Studying the Use of Developer IRC Meetings in Open Source Projects

Emad Shihab, Zhen Ming Jiang and Ahmed E. Hassan
Software Analysis and Intelligence Lab (SAIL)
Queen's University
Kingston, ON, K7L 3N6, Canada
{emads, zmjiang, ahmed}@cs.queensu.ca

Abstract

Open source developers communicate with each other via various online outlets. Thus far, mailing lists have been the main coordination mechanism. However, our previous study shows that the use of developer IRC meetings is increasing in recent years. In this paper, we perform a study on the IRC meetings of two large open source projects: the GTK+ and Evolution projects. We explore three dimensions: who participates in the meetings, what do they discuss and how do they run the meetings. We find 1) that a small and stable number of the participants contribute the majority of messages in meetings, 2) that there are commonly discussed topics as well as project specific topics 3) that meeting styles vary across different projects.

1 Introduction

Developers of Open Source Software (OSS) are distributed across the world [18, 25]. OSS developers rarely meet in person, instead they coordinate most of their activities through online outlets, such as mailing lists, emails, bug tracking systems, Internet Relay Chat (IRC) channels, IRC meetings or instant messaging (IM) [23]. Thus far, mailing lists have been the main communication medium for developers of open source projects [7]. Mailing lists provide an open communication medium where all developers can review code and discuss concerns [22, 26, 27, 30].

Developer IRC channels are common servers, open 24 hours a day, where developers can connect and discuss informally, with no agenda agenda [19]. IRC channels are mainly used to ask others questions that pop up at the spur of the moment or to get quick feedback. IRC channels are very similar to IM, except for the fact that messages, by default, are viewable by everyone logged into the channel.

On the other hand, developers use *developer IRC meetings* to hold focused group discussions in a short period of time. The meetings are chaired by a meeting chair and are usually used to discuss several maintenance and project related issues such as upcoming releases, major bugs or task assignments.

Prior studies examined the developer IRC channels of distributed industrial and open source development teams [13, 17, 21]. However, very little is known about developer IRC meetings in the software maintenance and engineering communities. This limited knowledge is mainly attributed to the fact that developer IRC meetings are rarely archived. This paper aims to highlight the wealth of information that can be obtained from developer IRC meetings. We explore the IRC meeting archives for two large open source projects (Evolution and GTK) along the following three dimensions:

1. **Meeting Participants.** We quantified the contribution of participants and their stability over time.

   > *We found that key project members participate in IRC meetings and that a small and stable group of the participants contributes the majority of the IRC messages.*

2. **Meeting Content.** We explored the characteristics of the topics discussed in the meetings.

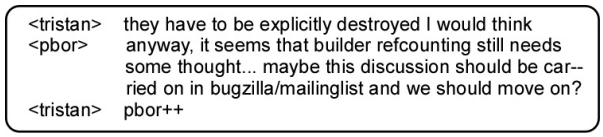
   > *We found that both projects share similar topics, such as bugs and patches, while having project specific topics that are consistent across time.*

3. **Meeting Style.** We characterized the communication style of the participants and studied their addressing patterns.

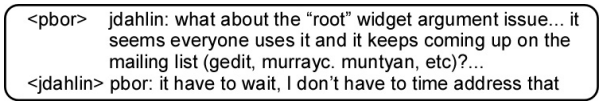
   > *We found that meeting styles vary across projects and that participants often address one another directly.*

To the best of our knowledge, this paper and our preliminary study [29] are the first to study developer IRC meetings in a distributed open source development setting.

Overview of Paper. Section 2 details the role of IRC meetings. Section 3 describes the studied projects. Section 4



(a) IRC discussion continues on bugzilla/mailing list



(b) Mailing list discussion being discussed in IRC meeting

Figure 1: IRC discussions complementing other coordination mechanisms

presents the process used to mine the IRC archives. We present and analyze our results in Section 5. We discuss the implications of our findings, possible future work and our threats to validity in Section 6. The related work is discussed in Section 7. Section 8 concludes the paper.

2 The Role of IRC Meetings

In our previous preliminary study [29], we showed that the usage of IRC meetings among OSS developers is growing over time.

In this paper, we sought to study the role of IRC meetings. We inspected the meeting logs for two large projects (Evolution and GTK+). We found that IRC meetings serve three roles:

1. *To complement other coordination mechanisms (e.g. mailing lists).*
   The excerpt in Figure 1(a) shows an IRC discussion that is moved to another coordination mechanisms (in this case the mailing list or bugzilla). Figure 1(b) shows a mailing list discussion that is brought up in the IRC meeting.
2. *To substitute physical meetings.*
   Figure 2 shows an excerpt that highlights the use of IRC meetings as a substitute to physical meetings. In the excerpt, some of the developers note their reluctance to travel to particular destinations (in this case the United States) and use the IRC meeting as a substitute.
3. *To answer questions and brainstorm.*
   Figure 3 depicts a small brainstorming session that is initiated by a question from the participant `federico`. During our study, we observed that many question-answer/brainstorming discussions are held in the developer IRC meetings.

