CodeSaw: A Social Visualization of Distributed Software Development

Eric Gilbert and Karrie Karahalios

University of Illinois
Urbana-Champaign, Illinois, USA
{egilber2, kkarahal}@cs.uiuc.edu

Abstract. We present CodeSaw, a social visualization of distributed software development. CodeSaw visualizes a distributed software community from two important and independent perspectives: code repositories and project communication. By bringing together both shared artifacts (code) and the talk surrounding those artifacts (project mail), CodeSaw reveals group dynamics that lie buried in existing technologies. This paper describes the visualization and its design process. We apply CodeSaw to a popular open source project, showing how the visualization reveals group dynamics and individual roles. The paper ends with a discussion of the results of an online field study with prominent open source developers. The field study suggests that CodeSaw positively affects communities and provides incentives to distributed developers. Furthermore, an important design lesson from the field study leads us to introduce a novel interaction technique for social visualization called spatial messaging.

1 Introduction

A recent study suggests that nearly 1.1 million software developers in North America participate in open source development [1]. Major companies and governments have adopted open source software for critical infrastructure. Projects like Linux, Apache and Mozilla have propelled the open source "movement" into the popular consciousness through their media attention.

Open source software development operates quite differently than traditional software development. Developers do not meet face to face. There are few schedules. In all but the most prominent projects, there is no plan. Developers choose what they work on and how much time they spend working on it. In almost all cases, developers do not get paid for their work. Traditional mechanisms for coordinating work (e.g., schedules, plans, face to face meetings) are absent [2], yet studies show that distributed development needs coordination more than collocated development [3]. We are just starting to learn what drives these communities.

At the same time, many open source communities also represent vibrant online social spaces [4]. Developers have heated emailed exchanges on the project mailing list about feature additions. Legal issues concerning software licensing get vigorously discussed. Code is checked out, checked in and reviewed by community leaders in cycles of iterative development. The code and email archives left behind tell the story.

Fig. 1. CodeSaw showing Gaim in 2004. Small timelines denote developers, with time progressing from left to right. The top of each axis represents code contributions; the bottom represents project communication. The user has compared two developers by dragging their timelines into the investigation area at the top. By hovering, the user sees the top 5 files changed by a developer.

Yet, for the most part, projects leave these archives untapped. The sheer size of the archives may play a significant role. A typical open source project can generate over 20,000 CVS code checkins. A project mailing list may hold years of conversations, comprising thousands of individual email messages. While a long history of group dynamics lives in these archives, the current state of technology leaves it buried.

In this paper, we present CodeSaw, a social visualization of distributed software development. CodeSaw combines code repository information with project communication to visualize a software community from two independent perspectives. By bringing together both shared artifacts (code) and the talk surrounding those artifacts (project mail), CodeSaw reveals group dynamics that lie buried in existing technologies. CodeSaw's interface allows users to compare developers, dig deeper into archives and leave messages on the visualization itself (Figures 1 and 4). CodeSaw underwent an iterative design process, and we evaluated CodeSaw in an online field study with lead open source developers.

CodeSaw can benefit both community insiders and outsiders. For outsiders, CodeSaw provides an opportunity to compare open source projects and dig deeper into particularly relevant ones. For project developers, the foundation of the community, Code-

Saw offers a chance to reflect on their community and their contributions. Especially because most open source developers do not get paid for their work, seeing their contributions in the community context can be powerful. One user study participant said that CodeSaw recalled an important time when she worked closely with another developer who had left the project. Another told us that CodeSaw made him feel "vindicated." Our contributions include: the CodeSaw visualization, an example of how CodeSaw can reveal roles in open source projects, the evaluation of CodeSaw and a novel interaction technique for social visualization called spatial messaging.

2 Related Work

We begin by reviewing related work in distributed software development, our target audience. Afterward, we review related work in visualization.

Distributed Software Development. Gutwin, et al. interviewed open source developers to learn more about the mechanisms that underlie their collaboration [5]. In their study, they found that developers do maintain awareness of one another, and that text-based communication (e.g., mailing lists and chat systems) is the primary vehicle for maintaining it. Along the same lines, LaToza's study claims that 40% of a developer's time is devoted to communicating about code [6].

