Success of Open Source Projects: Patterns of Downloads and Releases with Time

Ayelet Israeli and Dror G. Feitelson
School of Computer Science and Engineering
The Hebrew University of Jerusalem
91904 Jerusalem, Israel

Abstract

The success of software projects has many different facets: meeting user requirements, being developed within given budget and time constraints, and actually being used. We focus on one of them: the pattern of adoption by users, and its possible relationship with continued development. This is done based on readily available data for open source projects, namely their releases and downloads. Rather than just classifying projects as "successful" or "failure", we identify six distinct patterns of how the download rate changes with time that illuminate different aspects of successful or failed projects.

"You're not thinking fourth dimensionally!"

Dr. Emmett "Doc" Brown,
Back To The Future III, scene 8

1. Introduction

Software engineering deals with methodologies for building successful software systems. Thus a very basic requirement for software engineering research is to be able to identify and define the degree to which different projects are successful. For example, when one wants to argue for agile methods, it is useful to be able to say that a certain fraction of projects developed using such methods were successful, and that this fraction is higher than for competing methodologies.

A relatively straightforward metric for success can be adoption by potential users. In the case of commercial software, this metric translates into market share. For open source software it translates into downloads. However, the total number of downloads by itself does not provide a full picture. A better understanding of the success of an open source software project is obtained by studying the pattern of downloads over time.

By comparing the downloads of multiple projects on SourceForge, we find several recurring patterns that characterize projects that enjoy different levels of adoption by users. For example, one of the patterns is that of projects that enjoy transient success: they are very popular for a certain period of time, but then they are replaced by another competing project.

In some cases the pattern of downloads is related to the patterns of releases of new versions of the software, but in others it is not. For example, two patterns we saw are recurring peaks of intense downloads with each new release, and conversely, continuous activity of downloads that is hardly affected by new releases. We interpret the first as reflecting a saturated user base that anticipate and require new features, and the second as reflecting more sporadic use by casual users, which do not need to follow every new feature.

The next section reviews some previous work on characterizing the success of software projects. Section 3 outlines some methodological aspects of our work, and Section 4 presents the main results: six patterns of downloads that lead to different interpretations of success (or lack thereof). Section 5 concludes the paper. In addition, an appendix provides details about extraction of the data from the SourceForge database.

2. Metrics for Success

Assessing the degree to which a software project is successful is not always easy. Of course, there are those projects that are an unquestionable success, dominating their areas and used by millions, while at the opposite end are those that are canceled after costing millions of dollars, which can safely be classified as failures [6, 1]. But what about projects that are late, over-budget, or simply not used?

In 1992 DeLone and McLean devised a model for Information System success. This model contained six dimensions of success, ranging from system and information quality, through usage and user satisfaction, to impact on

the individual and the organization [4]. Ten years later, they revised the model based on how it was being used in practice [3]. Indeed, many others have used this model as a basis for discussing success of Information Systems, and have attempted to measure its various attributes.

While the model is in general applicable to Enterprise Information Systems (albeit perhaps not all its components are indeed easy to measure), this is not necessarily the case for open source software. For example, most open source projects are not developed within the context of an organization that will use them, so the whole issue of impact (or benefits) on the organization is mute. Therefore, metrics that are based on the team of developers may be more appropriate [8]. Based on such considerations, Crowston et al. have listed multiple alternative metrics, based on the free/open-source development process and on input from developers as to what they consider to be a success [2]. These metrics emphasize the process and artifacts produced, e.g. number of developers, time to close bugs, and the interest expressed by users (e.g. as measured by downloads and pageviews). Importantly, Crowston et al. also point out the importance of tracking metrics over time. However, their data only included five observations with uneven spacing over a period of about 5 years.

Perhaps the simplest metric for success of an open source project is indeed the number of times it has been downloaded. This metric has its problems, of course, as downloads do not necessarily translate into actual use. However, it is easy to measure, and is also used as a basic metric for ranking projects on sites such as SourceForge. This has led to research that attempts to find thresholds based on the distribution of downloads and thus classify projects into different levels of success [7, 5].

