2014 47th Hawaii International Conference on System Science

Forge++: The Changing Landscape of FLOSS Development

Megan Squire
Elon University
msquire@elon.edu

Abstract

Software forges are centralized online systems that provide useful tools to help distributed development teams work together, especially in free, libre, and open source software (FLOSS). Forge-provided tools may include web space, version control systems, mailing lists and communication forums, bug tracking systems, file downloads, wikis, and the like. Empirical software engineering researchers can mine the artifacts from these tools to better understand how FLOSS is made. As the landscape of distributed software development has grown and changed, the tools needed to make FLOSS have changed as well. There are three newer tools at the center of FLOSS development today: distributed version control based forges (like Github), programmer question-and-answer communities (like Stack Overflow), and pastebin tools (like Gist or Pastebin.com). These tools are extending and changing the toolset used for FLOSS development, and redefining what a software forge looks like. The main contributions of this paper are to describe each of these tools, to identify the data and artifacts available for mining from these tools, and to outline some of the ways researchers can use these artifacts to continue to understand how FLOSS is made.

1. Introduction

First emerging in the late 1990s out of the pairing of web hosting and free, libre, and open source software (FLOSS) development, software forges are centralized online systems that provide useful tools to help distributed development teams work together. Distributed development teams may find the tools provided by a software forge to be convenient to use, removing some of the administrative overhead of configuring and maintaining all those separate software packages. Instead, the development team can outsource those tasks to a software forge, and focus its limited time on writing code.

Tools provided by software forges can include version control systems, file download tracking, mailing lists and communication forums, bug tracking systems, web hosting space, etc. Many software forges were originally created with the needs of decentralized and highly-transparent FLOSS teams in mind. Early forges included Sourceforge and Savannah, later joined by special-purpose forges such as Rubyforge and Launchpad, and even later by the very large players such as Google Code and CodePlex (Microsoft). The work done by [1] shows a timeline of forge creation as well as a matrix of features provided by each of 24 separate forges during the period 1999-2011.

Public software forges are appealing for FLOSS studies because having access to the artifacts and metadata from lots of projects, all formatted similarly, makes them much easier to compare to one another (and to themselves over time). Software artifacts, in the form of source code, bug reports, communication archives (e.g. mailing list messages), and project metadata are available from the FLOSS software forges, and have been the basis for hundreds of FLOSS studies and the successful *Mining Software Repositories* working conference [2], now in its 10th year.

Given this history of forges being used for FLOSS research, the purpose of this paper is to describe three important developments that have occurred in the way FLOSS is made that will impact the use of forges and their importance going forward. First is the rise of distributed version control systems (DVCS) like git and related code repositories like Github. Second is the growth of community-oriented question-and-answer sites like Stack Overflow. Third is the increasing usage of pastebin tools like Pastebin.com and Github's Gist.

Each of these tools has changed how FLOSS is developed, and each has artifacts of its own that may be able to be mined for understanding about the software development process. Sections Two-Four will describe each of these three tools in turn. Each section will describe the purpose of the tool, the artifacts available for mining, and some ways to collect the interesting data. Because the focus of this paper is on FLOSS development, each section includes a discussion of site-specific issues with regard to "openness" (including both code and content licensing).

978-1-4799-2504-9/14 $31.00 © 2014 IEEE
DOI 10.1109/HICSS.2014.405

2. Github

Github (github.com) was started in 2009 as a hosted software forge for projects that use the Git distributed version control system, or DVCS. VCS-specific forges are not new; for example, Launchpad (launchpad.net) was begun in 2004 to host projects using the GNU Bazaar distributed VCS. When git was released, web sites like Github and Gitorious (gitorious.org) were created to serve as wrappers; they helped mask the complexity of git and ease the transition to this new tool.

In addition to version control, Github also has just a few other capabilities of a traditional software forge, namely issue tracking and user profiles. Like other software forges, Github has users, but unlike other forges, Github projects (called "repositories" or "repos") only exist when users create them or fork them from others. As such, they are always listed underneath a particular user. Every public repo belongs to a given user, and can be forked and changed by any other user, at which point the changes can be accepted for use by the original user or not. As a DVCS, git is quite different from the VCSs that came before it, and can be confusing to new users. However Github's web interface lowers the barrier to entry for a transition to git. In April 2013, the number of users registered on Github had risen to 3.5 million, with 6 million repositories. To help keep track of all these users and repos, Github's social features include the ability to *follow* the activity of other users, the ability to *create* teams of users (called organizations), and the ability to *watch* a project/repository. Github also offers a pastebin type of service called Gist (pastebins are discussed later in Section 4 of this paper) and Github Pages, which turn repo text into viewable web pages. Github does not have a few of the features found in many traditional software forges, such as the mailing list management software, forums, alternate VCS systems, feature request systems, or task lists, but it does have code reviews (in the form of commented pull requests), wikis, and issue tracking. And with the release of Gist and Pages, Github does seem to be inching toward adding more features, as long as those features can be conceived as a repository on the back-end.

