Boundary-Spanning Documents in Online FLOSS Communities: Does One Size Fit All?*

Carsten Østerlund & Kevin Crowston
Syracuse University School of Information Studies

Abstract

Online communities bring together people with varied access to and understanding of the work at hand, who must collaborate through documents of various kinds. We develop a framework articulating the characteristics of documents supporting collaborators with asymmetric access to knowledge versus those with symmetric knowledge. Drawing on theories about document genre, boundary objects and provenance, we hypothesize that documents supporting asymmetric groups are likely to articulate or prescribe their own 1) purpose, 2) context of use, 3) content and form and 4) provenance in greater detail than documents used by people with symmetric access to knowledge. We test these hypotheses through content analysis of documents and instructions from a variety of free/libre open source projects. We present findings consistent with the hypotheses developed as well as results extending beyond our theory derived assumptions. The study suggests new directions for research on communications in online communities, as well as advice for those supporting such communities.

1 Introduction

The information technology revolution has led to the proliferation of online communities and digital collaborations both within and across organizations. In many online communities, documents (considered broadly) constitute the primary (or only) means for knowledge sharing and exchange among collaborators. Research has suggested the importance of mutual knowledge [1], shared mental models [2] or common ground [3] as a basis for communication. Yet, community members often bring divergent understandings and knowledge from non-converging frames of reference to the production and use of documents, hampering communication. For example, a novice programmer with no history in a particular project may be able to get some sense of the work completed from a report written by a software engineer on the project. However, without knowledge of the organizational practices that went into creating the code and the report, the novice may be unable to use either to determine what to do to make a contribution. In contrast, an expert engineer with experience on similar projects may simply need a few key words to guide his or her future work. A single type of document does not fit both audiences. How then can IS researchers and practitioners best support such heterogeneous online environments with 1000s of users, some deeply involved, many only peripherally so?

In such situations, one is often advised to fall back on the age-old truism, "know your audience and write appropriately." But what does it mean to write appropriately for an audience? To following the saying, one needs to be able to address two questions. 1) Who is the audience? 2) How does one write appropriately? In this paper, we manipulate the first dimension by studying two types of relations among writers and their audience: relations characterized by symmetric access to knowledge vs. asymmetric access to knowledge. This manipulation allows us to answer the second question by examining what aspects of documents can be tailored based on the background knowledge of different audiences and so exploring strategies that online community members apply to "write appropriately" to these two different types of audiences. More specifically, we address the following question:

> What characterizes documents that link people with asymmetric access to background knowledge compared to documents used among people with symmetric access to knowledge?

Answering this question is important for understanding the nature of effective communication in online communities, especially as they grow and include participants that are more diverse., particularly if one hope to automate some aspects of document generation in online communities. Below, we develop a theoretical perspective on documents that leads to hypotheses for this question and present the design and results from a research project that tests these hypotheses.

* Previous versions of this paper have been presented at HICSS 2012 and as a research-in-progress paper at ICIS 2012. The current version includes a greatly expanded data set (14 projects rather than 2) and extended discussion.

2 Theory elaboration and hypotheses

As a basis for our study, we draw on three bodies of work that describe documents and how they might span groups: genre theory, work on boundary objects and classification and studies of provenance. The first perspective focuses on the common stock of knowledge people bring to document production and use specifically. The second addresses how artifacts, such as documents, can bridge people with little shared points of reference. The third speaks to how people preserve the history and genealogy of documents to alleviate a lack of shared reference points and background knowledge.

Genre theory. Document genre has been defined as typified communicative action invoked in response to a recurrent situation [4-6]. People engage genres to accomplish social actions in particular situations, characterized by a particular purpose, content, form and participants in specific times and places. Identification of a document's genre makes the document more easily recognizable and understandable, reducing the effort required to convey meaning. For genres to be of aid in communication though, they must be shared by members of the community [7]. Thus, the utility of genres depends on symmetric access to knowledge among a group of people. Community members familiar with a genre are likely to know the expectations implied. Conversely, people with little access to the background knowledge of the community are not likely to know the genre and in turn bring few if any expectations about what purpose, content and form a document in that genre is likely to convey and what set of participants have produced and use it at what times and places. Therefore, to facilitate communication among people with asymmetric access to knowledge, rather than simply drawing on a genre, a document must explicitly state its purpose, form, content, appropriate participants and time and place of the communication. These considerations lead to three general hypotheses:

Hypothesis 1: A document shared among people with asymmetric knowledge is more likely to explicitly state its purpose.

Hypothesis 2: A document shared among people with asymmetric knowledge is more likely to explicate the context of use in regard to appropriate participants, times and places of its production and use.

