# Describing the Software Forge Ecosystem

Megan Squire, David Williams
Elon University
{msquire, dwilliams22}@elon.edu

## Abstract

*Code forges are online software systems that are designed to support teams doing software development work. There have been few if any attempts in the research literature to describe the web of people, projects, and tools that make up the free, libre, and open source (FLOSS) forge ecosystem. The main contributions of this paper are (1) to introduce a classification of FLOSS-oriented forges according to their characteristics; (2) to describe the forge-level and project-level data and artifacts currently available at each FLOSS forge; (3) to show various patterns already discovered in the FLOSS forge ecosystem, such as timelines of creation or arrangements by size or feature; (4) to make some recommendations to forge providers and data collectors about how to expose the structure and information in the forges; and (5) to describe the effort needed to extend our publicly- available information about the FLOSS forge ecosystem into the future.*

## 1. Introduction

A software forge provides various tools and facilities to distributed development teams, such as source code control systems, mailing lists and communication forums, bug tracking systems, web hosting space, and the like. Because of the decentralized and public nature of free, libre, and open source (FLOSS) projects, many software forges were created with the needs of FLOSS teams in mind. However the concept of a forge is now used in numerous non-FLOSS projects, as well as in projects not designed for public participation.

Researchers who study the FLOSS phenomenon are often interested in public forges because of the relative ease of data gathering (access to lots of projects, all formatted similarly, making them easier to compare to one another). The purpose of this paper is threefold: to extend existing work on describing similar collaborative software development virtual spaces, to initiate a discussion about the current state of FLOSS forges, and to propose a way to expose the information inside of FLOSS forges to facilitate further study. We wish to provide an overview of the entire FLOSS forge ecosystem, including lesser-known forges, and the projects and users that inhabit them. Knowing basic facts about the forges, and being able to compare and contrast them will help open source researchers in several ways. First, when we use a common vocabulary for describing forges, we will be able to better explain why we choose to collect or use data from a certain forge versus another. Next, when we gain knowledge about what artifacts and data points are available in the different forges, we will be able to more efficiently put together a study with projects or developers from multiple forges. Finally, by exposing the data hidden in lesser-known forges, we will tap into a source of more varied and interesting research questions.

In Section 2 we give a brief review of work on software forges and collaborative development environments more generally. Section 3 describes our work to collect and store relevant facts about the forge ecosystem, including basic descriptions of the forge, project, and user entities. In Section 4 we attempt to draw comparisons between forges based on their characteristics, and we show it is possible to combine forge and project metadata to further describe the forge ecosystem. We also describe ways that forge operators could expose the information in their systems so that the data gathering for research would be easier and more effective. In Section 5 we discuss some of the limitations of our work, and we also present our plan for keeping this information current and for extending it to be more relevant.

## 2. Background and motivation

Despite the popularity and ubiquity of forges in the FLOSS development communities (and increasingly among teams engaged in making proprietary software too), among academics there have been relatively few efforts to study the forges themselves: what features are provided, how the projects use the forges, and what artifacts are available to study about the projects and users of the

forge. In fact, most academic work in this area does not use the term "forge" at all. There is something of a disconnect between the description of a "collaborative development environment" (or CDE, the closest relative of the forge as described in the academic literature), and the "code forge" as described in practitioner parlance. In this section, we first discuss the academic notion of a CDE, and then we discuss the reality of code forges as a facet of common FLOSS development practices.

## 2.1. Collaborative development environments

The notion of a CDE for software development was initially positioned in the literature as a Web-enabled and virtualized extension of the traditional developer desktop IDE (integrated development environment) [1]. Well-known IDEs include software packages such as Eclipse, Activestate Komodo, or Microsoft Visual Studio. The IDE typically provides features such as a text editor, shell, file uploads, compiler integration, interactive debugger, integration with bug tracking systems, integration with version control systems, etc. With the commercialization of the Internet in the mid- to-late 1990s, and the concurrent rise of FLOSS as a highly visible, viable model for software development, software development teams continued to become more geographically dispersed and dependent upon Internet-based tools for collaboration. The CDE was described by Booch and Brown [1] as "a virtual space wherein all the stakeholders of a project...labor together to ...create an executable deliverable and its supporting artifacts." A software CDE is, then, a set of tools that facilitates the same tasks as a software IDE (writing code, writing documentation, finding and fixing bugs, distributing releases) but does so in a way that meets the needs of distributed (over time and space) groups of developers.

Nearly ten years later, we want to know how well the actual implementation of CDEs has happened in practice. Have code forges extended or obscured the vision presented in early CDE papers? How can we reconcile the initial vision with the practical reality of a decade of FLOSS development in forges? Today, most of the commercial software CDEs described in the original Booch and Brown paper have either disappeared or gone through enough changes that they are no longer recognizable as originally described. However, their description of the CDE as "a hundred small things" rather than a monolithic killer app is still apropos, and their list of CDE features is still relevant (see especially Figure 7 in [1], which serves as the framework for Section 3 of this paper).