2.1. Github and "openness"

One of the important ideological ways that Github differs from prior software forges is that a user can start a public repo without specifying a software license. Sourceforge, Google Code, and many other FLOSS forges (large and small) that came before Github required that software be assigned a recognizable open source license at the project level, at the time the project was created. This license was intended cover any project code, including any code or downloads that were (accidentally or otherwise) released without a license. Some forges were no-cost when used by FLOSS projects, but other forges disallowed any non-FLOSS projects entirely. (See [1] for an accounting of FLOSS requirements and features by forge.)

On Github, this is not the case. Licenses can be uploaded as files, or included inside source code files that are then uploaded, but these actions are optional. On Github a user is not required to select a FLOSS license to apply to an entire project. In fact, a 2013 report by the Software Freedom Law Center [3] asserted that only 15% of projects on Github included a recognizable FLOSS license in their top-level directories. Furthermore, of that 15%, the vast majority used permissive, non-viral (or non-reciprocal) licenses like MIT or BSD. In addition, many projects made up their own licenses, or used license terms that were self-contradictory. The Github Terms of Service (ToS) includes no default license [4], so copyright provisions are in effect. The result of this is that unlicensed code is not legal to copy; using unlicensed code found on Github (for example by forking a project using the Github one-click interface) is a breach of copyright. After Simon Phipps suggested several fixes for this problem, the Github VP of Marketing, Brian Doll, acknowledged that licensing may have to be part of the project creation process in the future [5].

2.2. Github data

We have shown that Github differs in important ways from the forges that came before it: in functionality, in size, and perhaps in ideology. Nonetheless, of the three tools mentioned in this paper, Github is the most like a traditional software forge in its artifacts and metadata.

Software artifacts and metadata about Github are plentiful and easily available, starting with the API available from Github itself [6]. Requests to the developer API are limited to 60 per hour for unauthenticated users, and 5000 per hour for authenticated users. The API allows access to Github events, projects (repositories), and users, as well as the issues, pull requests, commits, blobs (files), and trees (directory structure) inside of each repository. Githubarchive [7] uses the public Github web page about its API to create and archive a timeline of public events, starting in February of 2012. The GHTorrent toolset [8] uses a distributed web of clients to make the maximum requests to the Github API itself, in order to

construct a mirrored collection of events, repositories, and users. GHTorrent stores the data in a MySQL server and provides a web interface for querying, which we will use in Section 2.2.1 below.

Foreshadowing the rise of git and later Github, Bird, *et al.* [9] issued a warning to researchers about the differences between mining a distributed version control system like git and the traditional VCS (specifically, Subversion). They present lessons learned through the process of analyzing the source code in a DVCS, and while they do not specifically mention Github, the paper is still applicable to the source code artifacts there.

A number of software engineering research studies have already used the artifacts from Github in interesting ways, see [10][11][12][13]. These studies are part of the research community's history of making tools to study the "usual data sources" [14] of the software development process. For example in [10] the authors combine a social graph of Github users with their *commit* and *follow* actions. They then use the geographic data in the user profile to geolocate the users and make inferences about influence in the community. (We should note that these authors also built a crawling infrastructure to collect additional user profile data from what they uncovered with the Github-provided social graph.)

2.2.1. Github example Here we explore the Github metadata to get a sense of what is going on with the site and how it is being used. GHTorrent [8] is probably the easiest option for doing so, as of this writing. We connect to GHTorrent and issue some basic queries using a 'guest' login.

What are the top 10 programming languages in use on Github, and how many of each project is using each language?

```
SELECT language, count(*)
FROM projects
GROUP BY 1
ORDER BY 2 DESC
LIMIT 10;
```

Results:

language	count(*)
JavaScript	796944
NULL	563479
Ruby	479615
Java	361543
Python	314553
PHP	304015
C	169361
Objective-C	149638
C++	130660
Shell	85826

The programming language count is an appropriate first query, given our focus on software forges. In older, first-generation forges (e.g. Sourceforge in the mid-2000s), project leaders were expected to self-classify their project according to a "trove" of keywords, which included each programming language (and operating systems, licenses, intended audience, and a whole host of other pre-designated categories). Perhaps recognizing that the main weakness to this self-categorization system is the need for project owners to constantly re-classify themselves in order to keep their trove categories up to date, Github instead automatically reads the source files uploaded to the project site and figures out the percentage of the project devoted to each programming language. On the page for each repository, a horizontal stacked single-bar graph, (as shown in Figure 1) describes the current language composition for any given project.

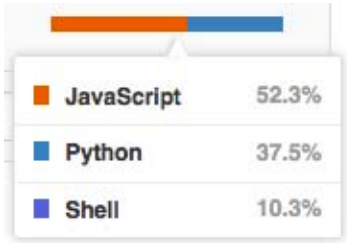

Figure 1. Github automatic programming language calculation

This automatic language classification is reflected in the above GHTorrent query through the programming language listing in the projects table. For example, the project shown in Figure 1 is in the 'projects' table next to its languages: JavaScript, Python, and Shell. When we count projects using these languages, the percentage of the codebase ("52.3% Javascript") was not considered. Looking at the query results, we see that Javascript is used in 796k projects, but each one could (in theory) be using Javascript for only 1% of its codebase. We further suspect that "NULL" could be the second-most frequent language because of README files (they exist in many projects but are not written in any programming language).

