Data Sets: The Circle of Life in Ruby Hosting, 2003-2015

Megan Squire
Elon University
Elon, NC 27244 USA
msquire@elon.edu

ABSTRACT
Many software development teams use web-based hosting services to help create, distribute, and maintain their software. Studying the way these hosting services are used can provide valuable insights into the behaviors of large groups of software developers and their projects. Traditionally, most analysis of metadata collected from hosting services has been conducted by specifying some short window of time, typically just a few years. To date, few - if any - studies have been built from data comprising the entirety of a forge's lifespan: from its birth to its death, and rebirth. Thus, the first contribution of this data set is to support the historical analysis of over ten years of collected metadata from the now-defunct RubyForge project hosting site, as well as the follow-on successor to RubyForge, the RubyGems hosting facility. The data sets and sample analyses in this paper will be relevant to researchers studying both software evolution and the distributed software development process, in particular those used in highly social coding environments.

CCS Concepts
• Software and its engineering → Collaboration in software development • Software and its engineering → Open source model • Computing Methodologies

Keywords
FLOSS, data, RubyForge, RubyGems, repository, historical.

1. INTRODUCTION
In the early 2000s, a rapid increase in the number of geographically and temporally distributed software teams meant that developers needed to perform their work in an asynchronous, location-neutral way [1]. Early project hosting services, such as SourceForge and GNU Savannah, sometimes called software forges or repositories, provided a low barrier to entry for development teams, in particular open source development teams, by offering space for file downloads, as well as version control systems, mailing list software, wikis, bug tracking software, and so on. Later forges included Google Code, Microsoft CodePlex, Debian's Launchpad, and Github. Numerous research studies have attempted to learn about a given software ecosystem by studying these forges, the projects hosted there, and the developers who participate [2][3][4][5][6][7].

For this paper, we focus specifically on building a large collection of data for two Ruby language hosting services that have not been studied before: RubyForge and RubyGems. Table 1 shows key events in the lives of these sites, including our collection of the data, and its subsequent donation to the FLOSSmole project. After describing our data collection and data model, this paper shows how our data can be used to provide a basic, detailed analysis of these two forges, both comparing RubyForge to itself as it changed over a decade, and also comparing RubyForge to its successor RubyGems. Finally, we conclude with some ways that this data can be analyzed in the future.

Table 1. Events in the transition from RubyForge to RubyGems

Date	Event
July 2003	RubyForge is launched
July 2006	First FLOSSmole collection of RubyForge data
April 2008	Github starts gem building services [8, 9]
Summer 2008	Gem hosting debate: RubyForge vs. Github [10]
April 2009	Nick Quaranto begins building Gemcutter as an alternative to gems.github.com [11]
Aug. 2009	Gemcutter launched, RubyForge gems imported [12]
Oct. 2009	Gemcutter is the new official default gem host [13], Github stops building gems [14], RubyForge to be phased out. [15]
Feb. 2010	Gemcutter becomes RubyGems.org, officially replaces RubyForge as gem host. [16]
Nov. 2013	Announcement RubyForge to be shut down. [17]
Dec. 2013	RubyForge project data frozen. No new changes.
May 2014	RubyForge.net shut down. Last FLOSSmole collection of RubyForge.
Nov. 2015	First FLOSSmole collection of RubyGems

2. DATA COLLECTION AND STORAGE
2.1 Data Collection
Our RubyForge data sets were made from 58 different automated collection sessions, conducted between April 2006 and May 2014. The RubyGems data sets were collected beginning in August 2015. The raw RubyForge and RubyGems data sets have been donated to the FLOSSmole repository [18], both as MySQL database tables, as well as in downloadable flat files [19,20]. To build the data sets initially we followed these three steps:

1. Generate a master list of projects from sites (RubyForge and RubyGems);
2. For each project on that list, collect the project's home page and store the HTML (RubyForge) or XML (RubyGems) for that complete page in a MySQL database;
3. Parse HTML/ XML for facts, and store those in the database;
4. Repeat steps 2 & 3 periodically until site goes defunct (for 10 years in the case of RubyForge; RubyGems still ongoing).