## 2.2. Code forges

Two recent events involving code forges also served as catalysts for writing this paper. In 2010, one of the keynote presentations at the International Conference on Open Source Systems (OSS2010) was about changes and developments over time in what was being called the FLOSS forge ecosystem [2]. The speaker referenced a Wikipedia page about software forges [12] and explained how the forges had grown over time and how some had become defunct or merged into others. Unfortunately, the list of attributes used on the Wikipedia page to describe forges was interesting but far from complete, and the page gave no explanation of the methodology for choosing which features to display. Like all of Wikipedia, the page only gets updated as readers choose to update it, so much of the data was outdated. Yet this was the best reference source available, and it was the only publicly-available, comprehensive attempt at creating a current classification of forges.

Following this presentation, we reviewed the FLOSShub.org research news portal and paper repository for information about how forges were being used to study FLOSS development. This paper repository includes bibliographic information (and pre-prints in some cases) for approximately 1100 papers about open source software development from various journals and conferences. We read through the assembled FLOSS papers looking for (a) whether forge data was being used, (b) whether the forge projects or developers were being studied in the aggregate, e.g. a multi-project study, or were single projects being extracted from the forge, (c) what artifacts were being studied (source code, bug tracking histories, metadata such as programming languages or license types, etc). We found and tagged 64 published research papers that had been written using project or developer data gathered from Sourceforge[1]. We observed that other FLOSS forges have hardly been studied at all in the literature. For comparison, among the FLOSShub papers, the next highest source of multi-project data was the project directory - not truly a forge - called Freshmeat, with four papers that used its data. FLOSS papers that focused on a single project (for example, case studies, code reviews, or surveys) were dominated by Apache with 55 papers, and Linux with 45 papers. Clearly Sourceforge is a popular source of FLOSS data, especially in multi-project studies or "breadth"

[1] http://flosshub.org/category/tags/sourceforge

studies (such as those focusing on license choice [3] or programming languages [4-6]).

Why do researchers turn to Sourceforge for project data? Of the papers that explained their rationale, the most common reasons given were (1) because the authors needed or wanted to study a comparatively large number of projects, and/or (2) because the parameters of the given study required a set of projects with predictable, exposed metadata and/or artifact data. Sourceforge exposes certain project and developer metadata in a format (HTML) that is predictable and easy-to-find. It did this early on in the history of forges (1999), it does so for all of its projects, and for most of the 2000s it housed more projects than any other forge. The implication is that gathering that amount of project data from another forge would be more difficult or impossible, because those forges lack the numbers that Sourceforge has, and because gathering that much project data by visiting individual project web sites and locating the variables needed to perform the study would be inordinately time-consuming and error- prone. Even papers that used semi-random sampling or that hand-selected a small number of projects from within Sourceforge relied on the fact that Sourceforge kept all project and developer metadata and artifacts exposed in a predictable way, making it a relatively straightforward source of data.

The lure of using Sourceforge as a data source is therefore very strong. This is the case despite the existence of multiple other publicly-available sources of collected data from standalone projects and forges [7,8], and despite the existence of publicly-available made-for-research collections of Sourceforge data [9], and despite warnings about the potential pitfalls in using Sourceforge data to begin with [10]. To complicate matters, many of the papers using Sourceforge data appeared to have collected their own data by writing web spiders to crawl the forge website instead of using one of the public collections [8,9]. But because that collection process was often troublesome to sustain over time, some of these papers were based on data that had been collected once at the beginning of a research study and never updated; indeed, some of the data sets were seven or eight years out of date (or more) by the time the papers were finally published in a journal. Very few of the papers donated the data they did collect to the FLOSSmole data repository for others to use after the fact, and very few of them provided their data for download at their own sites. Even fewer provided enough details to perform a useful secondary analysis or replication by other teams [11]. Our observation is that Sourceforge has become the "common fruit fly" of FLOSS research - accessible, inexpensive, and approachable - but is it the only organism in the ecosystem worth studying?

In the next section we discuss our efforts to collect data on the rest of the FLOSS forge ecosystem, the forge data collection process, and the model we use to organize our data.

## 3. Describing FLOSS forges and data

The CDE literature gave lots of ideas for ways to study CDEs in general, including CDEs developed for workers at different locations inside individual companies, and CDEs developed for non-software domains. However, we elected to keep this study relevant to software forges, and to FLOSS forges specifically. In this section we discuss the terminology we use to describe the forges, the method of gathering data about the forges, and the way we modeled and cleaned the data that we collected.

### 3.1. Forge entities

We elected to concentrate on three main entities within the FLOSS forge ecosystem: the forge itself, the projects hosted by the forge, and the people working on or using the projects. Some characteristics of each of these entities are as follows.

