Supporting Re-Use in DIY Software Projects: A Gray-Box Approach

Tao Dong
School of Information
University of Michigan
4322 North Quad
105 S. State St.
Ann Arbor, MI 48109-1285
dongtao@umich.edu

Jina Huh
School of Information
University of Michigan
4322 North Quad
105 S. State St.
Ann Arbor, MI 48109-1285
jinah@umich.edu

Mark W. Newman
School of Information
University of Michigan
4322 North Quad
105 S. State St.
Ann Arbor, MI 48109-1285
mwnewman@umich.edu

Mark S. Ackerman
School of Information and Dept. of EECS
University of Michigan
4322 North Quad
105 S. State St.
Ann Arbor, MI 48109-1285
ackerm@umich.edu

CSCW 2011, March 19–23, 2011, Hangzhou, China.
ACM XXX-X-XXXXX-XXX-X/XX/XX.

Abstract

DIYers who work with software often attempt to reuse others' work wherever they can as they seek to assemble, modify, and extend their systems. In this paper, we briefly discuss the challenges faced by software DIYers in the process of reusing others' software configurations through our study of the MythTV community. We also discuss the benefits of enabling users to engage with others' configurations as "gray-boxes," allowing them to pay attention to just the parts that must be opened up and modified and ignore the rest. We propose a new technical facility called Tailor Wear to give users guidance and hints about where and how to modify configuration artifacts by visually presenting the tailoring traces left by similar or selected peers.

Keywords

Software customization, collaborative help, hacking communities, recommender systems, knowledge sharing

ACM Classification Keywords

H.5 INFORMATION INTERFACES AND PRESENTATION (I.7)

General Terms

Software customization, collaborative help, hacking communities, recommender systems, knowledge sharing

Introduction

People engaged in Do-It-Yourself (DIY) software projects often take existing components and try to modify and tailor them for their own needs [2]. Relatively few people appear to create their own components from scratch. Instead, they reuse what they can.

While reusing existing components saves work, it can be an extremely frustrating experience, as anyone trying to configure complex systems can attest. What can be done to make it easier for people to re-use others' work for their own purposes?

To examine this question, we conducted a study of help interactions in the MythTV community[1] that aimed to understand the challenges MythTV users faced in giving and receiving help around configuration [4]. In this paper, we summarize some of the challenges that are faced by MythTV community members and describe our early work on developing technical facilities to address those challenges. In particular, we describe *Tailor Wear*, a technical facility we are currently developing to enable appropriation of peer-contributed configuration artifacts.

In the rest of the paper we first present our observations of the MythTV user community. We then describe the current design of Tailor Wear, and discuss how it can address the configuration issues in tailoring software components.

MythTV And Its Community

MythTV is a highly tailorable open-source home media system. Through modifying its configuration files, home media enthusiasts can enable and customize a vast number of features such as recording TV shows, watching DVDs, playing games, and receiving weather forecasts. Though flexible and powerful, configuring MythTV can get very complicated. First, the configuration space is huge and growing. Secondly, 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 subscribed to a standard or high definition TV service also affect the configuration of one's MythTV system.

In an earlier study, Huh, Newman, and Ackerman studied the MythTV user-mailing list where users receive technical support from peers [4]. This study analyzed a sample of approximately 4000 messages to identify patterns of help exchanges and identify challenges faced by list members. We further validated our findings by conducting a total of 12 interviews with MythTV users.

Our observation indicated that configuration artifacts such as settings files, logs, scripts, error messages, and the outputs of certain diagnostic tools played an important role in the collaborative help process. In many cases, providing and receiving help in the form of such configuration artifacts appeared to greatly improve the efficiency of sharing knowledge. If the shared artifact was compatible with the receiving user's system and relevant to their needs, they could employ

[1] http://mythtv.org

the artifact without completely understanding its internal logic or the work needed to recreate it. However, when the shared artifact was not a perfect fit, users faced considerable challenges that hampered their ability to reuse or modify components and configurations.

Challenges of Configuration-based Help

In [4] we describe a number of issues faced by users. Here we focus on just one of the challenges—the difficulty of determining whether an artifact could be used as-is or must be opened up and modified. 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 transferred and reused by anyone else requiring, at most, minor modifications. These could be appropriated by others as "black boxes," meaning that the receiver could make use of them without understanding anything about how they worked. However, users struggled when they had to open up a configuration artifact (e.g., a settings file), search for sections requiring adaptation, and finally make the right modifications. In such cases, the artifact became a "white box," requiring a high degree of comprehension on the part of the receiver in order to make use of it.

A critical problem for user in using configuration artifacts, then, is determining which part of the configuration needs to be modified and what part can be ignored. In the following example, Avenard was trying to find a certain line to modify in a device configuration file but became overwhelmed by the irrelevant complexity posed by other parts in the configuration file:

> After reading a lot about udev[2], 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)

As the above example shows, it is not always the case that components or configurations of components can be "black-boxed" (closed) or "white-boxed" (completely open). In our view, then, MythTV users, would benefit from some form of "gray-box" reusability. If done correctly, gray-boxing would allow users to simultaneously ignore details when possible, open up a configuration artifact completely if necessary, and deal with parts as required. Supporting gray-box reusability would facilitate sharing and learning how to modify reusable solutions.

Technical Support for Gray-box based Software Customization

Wulf, Pipek, and Won [5] argue that component-based architectures are particularly well-suited to presenting end-users with a gray-boxed view of tailorable systems. However, additional challenges arise when dealing with systems such as MythTV that were not designed to support gray-box reusability. As a starting point , we believe a technical facility to collect, monitor, and share configuration traces from various community members would be most helpful to allowing a wide range of users to hack. Users could see other users' revisions of and

[2] Technically, udev is part of Linux and not MythTV proper. However, modifying udev rules is a common activity for MythTV users. At least one page of the MythTV wiki is dedicated to udev rules (http://www.mythtv.org/wiki/Device_Filenames_and_udev) and udev is mentioned in hundreds of messages in the mythtv-users archive.

