Supporting Collaborative Help for Individualized Use

Jina Huh[1], Mark W. Newman[1,2], Mark S. Ackerman[1,2]
School of Information[1], Department of Electrical Engineering and Computer Science[2]
University of Michigan
{jinah, mwnewman, ackerm}@umich.edu

ABSTRACT
In this paper, we seek to advance the research around utilizing collaborative help for supporting individualized use of technologies. We do this by shedding light on the ways that users of MythTV, a highly flexible open-source software system for home entertainment enthusiasts, collaboratively help one another in maintaining their individualized MythTV systems. Through an analysis of the MythTV user community's mailing list archive, documentation, and wiki, along with user interviews, we discuss how the community utilizes configuration artifacts as proxies to easily mobilize and exchange knowledge. While exchanging concrete artifacts such as scripts and configuration files was seen to sometimes increase the efficiency of knowledge transfer, it also presented several challenges. Negotiating the transparency of configuration artifacts, navigating the customization and appropriation gulfs, and aligning usage trajectories all emerged as problematic areas. We discuss design implications that center around addressing these challenges. Our findings provide a a useful new perspective on how to support users in their individualized use of systems.

Author Keywords
Appropriation, configuration, individualized use, collaborative help, tailorability, customization, pervasive systems, MythTV

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION

As computing becomes an integral part of our everyday lives, the work of configuring an increasingly complex environment becomes more challenging for end-users. Many users turn to user communities to get help from other users with similar experiences or better knowledge.

Collaborative help that builds knowledge through iterative distillation over time allows common problems to become more concrete and easily supported [3]. However, such approaches tend to be most helpful for solutions to common problems. For many problem spaces, though, while individual problems occur less frequently, in aggregate they occupy the largest portion of the space. Yet the barrier to solving uncommon problems is high, since it requires one to find suitable knowledge and modify it appropriately for what is often an individualized problem. Accordingly, the high effort level prohibits end-users from developing personalized, tailored, or contextualized technology use, obstructing a goal the HCI and CSCW fields have been attempting to attain for many years [e.g., 12, 28].

A crucial research question, then, is how to leverage collaborative help so as to better support individualized use for end-users. This involves examining how to support end-users for uncommon problems and needs.

In order to address this issue of supporting collaborative help for individualized use, we examined the MythTV user community. MythTV is open-source software for home entertainment enthusiasts that allows users to record TV shows, watch DVD, play games, and watch weather forecasts. MythTV can be installed on a number of current computing platforms, and the various components that make up the system (e.g., tuner, graphics card) can be replaced with many alternatives. Due to the highly flexible nature of the system, the MythTV user community served as an excellent study site for examining the challenges as well as successes in how a community could leverage collaborative help for individualized use. We focused on the configuration problems that users faced, because MythTV's configuration was complex yet tractable. This highlighted an important class of problems that people have in their current computing environments.

This paper reports on our examination of the community's email archive, documentation, and Wiki, along with interviews with MythTV users. We discuss how the community was effectively utilizing configuration artifacts such as configuration files, error messages, and scripts as a way to efficiently exchange knowledge and collaboratively tailor solutions for individualized use. We also observed several challenges that the community faced in engaging with configuration-based help, such as *dealing with transparencies of configuration artifacts, navigating the customization and appropriation gulfs,* and *aligning usage trajectories,* which we will discuss at length below.

We will begin our discussion with an overview of the background literature, followed by descriptions of the MythTV community and our study design. We then report our major findings, followed by a discussion and design implications.

BACKGROUND

The specific problem studied in this paper is help for configuration. Balka and Wagner [5], in their discussion of configurability as appropriation work, examined a wireless call system in a hospital setting that needed to work with devices such as intravenous pumps and electronically equipped beds, as well as facilitate communication among organizational units throughout the hospital. Their study contributed to an understanding of configurability beyond a single system and cast configurability as a challenge present in end-users' everyday computing environments.

With the view that the work of configuring and tailoring for individualized use is a deeply embedded practice that end-users need to work out in their everyday computing environments, several studies have searched for ways to support users' work of individualizing technologies. This has been studied in CSCW and HCI as tailorability, appropriation, and customization. While there are subtle differences among the three terms, the common goal is to allow users to fit systems to their needs. Work in the area of tailorability found that tailoring often became a collaborative process between the developers and the users [10, 14, 29]. Other researchers noted the presence of a critical group, called variously user-designers, tinkerers, or gardeners. These people customized and tailored software to users' needs [28] and shared their work with users through macros, forms, and other routinized snippets of code. Similarly, appropriation has also had a long history in HCI. For example, Pipek et al [20] described end users' collaborative efforts in making software artifacts work for their context of use. Stevens et al [25] described collaborative sharing practices of Eclipse IDE's software modified artifacts at a mid-size software development company. Customization has also been studied as the activities that are necessary in making a device or a system to work in a particular environment down to the level of an individual user [17]. These studies reveal configuration as a social activity, and highlight the relevance of collaborative help for supporting such activities.

Thus the other line of work upon which we draw includes studies in collaborative help, which have investigated technical help at a more general level. In this work, we follow a line of HCI research into help, which sees it as collaborative and social, with expertise being situated and contextual. These studies include how people get help from other people through online chat [2], question-and-answer forums, phone support lines [11, 18], Internet resources such as how-tos [27], and FAQs [9]. Related particularly to the domain of open source software communities, studies have surveyed various discourse types in voluntary peer-to-peer help interactions [22, 23]. Our work builds and extends on the prior work on peer-based collaborative help as it relates to users' configuration and tailoring practices of computing systems.