The main deficiency of previous approaches based on downloads is that they only consider the total number of downloads for each project. However, the pattern of downloads is also important. For example, a project with a huge potential that garners 1000 downloads upon its first release but quickly fizzles out is probably a failure, whereas a niche project that consistently serves 20 downloads a month for four years, for a total of 960, is probably a success. As we show in Section 4, several such patterns may be identified, and each leads to a different interpretation of the degree of success. Thus our work serves to increase our understanding of how downloads can be used to assess success, and suggests an approach that may be applicable to other metrics as well.

3. Methodology

Our work is based on the database used in the SourceForge open source hosting site, which has been made available for research on open source software development [9].

The two types of data we use in this paper are project releases and downloads. Download data is available at a monthly resolution. This is much better than the developer data from [2] which had 5 data points in 5 years. However, there are some problems with the data. For example, many of the projects (although not all of them) recorded zero or very few downloads in June 2003, and there were also gaps in the data during 2005. Releases are indicated with an exact date, but we just use a binary scale of having one or more release in a given month. Information about the schemas used and the different problems with the database are detailed in the appendix.

At the time of writing, SourceForge hosts more than 145,000 projects. Nearly 67,000 of them have at least one download, implying that more than half have never been downloaded (possibly because they have never issued even a single release). Another problem is that many projects are relatively new, and have not had the chance to attract a following yet. To avoid these problems and the problems with the data in 2005, we focus on projects that existed during the five years from November 1999 to December 2004, and have at least one download.

We avoid using any additional filtering in order to avoid bias in our results. For example, Crowston et al. used only projects with 7 developers and 100 bug reports, in order to study the relationship between success and the dynamics of the developer/contributor community [2]. Not surprisingly, they found that the vast majority of their projects were successful, but noted that this is most probably due to the fact that less successful projects simply didn't make their selection criteria.

Another possible problem with our data is that we may not see all the downloads, if a project has other download channels in addition to SourceForge. Indeed, we have identified at least one case where a project migrated to another site, causing its SourceForge downloads to drop. However, this is probably not very common, and in any case, is not expected to have a big influence on the patterns of downloads discussed here.

4. Patterns of Downloads and Releases

By plotting the number of downloads in successive months over several years, six basic patterns were identified. Note, however, that when looking at a certain project, the graph may actually be a combination of two or three of these patterns.

Growing Download Rate

This category consists of projects which have continuous growth of monthly downloads over time. These projects

have a consistently growing user base, and are therefore identified as successful.

The most obvious example is emule (Fig. 1), which is by far the most downloaded project on SourceForge, with over 300,000,000 downloads (second place is a bit less than half, with 141,000,000 downloads). The vertical lines in the graph indicate the release dates. We can see here that new releases cause an increase of the downloads, which later decreases again, but the overall trend is a growing one.

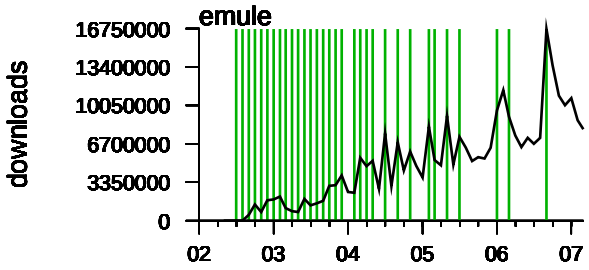

Figure 1. *emule: example of the "growing" category.*

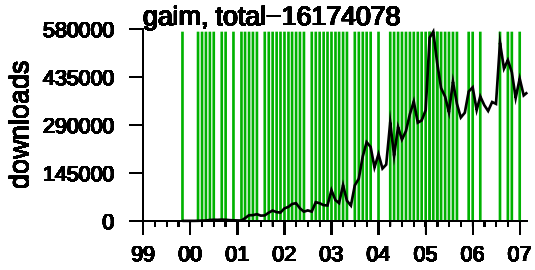

Figure 2. *gaim: growth followed by stability.*

Many projects do not grow over the full multi-year time span we are looking at, but only during part of it. An example is gaim (Fig. 2), which exhibited growing monthly downloads from 2001 through 2004, but seems to have stabilized since then. This might indicate that its user base is becoming saturated.

Downloads Related to Releases

"Release Early, Release Often" is a recurring practice in open source software development [10]. Naturally, each release engenders new interest by users.