3. Stack Overflow

Stack Overflow is a question-and-answer web site started in 2008 by software developers Jeff Atwood and Joel Spolsky. The salient features of Stack Overflow are that anyone can post questions, answer questions, edit questions or answers, and users gain reputation points and badges for doing combinations of those things. As such, Stack Overflow is not a software forge. However, it does include an enormous amount

of source code, code examples, and solutions to problems in multiple programming languages and development frameworks. As of June 2013, Stack Overflow contains 5.3 million questions with 10 million answers [15] from more than 2 million users. A study tracking developer behavior in Android estimated that software developers may be getting up to 50% of their technical documentation on Stack Overflow [16].

Even though Stack Overflow is not a software forge, it is being used in some forge-like ways. One example is how the Stack Overflow tagging system is used by development teams (both FLOSS and non-FLOSS) looking to outsource their developer support forums. (Tags are optional keywords that a user can add to his or her question to ensure that the question is classified quickly, and that it is easy to find by readers.) Some software development teams have decided that they like Stack Overflow tagged questions better for providing developer support than their own homegrown support systems such as online forums. For example, there is a Stack Overflow tag for the Google BigQuery project. There are currently 473 questions tagged with "google-bigquery". The web site for the Google BigQuery project [17] gives advice to developers to use the Stack Overflow tag for technical support with their API:

> *We support developers using the Google BigQuery API on Stack Overflow. Google engineers monitor and answer question with the tag google-bigquery. Please use this tag when asking questions. We aim to answer all questions in reasonable time.*

The final posting [18] on the old BigQuery Google Group for developer API support explains the rationale for moving support to Stack Overflow on May 1, 2012:

> *We are moving technical discussion to Stack Overflow because we think this will improve developer support, increase the speed that questions get answered, and improve the quality of answers.*

Numerous other commercial projects have also gone this route, including Facebook, Shopify, Youtube, SoundCloud, and Foursquare.

Stack Overflow also has hundreds of tags for many popular FLOSS platforms and tools: from languages (python, R, ruby), to database tools (MySQL, CouchDB, Hadoop), to operating systems (linux), and development tools (svn, git). But because Stack Overflow charges money to sponsor a tag to use it for providing official support (as with the Google tags or the Facebook tag), FLOSS projects without a budget are probably not replacing their own forge-based or homegrown developer support systems. Instead, Stack Overflow represents another communication channel for developers to collaborate and share knowledge. In the next section we discuss more about the relationship between Stack Overflow and the FLOSS ideology.

3.1. Stack Overflow and "openness"

We consider the "openness" of Stack Overflow in two ways: through its licensing and through community editing. First, in terms of licensing, Stack Exchange (and thus its sub-site Stack Overflow) has a default code license in place to govern sharing of any materials found on the site. This is in contrast to Github, which has no default license. The Stack Exchange Terms of Service [19] state:

> *You agree that all Subscriber Content that You contribute to the Network is perpetually and irrevocably licensed to Stack Exchange under the Creative Commons Attribution Share Alike license.*

Thus the CC-by-SA license [20] allows sharing and remixing of source code, for both commercial and non-commercial purposes. It also contains provisions requiring attribution of the original author in the new shared or remixed work, and that the new work must contain the same CC-by-SA license or similar. Users can also supersede this CC-by-SA license by attaching a different license to their own code at the time that it is posted.

Even though there is a default license on Stack Overflow, developers wanting to use code found on this site will still have a few issues to consider. Most critically, because the CC-by-SA license requires derivative works to carry the same license (or similar), developers will need to be careful of whether they are actually able to use that license on the code. Many projects already have a license (or license family) that they are committed to using, and it may be incompatible with CC-by-SA. Second, the requirement of CC-by-SA to attribute the source of the code is going to be challenging on a site with a potential for multiple editors per posting. Indeed, Stack Overflow has issued clarifications [21] for how to manage this situation. Editing, however, is the other important feature of Stack Overflow that keeps it an "open content" site. Stack Overflow encourages editing of questions and answers by other users in order to "make the post substantially better" [22].

In the next section we describe what types of Stack Overflow data are available and how to use them. As our motivating examples we first look at the role of source code on Stack Overflow, and we then attempt to show how much post editing actually goes on in the Stack Overflow community.

3.2. Stack Overflow data

Getting data from Stack Overflow is quite straightforward. First, there is an online SQL-based "Data Explorer" [15] which allows anyone to run arbitrary queries against the site data, to save, name and share queries, and to discuss queries with other users. For more in-depth analysis or offline processing, Stack Exchange also provides occasional flat file dumps of their entire (anonymized) data store suitable for import into a SQL database [23].

