not degrade. They remain the same, whereas the environment and equipment around the program continuously changes. Thus, adaptive maintenance is really a process of upgrading or improving the software to meet the needs of the changing environment. In the Clusterball case, such improvement was evident in, for example, additional support for new joysticks and enhanced support for video cards.

Third, in the process of perfective maintenance, innovative customer suggestions regarding the ranking system, the pregame chat, and balanced matchmaking were found. Taken together, these postings could be understood as recommendations for new system capabilities either by adding new functionality or to modify existing functionality. However, although there was indeed a demand for customer suggestions, this was also the maintenance category in which suggestions were largely ignored. This is somewhat unfortunate, as identifying new functionality is a major difficulty in conventional software maintenance (cf. Cusumano & Selby [34]), yet this is well catered for in this virtual community model, a feature of open source projects also [32].

Although being recognized as important in the Clusterball case, suggestions regarding additional play models, a viewer system, and changing weather in the venues were never implemented. This suggests that there were limitations in what the customers could influence. This is in accordance with one developer's view on the community and its importance in the maintenance process:

> I read the postings, but the main parts of the new features that are implemented are the result of our own ideas. We already know what we would like the game to be like.

Accordingly, one of the customers commented the following:

I don't think that I can influence the game itself, but more like little improvements and small features.

Although the preceding statements might suggest that there was no opportunity for customers to influence the maintenance process in terms of functionality addition or modification, it is important to recognize the context in which Clusterball was developed. Because Clusterball was not commissioned work but instead an in-house project, the developers had very strong ideas about the story line and the overall design of the game. As we indicated previously, in some situations, the developers already knew how they would like the game to evolve; and hence, there are reasons to believe that they did not want their own story line to be subject for too much external negotiation. Also, in developing software that is distributed to customers all over the world, there are additional actors such as distributors, publishers, and vendors to consider in the development as well as in the maintenance process. As recognized by Zachary [35], packaged software development is an activity driven by time-to-market demands and tight release schedules, making major changes or improvements difficult to attain depending on the surrounding circumstances in the software development environment. As illustrated in the Clusterball case, the process of software functionality addition or modification seems to be the process in which external actors, or strong in-house ideas, play an important role. As a result of this, many of the suggestions that were provided by customers were never implemented. However, although this implies that community use is limited in the perfective category of software maintenance, the community could still be used for the elicitation of innovative ideas important for future software improvement. Bergin and Keating [36] identified the need for an explicit model for software maintenance, and certainly, a model that would help the Clusterball developers harvest the wealth of useful suggestions from the virtual customer community would be very useful. This would also allow the strategic planning of which new features to implement, and some positive reinforcement could be provided to the customer community through, for example, the publication of a planned release schedule. In such a model, the community manager role within Daydream would also help ensure that such activities take place.

In looking back at the empirical material, there are certain benefits that can be associated with community use for software maintenance. A common feature in all maintenance categories is the community supportiveness, that is, community members' willingness to help other community members in solving problems. As shown in the Clusterball case, customers mutually engage [37] in helping each other, and customer–customer interaction is a prominent feature of the community. In a community, getting help means giving help, and the more people that get involved the better. This also reduces the workload of the developers somewhat because customers rather help each other before asking the responsible developer. In addition to this, the Clusterball case reveals community enrolment, that is, community members' willingness to engage new members to the community. Although being positive for the overall community atmosphere as well as for community activities such as tournaments, training sessions, and forum discussions, this also facilitates for the software firm in attracting new customers. In research within the field of packaged software, customer involvement has been recognized as important for improving the development process [20, 28]. Indeed, many of the best suggestions for product improvements often come from customers [38]. Recognizing this, the potential of having community members enrol new customers will benefit not only software development but also software maintenance in terms of an enlarged customer base and hence increased customer feedback on software adjustments and improvements.

Certainly, the degree of supportiveness and enrolment may vary over time and also between different communities, but the basic idea of having a group of people willing to support and involve each other is beneficial to the overall process of software maintenance. Part of the success of rapid application development approaches has been attributed to the fact that some of the development task, for example, documentation and testing, is devolved to the general customer community [39]. Likewise, it is often the case that developer–customer relations become frayed in traditional maintenance [9], and the positive atmosphere between developers and customers generated in the Clusterball virtual community bodes well as a model.

