Sketching and Drawing in the Design of Open Source Software

Eunyoung Chung[1], Carlos Jensen[1], Koji Yatani[2], Victor Kuechler[1], and Khai N. Truong[2]

[1] School of Electrical Engineering & Computer Science
Oregon State University
Corvalis, OR 97331-5501, USA
{chung, cjensen, kuechlej}@eecs.oregonstate.edu

[2]Department of Computer Science
University of Toronto
Toronto, ON M5S 3G4, Canada
{koji, khai}@dgp.toronto.edu

Abstract

In co-located software development, diagramming practices, such as sketching ideas out with a pen and paper, support the creative process and allow designers to shape, analyze, and communicate their ideas. This study focuses on the diagramming practices used in the design of Open Source Software (OSS), where the norm is highly distributed group work. In OSS, text-based communication (e.g., mailing lists) dominates, and sketching and drawing diagrams collaboratively remains difficult due to the barriers imposed by distance and technology. Previous studies have examined these practices and barriers in the context of individual projects. To understand how contributors across OSS projects use diagrams in design-related activities, we conducted a survey of 230 contributors from 40 different OSS projects, and interviewed eight participants. Our results show that although contributors understand the advantages of using diagrams for design-related activities, diagrams are infrequently used in OSS. This motivated us to examine how and why diagramming occurs, and the factors that prevent widespread use in OSS. Finally, we propose new ideas for supporting design activities in OSS projects.

1. Introduction

Research has shown that diagrams play an important role in software development as a way to represent knowledge and information, simplify complex information, and promote communication among people. Designers and developers extensively use diagrams for these purposes as well as to prototype and share ideas [10]. Specifically focusing on sketches, Tversky *et al*. noted that the designers' mind is expressed through such external representations, which also become a source of creativity [19].

Cherubini *et al.* showed that diagram use is very important in co-located development [3]. They showed that diagramming practices are primarily rooted in the use of tangible media, such as paper or whiteboard. Despite the well-documented advantages of sketching and drawing for design, these practices are not always adopted in a highly distributed environment.

Open Source Software (OSS) development is a notable example of a distributed development model. Previous research has shown that deliberations regarding crucial changes and problems in OSS design are often performed through text-based channels — mailing lists or forums [1, 12], rather than through shared diagrams as is more common in co-located development. This raises a number of interesting questions about diagramming practices in OSS projects: When are diagrams used? How is their use different in OSS compared to co-located settings? What is the reason for these differences?

Motivated by the lack of data about diagramming practices in distributed development, Yatani *et al.* studied how and why Ubuntu contributors used diagrams in their development process [20]. They showed that contributors express ideas using freehand sketching to explore different designs[1] as well as to get feedback from community members, albeit not as frequently as their co-located counterparts. Many of the barriers that their participants encountered were related to the technical limitations of current computer-based diagramming tools, and how they conflicted with the OSS workflow [20]. As Yatani *et al.'s* study was only exploratory in nature and focused on a single OSS project, we decided to follow up with a much broader study that would explore OSS diagramming practices and barriers in greater depth, especially as they relate to design, across a broad set of OSS projects.

In this study, we used a mixed methods approach to gain a better understanding of diagramming practices in OSS, particularly practices around design-related activities. We deployed an online survey and recruited 230 respondents from 40 different OSS projects. Our

[1] In this paper, as in *Yatani et al*'s paper., "design" refers to designing interfaces entangled with interaction between subsystems including system behaviors and appearances [18, 20].

key finding from the survey is that although participants from a variety of OSS projects appreciated diagrams, they did not use them effectively in their design practice. We then conducted semi-structured interviews with eight survey respondents to deepen our understanding of their motivations and practices related to diagrams. These interviews revealed how and why diagrams are created and used, as well as the challenges that hinder diagram use in OSS development.