As stated, we have added the time dimension to the downloads rate. We have also integrated the version releases into our data. By doing this, it is easy to observe peaks in the download rate which are synchronized with a release. If most of the downloads are in fact associated with releases, we believe that this may reflect a saturated user base, which follows updates and improvements. This indicates that the users actually use the project, since upon a new release, we see growth in the downloads, probably because the users want to be kept updated. Another possible explanation is that with the new releases there is more publicity, and the word-of-mouth of the new release of the application enlarges the user base. For example, this seems to be happening in emule (Fig. 1) for practically all releases since 2004.

The example in Fig. 3 is of a graphical user interface to gdb, the GNU debugger, running under KDE. Here we also see that following every release there is a peak in the downloads rate. The peak might be at the same month of the release, or on the following month, due to the fact the releases are daily and the download data is monthly.

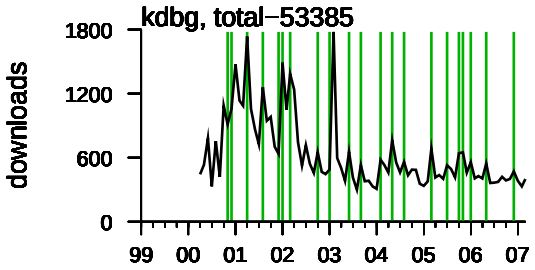

Figure 3. *KDbg: continuous downloads with peaks at release dates.*

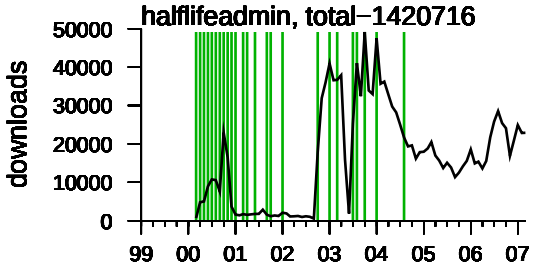

Figure 4. *HLA: large change after the Sept. 2002 release.*

Another pattern is that of a step-release. This is a release that leads to an abrupt and continuous increase in the download rate, to the degree that it can be said that this release "made the system". In the example of Fig. 4 we see that

such an effect occurs after the September 2002 release. A similar effect is also seen in Fig. 6.

It should be noted that not all releases always lead to a peak in the rate of downloads. There are often many releases during the life of the project, indicative of a "living" project, but with no distinct correlation between releases and peaks in the downloads rate. One reason why the download rate may not be directly affected by the release is that the releases are extremely frequent, and most users do not really need such frequent updates. For example, this may be happening in the gaim example (Fig. 2), where growth is continuous rather than being tied with specific releases.

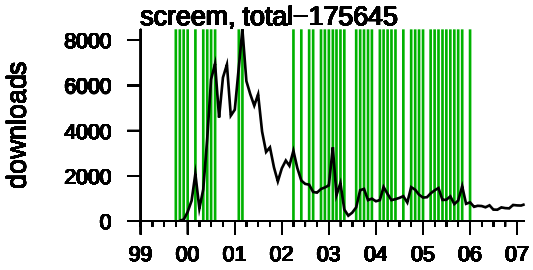

Figure 5. *screem: multiple releases with constant downloads rate.*

Another example is screem (Fig. 5), a project that had a very successful period in 2000–2001, but later the download rate declined. However, during 2002–2005 it exhibited a strong "liveliness" based on frequent releases, coupled with a steady download rate.

The above observations lead to an interesting distinction between "activity" and "success". In principle, a project can be very active, with multiple developers churning out frequent releases, but be completely ignored by users. On the other a project with few releases may have many users that continue to use it. The question is whether in general these two axes are indeed independent, or whether there is some correlation between them.

Constant Download Rate

This category contains projects that are relatively stable over time. There are a few possible explanations for this. One, noted above, is that there might be frequent releases which generate a largely constant curve. Another possibility is that the application is a niche application and its user base is saturated.

One example for a constant download rate is HLA in Fig. 4, where the last release was in 2004 but the download rate has stayed 15,000–25,000 a month ever since. The example in Fig. 6 is TCL (Tool Command Language) which is an interpreted language and interpreter. This language was created in 1988, and this specific project became popular towards the end of 2001. After an initial surge of downloads, it has seen a steady rate of downloads ever since. This example can match either explanation.

