Online Customization Sharing Ecosystems: Components, Roles, and Motivations

Mona Haraty
University of British Columbia
haraty@gmail.com

Joanna McGrenere
University of British Columbia
joanna@cs.ubc.ca

Andrea Bunt
University of Manitoba
bunt@cs.umanitoba.ca

ABSTRACT
The rise of open platforms and public APIs has enabled more users of customizable software to customize by installing plugins or add-ons that are created and shared by others. Despite the prevalence of online customization sharing, we know little about how and why online customizations are shared. Through interviews with 20 users of four diverse systems who have extensive experience with sharing their customizations, we reveal the concept of customization sharing *ecosystems*. These ecosystems include multiple components for hosting customizations, discussing, and managing them; the ecosystems are sustained through users acting in a diverse set of roles (e.g., sharers, re-users, reviewers, problem reporters, requesters, helpers, publicizers, and packers). Our interviews also reveal motivations for creating and sharing customizations online which overlap considerably with those in open source software. Our findings highlight tradeoffs and design considerations in these ecosystems.

Author Keywords
Customization sharing; customization; ecosystems

ACM Classification Keywords
H.5.m; K.4.3

INTRODUCTION
More and more users are taking advantage of software customizability to expand software's capabilities through additional features or to enable personalized workflows. Most of these users are benefitting without having the required skills or the time to create these customizations on their own. Instead, they are adopting customizations made by others, through plugins and other mechanisms. For example, 85% of Firefox users have chosen to customize by installing add-ons [30]. This phenomenon has been enabled by 1) software applications that are designed as open platforms that offer public APIs, thus allowing developers to create plugins and cross-application customizations using tools like IFTTT and Alfred, 2) customization sharers who are willing to create customizations, and 3) technologies that enable those sharers to share their customizations with other users. To support the important role of customization sharers[1], we need to understand what motivates sharers to create and then share customizations, what mechanisms they use to do so, and how those mechanisms either support or hinder sharing practices.

Sharers contribute to customizable software—often commercial—by extending its functionality. There is a vast literature on motivations for contributing to free and open source software (FOSS) and its infrastructure is designed to leverage those motivations. We were curious to see to what extent similar motivational factors drive creation and sharing of customizations for proprietary software.

Research on customization sharing has focused predominantly within organizational boundaries [6,12]. It remains unclear how the customization sharing practices translate from within-organization settings to online settings where sharers come from diverse contexts and may have other motivations to share. Little empirical research has been conducted to investigate online environments and mechanisms in terms of their conduciveness to customization sharing.

Our research takes that next step in understanding customization sharing practices beyond those within an organization. To understand what mechanisms sharers use to share their customizations and what motivates them to share, we conducted interviews with 20 customization sharers of four diverse systems: Sublime Text, Minecraft, Alfred, and IFTTT; the first two being customizable systems and the others being customizing tools used for creating customizations. Being a game, Minecraft adds an interesting perspective on sharing as will be seen. In fact, customizing is so commonplace in games that is considered an important part of *using* the system [7].

CSCW '17, February 25-March 01, 2017, Portland, OR, USA

ACM 978-1-4503-4335-0/17/03…$15.00
DOI: http://dx.doi.org/10.1145/2998181.2998289

[1] We use *customization sharers* to refer to people who create and publish their customizations, and we use *re-users* to refer to people who use sharers' customizations.

This study is, to our knowledge, the first to investigate online customization sharing, and makes several key contributions. Through documenting current customization sharing practices in four diverse systems, we reveal the concept of sharing *ecosystems*. These ecosystems are often complex, consisting of various components to support the different aspects of customization sharing and re-using: hosting customizations, discussing them, finding them, installing them, and keeping them updated. We encapsulate the roles that bring these ecosystems to life and show that they have some, but not full, overlap with the roles in FOSS projects. Our findings shed light on motivations to create and share customizations and the degree to which these motivations overlap with those that drive contributions in FOSS. Collectively, our findings point to implications for the design of customization sharing ecosystems, and highlight important design tradeoffs.

BACKGROUND AND RELATED WORK

Our research draws on the literature on customization and customization sharing. Sharing customizations that involve programming has some similarities with developing and participating in FOSS projects, thus we also briefly review literature on roles and motivations in FOSS.

Types of customizations

The customization literature has identified a range of different customizations, although there is no standardized terminology. For example, Opperman and Simm talk of two broad customization categories: functionality and interface adaptations [27]. Bentley and Dourish similarly distinguished surface customizations, in which users select from a set of predefined options, from deep customizations, in which users, for example, add a new behavior to the system [1]. Haraty and McGrenere define advanced personalization/customization broadly as customization that goes beyond changing the look and feel, and involves changing functionality [9].

Motivations to customize

MacKay identified several triggers (motivations) and barriers to customizing one of the earliest Unix environments targeted at non-technical users. Some of the most common triggers were noticing one's own repeated patterns, retrofitting when the system changed, and seeing a "neat" customization; the most common barrier was lack of time [20]. Motivations of game customizers, however, are quite different. They consider customizing an artistic endeavor, allowing them to make games "their own" and thus increase their enjoyment of game play, and helping them acquire a job in the game design industry [28,31].

Customization sharing: benefit, roles, and medium

Several studies have documented customization sharing habits and the different types of users who are involved in the sharing process *within* an organization [8,12,18,21]. Most of these studies identified a continuum of three types of users: ordinary users, local developers (sometimes referred to as translators and tinkerers), and professional programmers or lead users. Lead users created customizations for their own use, and translators created simplified and task-specific versions of the customizations created by lead users [18]. Similar to translators, local developers in Gantt and Nardi's study also created customizations for the employees of their organization [8]. Some local developers, referred to as gardeners, were paid to do so in certain organizations [8]. We further compare these roles with respect to our findings.

MacKay's pioneering study of sharing Unix configuration files and email filtering rules revealed the importance of sharing customizations by showing that only a small percentage of people customize, and most people prefer to ask others about a customization or to modify an existing customization file [18].

In a *within*-organization study of customization sharing, Draxler et al. investigated appropriation practices in Eclipse, a customizable development environment. They suggested three principles for supporting customization sharing: the ability to browse plug-ins installed by colleagues, providing an awareness of peers' customization activities, and the ability to install tools that are already in use by peers [6]. Murphy-Hill and Murphy found that peer observation and peer recommendation are programmers' primary means of discovering new plugins [25]; practitioners indicated a preference for peer interaction compared to other information sources such as forums and Twitter for discovering and learning about customizations.