2. Related Work

Designers typically have a strong need to communicate with stakeholders in order to try out solutions and minimize mistakes throughout the design process [17]. Diagrams are a crucial tool to support critical thinking, problem solving, decision-making, and communication [2, 6, 11]. Larkin claims that diagrams externalize the relationships between knowledge and information, and therefore help to organize pieces of information, build perceptual inferences, and construct ideas [11].

Many studies have shown how diagrams promote cognitive processes and social interaction between developers and designers. Blackwell lists some desirable properties of diagrams: encouraging novice programmers, promoting learning processes, and supporting communication [2]. Hahn and Kim state that diagrams have been extensively used in evaluation and analysis of system design, and for developing system behaviors [9].

Cherubini *et al*. described nine scenarios in which diagrams (visual representations of projects or design considerations) were actively used by co-located software development teams [3]. The nine scenarios are: understanding existing code, ad-hoc discussion, designing/refactoring, design review, on-boarding (helping new members acquire project knowledge), explaining to secondary stakeholders, explaining to customers, hallway art, and documentation. Their study showed that diagrams were not just used in official "design sessions". Developers relied on frequent "drop-in" meetings—informal ad-hoc sessions with other project members where ideas are shaped through informal sketches on a whiteboard—as a part of their creative process. Thus, diagrams were created and shared for many different purposes. Though developers and designers sometimes invested effort in refining diagrams, in most cases, diagrams were transient and therefore, not archived or modified for the future use.

Diagrams can also be highly sophisticated. Myers *et al.* showed that designers occasionally used diagramming tools to create and demonstrate system behaviors [13]. Such interactive prototypes of a system facilitate communication among designers, and enable them to explore navigation and simplify system behaviors.

Many have studied the practices and challenges with coordination, collaboration, and communication in highly distributed software development [15]. Although OSS contributors are often highly distributed and volunteer-driven, they successfully use Computer-mediated communication (CMC), mainly text-based communication, to coordinate their work [8, 14]. Barcellini *et al*. show that Python contributors communicated with each other through three channels: a discussion space (*e.g.,* mailing lists, forums, and chats), a documentation space (*e.g.*, blogs, wikis, and project websites), and an implementation space (*e.g.*, source code repositories) [1]. These channels, common to most OSS projects, support the contributors' collaboration on problem solving, clarification, decision-making, and design evaluation.

Yatani *et al*. examined how Ubuntu contributors used diagrams in their project [20]. They found that Ubuntu contributors occasionally used diagrams for five of the nine scenarios defined by Cherubini *et al* [3]. But, they did not find clear evidence of diagrams used for design-related activities. The results gained from this study show that OSS contributors understand the benefits of diagrams in design-related activities, and used diagrams, but not as actively as in co-located software development. Our results also indicate that diagrams used for design-related activities were published for different audiences for different purposes. Our findings provide further understanding about diagram use in OSS development.

3. Study Method

We designed our study to answer the following research questions:

1. How frequently do contributors use diagrams for design-related activities in OSS projects?
2. How do contributors create these diagrams?
3. How and why do contributors use diagrams?

In this section, we describe the two parts of our study: an online survey, and semi-structured interviews.

3.1 Survey

We conducted an online survey, running from January 2009 to June 2009, to collect quantitative data about diagramming practices and the attitudes of contributors in a variety of OSS projects. The survey was divided into three sections: Demographic questions, diagramming practices, and wrap-up.

In demographics, we defined 10 common roles in OSS projects (project management, coding package

maintenance, patch creation, testing, translation, community building, design, marketing, and user support), and participants could choose any number of these (at least 1). We asked about project membership, length of participation, formal CS training, and whether participants also work in co-located development.

We adopted the diagram scenarios from Cherubini's study to examine the different purposes for diagramming [3] (see above). Table 1 shows the questions from the diagramming practices portion of the survey. The survey took approximately 20 minutes to complete.

Table 1: Questions about diagramming practices.