MYTHTV AND ITS COMMUNITY

We chose the MythTV community as a study site for three reasons: 1) Each user's configuration of MythTV is often distinct from others', 2) the configuration is highly brittle due to changing computing environments, and 3) MythTV configurations are reasonably complex yet tractable. Accordingly, studying the MythTV user community helps us to gain insights designing collaborative help for configuring and tailoring activities in computing environments more broadly. Below we describe further what MythTV system is, with some technical details, followed by the typical challenges that MythTV users faced when creating and maintaining their systems. We then describe the MythTV community, which was the main source of our data and analysis.

MythTV system

MythTV (mythtv.org) is an open-source software system that allows users to perform a variety of tasks, such as record TV shows to their computers, play games, check weather, browse the Internet, watch streaming online videos, rip DVDs, and listen to music. The system can be installed on Linux, Mac, and Windows platforms. Alternatively, MythTV software can come in packaged versions where the operating system and the MythTV system are bundled so that users do not need to separately install MythTV. MythTV consists of a frontend, which is in charge of the user interface, and a backend, which deals with the database that contains recorded content. A given MythTV system can consist of multiple frontends and backends, and the frontend and backend do not have to be on the same machine. Each user needs to configure his own Mythbox(es) –the machine(s) running the MythTV system – by choosing a platform, graphics card, amount of RAM, CPU, tuner card(s), remote control, and monitor(s). Environmental factors such as which country the user lives in, whether the user is using a cable service or over-the-air service, and whether she is subscribed to a standard or high definition TV service also affect the configuration of one's MythTV system. Considering all the possible combinations of the above system components, each user's MythTV configuration is often unique or at least very uncommon.

MythTV community

Members of the MythTV community receive information and communicate through several channels; most notably the official website, mailing lists, IRC, the wiki, and forums. These communication channels mainly exist for knowledge sharing as well as maintaining and developing MythTV as an open-source project. The wiki is used for growing solutions about individualized problems and how-tos about various appropriations of MythTV. On the other hand, the MythTV documentation is primarily developed by the developers and used to document official installation procedures.

Because we wanted to learn about the MythTV community's current help practices as well as any challenges that arise, we focused on examining the archive of the mythtv-users mailing list (mythtv-users@mythtv.org), where most of the help interactions among users were happening. There were other small unofficial forums and websites, but the activity level in those places was substantially smaller that that of the mythtv-users list. To give a brief sense of the activity level of the list, it started in February 2003 with 785 posts in the first month and reached a maximum of 8082 in March 2004. Since then (until January of 2010) it has steadily been declining with an average number posting of 3,813 per month since the peak. There were 559 posters for December 2009 with a total of 3,293 posts.

For July 2006 (which will be analyzed at length below), due to a prevalence of self-disclosure in mailing list posts it was possible to infer that the members who spoke out on the users' mailing list were largely in their late 20s to 30s and were males working in technology industries or in staff jobs at broadcasting companies. Some of them came to the community to learn about Linux, some wanted to save money, and some came in for a hobby. Most were US residents, but there were a considerable number of Australians and British as well. We also observed a few users from India, South Africa, Germany, and Japan.

Membership

The formal member roles in the MythTV community consist of developers and users. Developers have their own mailing list (mythtv-dev@mythtv.org), but they often listen in on conversations in the users' mailing list to either update the progress of MythTV development (e.g., letting users know whether certain features will be in the next release) or to participate in discussions of whether certain features are worth putting into the development pipeline. Rarely do they offer technical help, which is done largely by experienced users. One of the interviewees told us that the community implicitly agrees that developers should spend their time on developing MythTV and users should contribute back by providing help for newer members and documenting solutions. The community welcomes newcomers, and often kindly points to the archived solutions when newbie questions are asked.

DATA AND METHOD

The total number of email messages in our data set were 288,983. We analyzed approximately 4000 messages, 3273 of which were from July 2006. Our sampling rationale was based on Herring's guideline for computer-mediated discourse analysis (CMDA) [1], which encourages the use of motivated sampling driven by research questions over random sampling that sacrifices context. Because our research questions involved how the community helps individualized use of MythTV, we largely examined periods where MythTV was stable enough for users to further tailor the system to their own use. To identify such periods, we informally reviewed message threads at the beginning and end of the archive as well as subject lines throughout the archive. This helped us to get a sense of the community's conversation changes over time. Based on this review, we decided to focus on July 2006, which offered a suitably stable but active period.

In our analysis, we looked for emerging themes, which were iteratively tested with more data as we advanced our analysis. Once the initial analysis was over, we went over the coding together, probing for any remaining questions. We then went back to the data and continued to question the themes that emerged, looking for any exceptions or hidden meanings that may have been overlooked.

In order to validate our findings, we contacted recent posters as well as those who were registered on the MythTV wiki. We conducted a total of 12 interviews, three by phone in 30 to 60 minute semi-structured interviews, and nine by email where the interviewee and the first author iteratively sent emails back and forth for further questions and clarifications. The interviewees were asked to describe their history of using MythTV, the kinds of help that they received from the community, any breakdowns in getting help, their use of the wiki, any challenges in maintaining their MythTV over time, and what they thought about what we had observed to that point about the community.

The following three sections go through our findings from the study. The next section introduces frequent problems that the members experienced and how community members helped one another to maintain their MythTV systems. In the section that follows, we discuss how the help is rooted in configuration artifacts, which is made possible by the innate flexibility of the MythTV system. The final findings section describes various challenges that the community has had to struggle through in order to support individualized use.