Several studies identified email as an effective way of sharing customizations *within* an organization [12,21]. Kahler's study of sharing Word add-ons via email found it effective for small work groups, but suggested that to scale *beyond* an organization, shared customizations need to have rich annotations and comments to provide context for others [12]. Studies of sharing customizations via wikis revealed users' dissatisfaction with allowing others to edit their customizations [13], and difficulty in knowing who can see their scripts and who is affected by their edits [17]. Altogether this suggests that designing mechanisms to support sharing customizations is not straightforward.

Online customization sharing has also been investigated in the context of remixing behaviors in maker communities [26], where little user activity was observed around generated (remixed/customized) designs.

FOSS: background, roles and motivations

FOSS projects often start with a single programmer solving his own problem, and then making the solution available to others. Once a FOSS project attracts developers who would like to contribute to the project, the owner becomes a coordinator [29]. In addition to the coordinator, the following roles exist in a typical FOSS project: core developers who write most of the code and review

submitted code, contributors who can become core developers after sufficient contributions (as voted by the core developers), problem reporters, user support, and end users [24].

Individuals have heterogeneous motivations to participate and contribute to FOSS. Although no one motivation dominates in the community [15], the promise of higher future earnings [11], the need to solve one's own problem [14], and intellectual curiosity [15] have been reported as the most important drivers to contribute to open source projects. Some contribute to improve their programming skills, some enjoy programming, some have a personal need for the code, some feel an obligation to the community because of using FOSS and believe that the code should be open, and some want to enhance their reputation [15].

To summarize, the existing literature on sharing customization focuses on understanding customization sharing for a single tool and/or within a single organization. Thus, the broader landscape of online customization sharing is relatively unknown. This paper builds on and extends this body of work by investigating *online* customization sharing practices for a variety of tools.

METHODS

We conducted a semi-structured interview study with 20 users of four systems with the goal of investigating the mechanisms they use to share customizations as well as their motivations for creating and sharing customizations.

Systems investigated

To find customizable systems that support sharing and re-using of customizations, we searched the Web for the following keywords: "share" plus each of "customization", "personalization", and "configuration" keywords. In addition, we asked friends and colleagues to introduce us to any customizable tools that they were aware of. From this initial list, we chose to review 10 systems that represented good coverage of the sharing mechanisms found. The 10 systems included: two blogging platforms (WordPress, Tumblr), two text/code editors (Vim, SublimeText), an application launcher (Alfred), an automation tool (IFTTT), a game (Minecraft), a task management tool (RememberTheMilk), a web browser (Google Chrome), and a desktop customization program (Rainmeter).

We identified key dimensions across which the shared customizations differed: their human readability, granularity, and authoring accessibility. The sharing mechanisms in these systems differed in where the customizations were shared, whether the platform ensured security of customization, whether it provided meta-data and supported commenting on shared customizations, and whether it allowed for customization requests. Then, to further investigate the characteristics of sharing mechanisms from sharers' perspectives, we chose four systems – Sublime Text, IFTTT, Alfred, and Minecraft – that represented the diversity across the ten systems in terms of the dimensions we identified. Each of these systems is briefly described below. We intentionally chose to include two *customizing* tools, ones that are used to customize other apps and services, namely Alfred and IFTTT. The other two systems that we chose are *customizable* systems: Minecraft and SublimeText.

System	List of active sharers	Response rate	#of Ps	Ps' Labels
Alfred	www.packal.org/contibutors	60%	6	Alfx
IFTTT	ifttt.com/top_chefs	50%	5	IFTx
MineCraft	www.curse.com/mc-mods/	36%	4	MCx
Sublime Text	packagecontrol.io/browse/authors	24%	5	SUBx

Table 1. Systems investigated in the study, URLs where we found lists of top customization sharers for each system, response rates, and how we refer to the participants from each.

Sublime Text is a customizable text/code editor. Creating an advanced customization in Sublime text is done through developing plugins in the Python language using the Sublime Text API and wrapping it into a package. For example, "All Autocomplete" is a package that extends Sublime's autocomplete by finding matches in all open files—instead of only the current one.

Minecraft is a game where users create worlds by breaking and placing construction blocks. Modifications (mods) of the *Minecraft* code add a variety of gameplay changes ranging from new blocks, to new items, to entire arrays of mechanisms to craft. For example, "Advanced Genetics" is a mod that gives the player and other entities in the game supernatural abilities such as teleporting or flying by injecting genes using a syringe. Creating a mod requires programming in Java.

Alfred is an application launcher and productivity tool. Users can automate their tasks by creating customizations (called *workflows*). Creating a workflow in Alfred involves trigger-action programming [32] in a visual programming environment, where users can connect triggers to actions. Creating an advanced workflow involves writing a script in a programming language of choice. An example of a workflow was "Movie Ratings" with which users can search for a movie and see its IMDB, Rotten Tomatoes, and Metacritic ratings. Workflows allow users to use and interact with their apps and web services more efficiently.

IFTTT is a web-based service that allows users to extend the functionality of their applications by creating "If This Then That" customizations, called recipes, which connect their different applications. Recipes are created using a visual programming environment. An example of an IFTTT recipe is one that connects one's Facebook to Dropbox by automatically saving new Facebook photos in which one is tagged to a Dropbox folder.

PARTICIPANTS

We recruited 20 participants who were actively sharing customizations in the four systems. To obtain their contact information, we used the "top sharer" lists from each system (Table 1). From these lists, we contacted those whose contact information was publicly available. The response rate ranged from 24% (for Sublime Text) to 60% (for Alfred) and this difference did not seem to be related to any characteristics of the systems or sharers. All participants were male (4 unemployed, 4 student/postdoc, and 12 developer/engineer) and were from eight countries. Their age ranged from 20 to 46, with 18 of the participants in their twenties or early thirties. They received $10 for their participation in the form of direct payment or donation to their favorite charity (three declined any compensation). See Table 1 for how we refer to the participants from each system.

Procedure

All the interviews were conducted via Skype. During the interviews, which were semi-structured, we asked participants about their experience with sharing their customizations. Specifically, we asked them to describe their motivation for creating and sharing customizations, their process of customizing and sharing their customizations, their interactions with users of their customizations, and their use of others' customizations. We personalized the questions for each participant based on their online sharing activities. The interviews lasted from 10 to 75 minutes, depending on the participants' willingness to talk about their various experiences. Interviews with IFFTT participants tended to be shorter than the others; as we will describe below, sharing in IFFTT is generally a simpler process (both technically and socially), meaning that those participants had less to talk about.