Hypothesis 3: A document shared among people with asymmetric knowledge is more likely to explicate the form and content of its communication.

Boundary objects theory. To further refine these hypotheses, we turn to Star and Bowker's work on boundary objects [8, 9]. Actors from different communities, with few shared points of reference and little common stock of knowledge, have to manage the tension between their divergent viewpoints. Star introduces the concept of boundary object to explain how such heterogeneous communities maintain productive communication. We posit that documents shared among groups with asymmetric access to knowledge may serve as boundary objects. Star describes four types of boundary objects. The first type, repositories, refers to collections of documents and so is not relevant for our discussion of individual documents, but the remaining three types offer some helpful ideas.

Star defines *coincidence boundaries* as common objects that have the same boundaries but different internal content. They arise when work is distributed over a large-scale geographic area. Star points to the state of California as a coincidence boundary for the collaboration among citizen scientists and professional biologists at UC Berkeley. The result is that work in different sites and with different perspectives can be conducted autonomously while cooperating parties share a common spatial referent. Extending Star's thinking, we suggest that shared documents can specify temporal or participatory boundaries.

Ideal types are documents such as diagrams, atlases or other descriptions that do not accurately describe the details of any one locality, thing or activity but are rather vague and abstract. However, it is this very quality which makes it useful to people with different points of reference and stocks of knowledge. Such a document offers a good-enough road map to demarcate general elements, processes or organization of the shared context while suppressing distracting or conflicting details. This argument suggests that people with symmetric access to knowledge do not need to use ideal type documents to facilitate their communication and collaboration. However, people who share little common stock of knowledge and exist at the periphery of the community may need them to navigate and be able to read and use a document. Together, coincidence boundaries and ideal types allow us to further articulate our second hypothesis:

Hypothesis 2: A document shared among people with asymmetric knowledge is more likely to explicate the context of use:

A. By specifying the appropriate participants, times and places of its production and use
B. Through ideal types, such as diagrams, atlas, road maps, which demarcate the specific elements or organization of the shared work.
C. By demarcating the boundaries of the shared work. These can be geographical or other

specific boundaries about the scope of the work required by the project and the specific document.

Finally, *standardized forms* include labels and other forms that offer a uniform way to index communicative content and form. While Star highlights how standardized forms delete local uncertainties from the shared information, we note the converse, that the standardized forms in fact articulate a basic structure for the document's content and form. This articulation might not be a very detailed prescription, but nevertheless, it specifies the information needed for the particular communicative relationship supported by the document. However, people with intimate knowledge of the work at hand have less need for standardized forms. They know what they have to get done and what information will be relevant to the work at hand. Based on the notions of standardized forms, we can refine our third hypothesis:

Hypothesis 3: A document shared among people with asymmetric knowledge is more likely to explicate the form and content of its communication by:

A. Bringing regularity in semantics and objects covered by one document to the next, e.g., through standardized forms that offer a structured way to index communicative content.
B. Requiring the users to make more details of their work visible in their descriptions.

Provenance theory. Historical documents offer an extreme case with a highly asymmetric relationship between what a document prescribes and the background knowledge users bring to its use. Thus, archivists have long been concerned with how best to preserve background knowledge to contextualize the use and meaning of historical documents. In particular, archivists keep track of a document's provenance, i.e., where something comes from, who created it and what sources it draws from [10] to preserve some of the background knowledge that might otherwise be lost over time and space. The notion of provenance has recently been adopted by computer science to better understand how information with multiple sources flow from one application and file to another, constantly getting recycled, reworked, and repackaged [11]. People holding significant background knowledge about a community may simply need to know the author, title and date to position a document in its historical context and the evolution of a project. In contrast, newcomers most likely gain little from a simple audit trail common to most blogging, software development and document management systems. With little background knowledge, such members require more details to understand how a document fits into the larger work process. We suggest that documents shared among groups with asymmetric access to knowledge will include more details about the provenance of their communication, to explicate their own history and thus allow the audience to better contextualize their use. This leads us to our forth hypothesis:

Hypothesis 4: A document shared among people with asymmetric knowledge is more likely to explicate the provenance of the communication by referring to:

A. Where the communication comes from (e.g., the document creator, sources drawn from).
B. The genealogy of the communication and ideas (e.g., who has accessed/used the document and what they did with it).

3 Design of the research

Setting. To test the hypotheses developed above, we sought a setting in which we could observe documents being used across groups with different kinds and levels of shared background knowledge. We chose to study documents used in Free/Libre/Open Source Software (FLOSS) development. Key to our interest is the fact that most FLOSS projects are developed by virtual teams comprising professionals and users [12, 13]. These teams are close to purely virtual in that developers coordinate their activity primarily by means of a variety of computer-mediated communication (CMC) tools [14, 15]. As development proceeds, evidence of the processes and interactions between tasks and participants is left in repositories of documents, such as email lists, issue trackers, source code management systems and so on. These channels are characterized by documents of different genres that make up the FLOSS genre repertoire.

