The Lives and Deaths of Open Source Code Forges

Megan Squire
Elon University
Elon, North Carolina, USA
msquire@elon.edu

ABSTRACT
Code forges are third party software repositories that also provide various tools and facilities for distributed software development teams to use, including source code control systems, mailing lists and communication forums, bug tracking systems, web hosting space, and so on. The main contributions of this paper are to present some new data sets relating to the technology adoption lifecycles of a group of six free, libre, and open source software (FLOSS) code forges, and to compare the lifecycles of the forges to each other and to the model presented by classical Diffusion of Innovation (DoI) theory. We find that the observed adoption patterns of code forges rarely follow the DoI model, especially as larger code forges are beset by spam and abuse. The only forge exhibiting a DoI-like lifecycle was a smaller, community-managed, special-purpose forge whose demise was planned in advance. The results of this study will be useful both to practitioners building collaborative FLOSS ecosystems, such as code forges, and to researchers who study the evolution and adoption of socio-technical systems.

Author Keywords
Open source; free software; FLOSS; code forge; diffusion of innovations; technology adoption; RubyForge; Google Code; SourceForge; ObjectWeb; CodePlex; Github; software evolution

ACM Classification Keywords
D.2.9. SOFTWARE ENGINEERING: Management; H.3.5 INFORMATION STORAGE AND RETRIEVAL: Online Information Services; Data sharing.

INTRODUCTION
Because teams of developers of free, libre, and open source software (FLOSS) projects are often geographically distributed around the world, many teams choose to structure their work in an asynchronous, location-neutral way. Early web-based FLOSS hosting services, such as SourceForge and GNU Savannah, offered features such as file downloads, version control systems, mailing list software, wikis, bug tracking software, and so on. In the early years of the FLOSS phenomenon, these *code forges* served an important role for developers by providing a low barrier to entry to coordinate team work, and they served end-users by providing a centralized place to find and communicate about a variety of different FLOSS projects.

By the mid-2000s, larger software companies began to create their own software forges, such as Google Code and Microsoft CodePlex. Non-commercial special-purpose forges were also created during this time frame, for example RubyForge was designed for projects written in a particular programming language (Ruby) and the ObjectWeb forge was designed for FLOSS middleware projects. Github was launched in 2008 to offer version control and some basic features such as wikis and file downloads, and is now by far the largest centralized software forge with over 21 million user accounts and 57 million repositories as of this writing. [1]

Though their intended audiences may differ, and the services provided by each code forge may be slightly different, the purpose of all FLOSS code forges is to host projects. Each time a project owner "chooses" to host their particular project on a code forge, this action is an indication that the code forge is still relevant in some way. Some of the oldest forges are still accepting new projects, while others have closed, merged, or otherwise transformed themselves as the FLOSS phenomenon has changed and matured. What do the project hosting rates look like in the years between a code forge's birth and its death? Do the code forges follow the same adoption or "diffusion" patterns found in other technologies, as project owners choose to adopt the technology or move to something else?

For this paper, we compare longitudinal data from six code forges to a "typical" technology adoption curve as presented in classical Diffusion of Innovation (DoI) theory. Basic models for technology diffusion were first described by Everett Rogers [2], who proposed a life cycle consisting of early adoption, adoption by the majority, then late adoption ("laggards"), and ultimately either discontinuance of the product or a saturated market. He applied this model to a variety of technologies in a variety of industries, refining the model to show that diffusion could be affected by many factors including social interactions and organizational dynamics. With time plotted on the x-axis and adoption of an innovation shown on the y-axis, Rogers proposed that the typical diffusion of an innovation over time will likely

resemble a normal distribution (or an S-curve if plotted with a cumulative x-axis). Figure 1 shows the two "typical" DoI technology adoption curves.

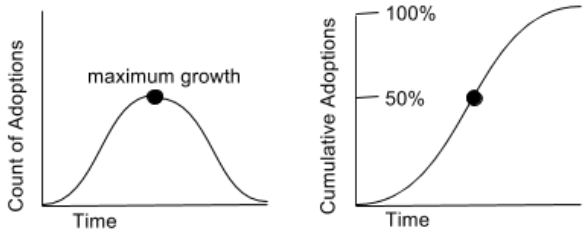

Figure 1. "Typical" Diffusion of Innovations (DoI) curves with periods of maximum growth shown (after Rogers, 1964).