The interviews were audio recorded and later transcribed in full. The transcriptions were qualitatively analyzed using inductive thematic analysis [3], focusing on data related to mechanisms used for sharing and re-using customizations, motivations to share, and social interactions around the shared customizations. The lead author initially coded the data. Identified themes were then refined collaboratively by all co-authors, with frequent revisits to the raw data by all.

FINDINGS

Our findings revealed that the mechanisms our participants used to share their customizations were supported by systems comprised of people and tools interacting with each other to support various sharing-related activities. We refer to these components and their interactions as *customization sharing ecosystems*. First, we describe these ecosystems in terms of their components, how people interact with the components to share and re-use customizations, and various roles that people play within the ecosystems. Then, we discuss what drives the ecosystems: what motivates people to create and share customizations, what makes a customization shareable, and how re-users trust and use the shared customizations. Throughout our findings, we compare and contrast our results with the prior work on customization sharing within organizations as well as the FOSS literature.

Customization sharing ecosystems

Ecosystems' components

Our findings reveal customization sharing ecosystems and that they consist of customizations, customization groups, customizable software, customizing software, discussion places, customization managers, customization repositories, and source code repositories. Not every ecosystem that we uncovered included all components. Table 2 summarizes each ecosystem based on these components. We briefly explain each next.

Customizations: Customizations in these ecosystems vary across two of the dimensions we identified in our initial review of customizable systems: the authoring accessibility and readability (see Table 2). In addition, we also found customizations differ with respect to their specificity to their author's needs. Later in the paper, we will discuss how these properties of customizations drive the customization sharing ecosystems by influencing re-users' trust in customizations and sharers' decision on whether to share a customization.

Customization groups: In the Minecraft ecosystem, our participants reported using *modpacks*, a group of mods that are put together to fit a specific need or a theme. In addition to simplifying downloading, one benefit of using modpacks instead of individual mods is that gamers normally play with lots of mods and the conflicts between the mods are taken care of in modpacks. We did not see the notion of grouping customizations in the other ecosystems.

Source code repositories: Places where sharers upload the source code of their customizations include online generic code repositories such as GitHub, and dedicated code repositories such as CurseForge (for games).

Customization repositories: Places where sharers upload their customizations. Different repositories offer different functionalities such as facilitating browsing and searching of their hosted customizations. Some provide meta-data on the customizations (e.g., number of downloads, ratings), and some ensure the security of customizations through a human moderation process.

Customization managers: Tools that connect the customizable software and the source code repositories. They allow re-users to browse and install customizations, and sharers to distribute updates to their customizations.

Discussion places: All the ecosystems include public discussion places such as forums, IRC channels, GitHub, and Twitter. Our participants reported announcing their

customizations, receiving feedback on their customizations, and supporting their customizations in these places. The feedback ranged from thanks or admiration, to feature requests, bug reports, contributions, and suggestions for improvements, both from users of their customizations and other sharers. Customization sharers, in both Alfred and Minecraft ecosystems, support each other in these places for developing better customizations. These discussion places are critical for ecosystems such as Minecraft where a substantial part of the modding knowledge is embedded within the community, according to MC2. Participants' attitudes towards the discussion components of the ecosystems varied (both within and across the different systems we studied). For example, IFTTT participants did not expect to support their customizations in any way: "*I wasn't really sharing the recipes to support and manage them, it was just kind of a throw it out there and if they want to use it they can"* [IFT2]. It is more common in the Alfred and Minecraft ecosystem to support their customizations, however, the desire to do so often depends on the nature of the requests and the customization: "*the one [workflow] that I received the most bug reports for was […]. I never really investigated [the reports] because it worked for me* [Alf5].

Degree of ecosystem component integration: three models

We briefly describe how our respondents reported interacting with the different components described above to share and reuse customizations in each ecosystem.

Considering how the ecosystems' critical components are integrated together, each ecosystem can be best described as one of the following three models: *a collection-of-islands* (the critical components are disconnected from each other), *a pipeline* (the critical components are connected such that to share a customization, sharers only need to upload its source code to a code repository and do not have to do anything else to facilitate re-using), or a *one-stop shop* (a single component does the job of all the critical components). Below, we illustrate how the models are instantiated within each of the four ecosystems.

IFTTT: The simplest among the systems we studied, the IFTTT website is the single place that supports all the processes of creating, sharing, browsing, searching, and installing recipes. Sharing/publishing a recipe is as easy as a single click in the process of creating a recipe, with no written code involved. Our participants reported finding shared recipes by searching or browsing the recipes. However, the ease of creating recipes caused some of our participants to create a recipe without searching first: "*I just have a need and start creating it. IFTTT process is so simple and quick that it's just as fast, if not faster, to just go ahead and make it and customize the field the way that I'm thinking than try to work off somebody else's existing recipes*" [IFT3]. This has led to many duplicate recipes on IFTTT [33]. There is no dedicated discussion place for IFTTT recipes, which does not appear to be missed by our participants: *"They [IFTTT recipes] are just such small things and such an almost incidental part of my day to day work. I can't imagine engaging in commenting back and forth on any of them"* [IFT3]. IFT2 and IFT5, however, mentioned receiving questions about their recipes on Twitter which we learned is used for requesting a recipe as well. In addition, users can tweet their created recipes from within IFTTT. Overall, this ecosystem is best described as a one-stop shop.

Minecraft: Our Minecraft participants reported uploading the source code of their mods in various code repositories such as GitHub (a generic code hosting service) and CurseForge (a dedicated one for games). Re-users can browse and download mods from mod repositories such as the Curse website, or install through a mod manager such as CurseClient. We found other customization repositories and customization managers for Minecraft, but none were used by our participants. According to our participants, users of their mods ask questions, report bugs, and request features mostly in the Minecraft forum, but also in the Curse forum and GitHub. Our participants find new mods to play by watching Youtubers who publicize and demonstrate how to use a mod or through modpacks that are listed and featured by the game launchers. Much of the Minecraft sharing ecosystem can be described as a pipeline, because once the customizations' source code are uploaded to the source code repository, they will be available on the customization repository and the customization manager to

Components | systems	**Minecraft**	**Sublime Text**	**Alfred**	**IFTTT**
Customizations investigated	Mod	Package	Workflow	Recipe
-Is authoring a customization accessible to non-programmers?	No	No	Depends on the workflow	Yes
-Are customizations human readable?	Yes if open source	Yes if open source	Yes	Yes
Customization group	Modpack	None	None	None
Source code repository	GitHub, CurseForge	GitHub	GitHub	N/A
Customization repository	Curse website	None	Packal, personal websites	IFTTT
Customization manager	CurseClient, FTB	PackageControl	None	IFTTT
Discussion place	Forum, GitHub	Forum, GitHub	Forum, GitHub	Twitter
Ecosystem model	Pipeline	Pipeline	Collection-of-islands	One-stop shop

Table 2. Summary of the ecosystems in terms of their components and properties

install, and the installed mods will be kept updated for their users as developers update them.