Researchers have also looked at open source communities from an organizational perspective. Mockus, et al. studied two well-known and successful open source projects, Apache and Mozilla [2]. In their study, they looked into the assignment of roles in each community. They found that a core group of about 10 - 15 developers used mailing lists to coordinate and assign work. Kraut and Streeter stressed the uncertainty, interdependence and informal communication of software development [7]. Noting the distinct problems of scale inherent in large software systems, they point out that "many software systems ... are very large and far beyond the ability of any individual or small group to create or even understand in detail." In the context of maintaining large software systems, Singer found that code repositories (e.g., CVS) are important sources of information for programmers [8].

Our work leverages the lessons of these studies. For example, incorporating Gutwin's analysis, CodeSaw uses mailing list email communication in its visualization. CodeSaw aims to address the need for a visualization that promotes awareness, both social and work-related, among distributed developers.

Visualization. A number of projects have visualized software projects [9–14]. SeeSys [9] takes an overall project approach, using a treemap visualization of the source code file system. SeeSys works particularly well for very large projects that are also relatively stable. Augur [11] adopts a line-oriented style, presenting one user with an integrated visual-diff display of CVS records. Augur, as in [12, 13], also gives users a social network view of the code and a temporal view. While Augur shows a line-oriented temporal history, CodeSaw abstracts that information. CodeSaw shows one year in the life of a project, and is not tied to any one file. CodeSaw also differs from Augur by adding

spatial messaging and email communication, social features, to its visualization. We designed CodeSaw as a community mirror, a visualization for an entire group.

Collaborative contribution has been studied outside of the source code context. History Flow [15] and Authorlines [16] visualize Wikipedia and Usenet postings, respectively. History Flow focuses on the interaction of collaborators on one Wikipedia entry. Authorlines deals with one user's posts to Usenet.

CodeSaw builds on the research presented here. CodeSaw seeks to visualize one year in the life of a project in an informative and intuitive way. Secondly, CodeSaw aims to present an aesthetically intriguing visualization. An aesthetically intriguing visualization can invite play, an important part of interacting deeply with data [17]. CodeSaw primarily differs from existing work by visualizing two important and contrasting information sources: code repositories and project communication. CodeSaw also presents a new, social approach to visualizing software development, exemplified by its spatial messaging and its focus as a community mirror.

3 CodeSaw's Iterative Design Process

Exploring the Design Space. To establish baseline requirements for CodeSaw, we conducted informal interviews about the collaborative software process with eight colleagues at our university. Five of our colleagues had worked on open source projects before; the other three had worked on many academic or commercial software projects.

Results. From our interviews we observed the following:

- *Developers felt disconnected from the rest of the community.* While working hard on one area of the code base, a developer may have a difficult time appreciating the work happening in other areas. This observation motivates tools that promote awareness and sociality inside the community. Such a tool could help confer value onto developers' contributions. However, developers noted that one metric alone would not capture the multiple dimensions of their work.
- *Developers used the project mailing list to maintain awareness.* This observation echoes the findings of Kraut and Streeter [7]. Developers used informal communication on the project mailing list to coordinate work and to get to know one another.

Iterations. CodeSaw evolved from a prototype to its current form over the course of four major iterations. After the development of each iteration, we conducted formative evaluations with between three and four software developers. In these sessions, developers used CodeSaw to explore a typical open source community, thinking aloud during the process. While it might be useful to present each of the iterations, for the sake of brevity we condensed this discussion into the following section. Next, we present CodeSaw's design rationale as a series of design implications learned from our formative evaluations.

4 CodeSaw

CodeSaw shows up to eight developers over the course of one year (Figure 1) [18]. Prior work pointed out that most projects have around 10 core developers. Our own experiments support this. In trials with 20 randomly selected open source projects, not one had more than 8 core developers.

4.1 Design Rationale

Design Implication: Focus on people. Users of the prototype started by asking simple social questions. "When was I most active?" "Who writes more code than me?" "What was going on on the mailing list then?" From the start it was evident that the prototype had difficulty answering these simple questions. We noticed that users of the visualization asked questions that revolved around people: themselves and the others working in the periphery. Therefore, CodeSaw focuses first and foremost on the people in the community.