Q1. **Please indicate how often you currently engage in each of the nine activities.** (Daily / Weekly / Monthly / Yearly / Never)
Q2. **Please indicate the frequency of your diagram use for the nine activities.** (All the time / Very often/ Sometimes/ Rarely / Never)
Q3. **Please indicate which medium you use to create diagrams for the nine activities.** (Paper sketch / Software tool / Blueprint / ASCII art / Diagram not created)
Q4. **Please indicate how much you agree to use diagrams for the nine activities.** (Strongly agree / agree / neutral / disagree / strongly disagree)

We recruited participants by posting to the mailing lists of 40 OSS projects of different sizes. Examples of our selection of projects are KDE, Gnome, Ubuntu, Fedora, Firefox, Open Office, Gentoo, Apache, OLPC, Debian, Mono, and Myth TV. We did not rigorously screen and select the projects with any specific criteria because we wanted to broaden the scope of the study and gain diagramming practices in different projects.

A total of 230 people completed the survey, five of which were randomly selected to receive a $30 gift certificate as compensation.

3.2. Follow-up interviews

We performed eight follow-up interviews with a representative selection of the survey participants to gain a better understanding of specific diagramming practices. Four interviewees explicitly stated they were involved in design activities in our online survey at the time we interviewed them. All but one participant had some experience with design. During the interviews, we asked participants questions about their experience with diagramming in OSS projects, as well as clarifying questions about their survey responses. All interviews were conducted over the phone, and lasted 50 minutes on average. All conversations were recorded and transcribed. We compensated all participants with a $30 gift certificate.

These interviews and the open-ended questions in our survey provided us with qualitative data to draw a more in-depth understanding of diagram use in design-related activities. We extracted 77 quotes related to design from the recorded interviews. Three of the authors conducted an iterative open coding [16] of the extracted quotes, and constructed the code set in Table 2. We identified recurring themes and events associated with diagramming and design-related activities, the roles of diagrams in the OSS workflow, and requirements for future tools.

Table 2: Coding schemes on designing activities.

Theme	Sub-theme	Agreement	Cohen's Kappa
Creation: Methods	Pen and paper	0.99	0.8
	Software tools	1	1
	Collaborative tools	0.99	0.9
Creation: Purposes	Getting feedback	0.97	0.88
	Own understanding	0.99	0.93
	Aid for others	0.94	0.78
	Documentation	0.99	0.82
	On-boarding	0.99	0.85
Update	Updating	0.99	0.85

4. Results

4.1 Demographics

Our participants claimed to assume 4 different roles on average within their OSS projects. Code development (66%), bug reporting, and testing (both 61%) were the most common roles. Only 19% of participants identified themselves as designers. However, our question about the frequency of the activities revealed that 40% of the participants (see Figure 1) were involved in design/refactoring and design review on a daily or weekly basis. Unlike in co-located development teams, roles in OSS projects are more loosely defined and fluid [14]. Contributors are more likely to engage in a number of activities, and move across roles as the project and their interests evolve.

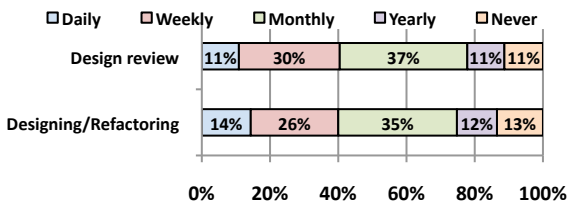

Figure 1: Frequency of design-related activities.

Table 3: Diagram use for design-related activities: Developers with CS vs. non-CS background.

	CS background		No CS background	
	Designing/ Refactoring	Design review	Designing/ Refactoring	Design review
All the time	8%	9%	6%	2%
Very often	21%	19%	9%	11%
Sometimes	27%	27%	23%	26%
Rarely	14%	13%	26%	19%
Never	30%	32%	36%	43%

Table 4: Diagram use for design-related activities: Co-located vs. non-co-located developers.