Sublime Text: Our Sublime Text participants reported uploading their packages' source code to GitHub, and request their package to be listed in the Sublime Text's customization manager (Package Control), which is integrated with Sublime Text such that users can search, install, and update packages from within Sublime Text. Despite not having to leave Sublime Text for finding new packages, our participants reported going to Package Control for that purpose, because usage data such as number of installs—which they find helpful in deciding between similar packages—are available only there. Discussions around a package include bug reports and feature requests and they happen in GitHub. The Sublime Text sharing ecosystem is also best described as a pipeline, because once sharers upload their packages to GitHub, they will be available to install both from the customization manager and from within Sublime Text, and the installed packages will be automatically updated for their users when updated by their authors.

Alfred: Our Alfred participants reported uploading both their workflows and their source code separately in two disconnected places: GitHub, and Packal—a website that is supposed to be the central repository for Alfred workflows. In addition, they announce their workflows in the Alfred forum's "share your workflow" thread. Bug reports and feature requests are received in both the forum and GitHub. Our participants reported finding new workflows by regularly checking the forum, rather than through Packal even though the latter is designed specifically for this purpose. The Alfred sharing ecosystem can be best described as a collection-of-islands, since no integration exists between its components.

Some of our participants commented on how they used to share their customizations in the past and how that has changed. We found that the way sharing is supported in each system has evolved over time, and the evolution, except for IFTTT, has been quite organic. Customization sharers used to share their customizations in forums. Over time, users or third party developers began to contribute to the sharing process by developing dedicated tools that facilitate sharing and reusing of customizations. These contributions have given rise to what we refer to as ecosystems. For example, Sublime Text's customization manager was developed by a Sublime Text customization sharer as a way to distribute updates to his package [2].

Roles in the ecosystems

Through our interviews with customization sharers, we also gained insights into roles other than sharers that occur in the sharing ecosystems. The descriptions above point to different activities various people perform in these ecosystems. Here, we consolidate this discussion into a set of roles: customization sharers, reviewers, re-users, problem reporters, requesters, helpers, publicizers, and packers. Some of these roles were common in all the four ecosystems, while others were only found in one or two ecosystems. As we describe below, these roles have some overlap with those previously identified in both within-organization customization sharing and FOSS projects. The roles consist of two main categories: sharers and re-users, each of which has subcategories. Reviewers are a subset of sharers, and problem reporters, requesters, and helpers are subsets of re-users. Publicizers and packers are two secondary roles.

Customization sharers: People who author and share customizations online with others. Sharers communicate with and help interested users to use their customizations. Some of our sharers also reported creating customizations for others upon request. As a result, customization sharers often have a more complex and multi-faceted role than those identified in previous studies of within-organization sharing, for example, combining the roles of *lead users* and *translators* in MacKay's study [18], and *local developers* and *gardeners* in Gantt and Nardi's study [8]. Using FOSS terminology, sharers often take on the combined roles of *owner*, *core developer*, and *contributor*.

Reviewers: A subset of customization sharers in the Alfred ecosystem play this role, by providing feedback to other sharers in the discussion places for the purpose of assisting the creation of higher quality customizations. They do so completely voluntarily upon requests: "*when it's a new user who is saying this is my first workflow, even if I don't use it or intend to use it. I always download it and see how it's constructed to be able to say what would have I done differently here […] to incentivize them to work for better solutions in the future"* [Alf3]. This feedback-driven role differs from the role of local developers in [8] who consulted with end users to *create* customizations to suit their needs. The reviewer role is common in FOSS projects [24], however, reviews there tend to be the necessary prerequisite to having a piece of code included in the central code base. The fact that review sometimes takes place in customization sharing ecosystem, even though customizations neither become part of an official code base nor have to be used by other users, was an unexpected finding.

Re-users: People who re-use customizations created by others. Some re-users play other roles such as problem reporters, requesters and helpers which we describe below.

Problem reporters: A subset of re-users of a given customization report its problems. They do so in various discussion places, some of which, like GitHub, are more suited to the task of bug reporting. The inclusion of and the support for problem reporters is one of the benefits of sharing customizations online: *"Those [bug reports] are very good because it fixes the bug for me and everyone else"* [Alf3]. Sharers commented that a system like GitHub

makes it easier to track and organize the reported problems compared to forums, where the problems are buried in other posts leading to redundant reports and responses. Despite this preference, many bug reporters continue to use the forums. The same role of problem reporter exists in FOSS projects, where users of the software are relied upon to report problems, however, we did not see an analogous role reported in studies of within-organization customization sharing.

Requesters: A subset of re-users who solicit a customization from others. Alf1 reported receiving a direct request from someone, and IFT4 responding to a public request on Twitter from someone he follows on Twitter. While this role has not been explicitly identified in prior studies, as mentioned earlier, local developers in [8], and translators in [18] created customizations for their colleagues sometimes in response to requests.

Helpers: A subset of re-users help other re-users who have difficulty using a customization. Some sharers reported relying on these helpers: "*If I have a relatively new mod, it's usually like you answer questions and help them out but once the mod gets bigger you have people who already know about the mod and know how to solve its problems. They usually take care of answering all the questions*" [MC4]. Helpers monitor the discussion places, and provide answers to users' questions. In contrast, within an organization, employees direct their questions about a customization to someone who is likely to know the answer [8,18]. Helpers' job is similar to the mundane but essential task of user support in FOSS projects [14].

Publicizers: In the Minecraft ecosystem, a few famous YouTubers publicize mods by demoing them. They not only create awareness of new mods, they also make it easy for others to use them. This role is similar to FOSS advocates who blog about various FOSS projects to raise awareness of them.

Packers: In the Minecraft ecosystem, a group of people — called modpackers — put mods together and take care of the conflicts between the mods. In the same way that translators in [18] created task-specific sets of customizations by reusing the customizations created by the lead users, modpackers create theme-specific sets of mods using the mods created by mod authors.