A particular interest is how the use of these varied documents depends on the patterns of relationships among members of a FLOSS team. Several authors have described successful FLOSS teams as having a hierarchical [16] or onion-like structure [17-20]. At the centre are the *core developers*, who contribute most of the code and oversee the design and evolution of the project. They are the only participants with the right to commit code. Surrounding the core are perhaps ten times as many *co-developers*. These individuals contribute sporadically by reviewing or modifying code or by contributing bug fixes. The co-developer group can be much larger than the core, because the required level of interaction is much lower. However, this lower level of interaction leads to the co-developers sharing

less background knowledge than the developers do. Surrounding the developers are the contributors or *active users*: a subset of users who use the latest releases and contribute bug reports or feature requests (but not code). Users interaction with developers is often channeled through a constrained set of genres. For example, questions and bug reports from users are valued, but only if presented in the "right way" [21]. Since they are not otherwise involved in development, we hypothesized that active users share even less background knowledge with developers.

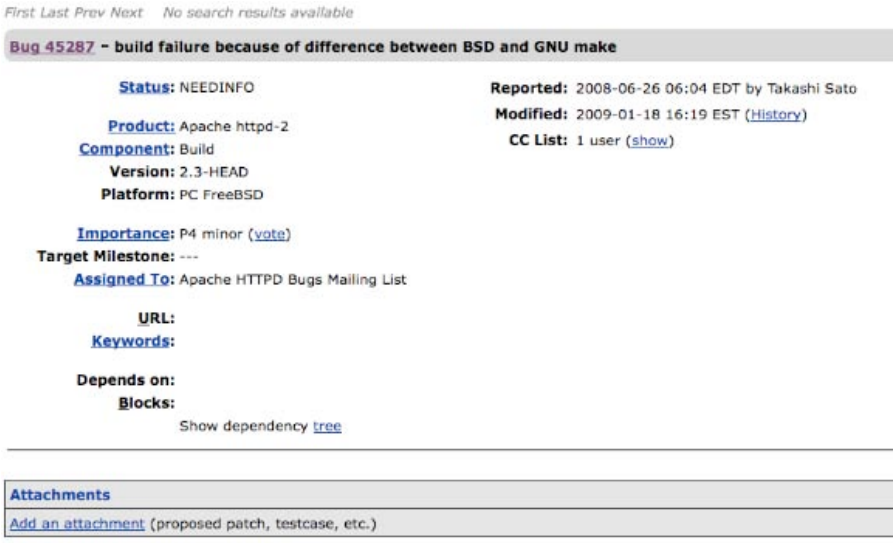


Figure 1. Example bug report from the Apache httpd project (from https://issues.apache.org/bugzilla/show_bug.cgi?id=45287).

Sample. FLOSS projects create a variety of documents, including code, documentation, feature requests, bug reports and so on. To emphasize our initial theoretical comparison, we chose three kinds of documents with audiences with different degrees of asymmetric knowledge, specifically *bug reports*, *source code patches,* and *commit messages*.

Bug reports (e.g., as shown in Figure 1) are used to report problems with a system. They can be created by both end users and developers, but are intended for developers, since developers are the only ones who can actually fix bugs. A bug report can include discussions between users and developers, e.g., if developers request more information to characterize the bug. As a result, this kind of document often spans two distinct communities (users and developers) who have little shared background knowledge. Projects often maintain a bug reporting system and provide explicit instructions about how and when to report a bug.

The second kind of document we considered was a source code patch. FLOSS projects grow through a process of incremental development as various developers contribute code that fixes a bug or implements a new feature. These code contributions are shared with the other developers in the project in the form a file representing the changes that were made to move from one version of the source code to another, called a patch file. These patches can be applied to the source code files maintained by other developers even if those developers have made some changes of their own, as long as the changes do not directly conflict. Patches are created by and used primarily by developers, meaning that this kind of document is shared amongst people with considerable shared background knowledge. (The size of a patch file precludes including one here as an illustration.)

The third kind of document we considered were source code commit messages (as shown in Figure 2). Most FLOSS projects use a source code control system (SCCS) to maintain the source code for a project. The SCCS keeps track of the various versions of the code and allows privileged developers (i.e., only the core developers) to store patches that are then shared with other developer. When a patch is added (or "committed") to the SCCS, it is usual for the core developer to write a short log message describing the

Figure 2. Example source code control system check in message from the MythTV project (from http://svn.mythtv.org/trac/changeset/24896).

change. These exchanges are shared among developers, i.e., people with symmetric access to knowledge. Bug reporting systems can be made to interoperate with the SCCS so the commit message for changes that fix bugs can be linked to the bug report and vice versa.