	Co-located		Non-co-located	
	Designing/ Refactoring	Design review	Designing/ Refactoring	Design review
All the time	13%	14%	4%	2%
Very often	21%	20%	17%	16%
Sometimes	28%	29%	25%	25%
Rarely	15%	12%	18%	16%
Never	24%	26%	36%	41%

The majority of participants (80%) had a Computer Science (CS) background, in this case defined as having taken some formal CS classes. We compared the frequency of diagram use in two design-related activities (design review and design/refactoring) between those with and without a CS background (Table 3). For a statistical test, we used a Mann-Whitney's U test, appropriate for comparing two independent sample groups of ordinal data. The Mann-Whitney's U tests found significant effects of a CS background on frequency of diagram use ($Z=-1.94$, $p=.05$, the effect size $r=.13$ in designing/refactoring; and $Z=-2.18$, $p<.05$, $r=.14$ in design review). This may imply that contributors with a CS background were likely more aware of the advantages of diagramming, and the tools and techniques for doing so than their colleagues, and therefore more willing to do so.

Forty-four percent of our participants also work in co-located environments. We do not have data on whether this was co-located proprietary software or OSS development. However, we expected these contributors to carry over practices in co-located development into their OSS work. Table 4 shows the comparison of the frequency of diagram use between contributors who were involved in co-located development and those who were not. A Mann-Whitney's U test revealed that co-located contributors used diagrams significantly more often than non-co-located contributors ($Z=2.78$, $p<.05$, $r=.18$ in designing/refactoring; $Z=3.41$, $p<.05$, $r=.22$ in design review).

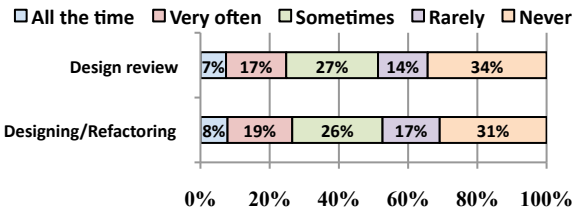

Figure 2: Frequency of diagramming practices in design-related activities.

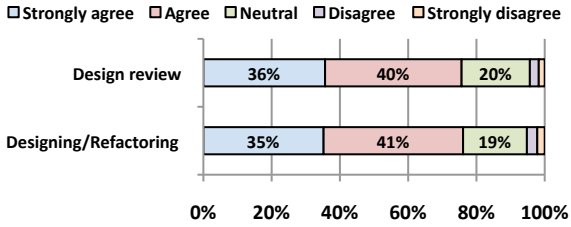

Figure 3: Attitudes toward diagramming practices in design-related activities.

4.2. Attitudes to and frequency of diagram use

We examined how often participants used diagrams for design-related activities. As shown in Figure 2, 24% of the participants answered that they diagrammed "all the time" or "very often" for design review, and 27% of participants answered that they diagrammed "all the time" or "very often" for designing/refactoring. These contrast with the perceived value of diagramming among participants (see Figure 3). While a minority claims to diagram regularly, 76% agreed that using diagrams for design related activities has value.

In the following sections, we discuss the practices and challenges around using diagrams in the design of OSS by examining the results from our interviews with eight participants.

4.3 Methods for creating diagrams

In our survey, software tools as well as pen and paper were the most frequently-used means for creating diagrams (33% with pen and paper, and 27% with software tools in design/refactoring scenario; and 25% used pen and paper, and 29% using software tools for design review; see Figure 4). Approximately 30% of our participants claimed not to have created diagrams for these scenarios.

Three interview participants reported that they used a whiteboard or paper during face-to-face meeting. For large OSS projects, such as Ubuntu, contributors often hold summits where contributors and users physically gather in one place to share ideas and brainstorm for the next software release. Our participants also created

freehand sketches with pen and paper for their own use. They drew icons, pictures of the system, and interactions between subsystems.

"[W]hen there is something complicated I tend to draw diagrams […] on paper, write a few notes, and sketch."