To summarize, compared to customization sharing in organizational settings, we found more roles in the online settings, many of which have analogies to those needed to build and maintain a FOSS. The expansion of roles over the within-organization setting could be attributable to the various discussion places facilitating online peer interactions. The transparency of the online interactions could contribute to online reputation building. We return to ways to better support these emerging roles in the Discussion.

What drives the customization sharing ecosystems

Prior studies of customization sharing have shown that only a small percentage of users of customizable tools create and share customizations with others [18]. Understanding the motivation of this small group is crucial for supporting them effectively, hence keeping the ecosystems alive. The other factor that plays a role in driving these ecosystems is re-users' trust in shared customizations. After all, if no one other than the original authors use their customizations, sharing becomes worthless. In this section, we describe our participants' motivation to create and share customizations, describe the characteristics of *un*shared customizations, and report on the characteristics of the ecosystems that help this particular group of re-users to trust and use the customizations shared by others.

What motivates sharing customizations

We found that a combination of motivations drive customization sharers' behaviors. While our participants from different ecosystems shared common motivations, some motivations were more pronounced in some ecosystems than others. We elaborate on the properties of ecosystems that contributed to such differences in the Discussion and describe the motivations below.

Being influenced by a sharing culture, particularly open-source culture: Many of our participants across the ecosystems share their customizations because they embrace a sharing culture: *"I guess it's more of a fundamental piece of myself where I really like open source software. I like the idea and motivation behind it, of always sharing things. So, I'm pretty public on trying to share as much as possible that I can with whether it's stuff that I do outside of IFTTT but also recipes"* [IFT2]. The sharing culture seems to be influenced by the use of FOSS. Sub4, Sub5, and Alf3 talked about their reliance on open source in their job as their motivation to share their customizations: "*Nearly everything that I rely on for my job is open source, […] so it would be just silly not to open source it*" [Sub4]. This sense of obligation to the community has also been identified as one of the motivations for contributing to FOSS [15].

Building reputation: Two Alfred participants referred to reputation building as part of their motivation. Alf4 and Alf1 specifically mentioned that their customizations contributed to their GitHub profile: "*To be honest, it's also good to have a GitHub profile every now and then, because then you get attention. That's also something that I don't want to play down; […] It's just a plus, it's nice to have. I can just put it on GitHub and other people might like it and is good for me as well*" [Alf4]. Sharing customizations on GitHub seems to be one way of managing one's activities to form good impressions, since online activity traces in GitHub are used for recruiting [22], as well as for forming impressions about ones' expertise [23]. This is similar to

self-marketing that promises future monetary awards, one of the motivations for contributing to FOSS [10,16].

Having an online backup of customizations: A side benefit of sharing a customization is that it gets backed up online: *"Some is just so that have it some place. If I stop using a particular workflow and later on I wanted it back again, if I have deleted it, I can just pull it back down from Packal [Alfred's unofficial customization repository]"* [Alf1].

Zero or minimal effort needed for sharing: IFTTT has made it extremely easy to share by adding only a single click to the process of creating a recipe. Such ease of sharing in IFTTT affected IFT3's decision to share: "*IFTTT makes it relatively easy to publish those [recipes]. So, it felt relatively inconsequential for me to just hit the share button and let it go out*".

We also asked about motivation for creating customizations in the first place which we describe below.

Having a personal need: This is the dominant motivation across the participants in all the ecosystems (except for Minecraft): *"all of them [workflows] are meant to scratch an itch"* [Alf3], *"I usually find deficiencies in my workflows and try to find ways to improve them"* [SUB5], "*to make things a little easier for me*" [IFT4]. Personal need for a solution has also been identified as one of the most common triggers to customizing [20], and one of the most important drivers of contributions to FOSS [14].

Increasing enjoyment of the game: Echoing prior findings [28,31], this is the main motivation for our Minecraft participants: "*The main reason is I just really enjoy the game. With any game that you enjoy you just find ways that you could improve various things*" [MC2]. This motivation is so strong that even lack of programming knowledge has not been a deterrent: *"My only [programming] background is what I've done with Minecraft. When I would watch tutorials they would suggest learning Java first but I went against that and just learned as I went along"* [MC1].

Self-development: Similar to some FOSS contributors [15], learning or practicing one's programming skills is a motivation for creating their *first* customizations for a few participants: *"my motivation was just learning about programming. I basically learned programming building Alfred workflows"* [Alf5], *"I always wanted to do something with my knowledge of java, and when a friend told me "You should do something like that [creating mods]" while playing Minecraft, I started"* [MC3].

Responding to others' requests for a desired customization: Although uncommon, this is another motivation to create a customization: "*A guy on Twitter that I follow was mentioning that he wanted to do something and I made this [recipe name] recipe and sent that to him and he was pretty appreciative of that*" [IFT4].

Job's responsibility: Finally, IFTTT's community managers create customizations as part of their jobs, somewhat similar to the gardeners [8] and the paid contributors to FOSS [16].

Unshared customizations: what, when, and why

Many participants mentioned that it is silly not to share a customization once they create it. Despite that, all of the participants, except for the Minecraft participants, reported creating some customizations that they chose not to share. We describe the characteristics of customizations that our participants referred to as influencing their decision about sharing.

Customizations with private information are not shared. Being unsure as to how to anonymize a customization or hide private information is a reason for not sharing customizations: "*some of them [I didn't publish] probably because they had more private information and I wasn't really sure how to anonymize them. They referred to a specific directory somewhere on my computer, or they referred to a RSS feed that were private to me*" [IFT3].

Overly specific customizations are not shared. All the IFTTT and Alfred participants mentioned that if their customization is very specific to their needs, they will not share it: "*I have a few that I don't think that as many people would find it useful or might have something custom to the way I manage folders, Google drive, or something like that*" [IFT2]. It is possible to make a specific customization useful to others, but it can require extra effort as Alf6 put it: "*To make a workflow usable to everyone, there is certain level of quality that you have to reach. You need to write a bit of documentation, you have to add the configuration UI.*" Such effort can be more than some are willing to invest: "*if it would take more work to make them good enough whether it's pretty or simple so everyone can use, I won't share because I'm lazy*" [Alf5].

Too straightforward customizations are not shared. Our participants also tended not to share customizations that are "too straightforward" to create: "*they were too straightforward to share; basic stuff like if there is a new post in the RSS feed, email it to me. I feel like because most IFTTT users know that functionality exists, it's probably not necessary to share that*" [IFT3]. In the case of Alfred, "too straightforward" to create workflows are the ones that could be created by the Alfred GUI without coding.