Social Data Analysis. During formative evaluations, users would often call over other people to look at their discoveries. In one instance, a user noted that a particular developer only contributed code during the summer, wondering "if this developer might be a student." In another instance, a user commented that with the exception of one developer, no one had contributed to a particular project during the last week of December. Talking about the one developer that contributed, the user said, "the holidays must not have been good, since this guy worked so much." The developer contributed only code during this period; the mailing list was silent, implying that the developer worked alone. We feel that our focus on people directly lead to the social data analysis we observed.

Design Implication: Allow users to explore context. While early CodeSaw iterations achieved a simplicity that users liked, some felt that it hid too much information. They asked for a way to uncover some of the information that CodeSaw aggregated or discarded. For example, a number of designers wanted to know on what files the developers were working, or what phrases the developers were most commonly writing in emails at the time of a release.

To uncover the email discussion, CodeSaw shows excerpts from the email dialog on the project mailing list. In the same way that the source files are listed by their activity levels, email words are sorted by their frequency of appearance in the email exchange (Figure 2). The choice to include salient words from an email dialog mirrors the design choices made in the recent Themail [19]. In Themail, Viégas, et al. used prominent words from email exchanges to paint a portrait of relationships. In addition, users wondered aloud many times if release dates coincided with spikes in activity. To provide this contextual information, CodeSaw plots release names along the bottom of the visualization. This simple addition provides some interesting insights about releases and milestones.

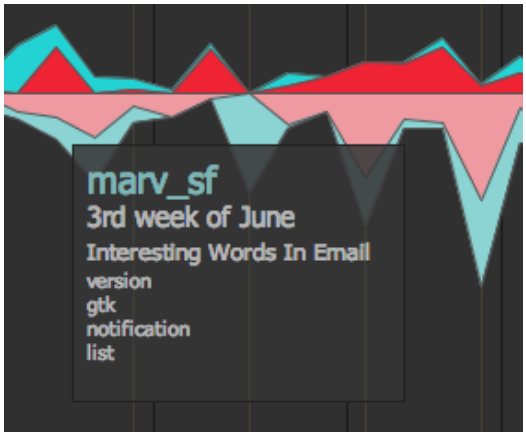

Fig. 2. CodeSaw exposes project email dialog to reveal informal communication. This design choice helps users explore the online social space. In this example, a user has hovered over the timeline of particular developer to find out what he was saying on the mailing list.

Design Implication: Only visualize essential information. The CodeSaw prototype used a large portion of the available information in the visualization: number of lines added, number of lines subtracted, recency of changes, etc. Initially, we hypothesized that code developers would want such a high level of detail. However, users told us that it exposed too much information. If they wanted such high levels of detail, users could more easily go back to the CVS archives and the code itself. Only the information items indicated as very useful by users made their way into CodeSaw.

Design Implication: Keep it quick and simple. In early iterations, a technical constraint made users wait for all of the data to load from the database before showing the visualization. CodeSaw requires a number of complex database queries that can take a considerable amount of time. Users often felt frustrated with the long time they had to wait to see anything. Following the lead of Wattenberg in NameVoyager [20], the data for all eight developers loads in parallel. The small timelines animate from left to right as data fills into them. Many have commented on the aesthetic beauty of this technique.

CodeSaw presents an easily graspable concept for code contributions: raw number of code lines added. Although this choice throws away some important information about a developer's actions, most designers and developers considered it a fairly accurate representation of work. We made a similar decision with email. CodeSaw goes with a simple metric, the number of words written in email on the project mailing list.

This simplified representation is very powerful. Most existing tools focus on only one archive. CodeSaw, on the other hand, incorporates two archives, visualizing the community from two important and separate perspectives. CodeSaw intuitively made sense to users, since it did not incorporate complex measurements that were hard to understand quickly. Unlike our experience with early iterations, many people said that they would not need to go back to CVS to see the things that CodeSaw told them.

4.2 The Visualization Details

The area under each triangle is calculated as follows. The raw number of code lines added to a source file by a developer determines the area under the top triangles. The raw number of words written on the project mailing list determines the area under the bottom triangles.