3 Studied Projects

In this paper, we study the IRC meetings of the Evolution and GTK+ projects. We study these two projects for the following three reasons: 1) both projects are large open source projects that have a rich history (more than 8 years of development), 2) both projects cover two different domains (mail client and graphical library), and 3) both projects regularly hold and archive their developer IRC meetings.

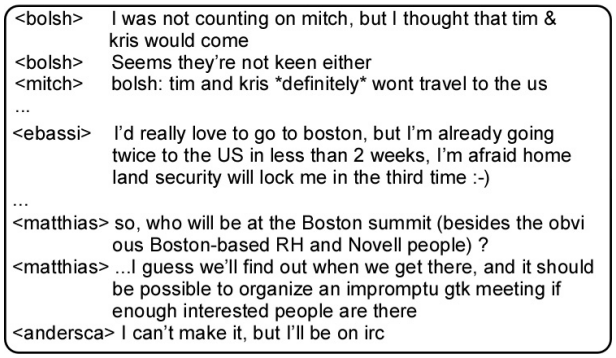


Figure 2: IRC meetings as a substitute for physical meetings

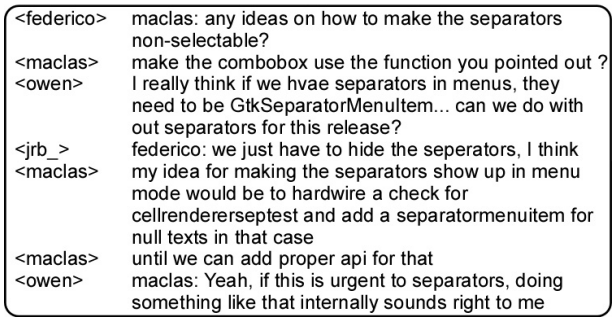


Figure 3: IRC discussions for brainstorming

Evolution is the mail client of the GNOME desktop. It provides email, address book and calendar functionality [2]. The GTK+ project is a library used to create graphical user interfaces (GUIs) for Windows and UNIX like platforms [4]. Developer IRC meetings are held by the GTK+ and Evolution development teams on a regular basis to discuss various project related issues, (i.e. bugs, release schedules and task assignments). However, the meetings are open to anyone interested in the project. We obtained the developer IRC meeting logs that are held in `#gtk-devel` [3] and `#evolution-meet` [1]. The number of meetings held, the number of different participants attending the meetings and the number of message lines is shown in Table 1.

	No. of meetings		No. of participants		No. of message lines	
Year	Evolution	GTK+	Evolution	GTK+	Evolution	GTK+
2004	-	34	-	44	-	5,105
2005	21	38	65	66	4,442	5,991
2006	-	13	-	24	-	1,252
2007	10	8	23	37	1,281	1,941
2008	31	14	28	44	3,725	2,903

Table 1: IRC meeting statistics for the studied projects

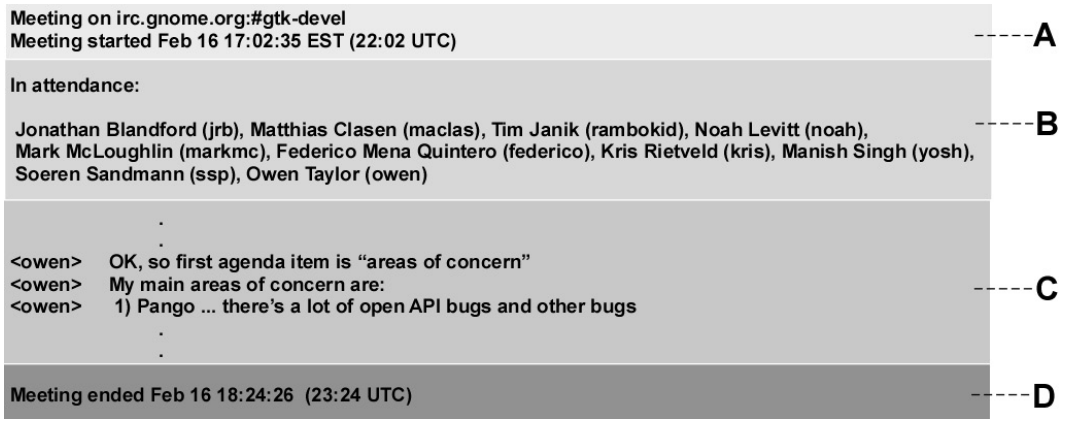


Figure 4: Sample IRC meeting log

The Evolution project started archiving their IRC meeting logs in 2005. The logs for 2006 were not available on their web site, therefore, in our study we use the IRC meeting logs from the years 2005, 2007 and 2008. The IRC meeting logs for the GTK+ project were archived since 2004 till present. In our study, we considered the logs from the time they were first archived till the end of December 2008.

We studied 105 of the 107 IRC meetings for GTK+ (two of the GTK+ meetings were summarized and no meeting logs were made available) and 62 IRC meetings for Evolution. In total, we studied 26,640 IRC messages (17,192 for GTK+ and 9,448 for Evolution) from 167 meetings, held over a period of 8 years (5 years for GTK+ and 3 years for Evolution). A total of 218 (127 for GTK+ and 91 for Evolution) unique participants contributed to these meetings.