In assessing the adoption or lifecycle of a technology, the steepness or shallowness of the S-curve is interesting, as is the point at which the innovation declines following a period of maximum adoption. By studying the adoption curves, we may find that some code forges may reach their peak (maximum adoption) earlier or later than expected. Some forges may be kept alive well past their expected lifespan. For code forges that are not dead yet, a partial adoption curve may exhibit clues for what is to come.

Thus, this paper begins a data-driven, historical analysis of the diffusion/adoption curves of code forges. We study six in detail: RubyForge, Google Code, SourceForge, CodePlex, ObjectWeb, and Github. Our questions are:

- RQ1: What do the adoption curves for each code forge look like?
- RQ2: What factors, if any, alter the curves or affect the adoption patterns between the forges?
- RQ3: Do all code forges exhibit the same patterns of birth, growth, and death as would be expected from traditional DoI theory?

To answer RQ1, for each of six code forges, we gather metrics to describe its adoption rate and we plot the adoption rate graphically. For RQ2, we outline the various details (e.g. spam) that may explain the shapes of the curves. To answer RQ3, we compare the shape of these curves to what DoI theory would predict and discuss the possible reasons for any differences. Finally, we explore the limitations of this work and present ideas for how to advance this work in the future.

TECHNOLOGY ADOPTION DATA

FLOSSmole [3] data is used in this paper to describe the adoption rates - via new project registrations - of RubyForge, Google Code, CodePlex, and ObjectWeb. SourceForge Research Data Archive (SRDA) [4] data is used to describe the lifecycle of SourceForge. GHTorrent data [5] is used to describe the lifecycle of Github. For each forge, the data, queries, and calculations used in this paper are available for download in the FLOSSmole data repository, at http://flossdata.syr.edu/data/forgeStudies/2017deathOfForges.

RubyForge

RubyForge was launched on July 16, 2003 as a Ruby language-specific hosting site for FLOSS projects. It included collaboration tools such as file downloads, source code control software, bug tracking, and mailing lists. Project-level metadata collected from the 10 years of RubyForge's existence was gathered and described in our prior work [6]. Examples of project-level metadata we collected for RubyForge includes: project name, project owners/developers, project description, project license, project registration date, and so on.

Figure 2 shows a visualization of the RubyForge monthly new project registrations found in the [6] data set, beginning with its launch in 2003 through its shutdown in 2014. The dates of two important events in RubyForge's history are overlaid on the graph: the launch of Github's gem builder in 2008, and the 2009 launch of Gemcutter (eventually renamed RubyGems). Github was a significant competitor to RubyForge, and Gemcutter/RubyGems was specifically designed by the RubyForge team to be a replacement for RubyForge. As the graph shows, the years of most intense growth at RubyForge were between 2006-2009.

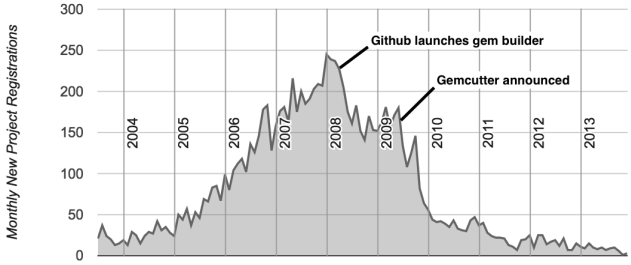

Figure 2. Monthly new project registrations on RubyForge, 2003-2013, with key dates shown

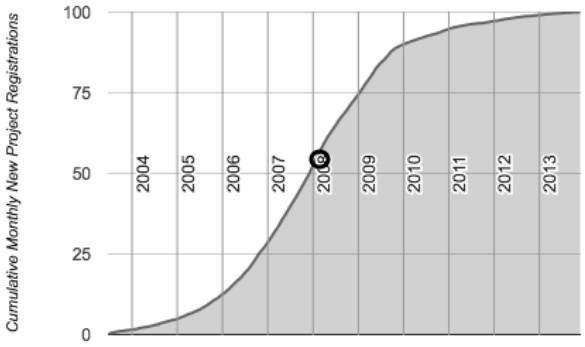

Figure 3. Cumulative monthly new project registrations on RubyForge, 2003-2013, with month of maximum growth shown (January 2008)

Figure 3 shows the same data, but with monthly contributions comprising a cumulative total. This graph shows that RubyForge began its decline at 52% saturation

of the "market". Following the launch of a competitive site (Github) and its own planned replacement (Gemcutter/RubyGems), RubyForge began a slow decline. RubyForge was eventually closed to new projects at the end of 2013, and shuttered for good in May of 2014, having hosted a total of 9,898 projects over its lifetime.