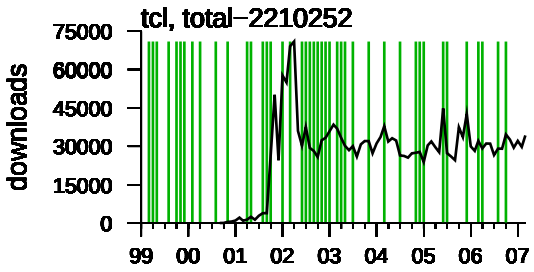

Figure 6. *tcl: example of continuous downloads — high average.*

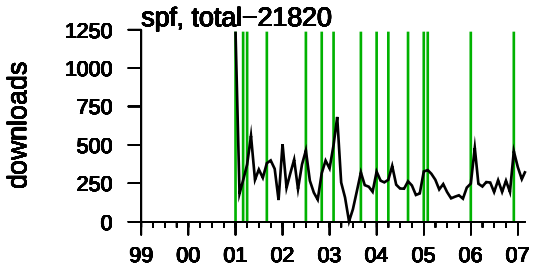

Figure 7. *spf: example of continuous downloads — medium average.*

Two more examples, the spf project (Fig. 7) and the JOODA project (Fig. 8), demonstrate continuous downloads with different levels of averages — while spf has a monthly download average of 250, JOODA has an average of only 25 monthly downloads.

Another explanation for constant downloads might be sporadic use of the application. The bpe project (Fig. 9) is a screen-oriented, curses-based binary editor. The usage pattern can be explained by such sporadic usage.

These examples motivate a possible distinction between different levels of continuous downloads. A project that exhibits a continuous but low level of downloads may be a niche application that only has limited potential users, but is still a success as witnessed by the fact that the potential users continue to use it. To judge whether this is the case it

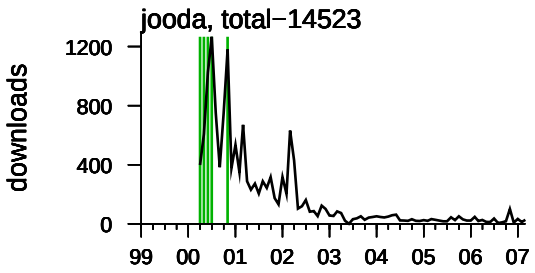

Figure 8. *JOODA: example of continuous downloads — low average.*

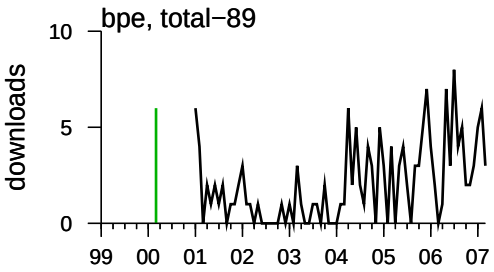

Figure 9. *bpe: example of continuous sporadic use.*

may be useful to compare the constant rate with the initial peak, based on the assumption that the initial peak reflects the *potential* user base. For example, for JOODA (Fig. 8) the constant rate of ∼25 is much lower than the peak of 1250, so this constant rate most probably does not indicate success. But for bpe (Fig. 9), with only one release and no peak, even a much lower rate might indicate modest success.

Limited Time Span

This category deals with projects that were downloaded at a high rate for some time, and then their download rate has then dropped to almost none. Here, four different cases have been identified:

1. Irrelevant Projects: The project was a leader at the time when it was relevant. Nowadays, it is irrelevant. Hence, we see a decline in the downloads rate. The example in Fig. 10 is of Linux PCMCIA Card Services. This application introduces Linux support for PCMCIA and CardBus devices, including kernel services, client drivers, and user-level utilities. The kernel components are deprecated for 2.4 and later kernels. The user-level tools are deprecated for 2.6.13 and later kernels. Since these are deprecated we expect the download rate to decrease.