7. CONCLUSIONS

In this study, we examined the specific use of virtual communities for involving distributed customers in the maintenance process of packaged software. On the basis of an empirical study of a computer game community, we illustrated how virtual communities can be used in the process of corrective, adaptive, and perfective maintenance.

In the corrective process of software fault repair, virtual communities facilitate bug tracing activities by allowing for the continuous elicitation of particular software errors in relation to individual customer configurations. Also, detailed troubleshooting help to refine the precise nature of the bug and help to pinpoint its exact location. In the adaptive process, virtual communities bring forward customers' view on future versions and adjustments necessary to adapt the software to requirements put forward by other equipment. Finally, in the perfective process, virtual communities allow for the elicitation of innovative ideas for future improvements and new functionality. However, although we suggest that virtual communities prove useful for involving customers in all three categories of software maintenance, it seems that customer impact is most evident in relation to corrective and adaptive maintenance. Due to external requirements as proposed by actors such as distributors, publishers, and vendors or strong in-house ideas as proposed by the software developers themselves, customer suggestions have less direct impact in the category of perfective maintenance. In accordance with Bergin and Keating [36], we suggest that an explicit model for software maintenance could allow for the suggestions for new functionality to be better recorded and for positive reinforcement to be provided to the virtual community.

Finally, in the study, we suggested that there are certain benefits that can be associated with community use for software maintenance. First, community supportiveness, that is, community members' willingness to help other community members in solving problems, was identified as important for reducing the workload of the developers. Second, community enrolment, that is, community members' willingness to engage new members to the community, was identified as central for attracting new customers to the community and hence provides software developers with increased customer feedback in their everlasting process of software maintenance.

REFERENCES

[1] G. Alkhatib, "The maintenance problem of application software," *Journal of Software Maintenance: Research and Practice,* vol. 1, no. X, pp. 83–104, 1992.

[2] B. Boehm, *Software Engineering Economics.* Englewood Cliffs, NJ: Prentice Hall, 1981.

[3] B. P. Lientz and E. B. Swanson, *Software Maintenance Management.* Reading, MA: Addison-Wesley, 1980.

[4] J. R. McKee, "Maintenance as a function for design," In Proc. of AFIPS National Computer Conference, Las Vegas, NV, 1984.

[5] C. Babcock, "Staffers seek bolstered image," *Computerworld,* vol. 21, no. 20, p. 8. 1987.

[6] N. Schneidewind, "The state of software maintenance," *IEEE Trans. Software Eng.*, vol. 13, no. 3, pp. 303–310, 1987.

[7] C. McClure, *Managing Software Development and Maintenance.* New York: Van Nostrand Reinhold Company, 1981.

[8] I. Vessey and R. Weber, "Some factors affecting program repair maintenance: An empirical study," *Comm. of the ACM,* vol. 25, no. 7, pp. 128–134, 1983.

[9] D. Smith, *Designing Maintainable Software.* New York: Springer, 1999.

[10] S. Hirt and E. B. Swanson, "Emergent maintenance of ERP: New roles and relationships," *Journal of Software Maintenance and Evolution: Research and Practice,* vol. 13, no. X, pp. 373–397, 2001.

[11] D. Schmidt and A. Porter, "Leveraging open source communities to improve the quality & performance of open source software," in *Making Sense of the Bazaar: The 1st Workshop on Open Source Software Engineering, 23rd International Conference on Software Engineering,* J. Feller, B. Fitzgerald, and A. van der Hoek, Eds. Toronto, Ontario, Canada: PUBLISHER NAME, May 2001, pp. 52–56.

[12] K. M. Nelson and J. G. Cooprider, "The relationship of software system flexibility to software system and team performance," in *Proc. of the 22nd ICIS,* New Orleans, LA, 2001.

[13] B. Nuseibeh and S. Easterbrook, "Requirements engineering: A roadmap," in *Proc. of ICSE,* Limerick, Ireland, June 2000.

[14] H. Chesbrough, *Open Innovation.* Cambridge, MA: Harvard Business School Press, 2003.