Google Code

Google Code (http://code.google.com) was launched in 2006 as a free-of-charge hosting site for any software project using an OSI-approved FLOSS license. The site offered basic hosting services, including wikis, source code control software, bug tracking services, and file downloads. FLOSSmole began collecting project-level metadata from Google Code in 2010, and continued to collect this data through the closure of Google Code in 2015.

Unlike RubyForge, project creation dates were never published on Google Code's public site as part of the project-level metadata. However, on June 25, 2015 Google Code donated to FLOSSmole a list of projects and their creation dates [7]. Unfortunately, this data reveals that the registration date field was first added to the Google Code system on February 4, 2011, so only projects created in Google Code system after that date have a recorded registration date. Thus, we will only be able to show the rate of new project registrations for 2011-2015, or only the latter half of the Google Code lifespan.

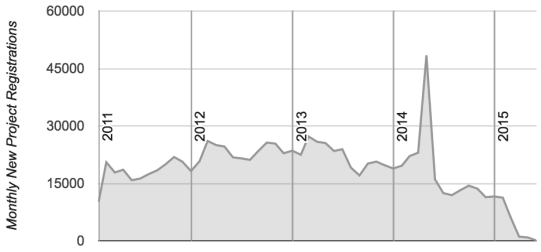

Figure 4. Monthly new project registrations on Google Code, 2011-2015

Figure 4 illustrates the rate of new projects being added to Google Code from February 2011 through its closure in March 2015. Monthly new project registration counts appear to have been fairly steady between 2011 and the beginning of 2014, with about 15,000-25,000 new projects created per month. However, in May 2014, a startling 48,000 new projects were created. It seems that spammers were using automated project creation scripts ("bots") to create fake projects on Google Code for the purpose of search engine optimization (SEO). The *File Downloads* feature in Google Code had been similarly abused by malware and illegal file distributors a year before [8], resulting in that feature being removed entirely for all users. As Figure 4 shows, in the months following the May 2014 spam deluge, anti-spam measures were put in place, but new project creations never again reached their pre-deluge levels. Nonetheless, on March 12, 2015 Chris DiBona announced [9] that the code forge would shut down completely. He explained,

"As developers migrated away from Google Code [to Github], a growing share of the remaining projects were [sic] spam or abuse. Lately, the administrative load has consisted almost exclusively of abuse management."

DiBona directly blames the demise of Google Code on both the spam/abuse problem and the existence of competitive services such as Github.

CodePlex

Microsoft created CodePlex (http://codeplex.com) in 2006 as an open source project hosting facility. It offered several flavors of source code control software, as well as discussion forums, issue tracking, and the like. CodePlex was one of the only code forges to support Microsoft's Team Foundation Server product, which made it a popular choice for developers using TFS-friendly platforms, including Visual Studio. In 2015 many of Microsoft's own developers began hosting their projects on Github [10], and in 2017 the company announced that CodePlex would be closed [11].

At the time of the shutdown announcement, we were able to collect the CodePlex home page and "Change History" pages for 108,619 projects and donate these to FLOSSmole. To estimate the registration dates for these projects we used the Change History page to extract the month and year when the first change was made (month and year are shown, but specific dates are not). Next, to distinguish between spam and legitimate projects, we used the number of all-time downloads as a proxy: projects with no files ever posted, or projects with an all-time count of zero for their last download, are likely spam or test projects.

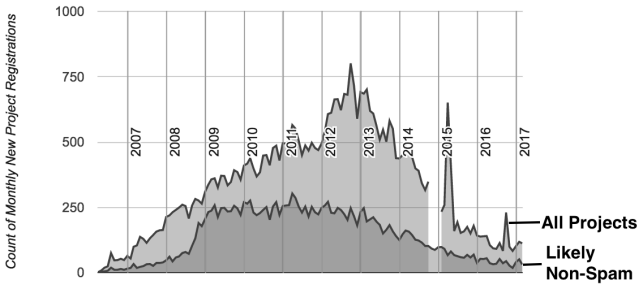

Figure 5. Monthly new project registrations on CodePlex, 2006-2017

Figure 5 shows the project creation rates over time for all projects, and the count of likely non-spam projects overlaid. We had to remove the months' worth of "all projects" data points from the chart (November 2014-January 2015, shown in white on the chart) because the project numbers were so high as to make the chart completely unreadable. In December 2014, for example, 47,945 projects were created. This is more than 130 times the number of projects that were created in a normal month in 2014. After a two-month break in February and March 2015, April 2015 once again

shows a spike in zero-download projects, but this time the surge was only about 3 times greater than surrounding months.