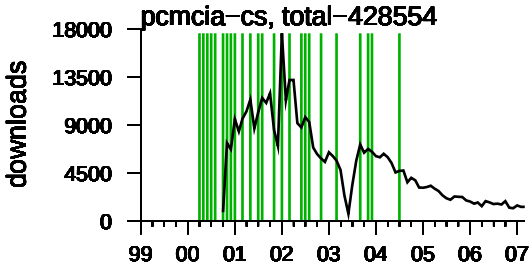

Figure 10. *PCMCIA: a successful project that lost relevance.*

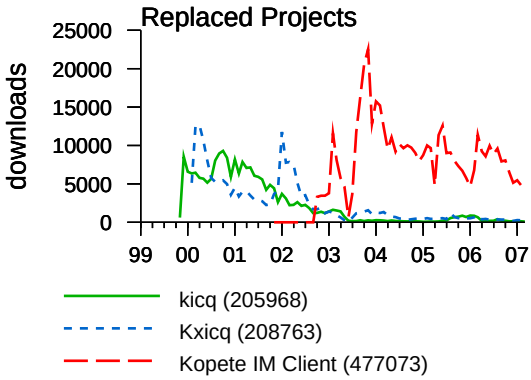

Figure 11. *ICQ clients: successful projects that replace each other.*

2. Replaced Projects: The project was a leader for some time, but was then replaced by another project. The example in Fig. 11 displays this phenomenon for ICQ clients for KDE (K desktop environment for Unix). Initially there were two open source implementation — kicq and Kxicq — which had many downloads (more than 200,000) and competed against each other for about three years. Then, in September 2002 a new application with the same purpose was introduced, and ever since it has become the leader. Currently, it was downloaded over 470,000 times in the range of less than 5 years. The other two projects download rate was reduced to almost 0.

3. Fashion / Periodic hits: An application which was pop-

ular for some time, but is no longer popular. This doesn't have to do with outside effects like a better replacement or environmental changes, but just due to fashion. The example in Fig. 12 is of Tux Racer. This is a simple OpenGL-based racing game featuring Tux, the Linux Penguin. The object of the game is to slide down a snow- and ice-covered mountain as quickly as possible, avoiding the trees and rocks that will slow you down. Naturally, new game's popularity expires over time.

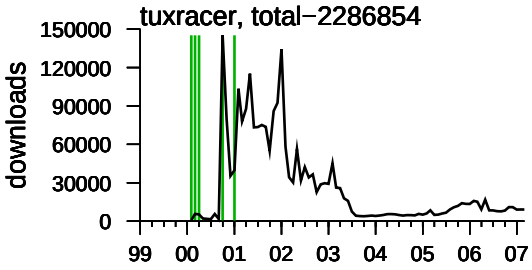

Figure 12. *Tux: a successful project that lost popularity.*

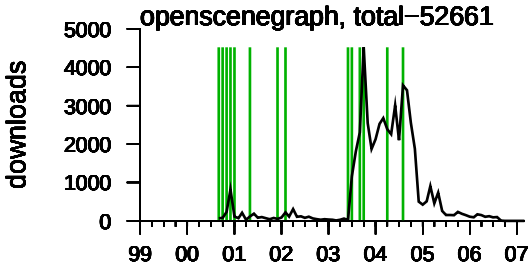

Figure 13. *OSG: a successful project that moved to another site.*

4. Moving Projects: We also found projects that follow the described pattern, and the reason for the sudden drop is moving the project website from SourceForge to a different location. The example (Fig. 13) is of Open Scene Graph application, which is an open source high performance 3D graphics toolkit. In November 2004 a new site was launched, and at about the same time the download rate from SourceForge dropped dramatically. The new site is http://www.openscenegraph.com/.

The Limited Time Span downloading pattern most resembles the classic marketing Product Life Cycle model. This model captures the sales pattern of a product over time. According to this model, the product goes through four sales stages: The first stage is Introduction, in which the sales are pretty low, and are growing slowly. Customers are not yet familiar with the product and have to be either real innovators or to be promoted in order to buy the new product. The second stage is the Growth stage, where sales increase rapidly and significantly, due to the growing public awareness. In this stage competition also starts, after the competitors have identified the potential of the new product. The third step is Maturity, in which sales reach their peak and then start to diminish. The last step is the Decline stage, at this stage the sales either decline or stabilize to a constant level. The idea is that the cycle captures the market potential of the product, and at the end of the cycle, the product has been fully adopted by it's potential customers.