3.2.1. Data about Source Code Using the flat file dump from August 2012, we were able to calculate a few interesting metrics about the pervasiveness of source code in Stack Overflow postings. First we separated each post into plaintext and code. We then calculated the length (in characters) of each of these parts, and calculated a code-text-ratio for each post[1], storing the results in a new table called new_posts_meta.

(1) How many Stack Overflow postings have source code?

```
SELECT count(*)
FROM new_posts_meta;

SELECT count(*)
FROM new_posts_meta
WHERE plaintext_code_length > 0;

SELECT plaintext_code_length, count(*)
FROM new_posts_meta
GROUP BY 1 ORDER BY 1 ASC;
```

Results:
Total Postings: 10,338,371
Total Postings with code: 5,899,791

What is the distribution of posts containing different amounts of code? In Figure 2, we binned each amount of source code (50 character bins) and counted the number of posts per bin. (The number of posts with more than 1000 characters of source code continues to decline as the character count increases, with only one or two posts in the 17,000+ character range.)

Our example with code counting is a simple one, designed to show how Stack Overflow can be mined for patterns about software development, the same way other forge artifacts have been mined in the past. Since the Stack Overflow data dump provides the entire body text for every posting on the Stack Overflow site, the possibilities for text mining are numerous. In fact, the Mining Software Repositories challenge for 2013 [24] was to use Stack Overflow data to find interesting patterns about software development. In the next section we discuss using Stack Overflow data to learn more about its characteristics as an open content community.

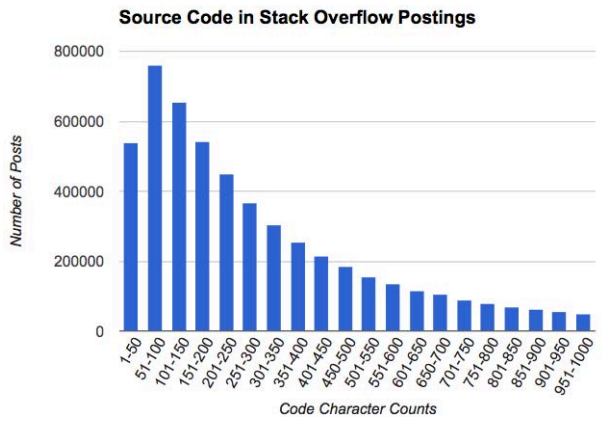

Figure 2. Number of posts by character count

3.2.2. Data about Editing Here we use Stack Overflow Data Explorer and the data dumps to investigate the extent and impact of editing. How common is editing on Stack Overflow? Is the amount of editing increasing over time? We run a series of queries as follows. Unless specified, all queries use live data from the Data Explorer as of June 2013.

(2) How many total questions and answers are there on Stack Overflow?

```
SELECT count(*)
FROM posts
WHERE PostTypeId=1;

SELECT count(*)
FROM posts
WHERE PostTypeId=2;
```

Results:
Questions: 5,171,392
Answers: 9,515,391

(3) How many of the questions have ever been edited? How many of the answers have ever been edited?

```
SELECT count(*)
FROM posts
WHERE postTypeId = 1
AND lastEditorUserId IS NOT NULL;

SELECT count(*)
FROM posts
```

[1] On Stack Overflow, code is separated from the rest of a posting through the use of a special "code" delimiter. This delimiter puts the code in a non-proportional font and highlights it with a gray background for easy reading. (Site users who find posts that contain code and no delimiter will often edit the posts to include it. This is an example of "making a post better", and as such, the user will earn reputation points for their edit. More on editing in the next section.)

```
WHERE postTypeId = 2
AND lastEditorUserId IS NOT NULL;
```

Results:
Questions edited: 2,522,694 (49%)
Answers edited: 2,225,720 (23%)

(4) Using the August 2012 data dump, how many questions and answers have been edited?

Results:
Total questions: 3,453,742
Questions edited: 1,632,131 (47%)
Total answers: 6,858,133
Answers edited: 1,478,465 (22%)

It is not surprising that questions would be edited more than answers, since there are many more answers than questions, and not all of them are very good. And since one of the main reasons to edit a posting is to correct information that has gone out of date, it stands to reason that over time the percentage of edited posts will rise slightly (irrespective of other reasons why user editing activity could increase).

Since editing posts is one way to gain reputation points on Stack Overflow, and editing is an important aspect of its open content mission, we are interested in finding the number of users who are editing, and whether editor counts are increasing over time.

(5) How many users are in Stack Overflow (June 2013)?

```
SELECT count(id)
FROM users;
```

Results:
Total users: 2,075,879

(6) How many of those users have ever actually edited anything? (The PostHistoryTypeId for editing actions is either 4, 5, or 6.)

```
SELECT count(distinct UserId)
FROM PostHistory
WHERE PostHistoryTypeId=4
or PostHistoryTypeId=5
or PostHistoryTypeId=6;
```

Results: 479,891 (23.1% of total users were editors)

(7) Using the August 2012 data dump, how many users have ever actually edited anything?