For both RubyForge and RubyGems, in step 2 we only collected public-facing data, such as what anyone would see through a non-authenticated web browser. It may be helpful to some readers to know that the RubyGems site now makes available PostgreSQL dumps of some of their site data on a weekly basis [21].

2.2 Data Models
Tables 2 and 3 show the database tables and column names of the collected, parsed, and stored RubyForge and RubyGems data. Each time data is collected, it is timestamped and added to the

data that is already there, thus providing a long-term archive of how the site looked at each moment in time.

Table 2. RubyForge Database Tables

Table Name	Description
rf_project_indexes	Unparsed, raw index html files.
rf_projects	Project name and basic facts about the project (e.g. date registered, URL)
rf_developers	List of developers currently working on any project in the RF system.
rf_developer_proj.	Connects developers to projects.
rf_project_desc.	Textual description of project.
rf_environment	Computing environment(s) of project.
rf_intended_aud.	Intended audience(s) of project.
rf_licenses	License(s) assigned to the project.
rf_natural_lang.	Natural language(s) of project.
rf_operating_sys.	Operating system(s) of project.
rf_prog._lang	Programming language(s) of project.
rf_status	Project status, e.g. Alpha, Beta, Production.
rf_topic	Project topic(s), e.g. Games, Rails, Internet.

Table 3. RubyGems Database Tables

Table Name	Description
rg_project_pages	Unparsed, raw html and RSS files.
rg_project_facts	Gem name and basic facts about each gem, from the gem's main page.
rg_project_owners	Owner(s) of gem
rg_project_authors	Author(s) of gem
rg_project_rtdep	Gems claimed as runtime dependencies
rg_project_devdep	Gems claimed as dev dependencies.
rg_project_links	Links for a gem (e.g. homepage).
rg_project_versions	Versions of gem that have been released.
rg_proj._create_dates	First known release date for each gem.

3. DATA ANALYSIS

In this section we give some basic examples of site-level, project-level, and person-level analyses that can be performed with this data.

3.1 Site-level Analysis

Counting Projects: RubyForge. The final number of projects that were being hosted on RubyForge at the time of its shutdown was 9,603. Figure 1 confirms that the rate of RubyForge project creation slows dramatically when Gemcutter was announced. Figure 2 shows new project registrations each month. We again see a sharp drop-off in new project registrations, corresponding to the release of Gemcutter.

Counting Projects: RubyGems. As of December 15, 2015 there are 109,822 gems on RubyGems.org. To find their creation dates, we used the list of gem versions that we had collected for each gem, and found the earliest version listed for each, removing those with errors for dates (for example, they had years of 1970 or 2051). Figure 3 shows the results.