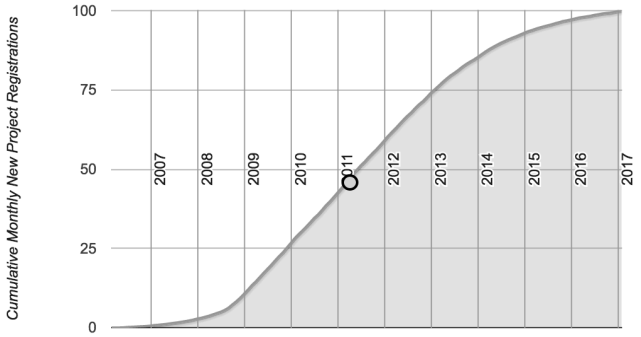

Figure 6. Cumulative monthly new project registrations (only likely non-spam") on CodePlex, 2006-2017, with month of maximum growth shown (April 2011)

We can also view the non-spam new project registrations on a cumulative basis, as shown in Figure 6: how much does each month contribute to the cumulative total of new projects created? Once we have calculated the cumulative contribution of each month to the total, we can attempt to find the last month of growth before the subsequent decline. Figure 6 shows that CodePlex reached its month of maximum growth at about 45% saturation, with steady growth of about 250 likely non-spam projects per month between 2009-2013. The site had a two-year ramp up, but a four-year decline.

SourceForge

SourceForge (http://sourceforge.net) was started in November 1999 as a hosting facility for FLOSS projects. Unlike CodePlex, Google Code, and RubyForge, the site is still in operation as of this writing. SourceForge still offers file downloads, source code control software, mailing lists, discussion forums, wikis, and so on. As with CodePlex and Google Code, SourceForge also allows new project creation by way of a web form, and thus - without taking additional measures - is vulnerable to being hijacked by spam project creation bots.

In 2003 SRDA began publishing a version of the data for researchers [4], but the last time SourceForge donated their data to the SRDA team was in September of 2014 [12]. Thus, in this study, we will only be able to show the rate of monthly new project registrations through September 2014. An additional wrinkle is that it is much more difficult to find spam projects on SourceForge using the downloads metric, as we did with CodePlex, due to inconsistencies in the way data was donated to SRDA over the years [13], and the lamentable fact that there seems to be no accurate mechanism for querying downloads after 2009. Therefore, we differentiate between all projects and and likely non-spam projects using the amount of text in the *description* field, since legitimate projects will include a description, even if it is short. Figure 7 shows the count of monthly new project creations starting in 2000, with counts for all projects and the subset of likely non-spam projects shown overlaid.

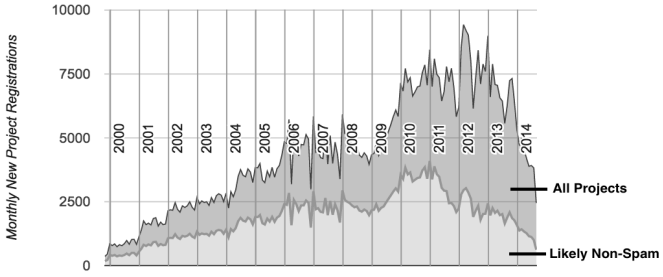

Figure 7. Monthly new project registrations on SourceForge, 2000-2014

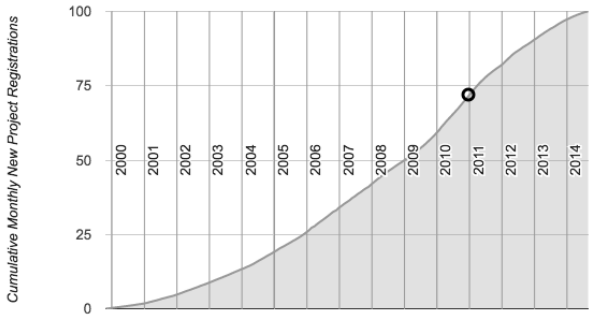

Figure 8. Cumulative monthly new project registrations (only "likely non-spam") on SourceForge, 2000-2014, with month of maximum growth shown (January 2011)

Figure 8 shows the rate of new non-spam project creation on a cumulative basis, with January 2011, the month of highest growth, circled. Recall from Figure 6 that early-2011 was the same time that CodePlex peaked as well.