[15] E. S. Raymond, *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary.* Cambridge: O'Reilly, 1999.

[16] J. Feller and B. Fitzgerald, "A framework analysis of the open source software development paradigm," in *Proc. of the 21st ICIS,* Brisbane, Australia, 2000.

[17] R. Pressman, *Software Engineering: A Practitioner's Approach,* European ed. New York: McGraw Hill, 1997.

[18] I. Sommerville, *Software Engineering.* Harlow, England: Addison-Wesley, 2001.

[19] T. A. Byrd, K. L. Cossick, and R. W. Zmud, "A synthesis of research requirements analysis and knowledge acquisition techniques," *MIS Quarterly,* vol. X, pp. 117–138, Mar. 1992.

[20] S. Sawyer, "Packaged software: Implications of the differences from custom approaches to software development," *European Journal of Information Systems,* vol. 9, no. X, pp. 47–58, 2000.

[21] M. A. Smith and P. Kollock, *Communities in Cyberspace.* New York: Routledge, 1999.

[22] B. S. Butler, "Membership size, communication activity and sustainability: A resource-based model of online social structures," *Information Systems Research,* vol. 12, no. 4, pp. 346–362, 2001.

[23] S. Sharma, V. Sugumaran, and B. Rajagopalan, "A framework for creating hybrid-open source software communities," *Information Systems Journal,* vol. 12, no. X, pp. 7–25, 2002.

[24] R. T. Fielding, "Shared leadership in the Apache project," *Comm. of the ACM,* vol. 42, no. 4, pp. XX–XX, 1999.

[25] S. Hermann, G. Hertel, and S. Niedner. (2000). Linux study: First results. [Online]. Available: http://www.psychologie.uni-kiel.de/linux-study/writeup.html

[26] W. Scacchi, "Understanding the requirements for developing open source software systems," *IEEE Software,* vol. 149, no. 1, pp. 24–39, 2002.

[27] R. Pavlicek, *Embracing Insanity.* New York: Sams, 2000.

[28] M. Keil and E. Carmel, "Customer-developer links in software development," *Comm. of the ACM,* vol. 38, no. 5, pp. 33–44, 1995.

[29] R. B. Hamman, "Computer networks linking network communities," in *Online Communities,* C. Werry and M. Mowbray, Eds. London: Prentice Hall, 2001.

[30] N. Baym, "The emergence of on-line community," in *CyberSociety 2.0: Revisiting Computer-Mediated Communication and Community,* S. Jones, Ed. Newbury Park, CA: Sage, 1998, pp. 35–68.

[31] G. Walsham, "Interpretive case studies in IS research: nature and method," *European Journal of Information Systems,* vol. 4, no. X, pp. 74–81, 1995.

[32] A. Mockus, R. Fielding, and J. Herbsleb, "A case study of open source software development: The Apache server," in *Proc. of 22nd ICSE,* 2000, pp. 263–272.

[33] P. Norvig and D. Cohn, "Adaptive software," *PC AI Magazine,* vol. 11, no. 1, pp. XX–XX, 1997.

[34] M. Cusumano and R. Selby, *Microsoft Secrets.* London: HarperCollins, 1997.

[35] G. Zachary, *Showstopper: The Breakneck Race to Create Windows NT and the Next Generation at Microsoft.* New York: Free Press, 1994.

[36] S. Bergin and J. Keating, "A case study on the adaptive maintenance of an Internet application," *Journal of Software Maintenance and Evolution: Research and Practice,* vol. 15, no. 4, pp. 245–264, 2003.

[37] E. Wenger, *Communities of Practice: Learning, Meaning and Identity.* Cambridge, England: Cambridge University Press, 1998.

[38] E. Von Hippel, "Lead users: A source of novel product concepts," *Management Science,* vol. 32, no. 7, pp. 791–805, 1986.

[39] B. Fitzgerald, "A preliminary investigation of RAD in practice," in *Methodologies for developing and mmanaging Emerging Technology Bases Information Systems,* A. T. Wood-Harper, N. Jayaratna, and J. Wood, Eds. CITY, England: Springer-Verlag, 1997, pp. 777–87.