4 Mining IRC Archives

In this section, we detail the approach used to process the archived IRC data. In a nutshell, we first examine the IRC logs to determine the different types of message formats. Next, we build a message parser which extracts the individual messages. Then, we resolve multiple name aliases and reconstruct the individual IRC messages. Finally, we store the data in a database for further examination. Each of these steps is detailed in the following subsections.

4.1 IRC Logs

Each IRC meeting is logged in a separate text file. A sample meeting log is shown in Figure 4. After obtaining the IRC meeting logs, a manual inspection yielded two types of IRC meeting logs. In the GTK+ project, the majority of the logs were comprehensive and included the start time (denoted as A in Figure 4), the list of attendees (denoted as B), the messages exchanged by the meeting participants (denoted as C) and the end time of the meeting (denoted as D). The IRC logs from the Evolution project were generally less descriptive and only the messages exchanged by the meeting participants (denoted as C in Figure 4) were included.

4.2 Message Parsing

We manually inspected the IRC logs and identified five different types of IRC message lines (as shown in Figure 5). In some cases, the month, date and time are included in the time stamp, while in others only the time is logged. In other cases, the time stamp was omitted altogether. It is worth noting here that this manual step is done only once, before we design and build our message parser.

After successfully identifying the different message formats, we built an IRC message line parser. The IRC message parser uses regular expressions to handle the different message formats. Given the limited number of IRC meetings in both projects, we were able to verify the accuracy of our parser for each meeting. We found this approach to work well in our case since we only had to deal with five different message formats.

4.3 Multiple Name Resolution

Participants of IRC meetings assign themselves nicknames before they join and sometimes change their nicknames during the meeting. Therefore, there can be multiple nicknames (aliases) for the same person. This so called multiple alias problem is similar to the multiple alias problem observed in mailing lists [8].

Date	Name	Message
Jun 05 21:05:40	<bob>	agenda for the meeting ...
	<bob>	agenda for the meeting ...
[21:05:40]	<bob>	agenda for the meeting ...
21:05	bob	agenda for the meeting ...
21:05	<bob>	agenda for the meeting ...

Figure 5: Different IRC messages formats

For example, the participant `jrb` uses five different aliases:

```
jrb
<jrb>
<jrb_>
<jrb_meet>
<jrb_sick>
```

Using name similarity heuristics, we resolved the majority of the aliases, however, manual inspection was required to resolve some of the rare cases. Furthermore, the majority of the IRC meeting participants use abbreviated names, therefore, methods such as the one proposed by Robles and Gonzalez-Barahona [28] may be used to accurately identify the participants' real names. Such identification becomes extremely important when multiple data sources (i.e. source code repositories, mailing lists and IRC meeting logs) are used together and one needs to create a unique identifier for each participant across all sources.

4.4 Data storage

After parsing the message lines, we reconstructed the IRC messages in preparation for storage in the database. Each IRC message contains three properties: `date`, `name` and `message`.

The information is stored in a PosgreSQL database for further use. The use of a database eases the exploration of the data at hand since we could rapidly explore different questions and generate specialized views to answer these questions.

5 Results and Analysis

Our findings are presented below. First, we discuss the participant's attendance, their contribution and their stability over time. Then, we present our findings on meeting content. Lastly, we study the meeting organization and addressing patterns.

5.1 IRC Meeting Participants

We sought to examine the people who attend the IRC meetings. By understanding who attends these meetings and their contribution and attendance pattern, we can better understand the use of these meetings and the benefit of studying the discussion in these meetings. Our study is structured along the following three research questions.

Who participates in the meetings?
For each project, the project documentation states that the meetings are attended by the "Evolution team" for the Evolution project and the "GTK+ team" for the GTK+ project. Team in this context refers to the core development team of the project. However, the meetings are open to everyone who is interested in the projects.

To investigate whether the core developers attended the IRC meetings or not, we obtained the names of the core developer team of the GTK+ project from the project's web site. Then, we manually linked the real names to their IRC nicknames as used in the IRC meetings. We were able to identify the IRC nicknames of 9 of the 10 core developer team members.

We measured the attendance of the core team during the year 2008 and found that out of the 14 meetings held in 2008, 8 of the 9 identified core team members attended at least 1 meeting and 50% of them attended more than half of the meetings held that year. Further, the average number of participants per meeting in the year 2008 is 10 participants. We can conclude that the core development team regularly attends the IRC meetings and that the meetings are popular (i.e. frequently attended) amongst other developers. We elaborate on this point and quantify the participant contribution next.

We did not perform the same analysis on the Evolution project since we were unable to obtain a list of its core developers.

How much do participants contribute?
The Pareto principle, states that the majority of the effects come from a minority of the causes. For instance, research shows that 20% of the code contains 80% of the bugs [11].

We conjecture that there exists a few participants (whom we call the *dominant group*) in IRC meetings that are responsible for most of the posted messages. In open source development, there is typically a central group that is responsible for important tasks, such as official releases [23]. They are members who are very knowledgeable about the project or members who are responsible to coordinate the project development efforts. This central body guides and supports newcomers and less active participants (whom we call the *casual group*). It is important to investigate whether a dominant group exists for two reasons: 1) participants can directly address their questions to the dominant group so they can receive more accurate and speedy responses and 2) the discussions of dominant group members are often used by others who are less knowledgeable about the project in the future.