For the period 2000-2014, SourceForge seems to follow the S-curve birth-growth-decline adoption pattern we saw with RubyForge and CodePlex, but with the added wrinkle of a large amount of spamming, just as we saw with CodePlex and Google Code. Another interesting feature of the SourceForge adoption curve is that there were two periods of growth: the first in 2006-2008, and the second, higher period in 2010-2011. After 2011, we see continued growth in the 'All Projects', but decidedly less growth in the 'Likely Non-Spam' projects. In mid-2013, SourceForge began bundling adware with project downloads [14], an unpopular business decision that may have caused further declines in legitimate, non-spam new project registrations. Again, the SRDA data set only goes through September of 2014, so subsequent activity, including after the company was sold yet again in 2016, cannot be shown.

ObjectWeb/OW2

ObjectWeb was begun in 2002 as a community for FLOSS middleware component projects. In 2006 it merged with

Orientware community and became OW2. Throughout this time period, ObjectWeb/OW2 has hosted a code forge (http://forge.ow2.org/). Compared to the other code forges in this study, ObjectWeb is quite small. Owing to its very specific niche in only hosting FLOSS middleware, in its 15 years of existence ObjectWeb has hosted fewer than 300 projects. Like RubyForge, the ObjectWeb forge uses a version of GForge, a fork of SourceForge's original forge management software. Thus, as with RubyForge and SourceForge (but unlike Google Code and CodePlex), every project's registration date is available as a default piece of metadata. Figure 9 shows the new projects added over time. Note that these are grouped on a quarterly basis rather than monthly because the monthly numbers were so small.

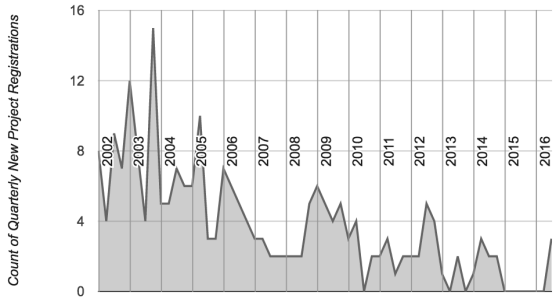

Figure 9. Quarterly new project registrations on ObjectWeb, 2002-2016.

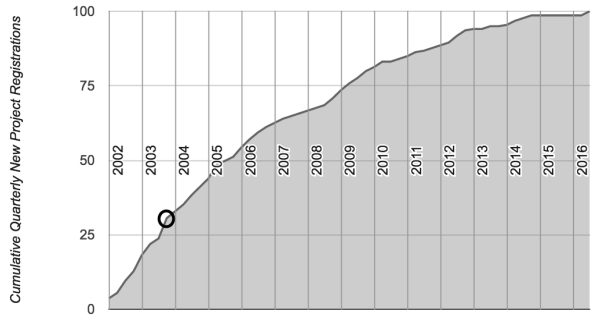

Figure 10. Cumulative quarterly new project registrations on ObjectWeb, 2002-2016, with quarter of maximum growth shown (Q4 2003)

Figure 10 shows the cumulative growth of new project registrations for each quarter. ObjectWeb had its maximum highest quarter early: at the 30% mark. The site registered 15 new projects in 4Q 2003, but since then, no quarter has yielded more than 10 projects. As of this writing, 7 of the last 12 quarters have had zero new projects added.

Github

Github (http://github.com) was launched in 2008 as a free-of-charge source code control repository and project hosting facility. The site offers users the ability to create code repositories, which can be copied, or *forked*, by other users, whose changes can be optionally integrated back into the main code body by way of a *pull request*. The site also offers wikis, issue tracking, code snippets (called *gists*), and documentation hosting.

The fact that Github has no license requirement for software hosted there (i.e. there is no FLOSS license requirement) means it is not strictly limited to hosting FLOSS projects. In fact, Github does not limit itself to hosting software at all. For an apples-to-apples comparison then, this may mean that Github adoption patterns cannot be compared to adoption patterns of the other five forges, all of which were strictly supposed to host FLOSS projects. Nonetheless, with 57 million projects, and both Google Code and CodePlex sending their users to Github after those forges closed, to *not* include Github in this comparison would be an oversight.