The Limited Time Span pattern of downloads seems to match the changes in sales predicted by the product life cycle. This might indicate that the model is relevant also for open source projects, and helps us to explain some of the phenomena: both irrelevant projects and fashion hits can be fully explained — the projects reached their full market potential, and therefore decline. The replacing project model can also be explained: In the growth stage competition starts to appear, and in this case the competitors became leaders, therefore causing the initial projects to move on to the next stage of the product life cycle, until they fade.

Downwards Trend

This pattern is a gradual decrease to oblivion of the download rate. The projects that are in this category have a decreasing trend over time. This can be part of another pattern, such as the tail end of a limited time span project.

In other cases theses projects were never successful. They started high and ever since have deteriorated. From this step onwards there are two possibilities for these projects — either to die (and reach essentially no downloads), or to remain in a constant (low) level. It is hard to determine why the project was not successful, it can be due to better competition, lack of interest in such an application, or maybe just a badly coded and buggy implementation.

The example in Fig. 14 is of a project with such a downward trend. This one is of a Internet-based communications program. It can be explained that it didn't succeed due to the saturated market of such applications.

Dead Projects

This category includes projects which were never really downloaded very much. We picked projects which have more than 1 download, but there are also many projects that have an average of about 1 download per month. The

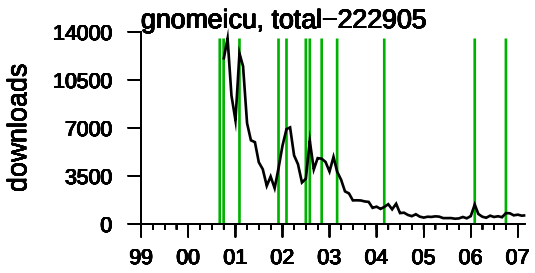

Figure 14. *A project that didn't take off despite several years of work and multiple releases.*

downloads seem as a "noise level", there are a few sporadic downloads over time. In some cases, the projects had a (much) higher download rate in the past. However, in contradistinction from the time-span category, this looks more like users trying it out and giving up, rather than continued success until the project is replaced.

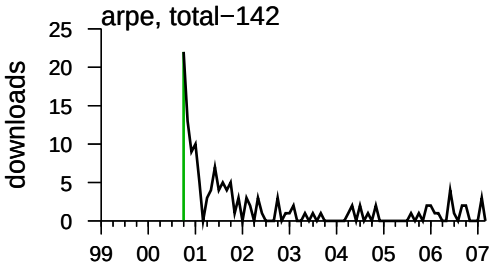

Figure 15. *A project that didn't take off and died after the first release.*

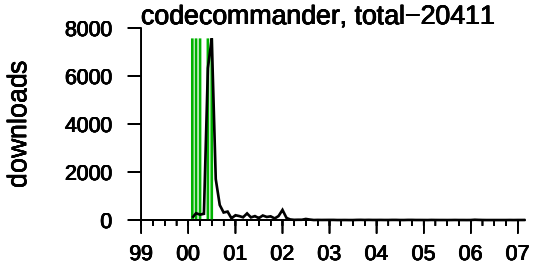

Figure 16. *A project that was successful for a few months but didn't survive.*

The example in Fig. 15 displays such a project. The application is for building and managing a Beowulf-style cluster: remote reboot and power control, and remote and centralized cluster management Web-based GUI. However, there are many such examples in SourceForge, and this is just a random one.

Another possible metric for dead projects is that the activity is observed only in a certain fraction of the months. For example, Fig. 16 displays such a project. The application is a code editor for Linux with many advanced features. The most successful month of the project were June and July 2000. In those two month, there were 21 different releases (out of 40 total releases within five month). But since mid 2000 there have been no more releases, and since mid 2002 most months show no downloads at all.

5. Conclusions

The question of whether an open source software project is successful has many facets, and can be answered in many different ways based on many different metrics. A prominent metric that is often used is downloads. However, the total number of downloads alone does not provide enough information to assess the degree to which an open source project is successful. Rather, we suggest that the time-dependent pattern of the downloads should be studied. We have identified six main patterns that characterize projects that enjoy different levels of success, and provide possible interpretations of why they emerge. The downloads of real projects may be a combination of several such patterns.