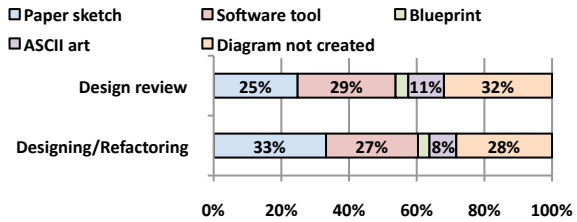

Figure 4: Methods to create diagrams.

In terms of tool use, participants used a variety of tools, including Gimp, Inkscape, Cmaptool, Dia, and Photoshop. However, their tool use does not seem as varied compared to co-located development where Myers *et al*. found that 16 different tools were used for different design process [13].

We also found different levels of formality in their diagrams. For instance, they often used rough sketches to create mockups in order to present their ideas to other developers. Formal diagrams like UML were created for documentation.

"*Very formal diagrams like UML that would be pretty rare only in this case. When refactoring it is common, or when designing something new."*

"If we're programming, it is very helpful for other people to follow design and UML guidelines."

Our interview participants sometimes used a collaborative tool when they needed synchronous communication with each other. Traditionally, visual communication among co-located designers and engineers has been based around tangible media, such as a whiteboard [3, 10]. However, such tangible media have no analogue in distributed situations. Two interview participants indicated that they used electronic sketching tools for remote design meetings. An example of such a tool is Dimdim [5], which provides a shared online space to chat, edit, and sketch.

"We actually tried to use 'Dimdim' to do like a virtual class room or like virtual whiteboard, so you can simply use a mouse and draw... Mainly for designing."

A collaborative diagramming tool allows developers to synchronously communicate with each other. However, our participants also felt that current collaborative tools did not always meet their needs. One participant explained that a developer who joined the meeting later could not see the artifacts created by others and thus failed to really join the discussion.

"We actually stopped using it (Coccinella [4]) because it is very confusing. What happened was you know, we had ten people in chat room and we were looking at the drawing, but then a person number 11 joined late. They cannot see anything that was drawn before."

In the survey, we also observed that diagrams were created with the Blueprint feature in the Launchpad hosting environment, home to Ubuntu. Our survey results showed that 70% of the participants who used Blueprint were also involved in Ubuntu.

The Figure 5 shows examples of diagrams created by our participants. Figure 5a is a freehand sketch with a pen and paper for icon designs. The participant who made this figure scanned the image, and posted it on her blog to get feedback from other contributors. Figure 5b is an interactive mockup created by a designer for Firefox. This mockup also contains comments about how to interact with the interface.

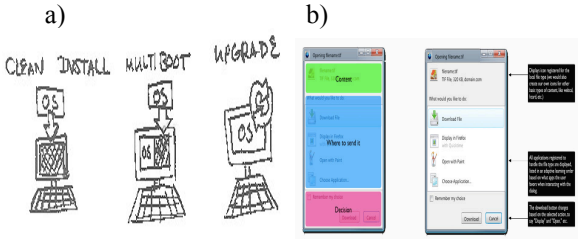

Figure 5: Sketches and drawings created by our participants.

4.4. Purposes for publishing diagrams

As shown in Table 2, we identified five purposes for creating diagrams from our interview data. We also found that diagram creation for these purposes is tightly coupled with publication. In this section, we look into each of the purposes more deeply.

4.4.1. Eliciting feedback

Eliciting feedback was documented in interviews. We also found that diagrams were circulated through mailing lists, Internet Relay Chat (IRC), blogs, and wikis for this purpose. Barcellini *et al*. found that discussion channels, such as mailing lists and chat rooms, were capable of fostering design discussions and critical evaluations in the Python community [1]. Our empirical data supports their finding by showing that diagrams were created to capture the attention of others and promote discussions by explaining and expressing them in an easily digestible form.

4.4.2. Enhancing one's own understanding

We observed that participants created diagrams to deepen their understanding of the systems they were developing, or to test ideas. This was also found by Cherubini *et al*. and Yatani *et al.* [3, 20], and we did not observe noticeable differences from their findings. Thus, we decided not to further analyze this scenario.