PROVIDING HELP

In this section we briefly survey general collaborative help interactions observed from the community. These largely echo observations reported in previous work, most notably work on help in open source software communities [22, 23] and are presented here to establish a backdrop for our subsequent discussion of configuration-specific help. First, we characterize the types of individualized problems to which MythTV users often had a hard time finding solutions.

Problems in individualized use

The fact that MythTV can support a wide range of individualized uses creates many challenges in solving technical problems. Beyond general troubleshooting problems, the most widespread problem we observed in the individualized use of MythTV was dealing with compatibility issues among hardware and software components. During the installation phase, finding the right set of hardware and software components such as tuner cards, graphic cards, CPU, operating system, drivers, and patches, all of which need to be compatible, is a challenge. Accordingly, successfully installing MythTV can take anywhere from a day to several months. Also, adding new features, upgrading components, or replacing parts of the system can break the system if there are incompatibilities among the replaced or outdated parts and the existing configuration. Users also encounter individualized problems through external reasons such as power outages, errors in the listing service messing up channel listings, and problems from moving across country. Because most MythTV users are US residents, users from other countries often suffer from having to independently develop region-specific resources.

In our observation, it was clear that a large number of problems from individualized use of MythTV described above were not solvable through official archived solutions such as documentation and FAQs. The community thus developed several standard help interactions on the mailing list similar to what Singh and Twidale observed in open source software communities [22, 23]. In addition, the MythTV community established a wiki to try to capture solutions to common problems related to individualized use. As we will discuss later, though, the usefulness of the wiki was limited.

Collaborative help interactions for individualized use

The challenges of contextualization has been extensively discussed in prior work on organizational memory [1]. When askers inquired for help on the mailing list, the context that makes up an individualized use – for example, hardware and software configurations, family members' use of MythTV, or geographical constraints – and the processes by which the problem occurred were often implicit. Thus the asker and the helpers had to iteratively give feedback, requesting any important information that may have been missing. Also, the implicit rule of the mailing list was that the asker would report back what worked and did not work, although this was not always done. The iterative interaction between members mostly consisted of requesting and providing diagnostic evidence, e.g., error messages, configuration files, query results, symptom descriptions, and data on results from tests. Using such evidence helped the community to detect the user's system configuration and their problem at hand, facilitating helpers in tailoring suggestions to the asker's particular situation rather than giving general advice.

Critiquing has been discussed as essential for giving tailored help [8]. Here, it was also a useful way for the helpers to give advice and solutions tailored for the asker's specific situation. In a critique, an experienced user might point out problems with the configuration that a user provided in their query; these problems may or may not have anything to do with the problem at hand. In these critiques, there was no defined set of "right" solutions, and each critique was tailored and tweaked for particular individuals.

CONFIGURATION-BASED HELP

Past work has briefly mentioned social sharing of configuration artifacts in the context of component-based software development [25]. Configuration artifacts in the MythTV user community played a novel role in facilitating collaborative help for individualized use. Due to the nature of MythTV, help was often based in the specific knowledge artifacts that defined one's configuration. Similar to [4]'s discussion of knowledge artifacts, knowledge in the MythTV user community was frequently shared in the form of concrete configuration artifacts, which here took the form of settings files, logs, scripts, error messages, and the outputs of certain diagnostic tools. Just as Nardi and Miller saw spreadsheets as "cognitive artifacts" that provided a point of cognitive contact that mediated cooperative work among spreedsheet users [16], configuration artifacts in the MythTV community can be seen as proxies that could transfer one's contextualized knowledge about a problem and the system setup in a simplified form. Unlike communicative artifacts discussed in prior work, however, some of the configuration artifacts in the MythTV user community were executable, providing "pluggable" solutions for users' problems while also serving as boundary objects for communicative purposes. While this made certain help interactions considerably more efficient, it also presented a new set of challenges. Since each user's configuration was different, a configuration-based solution often did not easily transfer from one user to another, or from one situation to another. Therefore, reusing the knowledge in a configuration artifact was often tricky, and a significant amount of translation work could be necessary to utilize others' configuration artifacts.

The notion of transparency—the ability of systems or parts of a systems to reveal their contents for inspection and modification—has been extensively discussed in the software engineering literature. "White-box reuse" [19] refers to reusing software artifacts through modification for new project requirements. On the other hand, "black-box reuse" [6, 15] allows software components to be reused "as is," without modification (or with only the customization of parameters to allow for limited flexibility).

In the next several paragraphs, we discuss the work of adjusting the transparency of configuration artifacts as a way of illustrating how configuration artifacts acted as proxies in sharing knowledge. Specifically, we discuss examples for two cases of adjusted transparencies: black-box configurations and white-box configurations. Most importantly, we discuss how the transparency of a configuration artifact was often undetermined and would need to change as the help interaction unfolded.

Black-boxed configurations. The ideal situation in sharing configuration artifacts was when they were in the form of scripts, code, or files that had the ability to be plugged in and/or executed by others with minor modifications. This was especially useful in adding a feature, adding a patch for a bug, fixing configurations, and copying recording profiles and other configurations from users who succeeded in accomplishing a setup. In the following example, Phil volunteered to share a perl script file he developed that could be used with MythStream (an optional feature to watch streamed online media on MythTV) to get on-demand video content from ABC Australia. Phil gave a brief introduction on what the script could do, as well as detailed instructions on what to install and where to put the script:

```
Aussies, I've written a couple of harvesters that can be used with MythStream to get on demand video content from ABC Australia. [...] They both use the perl module LWP::Simple so you'll need to make sure that's installed. Put them in your MythStream parser directory (in my case that's /home/MythTV/.MythTV/mythstream/parsers) and make them executable. Then add these lines to your streams.res file: [code lines omitted] Hope someone finds these useful. I find it great for getting news on demand. (ML: Jul 1, 2006, Phil)
```

No modification was supposedly necessary for other people to make the script to work as it did with Phil's Mythbox.

