PAID VS. VOLUNTEER WORK IN OPEN SOURCE

Dirk Riehle
Computer Science Department
Friedrich-Alexander University Erlangen-Nürnberg
Martensstr. 3, 91058 Erlangen, Germany
dirk@riehle.org

Philipp Riemer
Computer Science Department
Friedrich-Alexander University Erlangen-Nürnberg
Martensstr. 3, 91058 Erlangen, Germany
contact@philippriemer.de

Carsten Kolassa
Software Engineering
RWTH Aachen University
Ahornstr. 55, 52074 Aachen, Germany
carsten@kolassa.de

Michael Schmidt
Mathematics Department
Friedrich-Alexander University Erlangen-Nürnberg
Martensstr. 3, 91058 Erlangen, Germany
michael.schmidt.nbg@gmail.com

ABSTRACT

Many open source projects have long become commercial. This paper shows just how much of open source software development is paid work and how much has remained volunteer work. Using a conservative approach, we find that about 50% of all open source software development has been paid work for many years now and that many small projects are fully paid for by companies. However, we also find that any non-trivial project balances the amount of paid developer with volunteer work, and we suggest that the ratio of volunteer to paid work can serve as an indicator for the health of open source projects and aid the management of the respective communities.

Index Terms—Open source software, empirical software engineering, volunteer open source, paid open source.

1. INTRODUCTION

Open source software development has long become an important commercial activity. Corbet et al.'s analyses of recent Linux kernel releases show that a large part of this work is being carried out by developers using their companies' email addresses when submitting code [6], implying that they are paid for the work by their employers.

However, while commercial contributions to the Linux kernel have been widely acknowledged, little is known about the overall commercial contribution to open source projects in the form of paid rather than volunteer development work.

In this paper we show that open source has both strong and broad commercial support by companies paying developers to perform open source software development. Also, we suggest that understanding the relationship between paid and volunteer work in open source projects will aid project leaders in steering their community.

This work makes the following contributions:

- It shows empirically that open source has strong commercial support across a broad range of projects,
- it shows the possible range of healthy paid-to-volunteer work ratios to help project steering,
- it presents measurements of
    - how much open source work is being performed during working time,
    - how open source working time work has changed over the years,
    - the percentage of open source programmers that are paid programmers, and,
    - the distribution of volunteer vs. paid work across open source projects,
- using the Linux kernel specifically and a large sample of active open source projects (>5.000 projects).

The paper is organized as follows: In Section 2, we describe our research approach and define key terms. In Section 3 we present the main empirical results. In Section 4 we discuss our findings as well as their limitations. In Section 5 we review related work in and Section 6 we present final conclusions.

2. Research Approach

2.1 Definitions

We use the following definitions:

- An *author* of some piece of code is the creator of the code, i.e., the original developer.
- A *commit* is the process of putting some piece of code into a code repository.
- *Code repository* is used as a synonym for *configuration management system*.
- A *committer* is a software developer who has the necessary rights to commit to a code repository.
- *Maintainer* is a synonym for a committer (as used in Linux kernel development).
- A *patch* is a code contribution submitted to a committer for inclusion in the project.

A code contribution by an author indicates when the code was written and a commit by a committer indicates when the committer integrated the code into the code base. Author and committer are roles. Typically, in a two-step process, an author submits a patch and a committer integrates the patch into the main code base. An author, who is also a committer, can do both of these steps as one. The common case is that an author is not a committer, hence we separate both roles in our analysis of the Linux kernel.

Moreover, we define the following time-related terms using common governmental regulations in Western countries:

- *Working time* is the time from 9am to 5pm local time, Mondays to Fridays.
- *Spare time* is all the time that is not working time.

Consequently, working time and spare time depend on the time zone of the developer.

2.2 Data Sources

We use two data sources: The Linux kernel and the Ohloh projects. Our analysis of the Linux kernel development work is based on its public configuration management data found at Kernel.org [5]. Since 2005, it has been managed using Git, which in contrast to older configuration management systems lets us distinguish the authors of some code, i.e., the original developer, from the committer of the code, who integrated it into the kernel code base. For this work, we downloaded the whole configuration management history from 2005 to 2011.

Our analysis of open source projects is based on a 2008 snapshot of the Ohloh open source project database [20]. Using Daffara's definition of "active projects" [7], we find that our database snapshot contains 5,117 active open source projects. Daffara estimates that there were about 18,000 active open source projects in the world by August 2007, so our sample represents about 30% of the total active project population at that time. While not wholly representative for open source at its time, it is close nevertheless.