"We get user's immediate idea of what they want, and we catch a lot of problems before the real design of the application."

4.4.3. Aid for others' understanding

Diagrams were also created to aid others' understanding. In this case, diagrams were intended to help others understand the scope of the design and what contributors really needed to consider.

"It gives us a way of ensuring the design that we make is translated all the way down to code…"

One participant pointed out an advantage of diagrams; they can help bridge the language barrier for developers whose first language is not English. This observation is in line with findings by Myers *et al.,* and Olson and Olson [13, 15].

"A lot of this stuff is really technical and just talking through it is really difficult to do so that the other person understands. Especially if that person does not speak [good] English."

Face-to-face meeting can prevent or clear up misunderstandings. Myers *et al*. showed that non-verbal cues facilitated the process of resolving problems and misunderstandings in an offline meeting [13]. However, frequent offline meetings are not practical in most OSS projects, and not all contributors may be fluent in English. Although we could find only a small set of examples in our interviews, this anecdotal evidence seems to suggest that diagrams may help mitigate language and cultural barriers.

In co-located development, developers and designers typically have to present their ideas to different audiences, such as secondary stakeholders and users [3]. During these stages, diagrams are used to explain ideas to others. Our results also show that OSS contributors occasionally engaged in the same types of behavior to gain feedback on their ideas:

"If there is discussion about something on the mailing list and I think I have a good solution, I may do a quick diagram, or I may do quick mockup, and send it just to the list. And then, if people decide 'Hey, it might be a good idea, I wonder how others like it, and then I post to the blog. I post to the blog if I want it to be really public."

"I create a basic design, and then blog about it on Ubuntu planet. That gets a lot of viewers and users of Ubuntu. I got a lot of useful feedback from people saying whether they thought that part is useful or what feature should be added."

4.4.4. Documentation and on-boarding

In our interviews, diagrams were also created for the purposes of documentation and on-boarding. Participants referred to documentation presented on a wiki or diagrams saved in a Content Management System (CVS). One participant shared with us that his team stored design mockups on their wiki page. As a result, that page documented how the design has evolved.

"You can see current design and old… It is a very good tool for documenting changes you make with reasons why."

Although such a website may be useful for someone who is interested in joining the project, our participants also explained that they created diagrams explicitly for people currently in the process of joining the team.

"Occasionally, it [diagrams] is informative. If I am working on a project, and I design something, and there is a new contributor, then a diagram helps explain structures, so they can get an understand of how it works."

4.4.5. Relationship between creating and publishing diagrams

Four of the documented purposes of creating diagrams (eliciting feedback, aiding others' understanding, documentation, and on-boarding) were usually intended for publication. As we can see in the above sections, diagrams were also created for different audiences. Table 5 illustrates the relationship between the purposes for the diagram and the target audience as observed from our participants.

We believe that documentation and on-boarding are different from the other two purposes we identified. Diagrams for documentation and on-boarding are intended to be more archival. On the other hand, diagrams for getting feedback and supporting understanding generally aim at addressing problems of current interest, which can be described as "transient diagrams" [3].

Our results also indicate that OSS contributors often use communication channels other than mailing lists to circulate their diagrams. Mockus *et al*. showed that

analyzing, reporting, and discussing problems, and new features were often handled in mailing lists [12], but that posting diagrams was generally frowned upon. Our participants preferred to use blogs and wikis to contact users. By posting mockups on a blog, they felt that they were able to get feedback from users as well as developers in a casual manner.

Table 5: Motivating factors for publishing diagrams.

		Audience		
		Self	Project	Others
Purposes for publishing diagrams	Understanding	o	o	o
	Documentation	x	o	o
	Elicit feedback	–	o	o
	On-boarding	–	–	o
o: observed, x: not observed, –: not applicable				

4.5. Updating diagrams

We found that our participants did not update diagrams often. This is in line with what Yatani *et al.* found [20]. One reason for this may be that each diagram must be maintained in the project's repository for archival reasons and contributors would rather make a new diagram than update an old diagram.