White-boxed configurations. Providing help through sharing black-boxed configurations, however, breaks down when a configuration artifact fails to work for an individualized setting. For instance, UK residents who want to get content from the BBC through MythStream would have to open up Phil's script to be studied, understood, and modified.

In the following example, Hugh needed to understand one of his configuration artifacts, the xorg.conf file. Part of what a xorg.conf file does is manage configurations of advanced input devices and output to multiple monitors. Even though xorg.conf is part of the XWindows system and not MythTV, the MythTV official documentation provides a modified xorg.conf that allows using MythTV with two TV monitors. However, Hugh wanted to use a TV for MythTV and a CRT monitor for regular computing. Accordingly, he needed to modify the xorg.conf distributed in the official documentation, but had hard time making it work for his setup:

```
xorg.conf file [in the guide] is configured for TV out only and does not provide for a usable CRT/Monitor to do normal computing. I have tried modifying the xorg file using Jarrod's initial information and adding a second monitor, device and screen, without success. After several hours of experimentation I need some help/direction. (ML: Jul 7, 2006, Hugh)
```

Understanding and modifying the revealed information was a big challenge. Luckily, Goh, having had a similar experience, was able to help Hugh by walking through what he did to modify xorg.conf in setting up two screens each for computing and for watching MythTV, and referred to his resulting xorg.conf:

```
I've done something similar. Hopefully my experience will help you. […] Here's the process I followed to get this configuration to work:
[…] - Tweaked Jarod's example xorg.conf for the PVR-350 to fit my configuration (it became xorg.conf.tvout);
- Copied xorg.conf.tvout to /etc/X11/xorg.conf […]
- Merged xorg.conf.lcd and xorg.conf.tvout into xorg.conf.twinhead;
- This step required changing all instances of Screen0 in xorg.conf.tvout to Screen1.
[…] (Another online reference mentioned the need to add a "Load xtrap" line to xorg.conf to allow the mouse to traverse both screens, but I didn't find that necessary.)
My xorg.conf.twinhead file is included below. [the code of the script included in the message omitted] (ML: Jul 8, 2006, Goh)
```

As illustrated in the example above, a black-box configuration often needed to become transparent in order to make it work for an individualized use. The challenges lay in where to make it transparent, and how to deal with the information that was revealed through the process of converting a black-box into a white-box, or "white-boxing".

Configurations of undetermined transparency. For the most part, MythTV configuration artifacts in fact do not have determined transparencies of their own (they are all available for inspection with a text editor, for example). Rather, their effective transparencies are negotiated in use. Phil's perl script was technically white, but was shared with others as black. Hugh's xorg.conf was treated as black by the official documentation, but had to become white in order to work for Hugh's needs. One of the biggest challenges in configuration-based help was this process of black-boxing artifacts, then re-opening (white-boxing) and closing them again to be shared as black-boxed configurations for other potential users.

Furthermore, the critical problem in configuration-based help was not simply deciding whether to make configuration information black or white. Determining which part of the shared artifact and what other parts of the system's configuration needed to be transparent was critical. Avenard had trouble making his Mythbox recognize hardware devices in the same order each time he booted the system. One helper referred Avenard to documentation for udev rules (a Linux configuration subsystem that manages attached devices) and a previous mailing list thread that described how to set up the udev configuration to fix the problem. This udev configuration information offered in the previous thread could ostensibly be used as it was. However, for Avenard, following the instructions did not help. In order to diagnose his problem, he wanted to know more about which driver was actually handling his remote control device, which was information beyond what was described in the archived thread. He did not need to understand all of the udev rules—just knowing how to change a certain line of the udev rules file was enough for him:

```
After reading a lot about udev, and trying a few different configurations, I've been unable to get it to work as I wanted. I guess my problems come from
```

> that I do not know which driver is actually handling the IR interface... which makes it hard to guess the correct line in the udev rules. (ML: Jul 3, 2006, Avenard)

Notice the last comment about finding the correct line to fix in the udev rules. This nicely illustrates how the transparency of the udev rule needed to be componentized. That is, MythTV users often needed only some part of the configuration artifact to become transparent, not all. Also in needing to look at the driver that handled the infrared interface in his system, Avenard again did not need transparency of his whole system, but only enough of it to get his problem fixed.

In addition to the problem of adopting someone else's configurations as a proxy for their knowledge, configurations of different members, at any level of transparency, were also compared with one another to find similarities and differences so as to diagnose problems, play as benchmarks for performance tuning, or to prevent problems in the future. Configuration artifacts were used in different ways as part of the help interaction—as a reusable and modifiable object and a proxy for diagnosis and knowledge building through comparisons.

CHALLENGES IN SUPPORTING INDIVIDUALIZED USE

In this section, we discuss further how the process of help materials being generated on the mailing list and wiki encountered challenges. We focus on three challenges: identifying suitable solutions for individualized use, the contextualizing problems during help interactions, and maintaining solutions over the long-term.