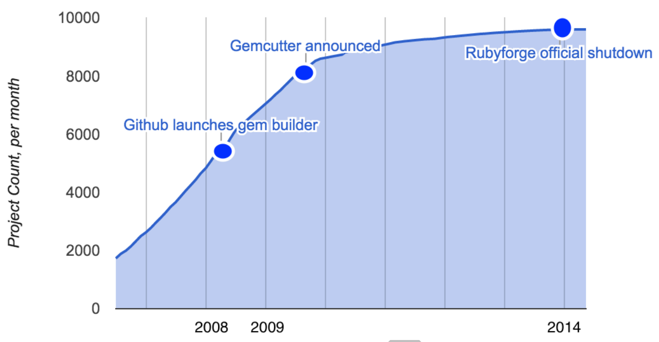

Fig. 1. Count of projects hosted on RubyForge 2006-2014

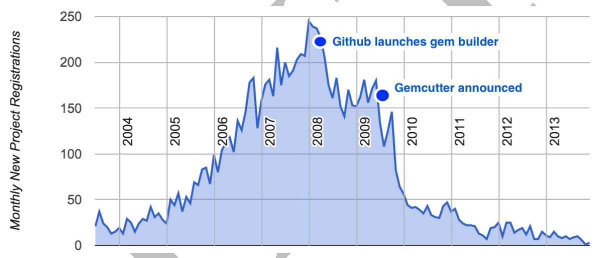

Fig. 2. Rubyforge new project registrations/month 2003-2013

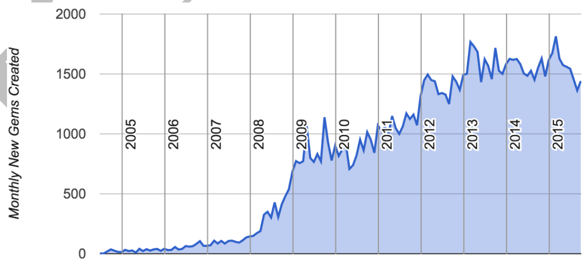

Fig. 3. RubyGems project registrations/month 2004-2015

3.2 Project-level Analysis

RubyForge Project Characteristics. The data set includes the types of environments, intended audiences, natural languages, operating systems, and topics that the project administrators chose to describe their projects. We compare these from our first collection in 2006 until the last collection in 2014. In Table 4, for space reasons, we have shown only the operating systems category and topics categories.

Table 4. RubyForge (Selected) Category Changes

2006				2013		
Operating Systems				Op. Systems		
	Count	% Total	% OS	Count	% Total	% OS
OS Ind.	620	53%	69%	2151	22%	68%
Linux	147	13%	16%	501	5%	16%
POSIX	113	10%	13%	412	4%	13%
Windows	77	7%	9%	278	3%	9%
OS X	59	5%	7%	239	2%	8%
Project Topics				Project Topics		

2006				2013		
	Count	% Total	% Top	Count	% Total	% Top
SW Dev.	206	18%	20%	734	8%	20%
Rails	90	8%	9%	403	4%	11%
Dynamic	57	5%	6%	148	2%	4%
WWW	51	4%	5%	261	3%	7%

License Use Over Time. There has been some hand-wringing of late in the FLOSS community about whether or not people are relying less on licensing in general [22], whether this is a good or bad thing [23], and whether it is the fault of Github [24]. Since we have a longitudinal data set for RubyForge, we can add to this conversation by showing whether unlicensed projects did in fact increase in number over the life of RubyForge (thus helping to confirm a general decline in people wanting to license their software), or whether the numbers of unlicensed projects stayed relatively steady for the life of the forge. Tables 5 and 6 show the overall license choice distribution for the two sites. By 2013, both sites show around 60% of projects using no license.

Table 5. RubyForge Licenses, 2013

License	Count	% Total	% of Licensed
None	5881	61%	-
MIT/X	1212	13%	33%
Ruby	670	7%	18%
GPL v2	661	7%	18%
BSD	348	4%	9%
GPL v3	298	3%	8%
LGPL	257	3%	7%

Table 6. RubyGems Licenses, 2015

License	Count	% Total	% of Licensed
None	61152	56%	-
MIT/X	40840	13%	33%
Apache 2	1708	2%	4%
BSD	558	.5%	1%
GPL v3	518	.5	%

However, Figure 4 shows the percentage of RubyForge projects that chose to specify a license, over time. In 2006, about 60% of projects were licensed, and the most popular license was the GNU Public License (GPL) version 2. In 2013, at the end of the RubyForge life span, only about 39% of projects were licensed, and the most popular choice was the permissive MIT license.

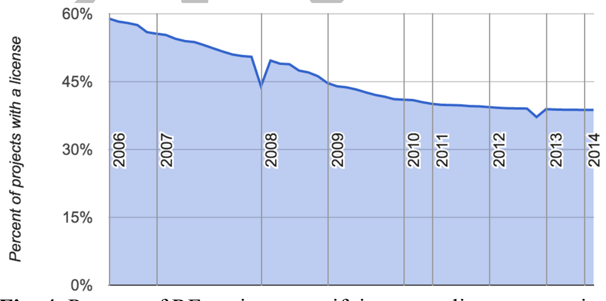

Fig. 4. Percent of RF projects specifying some license over time

3.3 Person-level Analysis

In this section we use the RubyForge and RubyGems data sets to learn about project teams. For example, from RubyForge, we can learn about the number of projects per developer, and discover the length of time between a developer's first and last created project. For RubyGems, we examine the number of owners and authors per gem, as well as how many gems each owner and author is listed as working on.