In both of the above cases, participants commented on wanting their customizations to be useful for a broader audience and that popularity (or lack thereof) of customizations could boost or hurt their ego: *"The one [recipe] with Nest thermostat, I didn't publish because it requires you have Nest. So, it just didn't seem like it would be necessarily popular. So, I publish the ones that could be used by a wider audience…if over a period of time they [recipes] remain relatively unpopular, I'll probably delete*

them. There is a little bit of ego to it, you know having a popular recipe is interesting. If something is sitting there and not being popular, I might unpublish it" [IFT1]. Unsharing a customization because it is not popular is an implication of customization repositories exposing the usage data of shared customizations. We discuss the trade-offs in making such data transparent in the Discussion.

Unlike Sublime Text, IFTTT, and Alfred, where private customizations are common, private Minecraft mods makes no sense to our Minecraft participants: "*that [creating a mod and not sharing it] wouldn't really make sense because you kind of make the mod for other people to use. Also the knowledge needed to create a private mod makes it not worth it to just create it for yourself, unless it's a very small thing*" [MC4]. The large investment to create a customization in Minecraft does not justify not sharing it.

Trust in shared customizations

Previous studies have shown that being able to trust shared customizations [6] and their sharers is critical [25]. Buggy customizations could break software, and improper treatment of user data could raise security concerns. We gained some insights into re-users' trust since our participants also re-used others' customizations. Our participants generally expressed either no or little concern in reusing shared customizations, because of the following characteristics of the ecosystems: human moderation, exposed popularity, customization readability, and sharer reputation.

Human moderation: The customization manager in Sublime Text (Package Control) and the code repository in Minecraft (CurseForge) provide human moderation on the customizations. This made our participants confident in the security of the customizations: "*I know from uploading my own mods they check things out before allowing others to download files"* [MC1].

Exposed popularity: Some participants pointed to popularity of a customization as a cue to its security: "*when the mod is really famous and thousands of thousands of people playing it, then I wasn't too worried. It can't be dangerous when so many people are playing it*" [MC2].

Customization Readability: Several participants mentioned that the availability and readability of customization files increases their confidence in the security of the file, even though they do not necessarily investigate every customization they use: "*the fact that all plugins' repo is freely available, they basically are code that you can read directly makes it more trustworthy in my opinion. I try to be careful, I usually try to have a peak on the code*" [SUB1].

Sharer Reputation: In the Alfred and Minecraft ecosystems, where there is a sense of community among users and customization authors, our participants mentioned knowing good authors whose customizations they trust: "*Now that a few years have gone by I know which developers can be trusted*" [MC1].

DISCUSSION AND IMPLICATIONS FOR DESIGN

An important finding of our study is the notion of customization sharing ecosystems: different tools and people in various roles working together to support various aspects of sharing and re-using of customizations. Considering the multiplicity of tools that are used to support sharing and re-using in each ecosystem, we could have not reached our current understanding of sharing practices had we only studied an individual component (e.g., a forum) within the ecosystem—an approach taken by some of the other studies of customization sharing [4,26].

Grounded in our findings, we discuss some implications for the design of customization sharing ecosystems, and highlight important design tradeoffs.

Both the pipeline and the one-stop shop can be appropriate approaches for customization sharing ecosystems; choosing one depends on the complexity of customizations. The degree of integration between an ecosystem's components impacts ease of sharing and reuse. Compared to the collection-of-islands ecosystems, both the one-stop shop and pipeline ecosystems make it easier for sharers to publish customizations and distribute updates, as well as for re-users to find customizations and keep them up-to-date. But a one-stop shop may only be applicable for relatively simple, lightweight-to-create customizations as in IFTTT, given that for advanced customizations that require programming, each of the components of the ecosystem such as discussion places, customization repositories, and code repositories are complex systems in their own right.

Using generic well-used code repositories such as GitHub has advantages. Most customization sharers in our study (except for IFTTT sharers) share their customizations' source code on GitHub. Sublime Text sharers have to do so in order for their customization to be listed in the customization manager, which is integrated with GitHub. However, Alfred and Minecraft sharers are not required to use GitHub, but choose to because: 1) of the importance of having a good GitHub profile (which ties into reputation building as one of the motivations for sharing), and 2) GitHub facilitates tracking and organization of bug reports and feature requests for their customizations (which points to its maturity as a tool, offering useful functionality).

Motivations to create and share customizations overlap considerably with those in FOSS, but there are some differences. Having a personal need, self-development, building reputation, and a sense of obligation to share because of using FOSS are common motivations across online customization sharing ecosystems and FOSS. This degree of overlap in motivations was not entirely expected because the contexts are different – although many shared customizations are open source, they are often contributing

to *closed* source commercial software. Indeed, this difference in context helps explain some differences in motivations. For example, while no one motivation tends to dominate in FOSS [15], we found personal needs to be the dominant motivation across the customization sharers. In addition, the zero to minimal effort needed for sharing a customization in some ecosystems actually motivated some of our participants to share. This motivation does not seem to exist in FOSS, and it could be explained by the intense peer review process there, which is required, since the contributions affect a common code base [29]. Such a process can, in fact, add to the effort needed to contribute to a FOSS project as it raises the bar for the contributions – they need to reach a quality level and meet project-specific code styles and conventions. In contrast, the review process is much lighter weight in customization sharing ecosystems, if it happens at all. Ecosystems could be designed to make sharing a customization almost as effortless as not sharing it. Indeed, we saw with IFTTT that some participants were motivated to share a recipe because it was so easy to do so.

Discussion places are beneficial to both sharers and re-users. MacKay identified a lack of feedback to lead users who created and shared customizations in her early within-organization study [19]. We found that discussion places in online customization sharing are helping to address this problem. Some of the key benefits of dedicated discussion places are building trust between the sharers and re-users, clarifying problems with customizations, and providing feedback to the sharers, which sometimes led to customization improvements. While having a discussion place is critical, we saw a tradeoff in having one versus multiple such places. In both the Alfred and Minecraft ecosystems, some users report bugs in one place (forum) and others in another place (GitHub). On one hand, this makes it hard for sharers to keep track of reported bugs. On the other hand, it lowers the barrier to bug reporting. For example, reporting an issue in GitHub requires an account. To resolve this tension, ecosystems could do more to integrate their various discussion places. For example, users could flag their forum post as a bug report causing the post to be filed as a bug in another component of the ecosystem (e.g., the issue tracker of the source code repository).