A software project is the term for the collaborative effort of designing and producing software artifacts to be used by people. A forge acts as a hosting service for projects. The forge provides different support functions to the project or to users in order to facilitate collaborative software development. Some of the features that a forge might provide could include communication features (mailing lists, discussion boards), collaboration features (revision control, bug tracking), or project metrics (activity ranking, statistics). The forge may specialize in some types of software projects (FLOSS only, certain programming languages, certain spoken languages).

Forges can have their own operating policies, their own set of features that they offer, and their own definitions of what types of projects and people they want to host. The artifacts created by a project may or may not be exposed to public view by the forge hosting that project. The metadata that a forge collects about the projects and people using it may or may not be exposed to public view. Each forge can decide what features to offer, how those features will be used by the projects that are hosted there, and what artifacts and metadata will be exposed about its projects and people.

The forge will usually give each project a unique URL so that it can be located. Projects usually have a name that is chosen at the time it is added to the forge. They may have lists of external URLs that represent other places on the Internet where this project lives. They may have an owner or someone in charge of them. They are often given a textual description. Projects may have various keywords or terms that further describe the usage of the software or the purpose of the project. These terms could include the programming language used, the license applied, the intended audience of the software, its latest release date, the activity level of the project, and any number of other facts about the project.

People in the forge ecosystem can be (1) directly affiliated with a project as part of the project team, or (2) users of a forge but unaffiliated with a project. Typically in the FLOSS ethos, users of a project do not have to be known (i.e. you do not typically have to register to download an open source project), but developers or members of a project will usually be known or their contributions will be listed somewhere. However, forges may differentiate between 'users', 'members', 'contributors', 'administrators', 'owners', 'developers', and any number of other roles or levels of participation.

### 3.2. Gathering forge data

To investigate the forges, projects and people in the FLOSS forge ecosystem, we used three main sources [12-14] to develop a list of several dozen candidates for our forge list. For the purposes of this paper, in order to be considered a FLOSS-oriented software forge, the Web site for the forge had to have a few key characteristics that were based on a modification of the description of a CDE given in [1]. Our minimum standards are as follows. The forge…

1. Must be designed to facilitate the process of software development by providing at least one software development tool and preferably multiple tools (e.g. group communication software, bug tracking system, revision control system) for use by geographically distributed teams over the Internet; and
2. Must provide a way for teams to identify themselves and their project(s), update and administer their own project, and distribute their own artifacts or product; and
3. Must have some sort of connections to the FLOSS community (either explicitly stated or implicit in forge policy); and
4. Must be, at the time of this writing, accepting new projects and actively maintaining the site for the projects hosted there.

Requirements 1 and 2 were added in order to remove the free software directories (e.g. Freshmeat, FSF Directory) and module repositories or archive networks (e.g. CPAN). Requirement 1 also meant removing the "open content" forges (such as KnowledgeForge), which were interesting but stretched the limits of this paper. Requirement 1 also specifically removes a number of "revision control only" hosting facilities (such as repo.or.cz, one of the first git hosts). Requirement 3 removes from the list those forges that are purely commercial in nature and have no substantive connection to any FLOSS community. Requirement 4 requires removing from the list any forges that were no longer operational or were in the midst of shutting down. (For example, at the time of this writing Project Kenai has been merged into Java.Net, MozDev is, according to their web site, in the midst of a "wind-down", and LuaForge is not accepting any new projects.)

For each forge we also recorded its parent company or organization, if any, and the self-reported date that the forge was established, if available. If the date of establishment was not readily available on the forge's own main web site, we attempted to find this date through news stories, press releases, or the like. Some additional information we gathered about the forges was liable to change over time: we also collected the number of users of the forge (self-reported) and the number of projects on the forge (self- reported or gathered from the web site's directory of projects). The final list of 24 forges we studied is shown in Table 1 in Appendix A. The SQL to build this list is shown in Appendix B, Listing 1. (The SQL statements needed to build the all the tables and figures in this paper are based on a data model shown in Appendix A, Figure 1, described further in 3.3. All SQL is given in Appendix B. The reason we give the SQL and data model is so that users who wish to access this data and run their own queries on our database located at FLOSSmole.org can get started quicker and understand what they are looking at.)

We understand that trying to identify and describe active software forges is something of a moving target, so our list of forges is never going to be complete or finished. However, in Section 5 we discuss some of the actions we are taking to keep the list current and relevant, and we list some of the limitations of the forge list we chose.

After making the list of forges, we then visited the web site of every forge and created an initial list of characteristics that forges seemed to have. We asked questions like: Does it have a browsable project directory? Are there standard guessable URLs for each project? What are the criteria for a project

joining this forge? (Is it FLOSS-only? Are there requirements about certain licenses? Can anyone create a project or is there approval required?) What collaboration features/tools are available? Does the forge offer paid accounts or upgrades? What is the general ethos of the place? (Is it FLOSS-friendly, business-friendly, affiliated with a larger umbrella organization?) Do many projects consider this forge to host their home page, or are they using this page to link to a different place where the real work is done?