We collected examples of bug reports, source code patches and commit messages from different FLOSS projects. However, to test our hypotheses, it was necessary to also look for explicit instructions or other discussions of how bug reports, patches and commit messages should be created or used. Figure 3 shows an example of instructions for creating a bug report; Figure 4, for creating a commit message. We collected instructions by searching the project websites for relevant documents. Two coders did the search; the choice of documents was confirmed through weekly discussion with the authors.

cURL ▸ Docs ▸ **Bug Report**

How, Why, And Where to Report Bugs

Of course there are bugs in curl and libcurl. Some are known, but most of them remain unknown to us until you let us know. We depend on bug reports from the users to find the problems. We are not likely to be able to fix bugs if we don't get to know about their existance first. Bugs tend to get fixed within a short while after we've been notified (at least when we agree about it being a bug and not a feature)!

Known Bugs

Some bugs are known to already exist. We try to keep track of them on a separate KNOWN_BUGS document.

How To Report

If you can't fix a bug yourself, file an *as detailed a report as possible* in the bug tracking system **or** mail it to a suitable mailing list. Bugs in or improvements to libcurl are best posted to the curl-library list, while bugs in or with the curl tool can be posted to the curl-users list.

If you opt to use one of the mailing lists, notice that you need to be subscribed first to get the mail through to the list!

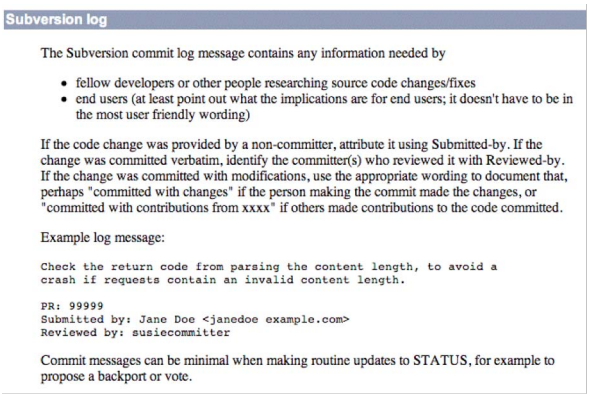
What To Report

When reporting a bug, try to include information that will help us understand what's wrong, what's expected to happen and how to repeat it. You should supply:

- your operating system's name and version number
- what version of curl you're using (curl -V is fine)
- what URL you were working with
- everything else you think might matter

Figure 3. Instructions for reporting a bug in curl (from http://curl.haxx.se/docs/bugs.html).

We chose documents by identifying all relevant documents from a purposeful sample of FLOSS projects. We decided to use purposeful sampling for three reasons. First, there is no complete sampling frame for FLOSS projects to support random sampling. Researchers often use forges such as SourceForge as a basis for sampling, but there are many different forges, and many interesting projects use their own infrastructure rather than a forge. Second, and more important, given the skewed distribution of project sizes, a random sample would have a large number of small projects and few if any larger projects. However, small projects would be less interesting for our purpose, as there would be less opportunity for communication across knowledge boundaries. Finally, for our initial goal of examining the validity of our hypotheses, it did not seem critical to be able to generalize to the entire population of FLOSS projects, which random sampling would support.



Subversion log

The Subversion commit log message contains any information needed by

- fellow developers or other people researching source code changes/fixes
- end users (at least point out what the implications are for end users; it doesn't have to be in the most user friendly wording)

If the code change was provided by a non-committer, attribute it using Submitted-by. If the change was committed verbatim, identify the committer(s) who reviewed it with Reviewed-by. If the change was committed with modifications, use the appropriate wording to document that, perhaps "committed with changes" if the person making the commit made the changes, or "committed with contributions from xxxx" if others made contributions to the code committed.

Example log message:

```
Check the return code from parsing the content length, to avoid a
crash if requests contain an invalid content length.

PR: 99999
Submitted by: Jane Doe <janedoe example.com>
Reviewed by: susiecommitter
```

Commit messages can be minimal when making routine updates to STATUS, for example to propose a backport or vote.

Figure 4. Example instructions for SCCS commit messages (from http://httpd.apache.org/dev/guidelines.html).