To identify the dominant group, we measured the top 20% most active participants in an IRC meeting based on the number of posted message on a yearly basis. For example for the GTK+ project in 2004, we took the top 9 (20% of the 44 participants) and labeled the participants as the dominant group. We studied the contribution by the dominant group in the GTK+ and Evolution IRC meetings. The results are shown in Table 2. We observe that on average the dominant group (i.e. 20% of the participants) contributes close to 80% of the IRC messages in both projects.

We compared the dominant group members for the year 2008 with the core development team members. We found that 60% of the dominant group for the GTK+ project is made up of members from the core development team. The

	Evolution		GTK+	
Year	**Dominant**	**Casual**	**Dominant**	**Casual**
2004	-	-	82.5%	17.5%
2005	80.0%	20.0%	81.7%	18.3%
2006	-	-	75.8%	24.2%
2007	83.6%	16.4%	74.9%	25.1%
2008	85.2%	14.8%	84.6%	15.4%
Average	**82.9%**	**17.1%**	**79.9%**	**20.1%**

Table 2: Percentage of messages contributed by the Dominant and Casual groups

	Evolution		GTK+	
Year	**Dominant**	**Casual**	**Dominant**	**Casual**
2004-2005	-	-	0.86	0.37
2005-2006	-	-	0.70	0.16
2006-2007	-	-	0.91	0.13
2007-2008	0.98	0.51	0.77	0.19
Average	**0.98**	**0.51**	**0.81**	**0.21**

Table 3: Cosine distance of the Dominant and Casual groups

other 40% of the dominant group are members that are not part of the core development team. This finding posts two interesting points: 1) the core development team is actively involved in the IRC meetings and 2) there are non-core members who contribute significantly to the IRC meetings and closely mingle with the core development team. Investigating the role of the active group of non-core members is an interesting question which we plan to study in the future. Nevertheless, the dominant group plays an important role in the IRC meetings.

How stable are contributions by the participants?

Now that we have determined that the dominant group contributes significantly to the meetings, we study the stability of the contribution by the dominant and casual group members. A relatively stable pattern of contribution by the dominant group (i.e. one that does not change frequently) is desirable because it means that dominant group members are actively involved in the meeting and contribute at the same rate over time, instead of contributing heavily in a few meetings and remaining silent in other meetings.

We use the Cosine Distance (CD) similarity metric to measure the variance in the rate of contribution by participants to the meetings. The CD similarity metric measures the similarity between the groups in two consecutive years. We chose to use the CD as the measure of similarity because it takes into account the presence of a participant (i.e. if they are there or not) and their level of contribution when measuring similarity. It outperforms other simple measures such as intersection or proportion which only measure the existence of a participant but not their level of contribution. The CD similarity is defined as

$$CD(P,Q) = \frac{\sum_x P(X)Q(X)}{\sqrt{\sum_x P(X)^2}\sqrt{\sum_x Q(X)^2}}$$

The CD metric takes as input two participation distributions – one for each of the years under study. Each distribution has the contribution of each of the participants for that year. So when comparing the dominant group from the year 2005 to the year 2006, the 2005 and 2006 participation distribution for the dominant group are used. The participation distribution records the percentage of messages contributed by each participant. A value of 0 for the CD metric means that the pattern of contribution by the group has changed drastically across two years. A value of 1 for the CD metric indicates that the pattern of contribution by group is very stable.

The results are shown in Table 3. We observe that on average, the contribution pattern by the dominant group is very stable. Overall, the contribution pattern by dominant group members is at least twice as stable as the casual group. This is a positive sign indicating that expert members, who are critically important to the IRC meetings are actively and consistently participating and imparting their knowledge in the meetings. We plan to explore, in more detail, the contribution variations of the developers in the future.

5.2 IRC Meeting Contents

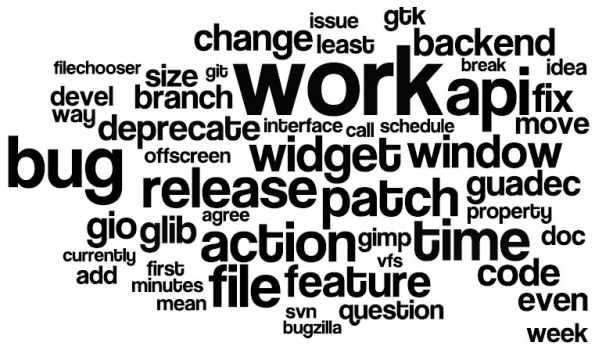
We study the content of the meetings to understand the value of conducting such meetings over other communication mechanisms such as mailing lists and bugzilla. We examine the topics discussed during the meetings. In particular, we examine the topics common across projects, the topics specific to a project as well as the time-specific topics.

What are they talking about?

We use the data from the IRC meetings to examine the discussed topics. Determining the different topics discussed in the IRC meetings is a good indicator of the use of the meetings and provides evidence of the usefulness of the information in these meetings, serving as an guidance for future exploration of such logs to support the maintenance and evolution of large software projects.