While we visited each forge, we viewed projects hosted on that forge to learn about them. We asked questions like: What metadata is available about each project? Is this is a predictable location on each page? Is there an API to extract information from the forge about a given project? Which artifacts are available about the project?

We also thought about what we could learn about people on each forge: What metadata is available about people who use this forge? Can we discern an affiliation between users and projects? Can we discern the various roles of the different people in the forge or working on a project? Some forges also support the idea of a team of people, or teams of teams. As discussed in Section 3.1, forges differ wildly in the terminology available to describe the people involved, and projects differ in their application of those terms to the people (for example, on some projects everyone is given a title of 'developer').

Once we had this long initial list of characteristics, we went back through the list of forges and revisited each of them in turn, trying to find out whether or not each forge could be described ("tagged") as having that characteristic or not. Sometimes this was easy to discern, other times it was not. Finally, we created parent categories of characteristics to try to group similar tags together. All the tags having to do with project metadata were grouped together, all the artifact tags were grouped together, etc. (We knew that each individual revision control system offered by a forge should be classified as a feature tag, but we broke them out into a separate category on their own because there are so many of them, and because the choice of revision control system is so central to the way a project is developed.) We also decided to focus most of our efforts on describing the forges and projects rather than the people, for various reasons we describe further in Section 5 (Limitations and Future Work).

### 3.3. Data model

Because we knew that the information about forges would change over time, we needed a persistent way to store multiple versions of this data. We decided that we would expand the forges table that already existed in the FLOSSmole database and add enough tables to describe the rest of the forge ecosystem characteristics. We thus developed the simple data model shown in Figure 1 of Appendix A. The two main entities in the model are forges and tags, and we used the join table of forge_trove to classify which forge had which tags. If the forge has a tag, there is a record in this join table. This table supports being changed over time, via the datasource_id column (a unique identifier referring to when the collection took place).

### 3.4. Data cleaning and checking

After compiling the list of tags and categories and searching the forges for evidence of these characteristics, we then looked for obvious errors, especially in tag combinations that didn't make sense. Things that made us suspicious included a forge listed as having mailing list archives as an artifact tag but without the corresponding mailing list feature tag, or several flossonly forges not requiring the license as part of the standard project metadata. We also looked for unintentional differences between forges known to be running the same underlying architecture (for example, inconsistencies between all the forges running variants of Gforge, Kforge and FusionForge). We removed tags that turned out to be uninteresting, or which lacked evidence in the population. For example, we removed a difficult-to-discern tag called ad-supported and instead used a tag called ad-free, which is much easier to observe in the forge population; we also combined tags for MySQL and PostgreSQL into a single tag called dbms.

As we found anomalies, we added our notes and caveats to a central list. For example, we were interested in knowing that not all FLOSS-only forges required the project to list its FLOSS licenses as metadata on its project page. Another surprise was that some forges provided almost no public-facing metadata about projects (typically these were the few forges that provided a wiki for free-form text as the home page). We also noticed that some forges allowed the projects to configure which metadata and artifacts were available, and which level of authentication was required to view these, with the result that a project might have a feature (such as mailing lists) but the associated artifact (the mailing list archive) was only viewable by certain registered users.

As we observed in our previous discussion of the popularity of Sourceforge data collection in FLOSS research (Section 2), the amount and quality of the

public-facing data in a forge will greatly affect our ability to use that forge for research. In the next section, we explore the forges in terms of their usefulness as sources of data for FLOSS research.

## 4. Exploring FLOSS forge ecosystem data

After gathering, storing, and cleaning the data as described in Section 3, we next developed a simple Web interface to better display the data[2]. The Web interface has two main sections: (1) a page with a grid showing basic forge facts and tags (as well as the anomalies and caveats we found when collecting the data), and (2) a page of simple example patterns discovered after organizing the data in this way. Our overall goal in exploring the data is to discover which forges hold the most promise for data collection.

### 4.1. Grid of forge tags

Because the classification data was both binary (yes/no) and categorized hierarchically, and because we had a large number of attributes and rows, we somewhat reluctantly decided to use a series of data tables to show the data collected. The tables we made are far too large to be reproduced in this paper, and may have been corrected or altered for better viewing, so we encourage readers to look at the data grid and our collection notes in the online format.

Organizing the data in this way means it is relatively easy for a research team to pinpoint forges based on some set of characteristics, for example "We are interested in FLOSS-only CVS forges that have more than 1,000 projects and which also have mailing list archives and bug tracker archives. The forge(s) must have a directory of projects and a predictable internal URL for each." (See Appendix B, Listing 2.)

### 4.2. Simple forge patterns

It is also fairly straightforward to highlight some simple patterns about the forge ecosystem itself. The first visualization we made was a timeline of forge creation, shown in Figure 2 (Appendix A). We used the *established* column in the main forges table to generate this diagram. We also wanted to know if older forges would necessarily be larger (*numProjects* column), and whether the very large forges have approval policies for new projects. The corresponding table and SQL are shown in Table 3 (Appendix A).