Results: 307,892 (editors)
1,295,620 (total users)
23.7% of total users were editors

Between August 2012 and June 2013, we do not see much of a decline in the percentage of total users who edit posts (23.7% vs. 23.1%).

Finally, we know that users are awarded badges on Stack Overflow as motivation, for performing various tasks deemed helpful to the site. Badges are awarded in three categories (gold, silver, or bronze) depending on how difficult they are to get and how important it is to encourage that particular behavior on the site. There are several badges related to editing the site, so to continue this exploration of the Stack Overflow data, we will examine the Strunk & White badge (silver), awarded upon editing 80 posts, and the Copy Editor (gold) badge, awarded upon reaching 500 posts. (There is also a bronze "Editor" badge awarded for a first edit. The numbers for that one are similar to those shown above in questions 5, 6, and 7).

First, with regard to motivation: we should note that some recent research [25] concludes that users do tend to quit performing the desired behavior upon earning a badge. (The specific badge in question in [25] was the Copy Editor badge. The authors found that users quit performing editing tasks directly after earning the badge. Authors of [26] also found that user behavior changes after earning a particular badge.) How many users have earned the different editing badges, and is that number going up over time, as a percentage of total users?

We can see from the answers to questions 8 and 9, numbers for each badge type as a percentage of the total user base are steady.

(8) How many users have earned each badge type?

```
SELECT count(*)
FROM badges
WHERE name='Strunk & White';

SELECT count(*)
FROM badges
WHERE name='Copy Editor';
```

Results:
Total Users: 2,075,879
Strunk&White: 4390 (.21%)
Copy Editor: 810 (.039%)

(9) Using the August 2012 data dump, how many users have earned each badge type?

Results:
Total Users: 1,295,620
Strunk&White: 2726 (.21%)
Copy Editor: 470 (.036%)

Stack Overflow is a non-forge web site with some "open content" and forge-like features. Two of these are the ability to share source code and the ability to edit posts. We described how to use the Data Explorer and the data dumps to study Stack Overflow artifacts and metadata. By doing so, we gain insights into how question-and-answer web sites are used to make software.

4. Pastebins

A pastebin is a web-based tool offering a simple paste-to-URL service. This means that a user (anonymous or not) can paste in any text to the web host (such as pastebin.com) and get back a URL pointing to a web page that includes the text as it was pasted. This saved text and its associated URL is called a *paste.* The paste will remain valid for a set period of time, usually specified by the user when the paste is created. When creating a paste, the creator has the ability to format the paste into a particular programming language, which will add the appropriate syntax highlighting and indentation to the paste. Programmers use pastebins to share code, error messages, and log files quickly and easily. Earlier alternatives to a pastebin were to (a) paste your (possibly long and complicated) code into a mailing list, forum, IRC channel where a discussion is taking place and have other users become annoyed, or (b) make a file, upload it to a public-facing server, retrieve its URL, and copy that into the mailing list message, forum, or IRC channel. Pastebins simplify this process for quicker and easier sharing.

At this point we should note that, largely because of their anonymity and ubiquity, pastebins can also be used to share non-programming content. [27] and [28] outline the darker side of pastebins and what they are used for in criminal activity.

There are hundreds of pastebin tools available today. There are three factors responsible for the fact that no single pastebin has become as popular as Github or Stack Overflow (in their respective categories). First, creating a pastebin is not terribly difficult since it does not require many features, and the requirements for authentication and security can be low (depending on whether the pastebin is offering private pastes or not). Thus, the number of pastebin web sites has proliferated. Second, companies or software groups have begun making their own internal pastebins, thus diluting the numbers of developers using any one particular public site. Finally, pastebins typically lack some of the social features of Github and Stack Overflow. This is because they have just a single primary functionality, which is to provide a URL to a piece of text. Thus pastebins have very small network effects (or, bandwagon effects) [29]. In other words, there is no real advantage to using the same pastebin as your co-worker; a particular pastebin does not increase in value the more that people use it (other than perhaps a small advantage conferred when sharing private pastes on some bins).

To gauge the impact of pastebins on the community of developers making FLOSS software, we searched across developer mailing lists to see how often these tools were being used to share code. We used searches on both MarkMail (markmail.org) and Marc.info (public software development mailing list aggregators, including for FLOSS), and found that pastebins started being discussed on developer mailing lists in the mid-2000s (a few dozen mentions each month, across several thousand lists) and by mid-2013 were mentioned on FLOSS lists upwards of 800 times per month. Figure 3 shows a graph produced by MarkMail for instances of the simple term 'pastebin' between 2001 and 2013. MarkMail reports roughly 57,000 messages across 8600 public email lists. (Not all the lists are about FLOSS development, but most are.)