Identifying suitable solutions for individualized use

MythTV is a complex multi-component system where each user's system is different from other individuals' systems, making it difficult to construct one-size-fits-all solutions. Existing solutions from the documentation, FAQs, the mailing list archive, MythTV wiki, and searching on the Internet would often have to be modified to fit with individualized MythTV systems. However, finding an appropriate solution (or set) to start with and adjusting that solution to fit one's individualized settings often requires a good amount of experience and expertise.

This could be a challenge for inexperienced users and represents an example of what Won et al [33] called the "customization gulf." Not only did it require knowing how and where to modify, but sometimes required figuring out one's own configuration information. The following example illustrates a user, Graeme, who had hard time using a set of instructions because the instructions did not work on his particular setup. Moreover, he did not even know how to bring up the specific information about his system configuration to even know which instructions to follow.

Graeme wanted to add an outdated tuner card, and he found instructions from the linuxtv wiki documentation on how to install the card in his Mythbox. When the instructions did not work, he assumed that it was because of the built-in modules in his kernel. However, he did not know how to check whether his assumption was true:

> It [the documentation] says for all devices I must modprobe [a program for loading modules to the kernel] i2c-core, crc32, firmware_class, dvb-core and dvb-pll. This works for all but crc32 and firmware_class. I understand that this could mean they are built into my kernel, but I don't know how to check that. I am running Fedora Core 4 with kernel 2.6.16-1.2115_FC4 (ML: Jul 21, 2006, Graeme)

Additionally, the vague instructions confused Graeme in terms of whether he needed to load all firmware or a specific one that was particular to the frontend information of his tuner, for which, again, he did not know how to bring up the information:

> This is confusing, because I'm not sure if I should load all of these [modules] or just the ones specific to my frontend/demodulator. I don't know which frontend/demodulator I have.

Then a helper taught Graeme how to get information on his built-in kernel modules as well as how to check the frontend/demodulator information—it often could be found on the card itself, or by running a command called dmesg. When Graeme checked, he did have the correct module built in to his kernel, but not the firmware_class, making it confusing why loading the module did not work for him. Also, he ended up taking a look at the card itself, only to find out that the frontend/demodulator information was obscured:

> I opened up the usb box, there is a conexant chip in there that starts with cx22. The rest of the numbers are obscured with heat sink compound.

Furthermore, the instructions directed him to use a firmware that clearly would have a compatibility problem:

> The linuxtv site advises me to use the Philips firmware file, but as I don't have any Philips chips, this must be wrong, no?

Graeme's case portrayed how challenging it can be to select and modify solutions that will work for one's specific configuration. Inability to understand one's own configuration settings, identifying unexpected constraints, and knowing the boundaries of how far the instructions could be applied to work in different configuration settings were clearly problems for inexperienced users.

Contextualizing problems during help interactions

Contextualization has long been discussed as a challenge in reusing information from knowledge repositories [1]. The mailing archive and the wiki of the MythTV user community were not the exceptions. Because MythTV configurations could be complex, those asking for help often had to choose what information to present about error messages, system configuration, and history of how their system changed over time. A MythTV wiki page on mailing list etiquette attempts to provide askers with guidance:

> Which MythTV version are you using? Please state whether you are using version 0.18, version 0.18.1, 0.19, 0.20, etc.

However, much of the page's instructions are ambiguous and rely on the user's discretion:

```
If your hardware or config details are unusual or
noteworthy and you suspect that information may be
pertinent, include it.
Include any relevant log file information like the
output from mythbackend, output from mythfrontend,
output from /var/log/messages, error message[s]
during compile. NOTE: Only include the relevant
information. It's okay to trim mundane stuff out of
logfiles.
```

These instructions suggest askers to provide information that they "suspect" might be helpful or pertinent, which are relative concepts that can result in varying outcomes depending on who is reporting the problem.

As in the example below, users' contextualizing was often challenged because of mismatched assumptions among the askers and helpers. Vamshi, who attempted to install MythTV in India, had hard time playing live TV. He assumed that this was at least partially due to him using a TV listings grabber for UK residents because the grabber for Indian residents was not yet available. Accordingly, the only contextual information he provided for his configuration was that he was using the grabber for UK residents. He also attached the error messages that he received when trying to populate the MythTV database. To this, different helpers solicited additional information depending on their beliefs about the cause of his problem:

```
Where did the messages you posted come from? Which
log file? They don't look like errors from
mythfilldatabase, they look like errors from
mythbackend. (ML: Jul 3, 2006, Phil)
```

While this helper focused on the connection between the frontend and the backend, another helper asked for configuration information on the capturing component:

```
What capture card are you using? What channels are
you expecting to receive, and do you have frequency
information for them? MythTV does need good data in
the channel database and watchTV can be unpredictable
if some channels are configured incorrectly. Maybe
you can configure your channels manually? (ML: Jul 3,
2006, Watkin)
```

This thread portrayed a typical challenge in queries: Despite the asker's trying his best to conform to the rules of etiquette, contextualization could require substantial dialogue between those helpers with misaligned assumptions to get to the context that they needed. This alignment work of assumptions, knowledge, and anticipation among members was a crucial challenge for sharing knowledge, as will be further discussed in the Discussion section.

Maintaining community knowledge over the long-term

Unlike the formal documentation and FAQs, the MythTV wiki was open to revision and inclusion of instructions by members of the user community. However, when users attempted to use these community-maintained instructions, two main challenges emerged: the obsolescence of solutions and missing context about solutions.