RubyForge: projects per developer. Here we will focus our attention on the last collection, since that data set has the most data in it (9,603 projects and 7,105 developers actually assigned to at least one project). The highest number of projects for a developer was 33, and only 77 developers worked on more than 10 projects. Of those developers with more than 10 projects, were they active for the entire lifespan of RubyForge? Were they early adopters who left during the 2009 timeframe like other users? Or did those frequent users continue to join projects late in the lifespan of the site? Figure 5 shows each developer with 10 or more projects, all the projects they worked on, and what date the project was created.

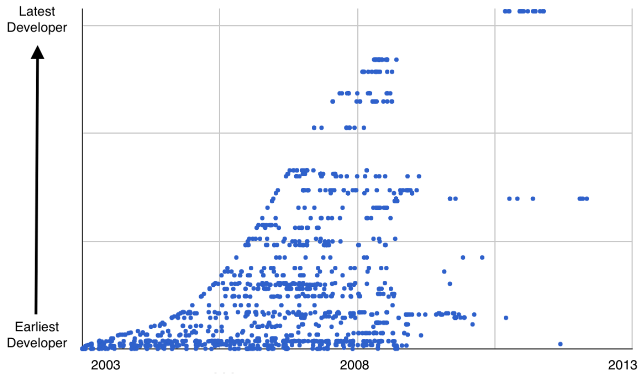

Fig.5. RubyForge developer create dates (>10 projects)

Figure 5 reveals that the vast majority of the developers working on 10 or more projects got started very early in the RubyForge lifespan. In fact, very few of them got started on RubyForge after 2007. Also, we can see that these heavy users / early adopters stopped creating new projects when Gemcutter was announced in 2009. Figure 6 shows the same developers with 10 or more projects, along with the length of time between when they created their first and last project. Users who joined later have more tightly spaced first and last projects than users who joined earlier.

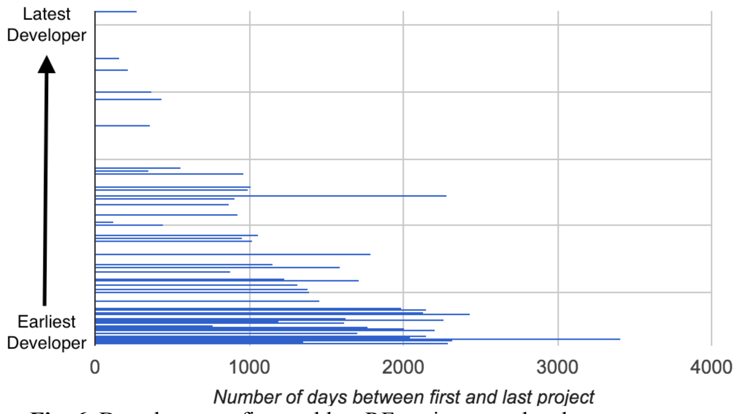

Fig. 6. Days between first and last RF project, per developer

RubyGems: Counts of Developers and Gems. Counting gems per developer on RubyGems is a little different since the nomenclature is not exactly the same as on RubyForge. On the RubyGems site, there are both gem authors and gem owners. Somewhat perplexingly, the RubyGems interface shows a username and gravatar for owners, but a regular text name for authors. For example, the owner with the highest number of gems is 'jrobertson' with 202 gems, but to find this user in the authors table requires knowing that his full name is 'James Robertson' (198 gems). Thus, because of this discrepancy, we have stored collected owners and authors in separate tables (as shown in Table 2). In the future we will need to develop a procedure for linking owners and authors.

4. FUTURE WORK AND CONCLUSIONS

As is usually the case with a plethora of longitudinal data, asking the first question leads to more and more questions. The RubyForge and RubyGems data sets are very rich and their long-term nature is very unusual for forge studies.

Questions for future work on RubyForge could include building social networks of developers over time, studying evolving team sizes, and looking at markers of quality in teams and projects. For RubyGems, we would like to study the interdependence of gems, as measured in their development dependencies and runtime dependencies, as well as their popularity. It will also be interesting to study the extensive version information available on RubyGems, for example counting versions per gem, or looking at release patterns such as days of the week or day-to-day release patterns in a given year. Connecting RubyForge projects to the equivalent gem on RubyGems is another challenge that should be tackled in the future. The advantage of connecting these is that we will have a continuous, unbroken chain of events for those projects that existed on both sites. Techniques that are used for entity matching [25] in this way can be applied to other long-term forges, such as SourceForge (which is over 10 years old as well) or, eventually, to Github.