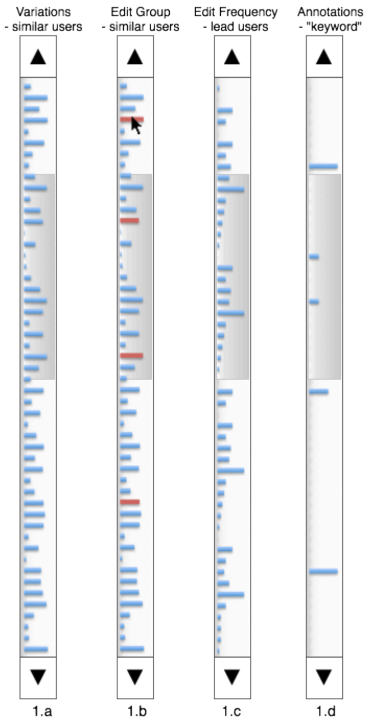

Figure 1. An example Tailor Wear visualized as attributes-mapped scroll bars. Each scroll bar is a bar chart showing a particular type of trace attributes. The above visualizations answers questions such as: which parts of the configuration are most identical and most varied across users (1.a)? Which parameters of the configuration are modified in a coordinated way (1.b)? Which parts of the configuration do the lead users frequently modify (1.c)? And which parts of the configuration are annotated most frequently with a certain keyword (1.d)?

annotations to their configurations. We are currently exploring a variety of techniques to present transformed, aggregated, or filtered traces. Our system is called Tailor Wear, inspired by the computational wear approach [1,3]. Computational wear employs the metaphor of physical wear that emerges on a document that is used over time. The smudges, crease lines, and annotations left by the users on the document can signify the patterns of usage and give new users hints about how to most efficiently consult the document.

Similar to the document wear metaphor, Tailor Wear allows users to explore the traces of configuration work left by similar or selected others who previously modified their own instance of the same configuration artifact. Traces are computational representations of users' revisions and annotations to a configuration artifact. Those traces are captured, aggregated, and re-presented to users without effort of their original creators.

As mentioned before, the configuration traces left by peers with similar experience or needs could be most helpful to guide gray-boxing. By restricting the traces to those of the similar others, users are more likely to see ready-made solutions to the problems they run into or a new feature they want to add.

It is challenging to present those traces in an informative, intuitive, and non-distracting manner. Users will want to view a subset of the configuration traces that are relevant to their problem. We are exploring a variety of visualization techniques to present transformed, aggregated, and filtered traces. We think the attribute-mapped scroll bars [3] (shown in figure 1) could be a sensible approach to display a variety of useful abstractions of accrued configuration traces. Alternatively, an attribute-mapped highlighting scheme or heat maps could serve a similar purpose.

As well, Tailor Wear can automatically suggest similar others from whom traces are transformed and presented in the user interface. We are exploring a number of different ways of determining similarity of users by comparing users' configuration artifacts, usage patterns of their system, previous configuration traces, and self-reported profiles.

To some users in a learning process, just seeing what the similar others have done might not be enough. Instead, they can choose to follow lead users to learn best practices. Such lead users or experts can be identified by a number of expertise finding algorithms [6] and new metrics developed by analyzing the usage data of Tailor Wear, i.e. who initiates changes that are adopted by others.

Finally, Tailor Wear also allows users to annotate their own and others' traces with comments, instructions, and potentially questions and answers. Those semi-structured annotations provide context for the shared tailoring traces, providing foundations for searching traces, embedding tutorials in configuration artifacts, and capturing feedbacks.

To sum up, Tailor Wear enables gray-boxing as a process of social navigation in editing configuration artifacts. Tailor Wear leverages tailoring traces of similar or selected peers to socially signify where attention should be paid, what can be ignored, and how appropriate changes could be implemented.

Conclusion

In this paper, we described a facility to provide help to software DIYers, based on our study of the challenges faced by members of the MythTV community. We discussed the benefits of enabling users to customize their configuration as a gray-box, with attention only paid to the parts that must be opened up and modified. A technical facility called Tailor Wear was proposed to give users guidance and hints about where and how to modify a configuration by visually presenting the tailoring traces left by similar or selected peers.

References:

1. DeLine, R., Khella, A., Czerwinski, M., and Robertson, G. Towards understanding programs through wear-based filtering. *Proceedings of the 2005 ACM symposium on Software visualization*, (2005), 183–192.

2. Hartmann, B.**,** Doorley, S. and Klemmer, S. R. Hacking, Mashing, Gluing: Understanding Opportunistic Design. *IEEE Pervasive Computing* 7, 3, (2008).

3. Hill, W.C., Hollan, J.D., Wroblewski, D., and McCandless, T. Edit wear and read wear. *Proceedings of the SIGCHI conference on Human factors in computing systems*, (1992), 3–9.

4. Huh, J., Newman, M. W., Ackerman, M. S. Supporting Collaborative Help for Individualized Use. Will appear in the proceedings of ACM CHI 2011.

5. Wulf, V., Pipek, V., and Won, M. Component-based tailorability: Enabling highly flexible software applications. *International Journal of Human-Computer Studies 66*, 1 (2008), 1-22.

6. Zhang, J., Ackerman, M.S., and Adamic, L. Expertise networks in online communities: structure and

algorithms. *Proceedings of the 16th international conference on World Wide Web*, (2007), 230.

[1]