Obsolescence of solutions. Obsolescence of information has been shown to be a challenge to the viability of Wikis [21], and this challenge was observed in our study as well. Once a user posts a solution to the MythTV wiki, due to the collaborative characteristics of wikis, the community officially maintains the solution rather than the original poster, creating interesting coordination problems in maintaining the solution. Solutions may not be sufficiently managed over time due to the ambiguous ownership of the solution, often giving the wiki the reputation of being outdated for topics that are not popular.

As a result, the MythTV wiki was often approached with a perception that it might be outdated. Not knowing how well updated a MythTV wiki page was one of the reasons the users turned to the mailing list:

```
I've just purchased a TV Tuner card (Yuan SmartVDO
EzDVD MPG150/160/600). The board is labelled MPG600GR
REV 1.1. I'm aware that on the ivtv wiki page it says
that this board is not supported due to the Phillips
SAA7174HL chip but I don't know how current that info
is. Has any body had any experience with this board
or chip? (ML: Nov, 16, 2004, PoorH)
```

On the MythTV wiki, a page can become obsolete not merely because no one cares about the page, but also because the page now has to serve the community as a whole. An author cannot fix the page to note how well the solution works on her current updated system because information up on the wiki that is outdated for her can still be relevant to others. The author only knows her situation and not others':

```
It [the wiki] tends to get updated by those who tried
to use it, found it was wrong, found out how to do
what they wanted to do, and went back and fixed the
wiki. Unfortunately those people (people like me) are
unable to remove the cruft because they do not know
if it is still valid for some people or not. (I:
Peale)
```

Comments and warnings on the wiki seemed to play an important role in validating information, but if too prevalent they also created trust issues for using the information.

Missing context-information about solutions. During the knowledge distillation process of moving information from the mailing list to the wiki, the solution becomes decontextualized. Information about how the solution emerged – what the original problem was that started the thread, how much interest the problem received, what detours were made in coming to the final solution, or at what point of time in the community's conversation the problem emerged – becomes lost. As a result, users may consider solutions on the wiki to be less useful than solutions discussed over the mailing list. During an interview, Kyle described how the discussions that revolve around coming to a particular solution were an important context that should not be abandoned when a thread gets distilled into the wiki:

```
A forum (the mailing list) is just different, people
are going back and forth, presenting arguments, etc.
With a wiki, you can't see which parts were debated
over, which were just stuck there, and who stuck
them. (I: Kyle)
```

Another piece of contextual information that gets lost is who initiated the solution, which would influence the credibility of the information. Kyle again said:

```
On the forum, if the owner of the project says
something, you think about it differently than if you
read it on the wiki. (I: Kyle)
```

Of course, the challenge is maintaining the intricate balance between revealing necessary information and preventing information overload.

In summary, we observed several challenges that are widespread in the MythTV community for getting collaborative help on individualized use: the set of skills required for modifying solutions to work for one's configuration setting that not all users possessed; ambiguities about which information should be put forward for contextualizing problems; rapid obsolescence of solutions described on the wiki; and problematic de-contextualization of the distilled knowledge on the wiki.

Next, we discuss what our findings mean, and how we can leverage collaborative help for individualized use.

DISCUSSION

In this section, we discuss three challenges that emerge from our findings: *dealing with transparencies*, *navigating customization and appropriation gulfs*, and *aligning usage trajectories*. We believe these issues are present in many systems with complex configurations. We discuss each in turn, followed by our design implications.

Dealing with transparencies

For MythTV, reusing black-boxed configurations was the easiest way to get help from others. However, as mentioned, these artifacts often required intense labor to understand how to reuse them and to then modify the artifact to work for the specific problem.

As discussed earlier, the transparency of a configuration artifact often switched between black and white for configuration-based help depending on whether the artifact could be used as it was or not. In addition, the configuration artifact in question needed to be understood within the overall configuration, which was often black-boxed. While in some cases white-boxing a configuration was not a difficult task, in other cases white-boxing was a skill to learn, as with Avenard who wanted to understand which driver was in charge of the IR receivers. The difficulty was in knowing which part of a configuration artifact should be opened up and how to utilize that information. As seen in Avenard's case of fixing his udev rules, the transparency of a configuration artifact had to be compartmentalized, opened up just enough to solve the problem at hand.

For MythTV, accordingly, customizing parameters of a black-boxed component was not always sufficient for dealing with the innumerable sets of configuration differences among the members' systems. More than mere parameterization was required to reuse a solution or a configuration artifact and in collectively diagnosing problems. At the same time, the complete transparency offered in white-box reuse was unnecessary and burdensome.

MythTV users, then, need some form of *gray-box reusability* [30]. For MythTV and probably other systems, configuration information is shared with great transparency, no transparency, and partial transparency, depending on the context of the problem. Gray-boxing would be a more systematic way of allowing users to simultaneously ignore details when possible, open up a configuration artifact completely if necessary, and deal with parts as required. While providing such facilities will be challenging, supporting graybox reusability would facilitate sharing and learning how to modify reusable objects solutions.

Navigating the customization and appropriation gulfs

For many MythTV users trying to solve individualized problems, finding the right solution to adopt and knowing how to go about appropriating it are technically challenging tasks. As discussed earlier, Won, et al. [33] referred to MacLean, et al.'s work [13] in describing the customization gulf, i.e., the considerable effort and skills necessary for moving beyond simple parameterization. Similarly, a significant amount of experience and skill was required in order for MythTV users to go beyond the simple tweaking of solutions in the official documentations and FAQs and in reusing the solutions available in the wiki and on the Web. For example, Hugh, who had to modify xorg.conf to make it work for his particular needs, needed someone to guide him through the "gulf" in modifying his configuration file.