2.3 Data Quality

Since 2005, the Linux kernel configuration management data (using Git) has been providing more precise information than traditional systems (CVS, svn). We can distinguish between authors and committers, and we can assess the exact time of a commit, whether a code contribution or code integration.

The 2008 Ohloh database snapshot is not quite as detailed. Collecting 8,705,118 commits from more than 9,192 projects, it does not directly provide all relevant data. Due to the diversity of configuration management systems used in open source, Ohloh cannot distinguish between an author and a committer; thus, we only have committer data at hand.

Another consequence of the variety of configuration management systems is that Ohloh stores all commit timestamps using UTC, ignoring the original time zone of a developer. However, for this work, we need the local time of a commit and hence the time zone.

We address this problem by using location data that Ohloh provides to determine the timezone of individual developers. By hand, we identified 580 committers (out of 45,870 distinct committer ids), constituting 1.3% of the committer population. Those identified committers performed 646,705 of the 8 million commits, totaling about 8% of the work being performed.

We call the set of identified committers the *known committer set* or the *known committers*, in short.

While we can argue that the original Ohloh data set is close to being representative of open source, the reduced number of known committers may not be. For one, we identified mostly committers of above average activity (1.3% of the population performing 8% of the work), so there is some bias. Thus, we need to understand whether this bias is relevant for the analysis presented in this paper.

For this, we ranked committers by number of commits and then binned the resulting committer sequence into 26 different bins. The 26 bins of known committers all have close-to-equal total numbers of commits and were suggested by R, the statistical analysis tool and environment we are using. Thus, the amount of work in each bin is about the same, but was performed by very different numbers of people.

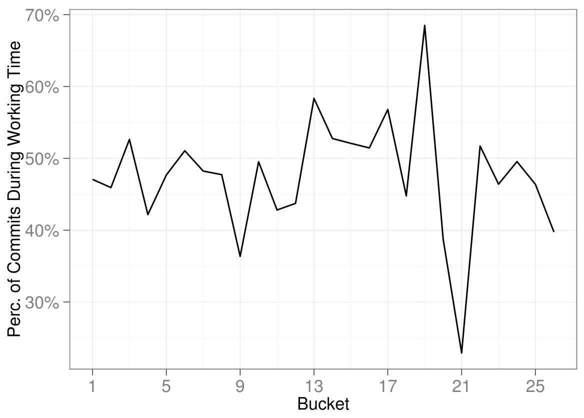

Figure 1. Paid-work-percentage of total work for 26 equal-commit-numbers committer bins (higher bin number means more committers in bin)

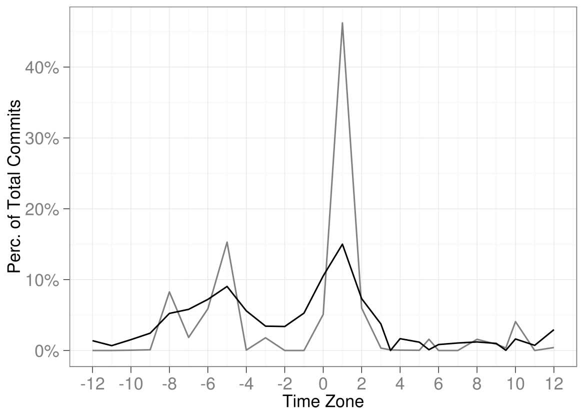

Figure 2. Distribution of percentage of total commits over time zones (light gray = known committers, dark gray = extended committers)

Assuming paid work to be work performed Mon-Fri from 9am-5pm (see below for definition and discussion), we can calculate the percentage of total work performed that is paid work. Figure 1 shows this paid-work-percentage (of total work) by committer bin.

With a null hypothesis that no trend (bias) is apparent (and alternative hypotheses that there is a bias), using a t-test and assuming a normal distribution, we cannot reject the null hypothesis at the 95% confidence level. We therefore have no reason to assume that reducing the overall committer set to the known committer set introduces a bias that impacts our analysis (but can also not exclude it).

In addition to the known committers set, we define an *extended committers set* or *extended committers*, in short. The extended committers set comprises all committers in the original Ohloh database, where the committer timezone is either known or assumed. The assumed timezone is determined using the following heuristic: We first condense all commits of a committer in the database into a single week. For all committers where we do not know the time zone, we match their week on an hourly basis with the weeks of the known committers set. Using a least-squares approach, we identify the time zone that has a minimum difference to the established data. This provides us with the most probable time zone for the not-known committers so that we can determine the local time for each commit (ignoring work while traveling).