Calculating new project registrations on Github is a bit different than doing so on the other code forges. First, Github is built around the idea of a *fork*, or a copy of an existing project. When a fork is created, should we count it as a new project or not? Figure 11 shows the rate of new project creation on Github, with all new project registrations shown on top, and the non-forked projects shown overlaid on the bottom. The data for this chart came from the GHTorrent [5] January 2017 data set.

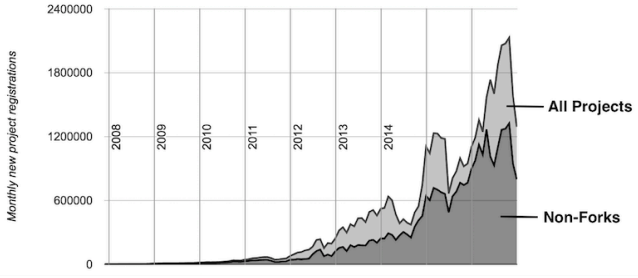

Figure 11. Monthly new project registrations on Github, 2008-2016

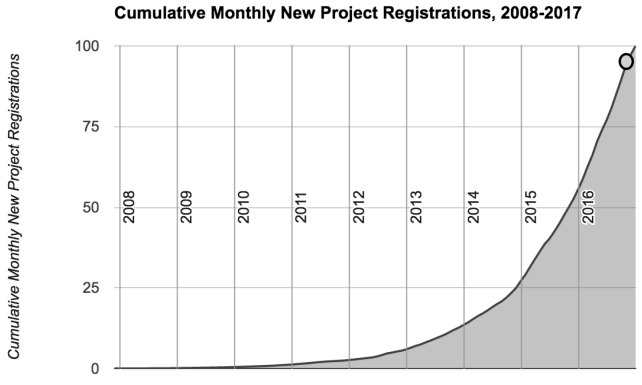

Figure 12. Cumulative monthly new project registrations (non-forks) on Github, 2008-2017, with month of maximum growth shown (November 2016).

There are two particularly interesting parts of Figure 11: the spike in new project registrations (both including forks and not including forks) during the first half of 2015, and a subsequent very large spike across most of 2016 in 'All

Projects' and to a lesser extent in the 'Non-Forks'. Both spikes are immediately followed by steep declines in the numbers of new registrations.

Figure 12 (previous page) shows the cumulative adoption of Github as a code forge, over time. Because Github is still accepting new projects, and has not yet shown a prolonged decline in new project registrations, its cumulative curve is decidedly less S-shaped than the other projects in this study.

DISCUSSION

RubyForge, Google Code, CodePlex, SourceForge, ObjectWeb and Github show very different lifecycle curves. The prior sections helped us begin to answer RQ1 and RQ2: *What do the adoption/diffusion curves for each code forge look like?* And *what factors, if any, alter the curves or affect the adoption/diffusion rates?* In this section, we begin to answer RQ3: *Do all code forges exhibit the same patterns of birth, growth, and death as would be expected from traditional DoI theory?* Here we summarize our findings of what the key differences are between the diffusion patterns exhibited by these different code forges and we compare them to the "typical" DoI adoption pattern.

Maximum Growth Points: Early Versus Late

Comparing the cumulative S-curves (Figures 3, 6, 8, 10, and 12) reveals that the different forges reached their period of maximum growth at different times in their life cycles. SourceForge reached its maximum growth month when 75% of projects were already created. This indicates a slow rise and rapid descent. Compare this to RubyForge, which reached maximum growth at 52%, close to the "expected" S-curve from DoI theory. CodePlex reached its maximum with a saturation of 45% of the total projects, indicating a slightly quicker rise and longer descent than RubyForge. ObjectWeb reached its maximum quite early in its life, at 32% saturation, and has been in a very slow decline since.

But because we lack newer data for SourceForge, and since ObjectWeb, SourceForge, and Github are still alive (i.e. they are still accepting new projects), we will find that the shape of their curves continues to change as time goes on. SourceForge exhibited a "second wind" of sorts in 2011, and (in theory) could again show fluctuations. Certainly this will be the case for Github too, since it is still alive and it only recently had its highest-growth month ever. ObjectWeb is the smallest forge in the study, and it has a very specific, niche mission, so its early rise may simply be an indication that its market was small enough to be easily and quickly saturated.