Given these considerations, projects were purposively selected to achieve variation on size, formality of organization (i.e., community-based vs. with a foundation or corporate involvement) and target audience (e.g., developer tools or code libraries vs. end-user programs). To improve comparability, we selected several projects from the same general domain, namely web services, software development and multimedia. Specifically, we examined:

1. WebKit (browser engine)
2. gcc (compiler)
3. ncurses (programming library)
4. Boost libraries
5. FFMPEG (digital video library and tool)
6. cURL (command line web tool)
7. wget (command line web tool)
8. Apache httpd (web server)
9. phpMyAdmin (web-based database administration tool)
10. VirtualBox (PC emulator)
11. OpenOffice (office suite)
12. Firefox (web browser)
13. MythTV (digital TV recorder)
14. Pidgin (IM client)

From the 14 project websites, we collected a total of 103 documents for analysis.

Coding. To test our hypotheses, we developed a coding system for the various document characteristics in the hypotheses (e.g., explicit statement of purpose or standardized forms). We started with the definitions of each of the concepts from the two theoretical sources. We then inductively coded a small set of documents to refine these definitions and to develop a coding scheme. We then applied this coding system to the collected documents. Coding was done in the NVivio program by two coders. Disagreements in coding between the coders were discussed to consensus; issues that could not be resolved were discussed at regular meetings among the coders and the authors to arrive at an agreed set of codes. The resulting coded document collection was then analyzed quantitatively and qualitatively.

4 Results

The section falls in two parts. First, we compare bug reports spanning active users and core developers (i.e., asymmetric access to knowledge) with commit messages shared among core developers only (i.e., symmetric access to knowledge). We provide illustrative examples of how these two types of document are consistent with our hypothesis developed above. Second, we compare bug reports versus source code patch-related documents. Our more detailed analysis reveal that while bug reports span active users, co-developers and core developers (i.e., asymmetric access to knowledge), patches involve not only core developers but also co-developers (i.e., asymmetric access to knowledge). The analysis extends our hypothesis and provides further insights into what it means to "write appropriately."

4.1 Active users and core developers

We first compared the documents associated with active users (i.e., contributors submitting bug reports) and core developers (i.e., those who can commit the patches to the SCCS). Specifically, we compared the instructions given for creating and using bug reports to instructions for commit messages. The difference was striking: across the 14 projects we reviewed, we found fewer than 10 documents explicitly addressing core developers with instructions on how to commit patches; several projects did not have any instructions for core developers and the communication around committing code. For projects that did have documentation, it often focused on security-related issues going beyond day-to-day code commits. In contrast, we found more than 50 documents across the 14 projects detailing how active users should communicate about newly-found bugs.

In the following, we provide illustrative examples of how the creation and use of bug reports and SCCS commit messages in the projects examined are consistent with the hypothesis developed above.

Hypothesis 1: A document shared among people with asymmetric knowledge is more likely to explicitly state its purpose.

Examining the instructions for filing a bug report for the curl project (Figure 3), we find that the purpose of bug reports is clearly stated: to let developers know about problems so they can fix them. The instruction pages for other projects are similarly explicit. By contrast, projects are less specific about the purpose of SCCS commits. When there are instruction pages for using the SCCS, e.g., in guidelines for the development process (Figure 4), these do not clearly state the purpose; rather, it seems to be assumed that the creator will know what information would be needed by other developers.

Hypothesis 2: A document shared among people with asymmetric knowledge is more likely to explicate the context of use:

A. By specifying the appropriate participants, times and places of production and use

Consistent with this hypothesis, we find that bug report instructions seem somewhat more explicit about participants, time and places of production. In part, these expectations are enforced by the technology, as systems enforce roles with particular privileges on documents, e.g., who can create, update, edit or dispose of certain kinds of documents. Again, the instructions for the commit messages specify less.

B. Through ideal types, such as diagrams, atlas, road maps, which demarcate the specific elements or organization of the shared work?

Comparison of the instructions for the two types of documents seems consistent with the hypothesis. The instructions for bug reports list what the creator and receiver of a document have to do in order to demarcate the shared work. By contrast, there is little discussion of what someone might do when reading a commit message, again reflecting an assumed shared understanding of the process.

C. By demarcating the boundaries of the shared work.

The instructions for bug reporting include descriptions of what is in scope and what is out of scope. For example, while a complex system such as MythTV is built from many components, users rarely perceive these internal components of a system, and so consider all bugs as originating with the application. Similarly, the designers of a system may have specific use cases in mind for the project, and may not be

interested in expanding its functionality beyond those. Therefore, bug-reporting instructions need to explain how to localize a bug and caveats about what kinds of bugs can be fixed and what kinds of new features will be considered. In contrast, the description of the commit message does not specify such boundaries.

Hypothesis 3: A document shared among people with asymmetric knowledge is more likely to explicate the form and content of its communication by:

A. Bringing regularity in semantics and objects covered by one document to the next, e.g., through standardized forms that offer structured way to index communicative content.