Customization packs can add value. The idea of customization packs and the role of packers, appeared only in the Minecraft ecosystem, but could be valuable for other sharing ecosystems. Grouping relevant customizations would make it easier for users to discover and use relevant customizations without having to worry about potential conflicts between the customizations: a concern raised in prior studies [25]. Further study is needed to understand the motivation of packers and how such a role can be encouraged and supported in other customization sharing ecosystems.

Demoing customizations should increase their adoption, and thus keep the sharers encouraged to share. We learned that many sharers care about the popularity of their customizations, yet they rarely publicize them. The publicizer role in the Minecraft ecosystem is the one exception. All our Minecraft participants reported discovering new mods mostly through publicizers and mod managers that feature popular and new mods. Publicizers effectively create awareness of customizations and demonstrate how to use them in video. Other ecosystems could leverage this approach by providing, for example, a video channel. Sharers could be incentivized to demonstrate their customizations, perhaps through a "weekly winner" mechanism. Altogether, publicizing should help increase the adoption of the shared customizations, which will in turn keep sharers, the most crucial role in these ecosystems, encouraged.

Trust deserves more attention. Our participants expressed almost no concern about using others' customizations. This was unexpected given that prior work had shown that people tend to trust their colleagues more than strangers when re-using customizations [5]. In retrospect, however, heavy sharers may not be representative when it comes to trust concerns, and so this finding should be interpreted with caution. The factors that engender their trust, however, point to possible areas for improvement. For example, many of the Sublime Text and Minecraft participants were confident in the security of others' customizations because, through their own sharing, they were aware of the moderation process. The visibility of this moderation for re-users, especially novice re-users, is questionable. Readability of customizations is another factor that aids trust, however, this will again be limited to people, like our participants, who can easily read others' customization code. Source code repositories could leverage the effort of those re-users who choose to read a customization's source code by allowing them to indicate their trust of a customization. Although we found that re-users rely on popularity in part as a proxy for trustworthiness, this penalizes newly shared customizations that have not yet had time to gain popularity. Publishing trust data could help these new customizations find an audience more quickly.

Providing formal support for the reviewer role would bring both benefits and costs. Related to reading customizations for their trustworthiness, we found that some sharers voluntarily review others' customizations to provide feedback. Formalizing the reviewer role within an ecosystem could be useful, but there are tradeoffs. It could encourage newcomers to attempt creating a customization (knowing feedback will come) and it will increase re-users' trust. The flip side, however, is the time it takes to review. Either reviewing needs to become easier, or an incentive structure needs to be in place to motivate users to contribute in this role.

Unpublished customizations may be a lost opportunity: In some ways, understanding when people do not share is as interesting as learning when they do. Reasons for not sharing included uncertainty about how useful a customization would be for others, too much effort to ready it for others, or it being too straightforward. This could be a lost opportunity. For example, a customization might indeed be deemed too straightforward for another top sharer to bother with, but what about a newcomer? In essence, the straightforwardness of a customization may be in the eye of the re-user. If the ecosystems could support sharers to announce a *possible* customization, in order for the sharer to assess interest, this could help the sharer decide if it is worth the effort to publish it, or even solicit effort from others who want to re-use it to do the "cleaning" and publishing.

Perceived difficulty/ease of authoring a customization affects both sharing and re-using: As mentioned above, when authoring a customization is perceived easy, the authors choose not to share. We also saw with Minecraft that the difficulty of authoring does not justify keeping a customization private once it is authored. In addition, some IFTTT participants mentioned authoring a customization without bothering to search existing ones, because the cost of searching and re-using was equivalent to the cost of authoring for them. In the latter case, the ease of authoring affected the decision about whether to reuse or to author.

Customization authors in different ecosystems decide about sharing their customizations in different points in time: In the Minecraft ecosystem, the decision to share appears to precede the authoring, since our participants reported authoring a mod only with the intention to share it. On the other hand, the majority of Alfred and Sublime Text participants reported authoring a customization to address a personal need, and sharing it once it proves its value or they think it might be useful for other people too. In IFTTT, authors often decided about whether or not to share a recipe while authoring a recipe. We attribute that to how IFTTT supports sharing, i.e. selecting a checkbox to make the created recipe public as part of the process of authoring it.

LIMITATIONS

The sample size in our study from a per-system perspective was small (five participants per system). We intentionally chose breadth over depth to explore differences in customization sharing mechanisms. Future studies with larger samples could further examine some of the system-specific dynamics and issues that we have uncovered.

Our sample included only people who had extensive experience with sharing customizations. Although this focus allowed us to collect rich data about the sharing ecosystems, it did not fully capture the perspectives of those who do not share but only reuse customizations. All of our sharers were also re-users and so we do capture some of the re-user perspective here. Future studies with re-users would provide additional insights into their motivations for re-use, their motivations for reporting problems with customizations, how they trust the shared customizations, and how the difficulty of the language for authoring customizations affects reusing practices.

CONCLUSION

In our efforts to understand how customization sharers go about sharing and what motivates them, our study uncovered online customization sharing *ecosystems*. We documented various components of these ecosystems, described the design of the ecosystems based on how their various components are connected, and discussed the tradeoffs among designs. We also identified various roles that occur in the ecosystems, compared and contrasted them with similar roles in customization sharing in organizational settings as well as in FOSS projects, and discussed how to provide support for those roles.

Our study has only scratched the surface of the online customization sharing landscape, and raises some interesting directions for future work. In addition to the future work described above, we would also like to track particular shared customizations from multiple ecosystems over time, to see how they evolve from the point of publishing. Uncovering how success is defined, and the factors that make some shared customizations more successful than others, will further contribute to our understanding of how to design customization sharing ecosystems.

ACKNOWLEDGMENTS

We thank all of our participants for volunteering their time and contributing to this research. We also thank Syavash Nobarany, Gail Murphy, Francesco Tordini, Izabelle Janzen and all the reviewers for their helpful comments on previous versions of this document. We are grateful for NSERC's support.