The timelines in CodeSaw employ Tufte's small multiples concept [21]. Small multiples are representations of information that repeat a common design and scale, inviting comparison. In Envisioning Information, Tufte writes, "for a wide range of problems in data presentation, small multiples are the best design solution." The small timelines in CodeSaw allow a user to compare developers. By adding any combination of developers to the detail graph, a user can learn much more about the people involved. While resembling a Zooming User Interface [22], our technique differs by reserving screen space for drag-and-drop comparison which affords any combination of timelines in the investigation.

5 Using CodeSaw To Find Trends And Roles

Having described CodeSaw's design, we turn now to using CodeSaw to find trends and roles in an open source community. In the example that follows we applied CodeSaw to the popular open source project Gaim [23]. Figure 1 shows Gaim broadly, looking at all core developers over the year 2004. Figure 3, on the other hand, shows a detailed view of two developers. Both views give us insight into the community.

A glance at Figure 1 tells us that Gaim was very active in 2004. A closer look at Figure 1, however, also shows us that one or two people do the majority of the work. We have seen this trend in all 20 open source projects we have analyzed with CodeSaw. A handful of leaders emerge who keep the project moving.

A closer look at the developers in Figure 3 offers insights into their behavior patterns. Until May, marv_sf wrote only mail (a). In the last few weeks of April, marv_sf and thekingant write a large amount of mail corresponding to a project release (b). CodeSaw denotes a project release by a thin gray line. Almost directly after April's mail surge, marv_sf starts coding for the first time, backed up by thekingant (c). marv_sf makes significant contributions to Gaim during these couple of months. Did thekingant convince marv_sf to code for Gaim?

By July, marv_sf has stopped coding for the most part. He contributes almost no code for the rest of the year, in fact (d). We can also see that thekingant and marv_sf tend to peak in project mail at the same times, usually connected to a release (e). In the first week of October, right before a release, both developers go silent–no code or mail (f). Did the project go down, or did both developers happen to take a break at the same time? Do they know each other personally?

Viewing CodeSaw from the outside, we can try to project a story onto the visualization. Did the surge in production by the thekingant inspire marv_sf? Did marv_sf finish a hard semester at school and suddenly find a lot of time on his hands? However, CodeSaw is primarily designed as a community mirror. From the outside we can learn important things about a project, but we do not get the critical context that only the