[2] Found at http://flossmole.org

### 4.3. Best forges for general data collection

As we explained in Section 2, our most pressing question is which forges will expose the best FLOSS research data. "The best" could be defined here in a number of ways depending on any given research agenda, so here we propose a list of forge attributes that would probably make for successful general-purpose data collection effort:

- Easy to find the entire list of projects (a browsable directory or other means to get a list of all projects); or
- A large number of projects to pick from; or
- Lots of exposed metadata for each project; or
- Lots of exposed artifacts for each project.

Table 1 (see Appendix A) confirms why Sourceforge endures as a popular choice for data collection. It has a high number of projects, lots of exposed metadata and artifacts. It appears that other good choices would be Google Code, Launchpad, Rubyforge, and CodePlex. Of these, the FLOSSmole data repository already regularly collects data from all but CodePlex. FLOSSmole has also collected from Github in the past, although that became significantly more difficult when they removed their project directory.

There are several other characteristics of forges, mostly determined by forge policy, that typically make automated data collection easier:

- If the forge provides an API into its data and/or RDF or DOAP (means less "spidering" of HTML pages, and all the risk and annoyance therein [10]);
- If the forge has an un-complicated user / team / project structure (minimal recursion); and
- If several forges are running the same underlying software (means opportunity for re-usable collection software and easily-comparable data models).

Some of the forges already have these characteristics, but others do not. One possible recommendation we can make to forge providers who wish to make their data more accessible in a very low- stakes way is that they should expose data following the model of the project directory Freshmeat, which releases project metadata regularly in RDF format. Or, follow the model of Sourceforge and Launchpad, which have an API into their data so that developers can directly request information on projects or users.

We also identified six forges running Gforge/ Kforge/FusionForge variants (these are viewable in Table 1), so it may be that a collection architecture written for one of these could be easily extended to the others. (FLOSSmole already collects from two of

these: Objectweb and Rubyforge.) Or, modifications of the open source Gforge/Kforge/FusionForge codebase itself could introduce new features that allows forge operators to publish their data, for example as RDF.

## 5. Limitations and future work

We are hopeful that the findings we describe in this paper will be the start of an ongoing discussion about the forge ecosystem. Here we outline a few limitations, and we make some suggestions for our own future work, as well as that of related studies.

One limitation of this study is that we defined software forges in such a way that we excluded some very interesting "forge-like" sources of high quality data. The Apache project, for instance, is an obvious source of FLOSS data that should probably be included on a future list of forges. But because all its sub- projects are part of one overall parent project, it escaped our initial definition of a forge. (Similar sites would be Funambol, Drupal, or JoomlaCode.)

A second limitation of our study is that we ran into difficulties with collecting and describing data about the people in the forge ecosystem. We were stymied by the vast differences between forges and between projects in describing the people associated with each. Launchpad provides an excellent example of the difficulties herein: Launchpad has users, users can be contributors to a project, projects have registrants, contributors, drivers, and maintainers (all of which can be users, contributors, or teams), teams have members, and members can be other teams. Other forges use completely different vocabularies to describe the roles of people in their system. We finally settled on two simple but flawed tags: memberlist and administratorlist. With these we can indicate whether it is possible to figure out who the members of a project are and who is in charge of the project. A more robust and accurate set of tags is certainly necessary to continue this work in the future.

An important practical limitation to this study, and especially to the statements made in Section 4 about "best" forges to collect from, is that we have not always investigated the exclusions specified by the forges in their robots.txt files or other site usage policies. We know that Sourceforge, for instance, has both a robots.txt and an additional site usage policy[3]. Launchpad provides an API into their data but also has certain requests they make about usage of that API[4]. These policies should be closely watched by anyone deciding to collect data from a forge, as they present practical limitations that may impact any perceived benefit of collecting from that source.

Other future work on this project should of course include updating the three online sources that we used to derive our initial forge listing, especially the Wikipedia page [12]. We corrected a few items on that page as we went along in this study, but more work can be done to keep the page current and organized.

Our own forge study can be updated at any time, following the same procedure outlined in Section 3. The items that can be updated include the initial list of forges, the tags used, and the parent categories. The data model already requires a unique identifier for the date, time, and person collecting the new data. Researchers wishing to complete a new or updated version of this study can add their results into the FLOSSmole database right alongside the existing data. All the data is available at any time to users of the FLOSSmole.org database.

Aside from the basic data collection updates, probably the most important contribution of future forge studies would be to find a way to facilitate cross- forge project comparisons. This paper provides an initial collection of data about forges, projects, and people (to some degree), but matching these entities across forges is still an elusive goal [15,16]. How can we use the data in the forges to tell whether Project A on one forge is the same as Project B listed on another forge? Just like most people don't live in the same house their entire lives, software projects move locations too. Projects, like people, change their names, change their affiliations and dependencies, and spawn offspring. Forges are littered with the detritus of deceased, missing, and relocated projects, and the projects that are there often do not keep information current. To tell the story of a software project over time requires that we are able to track a project across forges, and across these many changes.