As Hugh's example showed, the MythTV users often had to understand what we call the *appropriability of a solution*, knowing which existing solutions can work without modification and knowing whether a solution could be appropriated for an individualized use. We call this problem the *appropriation gulf (of solutions).*

This gulf was widened in the MythTV community, due to the wiki missing context about how up-to-date the solution might be, for whom the solution did not work, for whom the solution worked best, and in what circumstances the solution was originally created (all of which are generally better described in the mailing list archive than in the wiki). It was difficult for a user to see the decontextualized solution and then decide how he might adopt the solution for his particular setting. (This is when the user turned to the mailing list to get help, because it is hard to figure out the appropriability of potential solutions by oneself.) Thus an asker with a seemingly unique problem may not initially realize how he could utilize a solution for other problems.

Mailing lists or forums are better for doing this in that they allow people to creatively repurpose solutions for unanticipated problems. For example, one user posted on the mailing list the need for creating a quiet living room by moving his backend server to another room, meaning that he had to deal with the wireless (or wired) connections between the frontend and the backend. A second user replied that he used MythTV with his laptop through a wireless network. This helper was able to give advice about the resolution of movie files given the constraints of the wireless network. A third user posted a more advanced way of utilizing a wireless network for using MythTV with his truck. He was sending video files every night to the truck from his basement, a setup that could be utilized for other circumstances such as using laptops or creating quiet rooms. The asker did not initially ask about MythTV's use in laptops or trucks to solve his quiet living room problem. Rather, the helpers who understood the key technical challenges in making a quiet room were able to bring in appropriable solutions for that particular problem.

The customization and appropriation gulfs create a barrier for a user when attempting to move beyond appropriating official or "safe" solutions and to find potential solutions for his or her individualized use. Helping users understand what potential solutions might be appropriable and helping users then know how to appropriate those solutions would be useful. While we only have the example of MythTV in this paper, we believe this form of help is likely to be useful for systems that allow for flexible configuration and tailoring activities.

Aligning usage trajectories

Maintaining one's MythTV was not just individual work. As we have seen, it often involved understanding other users' experiences to solve a current problem or determine future plans.

Not all users were interested in changing their system, of course. Some users came to the mailing list for a one-shot troubleshooting purpose, while others repeatedly came back to upgrade, change, or maintain their system.

Those users who were interested in their system over time had to consider what we call *trajectory alignment*, aligning themselves with other individuals and with the community. We use the term *trajectory* from Strauss [26], who defines a trajectory as: "(1) the course of any experienced phenomenon as it evolves over time (an engineering project, a chronic illness, dying ...) and (2) the actions and interactions contributing to this evolution."

All users had to align what they wanted to do with others' trajectories of use. This largely happened in two ways. First, users needed to know what other users had done and were likely to do, especially those users who had similar configurations. Second, as users anticipated how they wanted their MythTVs to be, other users who already had gone through similar experiences could discuss what could be potentially done or what might be a potential problem. In this way, one's trajectory of use became intermeshed with others'. For example, when Vamshi was trying to figure out why he could not watch live TV, one of the helpers asked which capture card he was using, because the helper knew about the consequences of using different capture cards due to his past experience. Another example of trajectory alignment is with critiques, which warn users about combinations of components that might break their MythTV in future updates. This could help users to prevent any potential future trouble.

Additionally, individuals aligned their usage trajectories of MythTV with that of the community as a whole. Some users on the mailing list wanted to configure their systems to be in accord with the MythTV developers' plans for the future. Users did not want to upgrade to a new tuner card, for example, that the MythTV developers were not willing to support.

In summary, trajectory alignment is a conceptual description of an important aspect of work that is required for supporting individualized use, namely the coordination work among individuals that allows the community members to keep each others' MythTV systems updated and well maintained. The concept reifies the importance of coordination and translation work among multiple users' and developers' trajectories.

DESIGN IMPLICATIONS

As stated earlier, our goal is to inform the design of tools to help end-users configure and manage their systems more effectively. As Stevens [24] noted, integrating appropriation and design discourses into user interfaces is a critical design challenge. We looked at the MythTV community in order to understand how to help end-users take advantage of the knowledge and experience of others when contending with problems of their own.

We believe that the three challenges presented in the Discussion section—dealing with transparencies, navigating the customization and appropriation gulfs, and aligning trajectories—are problems that must be addressed to help users in configuring complex systems or environments.

Several challenges could be addressed through relatively straightforward mechanisms, in particular:

- Users would benefit from having additional information about configurations. Our data suggest that if it were in gray-box form, it would be more useful. Gray-box solutions could include training wheels [7] for configuration files, configuration artifacts that allowed annotations, or ontology-based templates.
- Helping users jump over their customization and appropriation gulfs would be also useful. A potential solution might include providing forums to allow novice users to select and discuss best practices. Also, finding ways to maintain the wiki solutions, perhaps allowing the users to note when solutions fail for them, would help users know what to do.

Ameliorating both of these issues would be easier if there were access to a large number of user configurations. There is no way for the MythTV community, at this time, to know what configurations actually exist. This is true for many other systems as well. Users cannot easily share complete configurations that work; configurations cannot be grouped for comparison or help purposes.

Having a database of these configurations would help users with their appropriation gulfs. One could follow expert users, for example, noting when someone with greater expertise changed or upgraded her configuration. One might look up similar users to learn what he can do; this would be useful especially for new users who are looking to find an appropriate MythTV configuration to build.