Spam Spikes: Larger Forges Versus Smaller Forges

When comparing the adoption curves, especially for the larger forges (SourceForge, CodePlex, Google Code), the spikes created by periodic spam attacks are impossible to ignore. These large, general-purpose forges with automated signups all experienced significant spam problems, particularly in the post-2012 time frame. For CodePlex and Google Code, 2014 and 2015 seem to have been particularly bad years. We do not have SourceForge data after 2014, so we cannot say whether the problem continued there as well. However, we do see that RubyForge and ObjectWeb never seemed to experience a spam problem, probably owing to their smaller size. In a small, niche community of only a few hundred or few thousand projects, it would be easy to spot an influx of spam projects.

The Tail: Slow Demise Versus Premature Death

Of the six code forges, Google Code and CodePlex are interesting in that their owners "pulled the plug" following a spam problem. Even with our limited data, Google Code appears to have done so rather abruptly, as it was still attracting many thousands of legitimate projects on a monthly basis. In contrast, our data shows that CodePlex was already very much in decline when Microsoft pulled its plug. It does seem to be the case that the forges with the larger parent companies (Google, Microsoft), where FLOSS is not part of the core business mission, were far less willing to prop up a service that was attracting spam. SourceForge, on the other hand, is still accepting new projects, despite similar spam and abuse woes.

After Death: Diversity Versus Monoculture

Github is without question the leading code forge at this time, regularly adding more projects *per month* than any other forge attracted in its entire lifespan. So it is no surprise that when Google Code and CodePlex died, they both provided migration scripts for existing projects to move themselves to Github. SourceForge also provided import routines so that Google Code and CodePlex projects to move to its servers, however neither Google Code nor CodePlex offered a SourceForge migration or export tool in return.

The size of Github is astounding – with or without the addition of projects from competitor forges – and the impact of such a monoculture on FLOSS practices could be significant. Some of our prior work has addressed the concern that increasing numbers of projects may be hosted on a FLOSS forge but remain unlicensed (or do not specifically choose a FLOSS license).[6] Github itself estimated in 2015 that 82% of its projects are unlicensed.[15] If Github is the most popular place to host a project, and its culture is one of apathy or hostility towards licenses, it is unclear how this could affect FLOSS in the long-term.

LIMITATIONS AND FUTURE WORK

We began this work by introducing a DoI-based methodology for graphing technology adoption rates. We estimated code forge adoption by using monthly (or quarterly) new project registrations. Limitations of this methodology include only having six forges in the study, and having incomplete data for some of the forges (i.e. Google Code lacks project registration dates before February 2011, and SourceForge data is unavailable after September 2014). Future work could address the small

number of forges in the study by identifying other code forges which make project registration data available, including Tigris, Launchpad, and so on. Another limitation of using DoI methods to measure code forge adoption is the very high number of illegitimate projects hosted on some forges. The inclusion of spam projects obscures the true adoption rate, so it is very important to accurately identify spam projects. Future work could include improving our spam detection methods for SourceForge and CodePlex, and designing a method to separate spam from non-spam projects on Google Code.

Another idea for future work would be to identify the Google Code and CodePlex projects that did end up moving to Github. This would allow us to see the impact that their closures had on Github's numbers, if any. In addition, as Github matures, it should remain an active place for future study of code forge adoption patterns. When will Github truly begin to experience a decline in new adoptions? How long will its eventual demise take and what will it look like? Will Github experience a deluge of spam or abuse of the same magnitude as other large forges experienced?

CONCLUSIONS

This paper attempts to begin a historical study of the lifecycles – birth, growth, and sometimes death – of FLOSS code forges. We describe the growth of a forge by tracking how many new projects were added to the forge over time. We compare the growth curves for six code forges (RubyForge, Google Code, CodePlex, SourceForge, ObjectWeb, and Github) to the "typical" or expected normal distribution for technology adoption as presented in classical Diffusion of Innovation (DoI) theory. We find that each code forge presents a different adoption pattern, for different reasons.