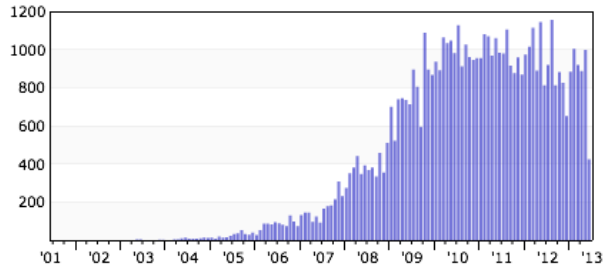

Figure 3. 'Pastebin' in email, 2001-2013

We compare this to the result of 'pastebin' searches on Google using Google Trends. (Figure 4) The sharp spikes at the right-hand side of the graph correspond to days when Pastebin.com was under a denial-of-service attack. In both graphs, 'pastebin' becomes a more popular search term over time, and may have leveled off somewhat over the past year.

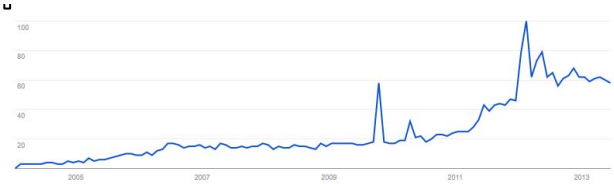

Figure 4. 'Pastebin' Google Trends, 2004-2013

4.1. Pastebins and "openness"

Pastebins certainly contribute to the same ethos of transparency and sharing as Github and Stack Overflow, but perhaps since pastebins are seen as a

waystation for code, and not its final destination, the pastebin hosts pay little or no attention to licensing issues. Copyright is, of course, still in effect, though anonymous posting and reading makes the infringement issue more difficult to navigate. But there is no default code license on pastebin.com, for example. Pastebin.com does outline its own obligations under the Digital Millenium Copyright Act (DMCA) to remove infringing copyrighted material. In addition, we found that the number of code samples submitted to pastebin.com that include their own code license is very small. In fact, when searching on pastebin for the phrase "license", the results showed more pastes about generating or sharing illegal license keys than actual source code being posted with a license.

In considering the "openness" of pastebins, we did uncover numerous FLOSS-oriented pastebins. For example, KDE has a pastebin (paste.kde.org), Oregon State University's Open Source Lab has a pastebin (pastebin.osuosl.org), and Github also released its own paste site called Gist (gist.github.com).

4.2. Data from pastebins

Pastebin.com is perhaps the largest of the independent pastebin sites. It reported in June 2013 that the site has reached as many as 15 million visitors per month, with more than 33 million pastes hosted on the site to date. Pastebin.com provides a very simple API [30], but compared to the ease of mining Stack Overflow or even Github, the options with this API are slim. The Pastebin.com API provides a few methods centered around the tasks of creating pastes and getting specific information about known users.

The history of repository mining shows that when researchers locate a potential source of interesting data, they will find a way to get it, even if the method is not elegant. In the early 2000s, Sourceforge was the most popular forge for FLOSS development. However it did not have an API or any officially sanctioned method of collecting project artifacts and metadata. So a few different research groups took on the challenge of collecting and storing Sourceforge artifacts and metadata for the entire research community to use [31][32], despite the many limitations of doing so [33]. Pastebin mining is similarly in its infancy, with just a few tools [34][35][36] available to help developers and researchers find, collect, store, and mine large quantities of pastes from different sites. Github itself stores its Gist pastes as repositories, so they can be mined in the same fashion as regular Github repositories.

4.2.1 Pastebin mining. The increasing popularity of pastebins necessarily changes the way FLOSS artifact mining will have to happen, especially in terms of mining email messages from mailing lists. (See [37] for a review of how FLOSS researchers mine email archives.) Consider the partial email exchange shown in Figure 5. This appeared in mid-2008 on the Apache-httpd-users mailing list.

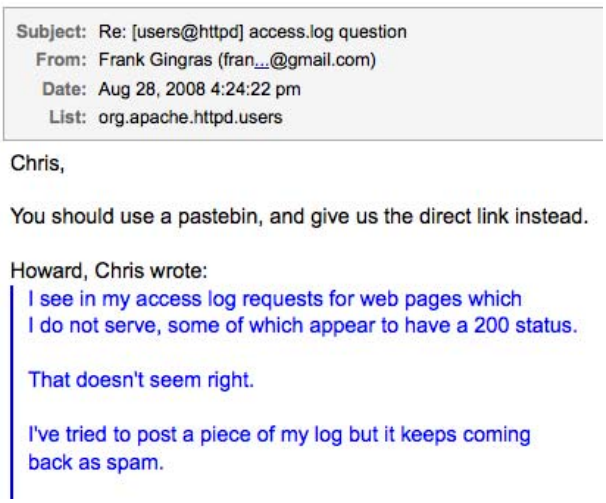

Figure 5. Email excerpt recommending pastebin, 2008

In Figure 5, the original poster, Chris, is requesting some help with understanding a log file, but the mailing list manager is rejecting his message for being spam. The respondent, Frank, directs Chris to use a pastebin site instead. We can confirm a few things from this exchange: log files can be interpreted as spam; spam is unwanted and blocked; links are welcome; pastebin is understood to be an easy way to post a link to a logfile.