In particular, using monthly observations over several years we come up with two novel findings. The first is patterns such as transient success, which were not recognized before. These are projects that enjoy a period of high success, but are then replaced by another project or just fall out of fashion. The second is the ability to compensate, at least to some degree, for the problem of projects that are directed at different communities of users and therefore have widely different potential market sizes. Observing consistent activity for a long time can be taken to indicate success, and even a saturated market, even if the total number of downloads is relatively low.

An immediate direction for future work is to automate the identification of these patterns. This will enable the mapping of projects to patterns to be determined at a large scale. For example, we would then be able to classify all the active projects on SourceForge, and see what fraction are successful in the different meanings of the term.

A more long-term goal is to integrate this work with other success metrics, such as those listed by Crowston et al. [2]. In particular, it seems that many of the other metrics can also be tracked over time, and the way they change

may be more indicative for success than the overall average value of the metric.

Appendix: SourceForge Data Extraction

Due to the large number of schema changes in the SourceForge database between August and September of 2005, and due to some changes between March 2005 and April 2005, the retrieval of the data from the database was not trivial.

The downloads data was extracted in two different ways: data until January 2005 (inclusive) was taken from the schema "stats_project_months" which is available only in the dumps until August 2005. This schema holds historical data of monthly downloads from the registration of the project in SourceForge until January 2005. Between January 2005 and August 2005, the "stats_project_months" table was not updated, moreover, it was dropped in the changes of September 2005. Other tables which seem to hold downloading data (for example "stats_project_all") were not being updated as well, and for the range of January-August still held data updated for January 2005.

Since April 2005, a new table was added to the database: "stats_groupid_alltime_agg". This table holds aggregated data of downloads since inception of each project. From April onwards, it was trivial to calculate the monthly downloads, simply by subtracting the aggregated data of two successive months. However, a gap remained between January and April, and for that time we could only extract the aggregate number of downloads for these three months. Another problem in the database was that the subtraction between September and October of 2005 resulted in a negative number of downloads in 3.2% of the projects.

For these reasons, we decided to concentrate on the range of November 1999 to December 2004. The examples in the paper, however, contain data until March 2007, which was extracted directly from the SourceForge.net site, and not through the database.

Extracting Data about releases was much simpler. The schema "frs_release" contains historical data of all the releases, in the SourceForge database date format. The date values are based upon the time functions on UNIX operating systems that measure the number of seconds from the epoch (midnight GMT, January 1st, 1970). So the data was not only daily, but much more accurate (in some cases there is a record of a few releases in one day). As stated, we gathered the daily data into monthly data by a binary scale of having one or more release in a given month.

References

[1] R. N. Charette, "*Why software fails*". *IEEE Spectrum* **42(9INT)**, pp. 36–43, Sep 2005.

[2] K. Crowston, J. Howison, and H. Annabi, "*Information systems success in free and open source software development: theory and measures*". *Softw. Process Improvement & Pract.* **11(2)**, pp. 123–148, Mar/Apr 2006.

[3] W. H. DeLone and E. R. McLean, "*Information systems success revisited*". In 35th *Hawaii Intl. Conf. System Sciences*, vol. 8, p. 238, Jan 2002.

[4] W. H. DeLone and E. R. McLean, "*Information systems success: the quest for the dependent variable*". *Inf. Syst. Res.* **3(1)**, pp. 60–95, Mar 1992.

[5] D. G. Feitelson, G. Heller, and S. R. Schach, "*An empirically-based criterion for determining the success of an open-source project*". In *Australian Software Engineering Conf.*, pp. 363–368, Apr 2006.

[6] H. Goldstein, "*Who killed the virtual case file?*". *IEEE Spectrum* **42(9INT)**, pp. 18–29, Sep 2005.

[7] F. Hunt and P. Johnson, "*On the Pareto distribution of Sourceforge projects*". In *Open Source Software Development Workshop*, University of Newcastle, Feb 2002.

[8] S. Krishnamurthy, "*Cave or community? an empirical examination of 100 mature open source projects*". *First Monday* **7(6)**, Jun 2002.

[9] G. Madey, "*Sourceforge.net research data archive*". URL http://www.nd.edu/~oss/Data/data.html, 2005.

[10] E. S. Raymond, "*The cathedral and the bazaar*". URL http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar, 2000.