To determine the most frequently discussed topics, we visualize the word tokens from the developer IRC meeting logs using a tag cloud [6]. A tag cloud counts the frequency with which a word is mentioned in the text and adjusts the size of each word according to its frequency. More frequently mentioned words are shown in bigger and bolder font in the tag cloud. Viewing the cloud, we can quickly

(a) Meetings in 2004



(b) Meetings in 2008

Figure 6: Tag cloud of top 50 words used in the GTK+ IRC meeting

Figure 7: Tag cloud of top 50 words used in the Evolution IRC meetings in 2008

spot the most frequently discussed words in the meetings and we can infer the most popular topics in the meetings.

Informal language and names of participants are commonly used in the meetings. The use of informal language serves as noise and makes it difficult to extract useful information from these messages. We stemmed all words, removed all names and stop words and created a tag cloud of the top 50 words using the exchanged IRC messages.

Figures 6(a) and 6(b) show the most frequently used words in the GTK+ meetings in the years 2004 and 2008, respectively. We find that general topics such as `bug`, `release`, `patch`, `API`, `work` and `time` are frequently discussed in both tag clouds. In addition, there are topics that are time specific and which change with time. For example, in the year 2004, the term `python` is discussed frequently, while in 2008 we see that `deprecate` is discussed frequently and the term `python` is no longer in the top 50 words. GTK+ is a graphics library so terms like API, windows, and button are project specific terms.

Figure 7 plots the tag cloud of the top 50 words for the Evolution project. Again, main topics such as `bug`, `release`, `patch` and `work` are frequently discussed. Further, project specific terms such as `mapi` and `exchange` are also frequently discussed in the Evolution IRC meetings. MAPI and Exchange are Microsoft mail technologies. These topics are very specific to Evolution which is a mail client.

Our findings post three interesting points: 1) there exist common topics, such as bugs and patches that are discussed frequently over time within projects and across projects, 2) there exist project specific topics that are discussed frequently over time and 3) there exist time specific topics that are discussed within a project during specific time periods. We discuss these three findings in more detail.

Common Topics Across Projects. As we have seen earlier, common topics such as bugs and patches appear in the tag clouds that were created from IRC meetings at different times and even in different projects. To investigate this finding in more detail, we manually inspected the IRC meetings. We find that common software development tools like bugzilla, and project management issues like bugs and patches are regularly discussed in IRC meetings.

To illustrate, we show excerpts from meetings in different years and from different projects (shown in Figure 8). In Figure 8(a), the issue tracking tool bugzilla, is referred to in GTK+ in the different years (across time) and in Evolution as well (across projects). Figure 8(b) presents the IRC discussions that are related to project management issues. These project management issues like patches and releases are also discussed in the IRC meetings across time and across projects.

Project Specific Topics. Topics that are specific to a project, but are not time sensitive are commonly discussed in IRC meetings as well. For example, we found that the term `widget` appears in the years 2004 and 2008 in the GTK+ project. Furthermore, the term `exchange`, which is short for Microsoft Exchange, and its associated API `mapi`, is used commonly in the Evolution project.

Figure 9 presents excerpts that support our findings. We find that widgets are being discussed in the GTK+ project in 2004 and in 2008 because GTK+ is a graphics library. On the other hand, Microsoft Exchange (a mail server) is discussed in Evolution (which is a mail client). The term `exchange` is also frequently discussed in Evolution in

GTK+ 2004

<federico> owen: do you want to go through bugzilla marking blockers as such, or adding them to the gtk-web list?

GTK+ 2008

<timj> - should we add a website bugzilla component?
<martyn> timj: yes please :)
<mitch> and there is an evil filechooser bug i reported to carlos
<mitch> as in really evil
<mclasen> mitch: thats in bugzilla

Evolution 2008

<jony> Will be creating a bugzilla component for 'evolution--mapi' .. and makefiles are kludgy now .. patches are welcome :)

(a) Related to the use of development tools

GTK+ 2004

<federico> this week I want to review PATCH bugs to reduce the pending count
...
<owen> markmc: I think 2.4.0 is going to be going released prematurely in the sense that the bug submission rate is still going up
<markmc> owen, I suppose the quality of 2.4.0 isn't a huge issue as long as more release follow quickly

GTK+ 2008

<mclasen> mitch: carlos had a set of patches for remote file han--dling
...
<bratsche> If anyone else wants to check out the current patch for #56070 (mclasen already said he will), I'd really like to get this committed before the release.

Evolution 2008

<srag> Great news that we were below 50 unreviewed patch es for the first time
...
<srag> Monday we have our stable release in 2.22 series... just make sure that you have committed all your patches there

(b) Related to project management

Figure 8: Common topics across projects

prior years as well (2005 and 2007). We were surprised to note the common use of the term `share` in the Evolution project. Further investigation shows that this is due to the chair of most Evolution meetings in 2008 using the phrase "anything to share with us?" repeatedly as he goes around the table asking for updates. This pattern of conducting a meeting is explored further in the following subsection on meeting style.

Time Specific Topics. Some of the topics discussed in the meetings are time specific. For example, there might be topics (e.g. features or bugs) that become popular during a certain period of time then are seldom discussed afterwards. There are also maintenance related discussions that become more frequent at different stages of the project (e.g. deprecated code during code clean up phases in a project and release plans and documentation around the time of a new release).