REFERENCES

1. Richard Bentley and Paul Dourish. 1995. Medium versus mechanism: supporting collaboration through customisation. In *Proceedings of the fourth conference on European Conference on Computer-Supported Cooperative Work* (ECSCW'95), 133–148. Retrieved October 31, 2012 from http://dl.acm.org/citation.cfm?id=1241958.1241967
2. Will Bond. 2015. About - Package Control. Retrieved May 6, 2016 from https://packagecontrol.io/about
3. Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2: 77–101.
4. Giorgos Cheliotis and Jude Yew. 2009. An Analysis of the Social Structure of Remix Culture. In *Proceedings of the Fourth International Conference on*

Communities and Technologies (C&T '09), 165–174. https://doi.org/10.1145/1556460.1556485

5. Sebastian Draxler and Gunnar Stevens. 2011. Supporting the Collaborative Appropriation of an Open Software Ecosystem. *Computer Supported Cooperative Work (CSCW)* 20, 4: 403–448. https://doi.org/10.1007/s10606-011-9148-9
6. Sebastian Draxler, Gunnar Stevens, Martin Stein, Alexander Boden, and David Randall. 2012. Supporting the Social Context of Technology Appropriation: On a Synthesis of Sharing Tools and Tool Knowledge. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12), 2835–2844. https://doi.org/10.1145/2207676.2208687
7. Jeff Dyck, David Pinelle, Barry Brown, and Carl Gutwin. 2003. Learning from Games: HCI Design Innovations in Entertainment Software. In *Graphics Interface*, 237–246. Retrieved March 25, 2014 from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.122.6347&rep=rep1&type=pdf
8. Michelle Gantt and Bonnie A. Nardi. 1992. Gardeners and Gurus: Patterns of Cooperation Among CAD Users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '92), 107–117. https://doi.org/10.1145/142750.142767
9. Mona Haraty and Joanna McGrenere. 2016. Designing for Advanced Personalization in Personal Task Management. In *Proceedings of the 2016 conference on Designing interactive systems*.
10. Alexander Hars and Shaosong Ou. 2001. Working for free? Motivations of participating in open source projects. In *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*, 9–pp. Retrieved April 10, 2016 from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=927045
11. E. Haruvy, Fang Wu, and Sujoy Chakravarty. 2003. Incentives for Developers. *Contributions and Product Performance Metrics in Open Source Development'. Working Paper. Available from: http://www. iimahd. ernet. in/publications/data/2005-03-04sujoy. pdf [accessed 13 August 2010]*.
12. Helge Kahler. 2001. More Than WORDs - Collaborative Tailoring of a Word Processor. *Journal of Universal Computer Science* 7, 8: 826–847.
13. Benjamin Lafreniere, Andrea Bunt, Matthew Lount, Filip Krynicki, and Michael A. Terry. 2011. AdaptableGIMP: designing a socially-adaptable interface. In *Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology* (UIST '11 Adjunct), 89–90. https://doi.org/10.1145/2046396.2046437
14. Karim R. Lakhani and Eric Von Hippel. 2003. How open source software works: "free" user-to-user assistance. *Research policy* 32, 6: 923–943.
15. Karim R. Lakhani and Robert G. Wolf. 2003. *Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects*. Social Science Research Network, Rochester, NY. Retrieved May 7, 2016 from http://papers.ssrn.com/abstract=443040
16. Josh Lerner and Jean Triole. 2000. *The simple economics of open source*. National Bureau of Economic Research. Retrieved January 14, 2016 from http://www.nber.org/papers/w7600
17. Gilly Leshed, Eben M. Haber, Tara Matthews, and Tessa Lau. 2008. CoScripter: automating & sharing how-to knowledge in the enterprise. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems* (CHI '08), 1719–1728. https://doi.org/10.1145/1357054.1357323
18. Wendy E Mackay. 1990. Patterns of sharing customizable software. In *Proceedings of the 1990 ACM conference on Computer-supported cooperative work* (CSCW '90), 209–221. https://doi.org/http://doi.acm.org.proxy.lib.sfu.ca/10.1145/99332.99356
19. Wendy E Mackay. 1990. Users And Customizable Software: A Co-Adaptive Phenomenon. Retrieved March 23, 2011 from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.7497
20. Wendy E Mackay. 1991. Triggers and barriers to customizing software. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology* (CHI '91), 153–160. https://doi.org/http://doi.acm.org.proxy.lib.sfu.ca/10.1145/108844.108867
21. Allan MacLean, Kathleen Carter, Lennart Lövstrand, and Thomas Moran. 1990. User-tailorable systems: pressing the issues with buttons. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '90), 175–182. https://doi.org/10.1145/97243.97271
22. Jennifer Marlow and Laura Dabbish. 2013. Activity traces and signals in software developer recruitment and hiring. In *Proceedings of the 2013 conference on Computer supported cooperative work*, 145–156. Retrieved December 17, 2015 from http://dl.acm.org/citation.cfm?id=2441794
23. Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. 2013. Impression formation in online peer production: activity traces and personal profiles in github. In *Proceedings of the 2013 conference on Computer supported cooperative work*, 117–128. Retrieved

December 16, 2015 from http://dl.acm.org/citation.cfm?id=2441792

24. Audris Mockus, Roy T. Fielding, and James D. Herbsleb. 2002. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 11, 3: 309–346.
25. Emerson Murphy-Hill and Gail C. Murphy. 2011. Peer interaction effectively, yet infrequently, enables programmers to discover new tools. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work* (CSCW '11), 405–414. https://doi.org/10.1145/1958824.1958888
26. Lora Oehlberg, Wesley Willett, and Wendy E. Mackay. 2015. Patterns of physical design remixing in online maker communities. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 639–648. Retrieved July 26, 2016 from http://dl.acm.org/citation.cfm?id=2702175
27. R. Oppermann and H. Simm. 1994. Adaptability: User-initiated individualization. *Adaptive User Support–Ergonomic Design of Manually and Automatically Adaptable Software. Hillsdale, New Jersey*.
28. Hector Postigo. 2007. Of mods and modders chasing down the value of fan-based digital game modifications. *Games and Culture* 2, 4: 300–313.
29. Eric S. Raymond. 1998. The cathedral and the bazaar. *First Monday* 3, 2. Retrieved April 10, 2016 from http://ojphi.org/ojs/index.php/fm/article/view/578
30. Justin Scott. 2011. How many Firefox users have add-ons installed? 85%! *Mozilla Add-ons Blog*. Retrieved May 19, 2016 from https://blog.mozilla.org/addons/2011/06/21/firefox-4-add-on-users/
31. Olli Sotamaa. 2010. When the game is not enough: Motivations and practices among computer game modding culture. *Games and Culture*. Retrieved May 12, 2016 from http://gac.sagepub.com/content/early/2010/03/16/1555412009359765.abstract
32. Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L. Littman. 2014. Practical Trigger-action Programming in the Smart Home. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14), 803–812. https://doi.org/10.1145/2556288.2557420
33. Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L. Littman. 2016. Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (CHI '16), 3227–3231. https://doi.org/10.1145/2858036.2858556