A help system using crowd-sourced data could also help align trajectories:

- One could answer what-if scenarios, based on what others had done. What if a user wants to use DViCO tuner card instead of the popular Hauppage card? A user could look up the tuner card in the database to see what people had used, since tuner cards are notorious for having compatibility problems with Linux. A help system might be able answer whether people had trouble after installing the DViCO or help the user find others who had used the card.
- The data could be used to align the anticipated trajectory for a user's configuration with others, creating proactive help. This kind of help would determine potential problems, based on what users with similar configurations had done or had used.

CONCLUSION

In this paper, we studied the MythTV community as a place to explore collaborative help for individualized use. We were able to observe how the community struggled to support individual users through configuration artifacts and noted three significant challenges that we believe are common to systems with complex configurations. Our next goal is to implement our ideas for configuration support, and examine whether what we learned from MythTV can be expanded to other technical communities.

ACKNOWLEDGMENTS

This work has been funded in part by NSF 0905460. We thank our participants for helping the project and members of MISC and the SocialWorlds group for their support.

REFERENCES

1. Ackerman, M. and C. Halverson. Considering an organization's memory. In Proc. CSCW. 1998. 39-48.
2. Ackerman, M. and L. Palen, The Zephyr Help Instance as a CSCW Resource, in Resources, Co-Evolution and Artifacts, M. Ackerman, Halverson, C., Erickson, T., & Kellogg, W., Editor. 2007, Springer: New York. p. 37-57.
3. Ackerman, M.S., Augmenting organizational memory. TOIS, 1998. 16(3): p. 203-224.
4. Alavi, M. and D. Leidner, Review: Knowledge management and knowledge management systems. MISQ, 2001. 25(1): p. 107-136.
5. Balka, E. and I. Wagner. Making things work. In Proc. CSCW. 2006. 229-238.
6. Brereton, P. and D. Budgen, Component-based systems. IEEE Computer, 2000. 33(11): p. 54–62.
7. Carroll, J. and C. Carrithers, Training wheels in a user interface. CACM, 1984. 27(8): p. 800-806.
8. Fischer, G., A.C. Lemke, T. Mastaglio, and A.I. Morch, The role of critiquing in cooperative problem solving. TOIS, 1991. 9(2): p. 123-151.
9. Halverson, C., T. Erickson, and M. Ackerman. Behind the help desk. In Proc. CSCW. 2004. 304-313.
10. Kahler, H., Supporting collaborative tailoring, 2001, Roskilde University, Denmark.
11. Kiesler, S., B. Zdaniuk, V. Lundmark, and R. Kraut, Troubles with the internet. HCI, 2000 15(4): p. 323-351.
12. Koch, M. and G. Teege. Support for tailoring CSCW systems. In Proc. CSCW. 1999. 146-152.
13. MacLean, A., K. Carter, L. Lövstrand, and T. Moran. User-tailorable systems. In Proc. CHI. 1990. 175-182.
14. Mørch, A. and N. Mehandjiev, Tailoring as collaboration. JCSCW, 2000. 9(1): p. 75-100.
15. Mørch, A., G. Stevens, M. Won, M. Klann, Y. Dittrich, and V. Wulf, Component-based technologies for end-user development. CACM, 2004. 47(9): p. 59-62.
16. Nardi, B.A. and J.R. Miller, An ethnographic study of distributed problem solving in spreadsheet development, in CSCW. 1990, p. 197-208.
17. Page, S.R., T.J. Johnsgard, C. Uhl Albert , and D. Allen. User customization of a word processor. In Proc. CHI. 1996. 340-346.
18. Poole, E.S., Chetty, M., Morgan, T., Grinter, R. E., and Edwards, W. K. Computer help at home. In Proc. CHI. 2009. 739-748.
19. Poulin, J., J. Caruso, and D. Hancock, The business case for software reuse. IBM SJ, 1993. 32(4): p. 567-594.
20. Rodden, T., A. Crabtree, T. Hemmings, B. Koleva, J. Humble, Kettle, P., Kesson, P, and R. Hansson, Between the dazzle of a new building and its eventual corpse. In Proc. DIS. 2004, p. 71-80.
21. Roth, C. Viable wikis. In Proc. WikiSym. 2007. 119-124.
22. Singh, V. and M. Twidale. The confusion of crowds: non-dyadic help interactions. In Proc. CSCW. 2008. 699-702.
23. Singh, V., M. Twidale, and D. Nichols. Users of OSS-How Do They Get Help? In Proc. HICSS. 2009. 1-10.
24. Stevens, G., Understanding and Designing Appropriation Infrastructures. Ph. D. Thesis. 2009. University of Siegen.
25. Stevens, G. and S. Draxler. Appropriation of the Eclipse Ecosystem. In Proc. COOP. 2010. 287-308. Springer.
26. Strauss, A., Continual permutations of action. 1993: Aldine de gruyter.
27. Torrey, C., McDonald, D.W., Schilit, B.N. and Bly, S. How-To Pages. In Proc. ECSCW. 2007. 391-410.
28. Trigg, R.H. and S. Bødker. From implementation to design. In Proc. CSCW. 1994. 45-54.
29. Wulf, V. "Let's see your search-tool!"—collaborative use of tailored artifacts in groupware. In Proc. GROUP. 1999. 50-60.
30. Wulf, V., V. Pipek, and M. Won, Component-based tailorability: Enabling highly flexible software applications. IJHCS, 2008. 66(1): p. 1-22.