As expected, a bug report document includes a number of structured fields. The number of fields is greatest for the most institutionalized project, Apache, which uses the bugzilla bug tracking system. Interestingly, the cURL project does not require a form but encourages submissions by email, asking only for some basic information. This difference may indicate the assumption that users of cURL are sophisticated enough to submit good bug reports without explicit guidance, since cURL is a command-line tool. By contrast, a SCCS commit message is just a plaintext field; the message provided can be long or short. Some projects do suggest including particular fields, e.g., a reference to the bug report that the patch fixes, but these are not required. Furthermore, exactly how the patch should be described is left to the developer.

B. Requiring users to make more details of their work visible in their descriptions.

The bug report document includes in addition to the fields describing the bug, comments made by developers or other users on the bug. These are frequently used to keep track of work status. Commit messages are also used as a way to indicate the work done, though this is often done in only a summary fashion and the messages can be quite cryptic.

Hypothesis 4: A document shared among people with asymmetric knowledge is more likely to explicate the provenance of the communication by referring to:

A. Where the communication comes from (e.g., the document creator, sources drawn from).

Somewhat consistent with the hypothesis, bug reports go into detail about the origin of the document. As illustrated in Figure 3, a bug report specifies details such as the operating system and version, and what sources the author consulted before composing the document. In addition, the author must register in the system, allowing others to track their documents. Patches committed to the SCCS also articulate the creator of the document, but the commit messages do not provide any further detail about the sources from which the author draws.

B. The genealogy of the communication and ideas (e.g., who has accessed/used the document and what they did with it).

Consistent with the hypothesis, we find that bug reports offer more details about their genealogy compared to commit messages. As illustrated by the Apache bug report in Figure 1, the document includes the history of the communication, developers cc'ed and its dependencies. In contrast, the committed patch (Figure 2) offer minimal information about the history of the communication, though the example in Figure 4 does hint to the document's genealogy by specifying who submitted the bug and who revised it.

4.2 Active users and co-developers

We next compared the documents associated with co-developers (i.e., contributors submitting patches) and core developers (i.e., those who can commit code). Specifically, we compared the instructions given for creating and using bug reports and patches. .A closer look at patch-related documents revealed that a majority of them addressed newcomers to the FLOSS project and not core developers or even co-developers. Typical documents would specify the communication process involved in patch creation and submission as a way to help a newcomer become involved in the endeavor. Many documents discussed both bug reporting and patches as a way to become involved, where submitting bugs was a first step followed by the second step, creating a patch. Some projects explicitly suggested that if you wanted to see your bug fixed you might as well do it yourself. Based on these findings we distinguished core developers with commit rights from co-developers who contribute code but do not have commit rights.

Looking across all the documents addressing both bug reporting (i.e., active users) and patches (i.e., co-developers) we found some sub-elements of our four hypotheses covered in more documents than others. For **hypothesis 1**, we found about the same number of documents explicating the purpose of bug reports (7) and patch submissions (6). For, **hypothesis 2** we found many more reference to the *place* for bug report communication (25) compared to other aspects of the communicative context (i.e., timing (0), participants (4), ideal types (1), and boundaries (9). In contrast, we found about equal number of references to the boundaries (9), participants, place (10), and ideal types (10) for communication about patches. When it comes to **hypothesis 3,** both bug report and patch-related documents explicated content expectations with a high frequency compared to expectations about format, visibility and the use of standardized forms. For instance, we found 56 descriptions of content

expectations for bug reports compared to 12 format descriptions, 11 standardized forms. Regarding patches, the numbers were 24 documents explicating content compared to 8 explicating format expectations and only 2 standardized forms. Overall content expectations were the most frequently explicated expectation. Finally, for **hypothesis 4**, documents explicating provenance were less frequent in particular for bug report (4). For patches we found 12 references to provenance.

5 Discussion

Our analysis compared two pairs of document types: 1) bug reports versus commit messages and 2) bug reports versus patches. First, we compared bug reports, which span active users, co-developers and core developers (i.e., high degree of asymmetric access to knowledge) with commit messages involving only core developers (i.e., symmetric access to knowledge). Through analysis of the 14 FLOSS projects, we found that documents supporting collaborators with asymmetric knowledge do seem to explicate their own use in more detail. Bug reports appear to do so by articulating or prescribing their own 1) purpose, 2) context of use and 3) content and form and 4) provenance in greater detail than commit messages used by core community members with symmetric access to project knowledge.