Our participants explicitly pointed out an issue with updating diagrams with repect to tools.

"I think [not updating diagrams] is mainly a tool issue because sometimes if you have initial design, it works. You encounter bugs and you go to quickly fix them. And, there isn't something that pulls in your development, explaining to you that 'you changed an important code part.'."

However, we also observed several notable cases of updating diagrams.

"These comments were on version number 3. I made changes based on this feedback to create version number 4. So, I have 4 versions probably I will end up with at least 6 versions. I am working on version 5 right now."

Gasser *et al.* described the process of continuous design, a common practice in OSS [7]. This is a process in which developers release prototype systems and iteratively revise them based on the feedback they get. These rapid release and feedback cycles make it difficult to keep documentation up-to-date. We believe that in order to encourage greater use of diagrams, we need a way of linking these to the source code in change-tracking systems.

5. Discussion

Our results show that OSS contributors value their diagrams for design-related activities, and use a variety of tools and ways to share these. Our participants showed very positive attitudes toward diagrams, which was somewhat surprising when compared to those reported in Yatani *et al.*, where some participants showed negative attitudes toward diagram use, specifically for design review [20]. We also found that contributors with formal training in CS and who were involved in co-located development were more likely to use diagrams in their OSS work. This also implies that future studies need to consider more carefully how the participants' background influences on their practices and communication in OSS activities.

5.1. Design implication for a future tool

OSS contributors do not seem to have a great deal of choices when it comes to diagramming tools for design. Several interview participants stated that OSS contributors tend to stick with OSS tools. In addition to the ideological and potential licensing issues involved, OSS projects heavily rely on volunteer effort. Thus, the tools that projects use should not be a huge burden on their contributors, especially in terms of monetary cost.

We found that because OSS contributors are not always geographically co-located, it is crucial to synchronize communication and build shared understanding of their project's progress. Therefore, tracking how the design evolves over time could help other contributors stay up-to-date on the status of the project.

Another important finding is that contributors will use different archives to publish their diagrams based on who the intended audience is. Based on this result, support for organizing diagrams for different audiences could help OSS contributors manage and highlight design-related information more effectively.

Our study also suggests that the integration of diagramming tools into the development infrastructure, such as CVS, could facilitate the design and review process. In such integration, revision control for diagrams would also be important so that contributors could see the evolution of their designs. This would also be a useful resource for those joining the team and needing to learn the history of the project. A diagramming tool for OSS projects should be designed to accommodate these different purposes

"It would be very helpful to have sort of integration in development environment and design environment. So, whenever you change something, you can look back whether diagram is correct."

5.2. Threats to validity

Each OSS project has its own characteristics and culture, and diagramming practices might vary across different OSS communities. We did not tightly control from which OSS projects the participants came because our intention was to span across many projects' population and gain insight about the various practices. About 85% of our survey participants were involved in multiple OSS projects (the average number of projects participants were involved in was 3). Although we were not able to focus on the diagramming practices of any particular project in detail, our results still provide a broad and general understanding of diagram use across the OSS community. Despite our relatively small sample for the interviews, our data converged after we finished the eighth interview. We believe that the results gained through the interviews cover many of the diagramming practices shared by other OSS contributors, which in turn, provides a deep understanding of the motivations and practices of contributors diagramming in OSS.

6. Conclusion

Although the importance of diagramming in software development is recognized, few studies have investigated the practices and problems of diagramming in distributed environments. We studied how and why contributors in various OSS projects use diagrams for design-related activities. Our results revealed that our participants have strongly positive attitudes toward diagramming, yet diagram use is not adopted as fully as in co-located development. We also found that OSS contributors used an analog medium for diagramming despite the problems associated with sharing such diagrams over the Internet. Our study fills out some of the gaps pointed out by Yatani *et al.*'s study [20], and contributes a further understanding of how OSS contributors, particularly those who engage in design-related activities, use diagrams.