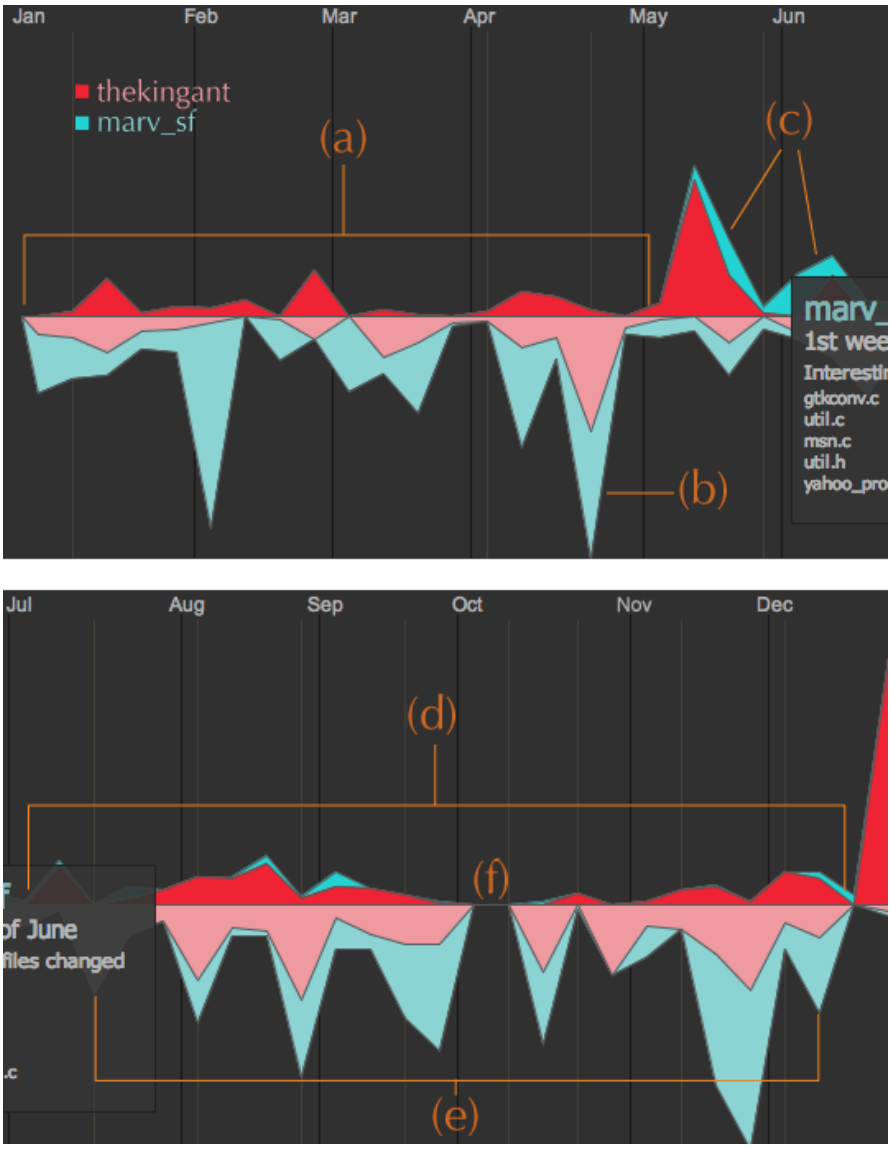

Fig. 3. A closer look at two core developers in Gaim. Until May, marv_sf writes almost no code (a). Then, following a surge in mail by both developers (b), marv_sf makes a big code contribution (c). However, his coding is short-lived (d). thekingant and marv_sf write mail at about the same time (e), and they take a break during the same week (f). What is their relationship?

community itself can provide. In this respect, we feel that CodeSaw achieves a good balance between revealing private information to the world and leaving enough of it ambiguous so as to not invade privacy.

6 Online Field Study

Since we wanted to test CodeSaw "in the wild," we distributed CodeSaw via the web to developers around the world. We wanted developers to integrate CodeSaw into their regular routines and projects, which we could not accomplish in the lab. We recruited by targeting project mailing lists and lead developers of open source projects. The projects were all hosted on SourceForge.net. In total, nearly 500 recruitment messages were sent over the course of one month. Subjects received a $20 gift certificate to an online retailer for their participation in the study.

Our approach allowed developers to see their own project history visualized. After interacting with CodeSaw from somewhere between 30 minutes to one hour, participants completed an online survey. We asked users to complete the survey no more than one hour after they finished using CodeSaw. Participants were free to continue using CodeSaw after completing the survey. Because we wanted to do an in situ study, and the participants were scattered across the world, we could not do interviews. So we replaced interviews with the best thing we could: an in-depth online survey. The survey asked participants about their satisfaction and enjoyment with CodeSaw. In addition, the survey asked participants about the effectiveness of CodeSaw in visualizing their community.

6.1 Participants and their projects

Nine participants took part in the evaluation of CodeSaw. Two were female and seven were male. The subjects ranged in age from 19 to 61. Five came from open source projects; four belonged to the same project at a research laboratory. The four from the research laboratory spent about half of their time in the same office and about half at a distance. They used email on the project mailing list to coordinate work. The research project had been ongoing for three years at the time of the study. In terms of code, the project consisted of 801 source code files.

Each of the five open source participants worked on a distinct project. Each project was among the top 200 most active projects hosted on SourceForge.net [24]. The duration of the projects ranged from one to four years. In terms of code, the projects ranged from 376 source files to 1785 source files, with a mean of 980. We chose these projects because they are representative of open source projects generally.

Each participant had developed software for more than three years. Six of the nine participants had used CVS for more than three years; the other three participants had used it for between one and three years.

7 Results

Overall, participants enjoyed using CodeSaw to investigate their project's history. When asked, using a 5-point Likert scale, how easy CodeSaw was to use, participants re-

sponded, on average 1.6 (1 being the most and 5 being the least). Not including watching a one minute introductory video, 8 of 9 participants said that it took them 10 minutes or less to get comfortable with CodeSaw, validating our design goal of a visualization that novices can access quickly. When asked how often they would use CodeSaw if it were deployed into their community, 7 of 9 said they would use it on a monthly basis or at release time. Since many projects go through 5 to 20 releases a year, we feel that this result is positive. Although participants were, for the most part, positive about their experience with CodeSaw, some pointed out flaws. We address these design lessons in the section entitled Incorporating Field Study Results.