- RubyForge is closest to the typical technology adoption curve presented by DoI theory. RubyForge shows growth following a classic bell shape, with a maximum growth period very close to the expected 50% mark. RubyForge was also the only forge in our study to be replaced by a newer product that was created by the same community.
- CodePlex was the next-closest in matching an expected DoI adoption curve, but with a slightly sharper rise and slightly slower demise. It experienced significant periods of spam and abuse near the end of its life.
- SourceForge had a slow birth, a period of growth followed by a "second wind," and then a rapid decline. Like some of the other large forges, it has also experienced significant spam and abuse at different periods in its history. It is the only forge to have been owned by multiple parent companies during its lifespan. But, with an incomplete data set and an ongoing lifespan, further study is needed to see how spam, ownership changes, and the like will affect its long-term adoption patterns.
- ObjectWeb had a very fast birth and growth, and a much slower decline. This forge is extremely small and is still living, so it is harder to draw conclusions.
- The Google Code adoption curve looks nothing like the others, mostly because we are missing data for the first five years of its life. We have no information about what the early adoption rates looked like on Google Code, and the remaining four years were marked by spam spikes followed by a very sudden termination of its life.
- Github is obviously still in the middle of its growth, but its early rise follows the same shape as many of the other forges. The question is whether it has reached a peak or whether it is still rising.

Diffusion of Innovation (DoI) models provide a simple explanation for how technological innovations are adopted over time. In this study, we track the adoption patterns of six different FLOSS code forges and find that while a few of them did indeed follow the classic DoI growth model, outside factors such as spam and abuse can prematurely hasten the death of a code forge or change its growth trajectory. In addition, the size of a forge, its ownership status or whether it is community-managed, and its relative age may also play a part in determining how the forge grows and changes over time.

ACKNOWLEDGEMENTS

We gratefully acknowledge the U.S. National Science Foundation (NSF-14-05643) for helping to support this work.

REFERENCES

1. Github. About. 2017. Retrieved April 18, 2017 from http://github.com/about
2. Everett Rogers. 2003. *Diffusion of Innovations*, 5th Edition. Simon & Shuster.
3. James Howison, Megan Conklin, Kevin Crowston. 2006. FLOSSmole: A collaborative repository for FLOSS research data and analyses. *Int. J. Info. Tech. and Web Engr* 1, 3: 17-26.
4. Yongqin Gao, Matthew Van Antwerp, Scott Christley, Gregory Madey. 2007. A Research Collaboratory for Open Source Software Research. In *Proc of the Int Workshop on Emerging Trends in FLOSS Research and Dev.* (FLOSS 2007).
5. Georgios Gousios. 2013. The GHTorrent Dataset and Tool Suite. *Proc. of the 10th Int. Conf. on Mining Software Repositories.* (MSR 2013). 233-236.
6. Megan Squire. 2016. Data Sets: The Circle of Life in Ruby Hosting, 2003-2015. *Proc of the 13th Int. Conf. on Mining Software Repositories*. (MSR 2016). 452-455.

7. FLOSSmole's Google Code data, http://flossdata.syr.edu/data/gc/2017.
8. Google Project Hosting. 2013. A Change to Google Code Download Service. *Google Open Source Blog*. May 20. Retrieved April 18, 2017 from https://opensource.googleblog.com/2013/05/a-change-to-google-code-download-service.html
9. Chris DiBona. 2015. Bidding farewell to Google Code. *Google Open Source Blog.* March 12. Retrieved April 18, 2017 from https://opensource.googleblog.com/2015/03/farewell-to-google-code.html
10. Kasey Uhlenhuth. 2015. We're Moving to Github! *C# Frequently Asked Questions Blog*. January 10. Retrieved April 19, 2017 from https://blogs.msdn.microsoft.com/csharpfaq/2015/01/10/were-moving-to-github/
11. Brian Harry. 2017. Shutting Down CodePlex. *Brian Harry's Blog*. March 31. Retrieved April 18, 2017 from https://blogs.msdn.microsoft.com/bharry/2017/03/31/shutting-down-codeplex/
12. SourceForge Research Data Archive Wiki. All Tables. Retrieved April 18, 2017 from http://srda.cse.nd.edu/mediawiki/index.php/All_tables
13. SourceForge Research Data Archive Wiki. Finding Data, Downloads. Retrieved April 20, 2017 from http://srda.cse.nd.edu/mediawiki/index.php/Finding_data
14. Roberto Gallopini. 2013. Today We Offer DevShare (Beta), A Sustainable Way To Fund Open Source Software. *SourceForge Blog.* July 1. Retrieved April 18, 2017 from https://sourceforge.net/blog/today-we-offer-devshare-beta-a-sustainable-way-to-fund-open-source-software/l
15. Ben Balter. 2015. Open source license usage on Github.com. *Github Blog.* March 9. Retrieved April 20, 2017 from https://github.com/blog/1964-open-source-license-usage-on-github-com