GTK+ 2004

<matthias> can we get libglade in shape for 2.6 ? I think it would be great to have a libglade release supporting most of the new widgets close to the 2.6 release

GTK+ 2008

<tbf> behdad: basically they just figure out the minimum size for all attached widget, and tweak the size--request each widget reports accordingly

Evolution 2008

<srag> lakhil, anything to share with us ?
<lakhil> patches and exchange back end testing, bug triaging

Figure 9: Project specific topics

GTK+ 2004

<mathrick> IDL would be implementation detail of C code
<owen> mathrick: Inherting C from Python is *hard*. I spent a day discussing that with hp once. I'm not sure it's worth trying to do.
...
<owen> mathrick: In python, I have all these wonderful intro-spection, dynamic object creation, etc, facilities. That I can use to easily bind to something defined with a typelib

GTK+ 2008

<mitch> i'd like to deprecate some tiny bits of gtktypeutils but that can also be done after the release
...
<timj> # 4 What is the prospect of creating a spin-off libgtk3deprecated that's not maintained by the Gtk+ core team.

Figure 10: Time specific topics

From the tag clouds for the GTK+ project, we observe evidence of the time specific topics. For example, in 2004 the term `python` is listed as one of the top 50 words used in the IRC meetings. In 2008, the term `python` is no longer listed as being in the top 50 words and the term `deprecate` now appears in the tag cloud. We extracted excerpts to help explain why these terms are popular during these specific times. It can be observed from Figure 10 that in the year 2008, the IRC meetings are discussing issues related to deprecating some of the GTK+ code. Through manual inspection, we observed that discussions related to code deprecation were becoming more popular over time. Also, in 2004, the GTK+ developers were discussing the pros and cons of using C versus Python while they were exploring the idea of adding support for python in the GTK+ library. With the decision to support python done in 2004, the popularity of the term `python` dropped.

5.3 IRC Meeting Style

In our study on meeting style, we examine how IRC meeting are run. In particular, we identify the different types of meeting styles and study the various addressing patterns.

	Evolution	GTK+
Agenda	9.2%	56.5%
Update	87.7%	25.0%
Other	3.1%	18.5%

Table 4: Percentage of agenda, update and other meetings in the Evolution and GTK+ projects

How are the meetings run?

In this subsection, we examine how the IRC meetings are run. We found three styles of meetings: *agenda meetings*, *update meetings* and *other meetings*. Agenda meetings are organized meetings that follow a specific agenda. Agenda items are discussed in the meeting and most of the agenda items are addressed before the meeting adjourns. Update meetings are meetings where participants meet to provide progress updates and discuss any difficulties they are facing. The other type of meetings are meetings that do not follow an agenda nor a update style. They are brief meetings that are put together to discuss a very specific point. From our observations, they are usually meetings that discuss a specific bug or release. In all of the meetings, there exists a meeting chair who coordinates the meeting.

To quantify the types of meetings, we manually inspect all of the GTK+ and Evolution developer IRC meetings and categorize them as follows: if a meeting followed an agenda, we classify the meeting as an agenda meeting, if a meeting chair was asking other participants for updates, then we classify the meeting as an update meeting and if a meeting is not classified by the previous two categories, then we include it in the other meeting category. In some rare cases, the meeting chair addressed an agenda and then went on to ask for updates from the participants. In such cases the meeting was counted as both an agenda and a update meeting.

We present the results of our findings in Table 4. The majority (87.7%) of the meetings held by the Evolution developers are update meetings; on the other hand, the majority (56.5%) of the meetings held by the GTK+ developers are agenda meetings.

We extracted some excerpts from the meeting logs and present them in Figures 11 and 12. In Figure 11 the agenda and agenda items are regularly being referred to in the GTK+ meetings. On the other hand, Figure 12 shows the meeting chair of the Evolution meeting constantly asking for updates from other participants. The frequent use of the phrase "anything to share with us" led to the word `share` being one of the most frequent words for the GTK+ project as noted in Figure 9.

How do participants address each other?

Participants can post messages to all the participants in the IRC meeting or directly address another participant in the channel. Direct addressing is used to ask a question or to ask for an update from a specific participant. Studying the direct addressing patterns has many advantages. For example, it can be used to identify key project members (i.e. managers often use direct addressing to ask questions to other team members) and to study the social connectivity of the development team (i.e. members that address each other frequently are most likely to work closest together) [12].