7.1 Community

Many of the participants reported seeing confirming and surprising aspects of their community in CodeSaw:

> *It was interesting to see the contributions over time in an easy graphical interface. I am not sure it was surprising, rather confirming.*
>
> *It gave me a better idea about what the overall community is doing.*

Other participants noted that CodeSaw exposed roles in the community:

> *It confirmed my long-held suspicions about the community. It was easy to see who was doing the development and who was doing the commentary. Often they were not the same people.*
>
> *It was interesting. At first I felt as though I had not contributed much, but then I realized that I had a 'surrogate'. I was working closely with another developer who committed the changes to the code base.*

While most participants felt that CodeSaw seemed most relevant to core developers, one participant felt that CodeSaw would be of better use to outsiders looking in:

> *As a developer I pretty much knew what was happening, but this would be more useful to an outsider who wanted to understand the activity of each contributor.*

In this case, CodeSaw could be of use to newcomers to a project. They could discover at a glance who contributes when and to what parts of the code base.

Two participants that worked on projects with less than 5 developers commented on the loneliness that CodeSaw engendered:

> *I feel a bit lonely. It doesn't reflect the community in a whole. There's a lot of people passing by in the forums. People sending bug reports and patches are not taken into account.*
>
> *Viewing/visualizing the [anonymous] traffic on the forums and the lists would be great.*

We did not explicitly design for projects with just a few developers. However, in retrospect it seems clear that CodeSaw does not serve these projects well. The participant behind the first comment above said, "I don't feel like using it again. As I'm almost the only contributor, it doesn't make much sense." The Incorporating Field Study Results section specifically addresses these feelings of loneliness.

7.2 Incentives for Developers

Participants also remarked on the incentives CodeSaw provides to developers:

> *I'd love to have it in the "activity page" of the project to show the "pulse". It might be interesting as an immediate reward for developers. CodeSaw might show the "most active people" in the community very easily, and it could show the history of the project (some project leaders might want to erase some parts of it though). :-)*

> *It made me appreciative of who was doing the work.*

When asked how CodeSaw made him feel about his contributions to his project, one participant simply responded, "Vindicated. :-)"

Two of the participants commented on the potential for CodeSaw as a project management tool, an application we did not explicitly design for:

> *For example, if one looks at <username> in 2005, he did no development work that year. If the project were waiting for him to do something and a milestone was not met, we could get after him.*

> *I found some surprises - some developers were not attributed as working on the code I *thought* they were working on in a given period. Others were not contributing much code during periods when I thought they should have been.*

No one worried about privacy or about a boss having this tool. Since only one participant described himself as a project manager, we found this result somewhat surprising.

8 Incorporating Field Study Results: Spatial Messaging

In the field study, we learned that CodeSaw helped users better understand their communities and provided incentives for developers. However, we also learned that CodeSaw created feelings of loneliness in communities with only a few developers. We view this as the major design lesson from the field study and hope that designers of other social visualizations can learn from our experience.

As a response to this lesson, we added spatial messaging to CodeSaw (Figure 4). Spatial messaging allows users to leave comments on the visualization *itself*. These comments are linked to visualization state (i.e., the configuration of timelines in the detail graph). When a user brings the visualization back into the state where the comment was made, the comment appears. We do not intend to replace traditional communication channels (e.g., the project mailing list) with spatial messaging. Developers do not need yet another communication medium to monitor. In fact, we explicitly designed spatial messaging to supplement existing communication channels. A spatial message automatically generates mail to the mailing list. The message includes a link that brings the visualization into the right state.

We designed CodeSaw as a community mirror for developers and users that rarely meet face to face, yet construct a vibrant online social space. Using spatial messaging, developers and users can reflect on their shared history in the same place where they see it. We feel that spatial messaging represents a novel interaction technique for social visualizations.