For text mining, the downside to using a pastebin is of course that the pastebinned text (the log file or code sample) is no longer included *with* the email message itself. This changes the nature of the artifact collection, storage, and cleaning process. It is more akin to having to deal with attachments, rather than just being able to process simple email text. Figure 6 shows an interesting turn of events from January 2013.

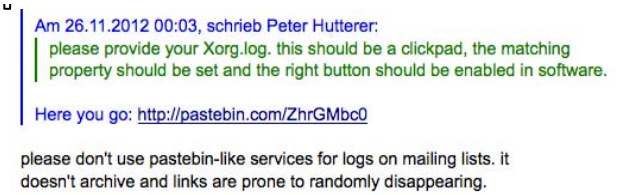

Figure 6. Email excerpt discouraging pastebin, 2013

In this exchange, a developer on the Freedesktop project list discourages another user from using pastebin to post logs. The reason given is that

sometimes the URLs become unavailable, making the problem/solution archive incomplete. From this exchange we learn that to (at least) one developer, the mailing list archives can be important archival tools. Additionally, we confirm that there is some awareness within the community that pastebins can change the way archiving works.

5. Conclusion

The main contribution of this paper is to describe three important tools used in the next generation of collaborative software development. FLOSS development is geographically and temporally distributed by its nature, so centralized virtual environments such as software forges are key to its development. For many FLOSS researchers, understanding software development in the 2000s had been an exercise in artifact collection and mining, mainly from forges and code repositories. Thus, this paper establishes that changes in forges, and in collaborative forge-like or forge-based tools, will affect the artifacts of the development process, which will in turn affect the way FLOSS is studied as a phenomenon.

We review the artifacts and metadata available in these new tools, and discuss some of the differences in the way each tool approaches "openness". We discover that the contemporary view of "openness" on Github, Stack Overflow, and pastebins may be centered on providing transparency or accessibility, rather than on offering specific rights as granted by particular software licenses.

In our analysis of functionality and artifacts, Github represents the next generation of the traditional software forge, albeit with some major differences in size and ideology. In short, Github is a forge built around a DVCS; it has very few of the other add-on tools traditionally offered by older software forges. We can say that Github represents a "forge light" approach: do a few things very well rather than many things poorly.

In contrast, Stack Overflow is a standalone tool, not a forge at all. Yet its functionality is replacing some traditional forge-based features (e.g. support forums and wikis). Stack Overflow has become a critical piece of development infrastructure, but at the same time, it is a separate entity, and not trying to be a software forge. If Github is able to succeed with a "forge light" approach, we wonder if that is partially because much of the communication and support load is being borne by Stack Overflow (and older technologies like mailing lists and IRC). While it is less convenient for users to have to split their development attention across multiple sites and media, perhaps this approach is appealing for its increase in functionality and lower overhead.

Pastebins are also standalone tools. But unlike Stack Overflow, which essentially replaced a feature formerly found on software forges, pastebins have no parallel in older software forges. In fact (perhaps ironically considering the "forge light" approach just discussed), Github is the one forge that *has* integrated a pastebin tool (Gist) directly into its feature offerings. To paraphrase the famous saying, "Software forges are dead. Long live software forges."

6. References

[1] M. Squire. "Describing the software forge ecosystem". In *Proc. Hawaii Int. Conf. Sys. Sci. (HICSS'45*). pp 3416-3425. 2012.

[2] 10th Working Conference on Mining Software Repositories. http://2013.msrconf.org/

[3] A. Williamson, "Licensing of Software on Github: A Quantitative Analysis" Online, http://www.softwarefreedom.org/resources/2013/lcs-slides-aaronw/. Accessed June 2013.

[4] Github. "Terms of Service". Online, https://help.github.com/articles/github-terms-of-service. Accessed June 2013.

[5] S. Phipps. "Github needs to take open source seriously". November 23, 2012. Online, http://www.infoworld.com/d/open-source-software/github-needs-take-open-source-seriously-208046. Accessed June 2013.

[6] Github. "Github API, v3". Online, http://developer.github.com/v3/. Accessed June 2013.

[7] I. Grigorik, "The Github archive," Mar. 2012. Online, http://githubarchive.org. Accessed June 2013.

[8] G. Gousios, "The GHTorent Dataset and Tool Suite," In *Proc. 10th Working Conf. on Mining Software Repositories (MSR 2013)*. pp. 233-236. 2013.

[9] C. Bird, P.C. Rigby, E.T. Barr, D.J. Hamilton, D.M. German, and P. Devanbu. "The promises and perils of mining git." In *Proc. 6th Working Conf. on Mining Software Repositories (MSR 2009*). pp. 1-10. 2009.