This paper presents two new data sets, a decade-long collection of metadata from the now-defunct RubyForge project hosting site, and the beginnings of a similar collection of data for its successor, RubyGems. We present a description of how to access and use the data, including some basic descriptive analyses of the data. Some of these analyses are only possible because the data set is longitudinal, for example knowing how license choices changed over a long period, or being able to see the patterns in project creation over a decade of the site's existence. This long-term, birth-to-death-to-rebirth focus is unusual among forge data sets. These data sets will allow researchers to be begin to design techniques for tracking projects over many years of development, and even as they move between forges.

5. ACKNOWLEDGMENTS

We gratefully acknowledge National Science Foundation (grant number NSF-14-05643) for helping to support this work.

6. REFERENCES

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

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

[3] Delorey, D.P., *et al*. Programming language trends in open source development: An evaluation using data from all production phase SourceForge projects. In *Proc. 2nd Workshop Public Data Software Dev. (WoPDaSD)*. Limerick, Ireland (2007).

[4] Krein, J.L., *et al*.. Impact of programming language fragmentation on developer productivity. *Int. J. Open Source Sw. & Proc, 2(2)*. 41-61. (2010)

[5] Krein, J.L., *et al*. Language entropy: A metric or characterization of author programming language distribution. In *Proc. 4th Workshop Public Data Software Dev. (WoPDaSD)*. Skovde, Sweden (2009).

[6] Vendome, C. A large scale study of license usage on GitHub,, In *Proc. 37th Int. Conf. Softw. Eng. (ICSE)*, 2, 772-774. (2015)

[7] Vasilescu, B., *et al*. Gender and tenure diversity in GitHub teams. *In Proc. CHI*. ACM. (2015)

[8] Github's RubyGem server, https://github.com/blog/51-github-s-rubygem-server

[9] History of the canonical gem host for Ruby gems, http://www.rubycoloredglasses.com/2012/05/history-of-the-canonical-gem-host-for-ruby-gems/ (2012)

[10] Gems from RubyForge and Github, http://www.infoq.com/news/2008/08/gems-from-rubyforge-and-github

[11] About RubyGems, https://rubygems.org/pages/about

[12] Roberts, R. Gemcutter A fast and easy approach to Ruby gem hosting. *RubyInside*. Aug 20. http://www.rubyinside.com/gemcutter-a-fast-and-easy-approach-to-ruby-gem-hosting-2281.html (2009)

[13] Cooper, P. Gemcutter is the new official default ruby gem host. *RubyInside.* Oct 26. http://www.rubyinside.com/gemcutter-is-the-new-official-default-rubygem-host-2659.html (2009)

[14] Wanstrath, C. Github gem building is defunct. Oct 8. https://github.com/blog/515-gem-building-is-defunct (2009)

[15] Schuster, W. RubyForge to be phased out. *InfoQ*. Oct. 26. http://www.infoq.com/news/2009/10/rubyforge-phased-out-rubygemsorg (2009)

[16] Blair, P. RubyGems.org Replaces RubyForge as Gem Host. *InfoQ*. Mar 30. http://www.infoq.com/news/2010/03/rubygems (2010)

[17] Evan Phoenix tweet, https://twitter.com/evanphx/status/399552820380053505 (2013)

[18] FLOSSmole, http://flossmole.org

[19] RubyForge Data on FLOSSmole, http://flossdata.syr.edu/data/rf/

[20] RubyGems Data on FLOSSmole, http://flossdata.syr.edu/data/rg/

[21] Rubygems.org data dumps, https://rubygems.org/pages/data

[22] Villa, L. Younger developers reject licensing, risk chance for reform. Feb 13. https://opensource.com/law/13/2/post-open-source-software-licensing (2013)

[23] Phipps, S. Why all software needs a license. *InfoWorld*. November 7. http://www.infoworld.com/article/2839560/open-source-software/sticking-a-license-on-everything.html (2014).

[24] McAlister, N. Study: Most projects on GitHub not open source licensed. *The Register.* April 13.

[25] Squire, M. Integrating projects from multiple open source code forges. *Int. J. Open Source Software & Proc.*, 1(1) 46-57 (2009).