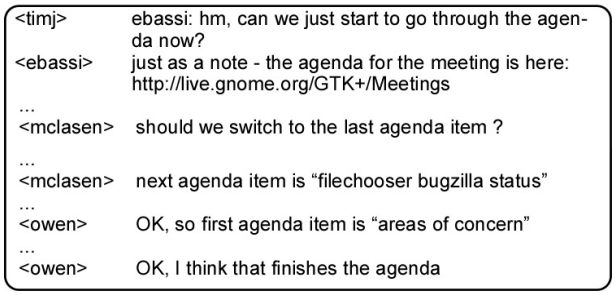


Figure 11: Agenda meetings in the GTK+ project

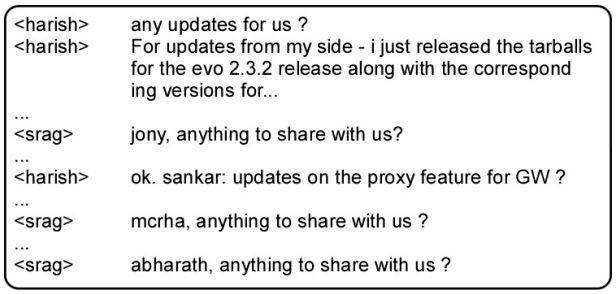


Figure 12: Update meetings in the Evolution project

Direct addressing is usually achieved by specifying the target's nickname followed by a semicolon or another type of punctuation [24]. Figure 12 shows an example where direct addressing is used. We measured and quantified the number of direct-addressing messages in the GTK+ and Evolution projects.

We found that a large number of the IRC messages are direct-addressing messages. 39.6% of the GTK+ messages and 54.4% of the Evolution messages are direct-addressing messages. Further, we found that the most active IRC meeting participants (i.e. the participants with the most messages posted) are also the participants with the highest number of direct-addressing messages. For example, the top 10 most active participants in the GTK+ meetings account for more than 66% of the direct-addressing messages. A similar trend is observed for the Evolution project as well.

We observe that, in agenda meetings, the meeting chair usually directly addresses other participants to discuss their agenda items. In update meetings, the chair would directly address participants to ask for progress updates.

6 Discussion

IRC meetings are an important coordination mechanism that needs to be studied further by the software maintenance community. IRC meetings allow us to study the dynamics of the open source development community. They provide us with features that are unmatched by other static coordination mechanisms such as mailing lists. In contrast to mailing lists, participants in IRC meetings must respond quickly instead of being able to think more about their replies like in mailing lists. Using IRC logs we can recognize the different personalities of OSS developers (e.g. more vocal versus deep thinker personalities). We can also observe firsthand the characteristics of good project leaders, expert designers, and great mentors based on their live interaction with others in the IRC meetings. Given the short-length and real-time medium of IRC meetings, we can better understand the priorities of open source projects. What are the topics that are worth raising through this limited time medium? Why are these topics raised in the IRC? How are these topics raised through other mediums like the mailing list. As one of the first exploratory studies of IRC meetings, we discuss interesting avenues for further research along the three dimensions used in our study: participants, content and style.

Meeting Participants. With a large number of meetings focusing on providing status updates, the meeting data could be used to identify domain experts. This first hand information can be leveraged to enhance cost and effort estimation. In addition, the same first hand information can also be leveraged to enhance current methods used to identify domain experts. We noted that some of the core members of project are not active participants in these meetings while others actively participate. It would be interesting to use the participation in IRC meetings as an indicator of future core members and of departing core members. Finally, the real-time exchange of IRC meetings provides us with insight about the personalities of developers.

Meeting Content. Our study on IRC meeting content reveals that IRC participants discuss topics that are time specific, project specific and topics that are common across time and projects. For example, we noted that both projects use bugzilla as their issue tracking system. We can use the data from IRC meetings to compare and contrast how these issues are reported, managed and reviewed across different projects. This type of information is valuable to studies on the open source development process. In addition, the information from IRC meetings can be used to study how key project members discuss and settle major project decisions in real-time instead of over mailing lists.

Meeting Style. Our study on meeting style shows that different projects run their meetings differently. Nowadays, many open source and commercial projects distribute their software development across the world. Studying the IRC meeting style can enhance studies which evaluate the coordination overhead of remotely located developers [9]. Furthermore, by studying the IRC meeting interaction, we can gain insight in best practices and options for future active project communication tools such as the Jazz platform [5].

Threats to Validity

The archived IRC meeting data used in our study does not contain private messages which may have been sent during meetings. This type of data is not archived. Furthermore, although our study focused on two large open source projects, our findings might not generalize to all open source projects.

7 Related Work

Previous work used information from IRC channels to study: the culture and beliefs in the open source community and coordination and communication networks that emerge in globally distributed commercial software development teams.

Elliott and Scacchi [15–17] used data gathered from GNU IRC channels to study the organizational cultural beliefs and values of free software virtual organizations and their affect on the software development process. They found that culture beliefs and values have a significant impact on the processes of free software development.

Gutwin *et al.* [20] examined how distributed developers maintain group awareness. They found that awareness is primarily maintained through IRC channels, Internet radio and mailing lists.

Cataldo *et al.* [14] used data from developer IRC channels to study task dependencies and the coordination activities performed by individuals. They proposed a technique that can identify the best "fit" individual for a task. They showed that their technique can reduce the amount of time required to perform a task.

Cataldo and Herbsleb [13] studied the evolution of communication networks of a geographically distributed commercial software development project. They found that over time a core group of developers emerged as a liaison between teams located in different locations. They also found that this core group contributed the most to the development effort. In addition, they found that members of the core group rotated in and out of the core based on the dependencies of their technical work at the time.

Our study differs from the previous work by Elliot and Scacchi [15–17] in that we focus on developer IRC meetings and not developer IRC channels. Previous work by Cataldo *et al.* [13, 14] and Gutwin *et al.* [14] also studies IRC channels, not IRC meetings. Further, in their studies, IRC data from a commercial software development team was used. There are differences between open source software development and commercial software development that may make their findings inapplicable to open source software development [23].