## 6. Conclusion

Our goal in this study was to collect enough basic information about FLOSS-oriented software forges to be able to describe them using a common vocabulary, compare them to one another, and ultimately identify which ones might hold data of interest to FLOSS researchers. To accomplish our goal, we established a list of forges, collected a host of metadata and statistics about each forge, designed a system to store

[3] See http://sf.net/robots.txt and http://sf.net/apps/trac/sitelegal/wiki/Crawler%20policy

[4] https://help.launchpad.net/TermsofUse

the data and accept new data donations into the future, created a web interface to display the data, wrote queries to display some basic summaries and statistics about each forge, and made some recommendations for the research community interested in using data collected from FLOSS-oriented forges. The entire corpus of data, including our collection notes and our web interface, is available via the FLOSSmole web site.

## 12. References


[1] Booch, G., Brown, A. W. (2003). Collaborative development environments. *Adv. in Comp.* (59).1-27.

[2] Oostendorp, N. (2010). The OSS forge ecosystem: Today and Tomorrow. *6th International Conference on Open Source Systems (OSS2010)*. Notre Dame, IN, USA. May 31, 2010. Retrieved from http://www.ustream.tv/recorded/7356043

[3] Lerner, J. and Tirole, J. (2005). The scope of open source licensing. *J. Law, Econ. & Policy, 21(1)*.20-56.

[4] Delorey, D.P., Knutson, C. D., Giraud-Carrier, C. (2007). Programming language trends in open source development: An evaluation using data from all production phase SourceForge projects. *2nd Wkshp on Public Data about Software Development (WoPDaSD2007)*. Limerick, IE. June, 2007.

[5] Krein, J.L., MacLean, A.C., Knutson, C.D., Delorey, D.P., Eggett, D.L. (2010). Impact of programming language fragmentation on developer productivity: A SourceForge empirical study. *Int. J. Open Source Software and Processes*, 2(2). 41-61.

[6] Krein, J.L., MacLean, A.C., Knutson, C.D., Delorey, D.P., Eggett, D.L. (2009). Language entropy: A metric for characterization of author programming language distribution. *4th Workshop on Public Data about Software Development (WoPDaSD2009)*. Skovde, Sweden. June, 2009.

[7] Gonzalez-Barahona, J.M., Izquierdo-Cortazar, D., Squire, M. (2010). Repositories with public data about software development. *Int. J. Open Source Software and Processes,* 2(2). pp. 1-13.

[8] Howison, J., Crowston, K., Conklin, M. (2006). FLOSSmole: A collaborative repository for FLOSS research data and analysis. *Int. J. Information Technology and Web Engineering*, 1(3). 17-26.

[9] Gao, Y., Van Antwerp, M., Christley, S., Madey, G. (2007). A research collaboratory for open source software research. *Intl. Wkshp. Emerging Trends in FLOSS Res. & Dev.t.* Minn., MN, May 2007.

[10] Howison, J., Crowston, K. (2004). The perils and pitfalls of mining SourceForge. *Workshop on Mining Software Repositories (MSR2004).* Edinburgh, Scotland, UK. May, 2004. 7-11.

[11] Robles, G. (2010). Replicating MSR: A study of the potential replicability of papers published in the Mining Software Repositories proceedings. *7th IEEE Working Conference on Mining Software Repositories (MSR2010)*. Cape Town, S. Afr. May, 2010. 171-180.

[12] Comparison of open source software hosting facilities. (n.d.) In Wikipedia. Retrieved June 1, 2011 from http://en.wikipedia.org/wiki/Comparison_of_open_source_software_hosting_facilities.

[13] Open Source Project Hosting. (n.d.) In Open Directory Project. Retrieved June 1, 2011 from http://www.dmoz.org/Computers/Open_Source/Project_Hosting/

[14] Public Git hosting sites. (n.d.) In GitHosting Wiki. Retrieved June 1, 2011 from https://git.wiki.kernel.org/index.php/GitHosting

[15] Conklin, M. (2007). Project entity matching across FLOSS repositories. *3rd International Conference on Open Source Systems*. Limerick, IE. June, 2007.45-57.

[16] Howison, J. (2008). Cros-repository data linking with RDF and OWL: Towards common ontologies for representing FLOSS data. *3rd Workshop on Public Data about Software Development (WoPDaSD2008).* Milan, IT. September, 2008.