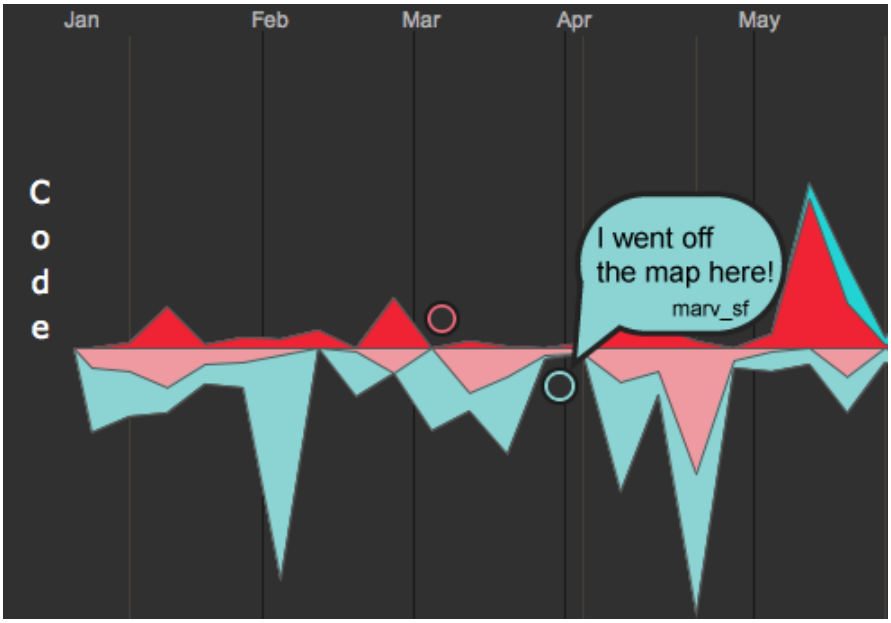

Fig. 4. CodeSaw incorporates spatial messaging to create a socially activated visualization. A user hovers over a spatial message, represented as a disk filled with the developer's color, to reveal its contents. Spatial messaging allows users to mark up a social visualization in a novel way.

9 Discussion

Overall, users found CodeSaw informative and intuitive. Users enjoyed interacting with CodeSaw and reported finding both confirming and surprising results. The field study suggests that CodeSaw positively affects users' conceptions of their own communities. In addition, the field study reveals that the visualization serves as a motivation to developers. This is key in open source communities, since developers volunteer their time. Based on user comments, we believe that introducing CodeSaw into an open source project may lead to increased production. We also believe that developers would feel more valued in the community. However, without a longitudinal study we cannot make concrete claims. We intend to follow up on this work with quantitative measures in a longitudinal study.

Since CodeSaw reveals deeply buried, although public, information, we expected privacy to be a serious concern. The field study showed otherwise. Not a single participant reported any concerns about privacy. It could turn out that under constant use and constant observation, CodeSaw would cause users to express privacy concerns.

Because we engaged in four iterations of a user-centered design process, we feel that CodeSaw reflects the needs of distributed software communities. As suggested by one of the participants in our field study, we plan to explore options for incorporating CodeSaw into real open source communities. Sourceforge.net is one option. A longi-

tudinal, ecologically-valid user study would provide valuable information to the HCI community and definitively answer the privacy questions expressed above.

Our field study provides a compelling analysis of the introduction of a social visualization into real-life open source communities. The field study also motivated the creation of a novel interaction technique for social visualization, called spatial messaging. Many visualizations revolve around scientific data or highly personal data, like email or health data. Since the data visualized in CodeSaw lies in a space between the familiar and the foreign, we believe our field study contributes in an often-overlooked area of visualization research.

10 Conclusion

We have presented CodeSaw, a social visualization of distributed software development. CodeSaw visualizes a software community from two unique perspectives: code repositories and project communication. Using two distinct project archives reveals patterns not available in current technology.

Following an iterative design process, we have explained the rationale behind CodeSaw as a series of design implications that can be used by other designers. Our online field study suggests that CodeSaw positively impacts distributed developers' notions of community, and that CodeSaw provides incentives for distributed developers. Following a design lesson from our field study, we have also presented a novel interaction technique for social visualization called spatial messaging. Spatial messaging allows users to leave comments on the visualization itself.

11 Acknowledgments

We thank the very busy developers in our study for contributing their time to our project. We also thank the UIUC Social Spaces group for their helpful feedback.