8 Conclusions

OSS Developers use Mailing lists, Instant Messenger, IRC channels and meetings to communicate with each other. Prior research focused on mining developer mailing lists (e.g. [10]) and IRC channels (e.g. [13, 17]). However, little work has studied the use of IRC meetings. At the same time, previous work [29] shows that IRC meetings are increasing in popularity (i.e. frequently attended) among OSS developer.

In this paper, we studied the developer IRC meetings from two large open source projects along three dimensions: meeting participants, content, and style. Our study highlights the usefulness of studying IRC meetings and provides interesting directions for future research.

References

[1] Evolution IRC team meetings. http://projects.gnome.org/evolution/meetings.shtml.

[2] The evolution project. http://projects.gnome.org/evolution/.

[3] Gtk meetings space. http://live.gnome.org/GTK+/Meetings.

[4] The GTK+ project. http://www.gtk.org/development.html.

[5] Jazz community site. http://jazz.net/.

[6] Wordle - beautiful word clouds. http://www.wordle.net/.

[7] O. Baysal and A. J. Malton. Correlating social interactions to release history during software evolution. In *MSR '07: Proceedings of the Fourth International Workshop on Mining Software Repositories*, 2007.

[8] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining email social networks. In *MSR '06: Proceedings of the 2006 international workshop on Mining software repositories*, 2006.

[9] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy. Does distributed development affect software quality? an empirical case study of windows vista. In *ICSE '09: Proceedings of the 31st international conference on Software engineering*, 2009.

[10] C. Bird, D. Pattison, R. D'Souza, V. Folkiv, and P. Devanbu. Latent Social Structure in Open Source Projects. In *FSE '08: Proceedings of the 2008 ACM SIGSOFT symposium on the Foundations of Software Engineering*, pages 24–35, 2008.

[11] B. Boehm and V. R. Basili. Software defect reduction top 10 list. *Computer*, 2001.

[12] G. Breach. I'm not chatting, I'm innovating! locating lead users in open source software communities. http://www.business.uts.edu.au/management/workingpapers /files/Breach2008.pdf.

[13] M. Cataldo and J. D. Herbsleb. Communication networks in geographically distributed software development. In *CSCW '08: Proceedings of the ACM 2008 conference on Computer supported cooperative work*, 2008.

[14] M. Cataldo, P. A. Wagstrom, J. D. Herbsleb, and K. M. Carley. Identification of coordination requirements: implications for the design of collaboration and awareness tools. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, 2006.

[15] M. S. Elliot. The virtual organizational culture of a free software development community. In *Proceedings of the 3rd Workshop on Open Source Software Engineering*, 2003.

[16] M. S. Elliott. Communicating and mitigating conflict in open source software development projects. http://www.ics.uci.edu/ melliott/commossd.htm, 2002.

[17] M. S. Elliott and W. Scacchi. Free software development: Cooperation and conflict in a virtual organizational culture. In *Free/Open Source Software Development, Idea Publishing*, 2004.

[18] J. Feller and B. Fitzgerald. *Understanding open source software development*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[19] D. M. German. The gnome project: a case study of open source, global software development. *Software Process: Improvement and Practice*, 2004.

[20] C. Gutwin, R. Penner, and K. Schneider. Group awareness in distributed software development. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, 2004.

[21] M. Handel and J. D. Herbsleb. What is chat doing in the workplace? In *CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*, 2002.

[22] A. E. Hassan. The road ahead for mining software repositories. In *Proc. FoSM 2008. Frontiers of Software Maintenance*, 2008.

[23] A. Mockus, R. T. Fielding, and J. D. Herbsleb. A case study of open source software development: the apache server. In *ICSE '00: Proceedings of the 22nd international conference on Software engineering*, 2000.

[24] P. Mutton. Inferring and visualizing social networks on internet relay chat. 2004.

[25] E. Raymond. The cathedral and the bazaar. *Knowledge, Technology & Policy*, 12(3):23–49, September 1999.

[26] P. C. Rigby, D. M. German, and M.-A. Storey. Open source software peer review practices: A case study of the apache server. In *ICSE '08: Proceedings of the 30th international conference on Software engineering*, 2008.

[27] P. C. Rigby and A. E. Hassan. What Can OSS Mailing Lists Tell Us? A Preliminary Psychometric Text Analysis of the Apache Developer Mailing List. In *MSR '07: Proceedings of the Fourth International Workshop on Mining Software Repositories*, 2007.

[28] G. Robles and J. M. Gonzalez-Barahona. Developer identification methods for integrated data from various sources. *SIGSOFT Softw. Eng. Notes*, 30(4), 2005.

[29] E. Shihab, Z. M. Jiang, and A. E. Hassan. On the use of internet relay chat (IRC) meeting by developers of the GNOME GTK+ project. In Proceedings of the 6th IEEE Working conference on Mining Software Repositories (MSR), 2009.

[30] P. Weissgerber, D. Neu, and S. Diehl. Small patches get in! In *MSR '08: Proceedings of the 2008 international working conference on Mining software repositories*, 2008.