Second, we compared bug reports (active users, co-developers, core developers) with patches (co-developers and core developers). In other words, these two sects of documents span participants with asymmetric access to knowledge; however, one can expect participants involved in source code patches to share more background knowledge than bug report participants. The comparison provided a more nuanced picture of what gets explicated among people with different types of asymmetric access to knowledge. We will discuss this second point in more detail.

5.1 The prominence of process

The coding revealed communication patterns not predicted in our initial hypothesis: A significant number of documents explicated the *process* of bug report and patch related communication. We found approximately 60 references to the bug report communication process and 70 descriptions of the patch communication process. Here, it is worth noticing that the ideal types found for patches (10) all depicted the communication process related to patch creation and submission.

Initially, we had not expected that FLOSS participants would explicate the communication process itself. Moreover, it emerged as the most frequently explicated expectation among people with asymmetric knowledge. In retrospect though, this focus makes sense theoretically. Both contemporary genre and boundary object literatures build on a practice theory foundation that stipulates that social structures and phenomena only exist as they get produced and reproduced in people's everyday social practices [22, 23]. We also note that we defined genres as "typified communicative actions invoked in response to recurrent situations" [5]. Consistent with both perspective, it is understandable that FLOSS core developers take the time to explicate the sequential ordering of FLOSS communication activities.

5.2 Context versus content and format

By comparing documents related to bug reports versus patches we notice a difference in how frequently documents explicate the context of communication (Hypothesis 2) compared to its content and form (Hypothesis 3). Documents targeting active users submitting bug reports tend to spend more time explicating content and form (Hypothesis 3) compared to patch related documents. The reverse is true for Hypothesis 2. In the results, we noticed comparatively more documents explicating the context of use associated with source code communication, with the exception of specifications of where communication takes place. The latter type of reference is highly prevalent among bug report documents.

One explanation might be that active users can submit bug reports by simply knowing where to do so, what content to provide and in what format. Core developers do not need to explicate their expectations for the communication further. The system automatically records provenance relevant information. Active users are prompted or even required to provide basic information about a bug through the standardized form that makes up the bug reporting system. Submitting a patch is more involved, unpredictable and requires a better understanding of the context of communication (Hypothesis 2). Developers cannot effectively engage in this type of communication without understanding where it takes place, who are involved, the boundaries of that work, and ideal representations of the communication process.

As participants move from a peripheral position as mere active users of bug reports to co-developers submitting source code patches the knowledge they require about communication practices changes. Knowing where to go and what to communicate about and in what format is the first step of a newcomer. Being more specific about the context of communication is the next step as one move further

toward the center of the FLOSS community. What stays constant in those early phases is a need to explicate the process of communication. By the time you become a core developer, we hypothesize, you know the ropes and you only need to explicate communicative expectations in unusual cases such as those relating to security breaches.

5.3 Beyond FLOSS

The approach developed in this paper contributes the general understanding of documents in online communities. We hope to extend the research beyond FLOSS teams, for example to online communities such as the Wikipedia community. Wikipedia does have an inner group that has intimate knowledge of the system and how the organization behind it works, and a larger peripheral group of participants with a much smaller stock of background knowledge. It would be interesting to explore why Wikipedia does not seem to require documents comparable to bug reports that bridge groups with asymmetric access to knowledge. Research could search for and describe other kinds of documents that bridge between these groups. It could be that there is no need to account for one's work in Wikipedia, as any member can commit a change to the core text. In contrast, only core developers can change the code in open source projects, thus requiring many would-be active users to rely on communication with others to accomplish their work. Power relations and access to execute actions may play a role in how much documents prescribe their use in various situations.

The present research also contributes to theory development by questioning some of the existing assumptions associated with document centric research. First, genre studies to date have tended to focus on groups with symmetric access to genre expectations. Future research could explore how genre expectations develop and are shared among people with asymmetric access to genre expectations. In short, how do genres work across various discourse community boundaries? One possible outcome is that documents spanning different asymmetries in background knowledge need to explicate different parts of the communication. Second, the inter-dependencies of boundary object and provenance theory calls for further exploration. In other words, our preliminary findings suggest that effective boundary object explicate their own provenance, i.e., go into some detail about their own history, allowing users of diverse communities to track the history of the object across the involved communities.

Finally, the research contributes to system design for online communities and digital collaborations. In particular, the extensive use of standardized forms for bug reports may provide some interesting insights. In healthcare, for instance, one finds a push for more standardized record keeping and information sharing. If it is mainly groups with asymmetric access to knowledge who benefit from using standardized forms, one may assume that resistance to standardized systems comes from members of groups with relative symmetric access to knowledge in their use of healthcare information systems. Using a standardized form that require high regularity in semantics and objects and great detail may seem like a waste of time for someone with a large stock of background knowledge in the specific area. A detailed understanding of what characterize documents that support collaborators with symmetric versus asymmetric access to knowledge could help create systems that tailor content to specific user groups.