## Appendix A. Figures and Tables

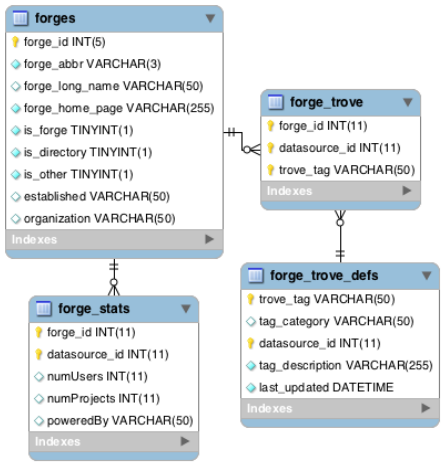


**Figure 1: Data Model**

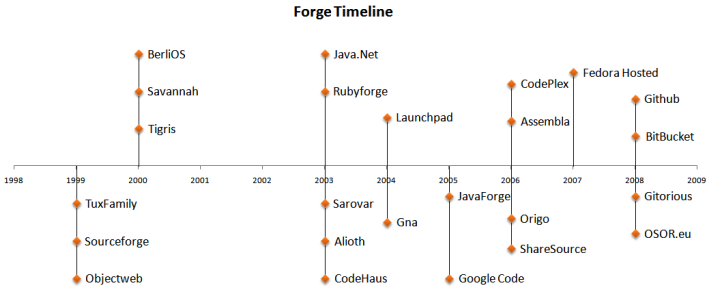


**Figure 2: Timeline of Forge Creation**

| Forge | Founded | Organization | Software | Num Projects | Num Users |
|---|---|---|---|---|---|
| Alioth | 2003 | Debian.org | FusionForge | 920 | 11053 |
| Assembla | 2006 | Assembla, LLC | Assembla | *NULL* | 300000 |
| BerliOS | 2000 | Fraunhofer FOKUS | FusionForge | 3600 | 0 |
| BitBucket | 2008 | Atlassian | *NULL* | *NULL* | 138000 |
| CodeHaus | 2003 | *NULL* | *NULL* | 288 | *NULL* |
| CodePlex | 2006 | Microsoft | *NULL* | 22559 | *NULL* |
| Fedora Hosted | 2007 | Red Hat, Inc. | trac | *NULL* | *NULL* |
| Github | 2008 | Github, Inc. | *NULL* | 2119817 | 758300 |
| Gitorious | 2008 | Shortcut AS | Gitorious | *NULL* | *NULL* |
| Gna | 2004 | FSF France | Savane | 1353 | 15199 |
| Google Code | 2005 | Google | *NULL* | 208664 | *NULL* |
| Java.Net | 2003 | Oracle | Kenai | 2057 | 604505 |
| JavaForge | 2005 | Intland Software | CodeBeamer | 494 | 23000 |
| Launchpad | 2004 | Canonical, Ltd. | *NULL* | 22912 | *NULL* |
| Objectweb | 1999 | OW2 | gforge | 191 | 19813 |
| Origo | 2006 | ETH Zurich | Origo | 6535 | *NULL* |
| OSOR.eu | 2008 | European Commission | gforge | 225 | 4696 |
| Rubyforge | 2003 | Ruby Central | gforge | 9206 | 88985 |
| Sarovar | 2003 | Linuxense.com | gforge | 514 | 69381 |
| Savannah | 2000 | Free Software Foundation, Inc. | Savane | 3334 | 51709 |
| ShareSource | 2006 | ShareSource | ShareSource | 600 | *NULL* |
| Sourceforge | 1999 | Geeknet, Inc. | *NULL* | 291600 | 2700000 |
| Tigris | 2000 | CollabNet | *NULL* | 799 | *NULL* |
| TuxFamily | 1999 | TuxFamily | vhffs | *NULL* | *NULL* |

**Table 1: Forges studied (SQL shown in Appendix B, Listing 3)**

| Forge | Est. | New Projects/Year | Approval? |
|---|---|---|---|
| Github | 2008 | 706606 | |
| Google Code | 2005 | 34777 | |
| Sourceforge | 1999 | 24300 | |
| CodePlex | 2006 | 4512 | |
| Launchpad | 2004 | 3273 | |
| Origo | 2006 | 1307 | |
| Rubyforge | 2003 | 1151 | approval |
| BerliOS | 2000 | 327 | approval |
| Savannah | 2000 | 303 | approval |
| Java.Net | 2003 | 257 | approval |
| Gna | 2004 | 193 | approval |
| ShareSource | 2006 | 120 | |
| Alioth | 2003 | 115 | approval |
| JavaForge | 2005 | 82 | |
| OSOR.eu | 2008 | 75 | approval |
| Tigris | 2000 | 73 | approval |
| Sarovar | 2003 | 64 | approval |
| CodeHaus | 2003 | 36 | approval |
| Objectweb | 1999 | 16 | approval |
| TuxFamily | 1999 | *NULL* | approval |
| Assembla | 2006 | *NULL* | |
| BitBucket | 2008 | *NULL* | |
| Fedora Hosted | 2007 | *NULL* | approval |
| Gitorious | 2008 | *NULL* | |