References

1. Orgell, E.: More than 1.1 million developers in north america now working on open source projects. http://evansdata.com/n2/pr/releases/dps2004.shtml. (2006)
2. Mockus, A., F.R., Herbsleb, J.: Two case studies of open source software development: Apache and mozilla. ACM Trans. Softw. Eng. Methodol. **11**(3) (2002) 309–346
3. Herbsleb, J.D., Grinter, R.E.: Splitting the organization and integrating the code: Conway's law revisited. In: ICSE '99: Proceedings of the 21st international conference on Software engineering, Los Alamitos, CA, USA, IEEE Computer Society Press (1999) 85–95
4. SourceForge.net: http://sourceforge.net. (2006)
5. Gutwin, C., Penner, R., Schneider, K.: Group awareness in distributed software development. In: CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work, New York, NY, USA, ACM Press (2004) 72–81
6. LaToza, T.D., Venolia, G., DeLine, R.: Maintaining mental models: a study of developer work habits. In: ICSE '06: Proceeding of the 28th international conference on Software engineering, New York, NY, USA, ACM Press (2006) 492–501

7. Kraut, R.E., Streeter, L.A.: Coordination in software development. Commun. ACM **38**(3) (1995) 69–81
8. Singer, J.: Practices of software maintenance. In: ICSM. (1998) 139–145
9. Baker, M.J., Eick, S.G.: Space-filling software visualization. Journal of Visual Languages and Computing (1995) 119 – 133
10. Eick, S.G., Steffen, J.L., Eric E. Sumner, J.: Seesoft-a tool for visualizing line oriented software statistics. IEEE Trans. Softw. Eng. **18**(11) (1992) 957–968
11. Froehlich, J., Dourish, P.: Unifying artifacts and activities in a visual tool for distributed software development teams. In: ICSE '04: Proceedings of the 26th International Conference on Software Engineering, Washington, DC, USA, IEEE Computer Society (2004) 387–396
12. Ducheneaut, N.: Socialization in an open source software community: A socio-technical analysis. Comput. Supported Coop. Work **14**(4) (2005) 323 – 368
13. Medynskiy, Y.E., Ducheneaut, N., Farahat, A.: Using hybrid networks for the analysis of online software development communities. In: CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems, New York, NY, USA, ACM Press (2006) 513–516
14. Bernard Kerr, L.T.C., Sweeney, T.: Growing bloom: design of a visualization of project evolution. In: CHI '06: CHI '06 extended abstracts on Human factors in computing systems, New York, NY, USA, ACM Press (2006) 93–98
15. Viégas, F.B., Wattenberg, M., Dave, K.: Studying cooperation and conflict between authors with *history flow* visualizations. In: CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, ACM Press (2004) 575–582
16. Viégas, F.B., Smith, M.: Newsgroup crowds and authorlines: Visualizing the activity of individuals in conversational cyberspaces. In: HICSS '04: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 4, Washington, DC, USA, IEEE Computer Society (2004) 40109.2
17. Vande Moere, A.: Form follows data: the symbiosis between design and information visualization. In: CAADfutures. (2005) 167 – 176
18. Gilbert, E., Karahalios, K.: Lifesource: two cvs visualizations. In: CHI '06: CHI '06 extended abstracts on Human factors in computing systems, New York, NY, USA, ACM Press (2006) 791–796
19. Viégas, F.B., Golder, S., Donath, J.: Visualizing email content: portraying relationships from conversational histories. In: CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems, New York, NY, USA, ACM Press (2006) 979–988
20. Wattenberg, M.: Baby names, visualization, and social data analysis. In: INFOVIS '05: Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization, Washington, DC, USA, IEEE Computer Society (2005) 1
21. Tufte, E.: The Visual Display of Quantitative Information. Graphics Press (1983)
22. Bederson, B.B., Hollan, J.D.: Pad++: a zoomable graphical interface system. In: CHI '95: Conference companion on Human factors in computing systems, New York, NY, USA, ACM Press (1995) 23 – 24
23. Gaim: http://gaim.sourceforge.net. (2006)
24. SourceForge.net: http://sourceforge.net/top/mostactive.php?type=week. (2006)