6 Conclusion

Online communities and digital collaborations bring together people with various access to and understanding of the work at hand. Yet, how do documents serve diverse users, many of whom are literally not on the same page? How does one write appropriately? The present research contributes to both scholarship and practice. First, the paper develops a framework based on three previously unrelated bodies of literature that characterize documents serving collaborators with asymmetric access to knowledge versus documents supporting those with symmetric knowledge. Drawing on document-centric approaches, we hypothesize that documents supporting asymmetric groups are likely to be more prescriptive and explicate their own use compared to documents supporting symmetric groups. Second, our work suggests that practitioners of online communities would benefit from explicitly considering 1) how much access to knowledge various participants hold, and 2) how prescriptive and explicit documents have to be to support those various groups. Systematic knowledge of what such document variations becomes essential for system developers hoping to support heterogeneous online communities.

7 References

[1] C. D. Cramton, "The mutual knowledge problem and its consequences for dispersed collaboration," Organization Science, vol. 12, pp. 346–371, 2001.

[2] J. A. Cannon-Bowers and E. Salas, "Shared mental models in expert decision making," in Individual and Group Decision Making, N. J. Castellan, Ed., ed Hillsdale, NJ: Lawrence Erlbaum Associates, 1993, pp. 221–246.

[3] H. H. Clark and S. E. Brennan, "Grounding in communication," in Perspectives on Socially Shared Cognition, L. B. Resnick, et al., Eds., ed Washington, DC.: American Psychological Association., 1991, pp. 127--149.
[4] C. Bazerman, "System of genres and the enactment of social intentions," in Genre and the New Rhetoric, A. Freedman and P. Medway, Eds., ed London: Taylor & Francis, 1995, pp. 79-104.
[5] W. J. Orlikowski and J. Yates, "Genre repertoire: The structuring of communicative practices in organizations," Administrative Science Quarterly, vol. 39, pp. 541-574, 1994.
[6] K. Crowston and B. H. Kwasnik, "Can document-genre metadata improve information access to large digital collections?," Library Trends, vol. 52, pp. 345-361, FAL 2003.
[7] J. Swales, Genre Analysis. Cambridge: Cambridge University Press, 1990.
[8] S. L. Star and J. R. Griesemer, "Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology 1907-39," Social Studies of Science, vol. 19, pp. 387-420, 1989.
[9] G. C. Bowker and S. L. Star, Sorting Things Out: Classification and its consequences. Cambridge: MIT Press, 1999.
[10] Y. L. Simmhan, et al., "A survey of data provenance in e-science," Sigmod Record, vol. 34, pp. 31–36, Sep 2005.
[11] H. Lonsdale, et al., "Cutting and pasting up: "Documents" and provenance in a complex work environment," presented at the Hawai'i International Conference on System Science (HICSS-43), Kauai, HI, 2010.
[12] E. A. von Hippel, "Innovation by user communities: Learning from open-source software," Sloan Management Review, vol. 42, pp. 82–86, Summer 2001.
[13] E. A. von Hippel and G. von Krogh, "Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science," Organization Science, vol. 14, pp. 209–213, 2003.
[14] E. S. Raymond, "The cathedral and the bazaar," First Monday, vol. 3, 1998.
[15] P. Wayner, Free For All. New York: HarperCollins, 2000.
[16] W. Scacchi, "Free/Open Source Software Development Practices in the Computer Game Community," IEEE Software, vol. 21, pp. 56–66, 2004.
[17] A. Cox. (1998, 22 March). Cathedrals, Bazaars and the Town Council. Available: http://slashdot.org/features/98/10/13/1423253.shtml
[18] C. Gacek and B. Arief, "The many meanings of Open Source," IEEE Software, vol. 21, pp. 34–40, 2004.
[19] J. Y. Moon and L. S. Sproull, "Essence of distributed work: The case of Linux kernel," First Monday, vol. 5, 2000.
[20] M. A. Rossi, "Decoding the “Free/Open Source (F/OSS) Software Puzzle”: A survey of theoretical and empirical contributions," Università degli Studi di Siena, Dipartimento Di Economia Politica, Working paper 424, 2004.
[21] E. S. Raymond and R. Moen. (2006). How to ask questions the smart way. Available: http://catb.org/~esr/faqs/smart-questions.html
[22] M. de Certeau, The Practice of Everyday Life. Berkeley: University of California Press, 1984.
[23] C. Østerlund and P. Carlile, "Relations in Practice: Sorting through practice theories on knowledge sharing in complex organizations," The Information Society, vol. 21, pp. 91-107, 2005.