**Table 2: Average num. new projects / year at forges, and approval policy (SQL shown in App. B, Listing 4)**

| forge_long_name | numProjects | directory | metadata | artifacts |
|---|---|---|---|---|
| Github | 2119817 | 0 | 3 | 1 |
| Sourceforge | 291600 | 1 | 15 | 8 |
| Google Code | 208664 | 1 | 7 | 4 |
| Launchpad | 22912 | 1 | 10 | 4 |
| CodePlex | 22559 | 1 | 9 | 6 |
| Rubyforge | 9206 | 1 | 16 | 7 |
| Origo | 6535 | 0 | 6 | 3 |
| BerliOS | 3600 | 1 | 16 | 5 |
| Savannah | 3334 | 1 | 7 | 4 |
| Java.Net | 2057 | 1 | 7 | 5 |
| Gna | 1353 | 1 | 7 | 4 |
| Alioth | 920 | 1 | 15 | 5 |
| Tigris | 799 | 1 | 6 | 5 |
| ShareSource | 600 | 1 | 7 | 4 |
| Sarovar | 514 | 1 | 16 | 5 |
| JavaForge | 494 | 1 | 7 | 5 |
| CodeHaus | 288 | 1 | 1 | 2 |
| OSOR.eu | 225 | 1 | 17 | 7 |
| Objectweb | 191 | 1 | 13 | 7 |
| Assembla | *NULL* | 1 | 4 | 7 |
| TuxFamily | *NULL* | 1 | 5 | 0 |
| Fedora Hosted | *NULL* | 1 | 1 | 3 |
| BitBucket | *NULL* | 0 | 1 | 3 |
| Gitorious | *NULL* | 1 | 5 | 2 |

**Table 3: Factors contributing to the relative benefit of collecting from forges (SQL in App. B, Listing 5)**

## Appendix B. SQL Statements (for interacting with this data on FLOSSmole.org)

**Listing 1: SQL used to build Table 1**

```
SELECT f.forge_long_name,
substring(f.established,1,4) as 'year',
f.organization, fs.poweredby,
fs.numProjects, fs.numUsers
FROM forges f
NATURAL JOIN forge_stats fs
WHERE is_forge = 1
ORDER BY 1;
```

**Listing 2: SQL used to build list of projects meeting example criteria specified in section 4.1**

```
SELECT f.forge_id, f.forge_long_name
  FROM forges f
NATURAL JOIN forge_trove ft
  WHERE ft.trove_tag = 'flossonly'
AND f.forge_id
IN (SELECT forge_id
  FROM forge_trove
  WHERE trove_tag = 'cvs')
AND f.forge_id
IN (SELECT forge_id
  FROM forge_stats
  WHERE numProjects >1000)
AND f.forge_id
IN (SELECT forge_id
  FROM forge_trove
  WHERE trove_tag = 'mailinglistarchive')
AND f.forge_id
IN (SELECT forge_id
  FROM forge_trove
  WHERE trove_tag = 'bugtrackerarchive')
AND f.forge_id
IN (SELECT forge_id
  FROM forge_trove
  WHERE trove_tag = 'directory')
AND f.forge_id
IN (SELECT forge_id
  FROM forge_trove
  WHERE trove_tag = 'internalURL');
```

*Resulting forges from this query: Sourceforge, Rubyforge, Savannah, Berlios, Gna.*

**Listing 3: SQL used to build Figure 2**

```
SELECT f.forge_long_name, substring(f.established,1,4) as 'year'
FROM forges f
WHERE f.is_forge = 1
ORDER BY 2;
```

**Listing 4: SQL used to build Table 2**

```
SELECT ft.forge_id, f.forge_long_name,
substring(f.established,1,4)
  as 'year',
ROUND(fs.numProjects/(2011-
substring(f.established,1,4)),0)
  as 'projPerYear',
ft.trove_tag as 'approval?'
FROM forges f
NATURAL JOIN forge_trove ft
NATURAL JOIN forge_stats fs
WHERE ft.trove_tag = 'approval'
UNION
SELECT distinct(ft1.forge_id),
f1.forge_long_name,
substring(f1.established,1,4)
  as 'year',
ROUND(fs1.numProjects/(2011-
substring(f1.established,1,4)),0)
  as 'projPerYear',
"" as 'approval?'
FROM forges f1
NATURAL JOIN forge_trove ft1
NATURAL JOIN forge_stats fs1
WHERE ft1.forge_id NOT IN(
SELECT forge_id FROM forge_trove WHERE
trove_tag = 'approval')
ORDER BY 4 DESC;
```

**Listing 5: SQL used to build Table 3**

```
SELECT forge_long_name,
numProjects,
count(if(trove_tag = 'directory',
1,NULL)) as 'directory',
count(if(trove_tag IN(
SELECT trove_tag
FROM forge_trove_defs
WHERE tag_category = 'project
metadata'), 1, NULL)) as 'metadata',
count(if(trove_tag IN(
SELECT trove_tag
FROM forge_trove_defs
WHERE tag_category = 'artifact'), 1,
NULL)) as 'artifacts'
FROM forge_trove
NATURAL JOIN forge_stats
NATURAL JOIN forges
GROUP BY 1, 2
ORDER BY 2 DESC;
```