[10] B. Heller, E. Marschner, E. Rosenfeld, and J. Heer. "Visualizing collaboration and influence in the open-source software community." In *Proc. 8th Working Conf. on Mining Software Repositories (MSR 2011*). pp. 223-226. 2011.

[11] X. Ben, S. Beijun, and Y. Weicheng. "Mining Developer Contribution in Open Source Software Using Visualization Techniques." In *Intelligent System Design and Engineering Applications (ISDEA*), pp. 934-937. IEEE, 2013.

[12] M. Allamanis, and C. Sutton. "Mining source code repositories at massive scale using language modeling." In *Proc. 10th Working Conf. on Mining Software Repositories (MSR 2013*). pp. 207-216. 2013.

[13] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. "Social coding in github: transparency and collaboration in an open software repository." In *Proc. of the ACM Conf. on Computer Supported Cooperative Work*, 2012. pp. 1277-1286.

[14] G. Robles, J.M. Gonzalez-Barahona, D. Izquierdo-Cortazar, and I. Herraiz. "Tools for the Study of the Usual Data Sources found in Libre Software Projects". *Int. J. Open Source Software and Processes*, 1(1). pp. 24-45. 2009.

[15] Stack Exchange. "The Stack Exchange Data Explorer." Online, http://data.stackexchange.com/stackoverflow/queries. Accessed June 2013.

[16] C. Parnin. "Api documentation". Online, March 3, 2013. http://blog.ninlabs.com/2013/03/api-documentati on. Accessed June 2013.

[17] Google. "Get Help Developing with Google BigQuery." Online, https://developers.google.com/bigquery/docs/support. Accessed June 2013.

[18] Google. "We're moving BigQuery developer support to Stack Overflow (using the tag "google-bigquery")." Online, http://groups.google.com/group/bigquery-discuss/browse_thread/thread/cf967e6914bdcfc2. Accessed June 2013.

[19] Stack Overflow. "Terms of Service". Online, http://stackexchange.com/legal. Accessed June 2013.

[20] Creative Commons. "Attribution-ShareAlike 2.5 Generic". Online, http://creativecommons.org/licenses/by-sa/2.5/. Accessed June 2013.

[21] J. Atwood. "Attribution Required," Stack Exchange Blog. June 25, 2009. Online, http://blog.stackoverflow.com/2009/06/attribution-required/ Accessed June 2013.

[22] Stack Overflow. "Privileges: Edit Questions and Answers," Online, http://stackoverflow.com/privileges/edit. Accessed June 2013.

[23] Stack Exchange Data Dumps. Online, http://www.clearbits.net/creators/146-stack-exchange-data-dump

[24] Mining Software Repositories Challenge, 2013. In *Proc. of the 10th Working Conference on Mining Software Repositories* (MSR 2013). pp. 53-100.

[25] S. Grant, and B. Betts. "Encouraging user behaviour with achievements: an empirical study." In *Proc. of the 10th Working Conference on Mining Software Repositories* (MSR 2013). pp. 65-68. 2013.

[26] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. "Steering User Behavior With Badges." In *Proc. WWW*2013, May 13–17, 2013.

[27] S. Matic, A. Fattori, D. Bruschi, and L. Cavallaro. "Peering into the Muddy Waters of Pastebin." *ERCIM News: Special Theme Cybercrime and Privacy Issues.* pp. 16. July 2012.

[28] G. Kontaxis, I. Polakis, S. Ioannidis, "Outsourcing Malicious Infrastructure to the Cloud," In *First SysSec Workshop (SysSec)*, pp. 35. July 2011

[29] J. Rohlfs. *Bandwagon Effects*. MIT Press. 2003.

[30] Pastebin. "Developers API". Online, http://pastebin.com/api. Accessed June 2013.

[31] J. Howison, M. Conklin, and K. Crowston, "FLOSSmole: A collaborative repository for FLOSS research data and analyses." *International Journal of Information Technology and Web Engineering*, 1(3), 17–26. 2006.

[32] Y. Gao, M. Van Antwerp, S. Christley and G. Madey, "A Research Collaboratory for Open Source Software Research", In *Proc. Int. Workshop on Emerging Trends in FLOSS Research and Development (FLOSS 2007)*, Minneapolis, MN, May 2007.

[33] K. Crowston and J. Howison. "The perils and pitfalls of mining sourceforge." In *Proc. 1st Workshop on Mining Software Repositories*. pp. 7-11. 2004.

[34] A. MacPherson. "PasteLert". Online, http://www.andrewmohawk.com/pasteLert/. Accessed June 2013.

[35] A. MacPherson. "PastebinParser". Online, http://www.andrewmohawk.com/pasteScrape/. Accessed June 2013.

[36] Corelan Team. "Pastenum". March, 2011. Online, http://www.corelan.be/index.php/2011/03/22/pastenum-pastebinpastie-enumeration-tool. Accessed June 2013.

[37] M. Squire. "How the FLOSS research community uses email archives". *Int. J. Open Source Software and Processes, 4(1).* pp. 37-59. 2012.