7. Acknowledgements

We would like to thank our study participants for their time and effort. We also thank the Linux User Group, Jose Cedeno, and Justin Gallardo for participating in our pilot study of the survey and interviews. We also thank reviewers for their valuable comments on our paper.

8. References

[1] Barcellini, F., Détienne, F., and Burkhardt, J.M. "Cross-participants: Fostering design-use mediation in an Open Source Software Community," In *Proc. of ECCE,* 2007, pp. 57-64.

[2] Blackwell, A.F., and Engelhardt, Y. "A meta-taxonomy for diagram research," *Diagrammatic Representation and Reasoning,* London: Springer, 2002, pp. 47-64.

[3] Cherubini. M., Venolia G., DeLine, R., and Ko A.J. "Let's go to the whiteboard: How and Why software developers use drawings," *In Proc. of CHI*, 2007, pp. 557-566.

[4] Coccinella, http://thecoccinella.org/

[5] Dimdim. http://www.dimdim.com/

[6] Funt, B.B. "Problem-solving with diagrammatic representations," in *Diagrammatic reasoning*, The MIT press, 1995, pp. 33-68.

[7] Gasser, L., Scacchi, W., Ripoche, G., and Penne, B. "Understanding Continuous Design in F/OSS Projects," *In proc. of ICSSEA*, Paris, France, 2003.

[8] Gutwin, C., Penner, R., and Schneider, K. "Group awareness in distributed software development," In *Proc. of CSCW*, 2004, pp. 72-81.

[9] Hahn, J., and Kim, J. "Why are some diagrams easier to work with? Effects of diagrammatic representation on the cognitive integration process of systems analysis and design," *ACM Trans. Of CHI,* vol.6, pp.181-213, Sept. 1999.

[10] Henderson, K. *On Line and On Paper: Visual Representations, Visual Culture, and Computer Graphics in Design Engineering*, Cambridge, MIT Press, 1999.

[11] Larkin, J.H., and Simon, H. "Why a diagram is (sometimes) worth ten thousand words," *Cognitive Science*, pp. 65-99, 1987.

[12] Mockus, A., Roy, F., and Herbsleb, J. "Two case studies of open source software development: Apache and Mozilla," *ACM Transactions on Software Engineering and Methodology,* vol.11,, pp. 1–38, 2002.

[13] Myers, B., Park, S., Nakano, Y., Mueller, G., and Ko, A., "How designers design and program interactive behaviors," In *Proc. of VL/HCC,* 2008, pp. 177-184.

[14] Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida K., and Ye, Y. "Evolution patterns of Open-source Software Systems and communities," In *Proc. of IWPSE, 2002*, pp. 76-85.

[15] Olson, G.M and Olson, J.S. "Distance matters," *Human-Computer Interaction*, vol.15, pp.139-178, Sept. 2000.

[16] Sharp, H., Preece, J., and Rogers, Y. Interaction Design : beyond Human Computer Interaction, 2nd ed, Wiley, 2007.

[17] Stappers, P.J. "Creative connections: User, designer, context, and tools," *Personal and Ubiquitous Computing,* 2006, pp. 95-100,

[18] Thimbleby, H., Blandford, A., Cairns, P., Curzon, P., and Jones, M. "User Interface Design as Systems Design," *In Proc of HCI,* 2002. pp.281–301.

[19] Tversky, B., Suwa, M., Agrawala, A., Heiser, J., Stolte, C., Hanrahan, P., Phan, D., Klingner, J., Daniel, M., Lee, P., and Haymaker, J. "Sketches for Design and Design of Sketches," *in Human Behavior in Design: Individuals, Teams, Tools*, 2003.

[20] Yatani, K., Chung, E., Jensen, C., and Truong, K.N. "Understanding how and why open source contributors use diagrams in the development of Ubuntu," In *Proc. of CHI*, 2009, pp. 995-1004.