The known committers set provides a sharp picture of time-zone-based weekly work activities, including paid and volunteer work, while the extended committers set provides a richer (more data), but more fuzzy picture of the weekly work activities of committers.

In the following, we present both the known and extended committer data side-by-side.

2.4 Data Interpretation

We would like to understand how much work in open source is paid work and how it is distributed across a wide range of open source projects.

As introduced above, we assume that work performed during regular working hours (weekdays, 9am-5pm) is work paid for by companies or paid for through self-sponsorship of the developer.

One possible objection to this is that not all cultures have working weeks of Mon-Fri, 9am-5pm. For example, some cultures work on Saturdays.

Figure 2 shows the distribution of commits over the different timezones of this planet.

All countries, mostly Islamic countries, that work on Saturdays, show very little open source activity (an interesting fact in itself). Thus, we feel safe to proceed with our definition of weekend and weekdays.

Also, one may argue that many people have working hours outside of 9am-5pm and that we are too conservative in our estimate then. For one, we'd rather estimate paid work conservatively, but we also shifted working hour definitions around, to 8am-4pm, 10am-6pm, 8am-8pm, etc. with no significant change in the results. Thus, we decided to stick to the most common workday definition.

3. Paid Work in Open Source

3.1 Total Work during Working Time

First, we investigate how much time is being spent on open source during regular working hours.

Figures 3-6 show the workweek on an hourly base for the years 2005-2011 for the Linux kernel and for the years 2000-2007 for the Ohloh data.

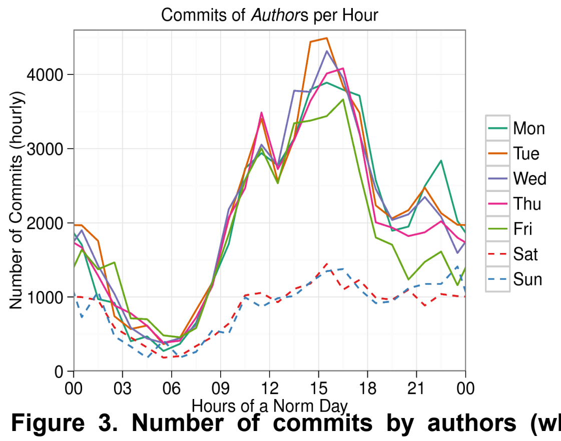

Figure 3. Number of commits by authors (when code is developed) per hour counted over all weeks 2005-2011 for the Linux Kernel

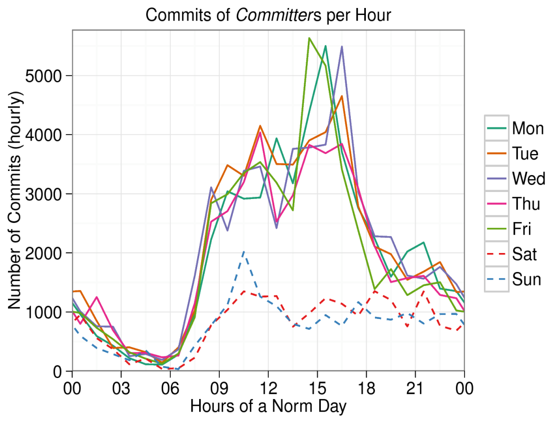

Figure 4. Number of commits by committers (when code is integrated) per hour counted over all weeks 2005-2011 for the Linux Kernel

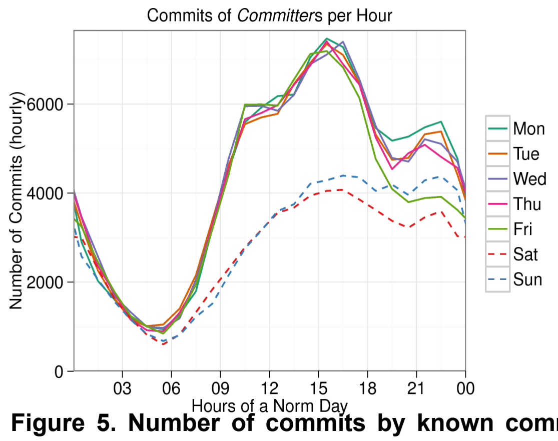

Figure 5. Number of commits by known committers per hour counted over all weeks 2000-2007 for the Ohloh projects

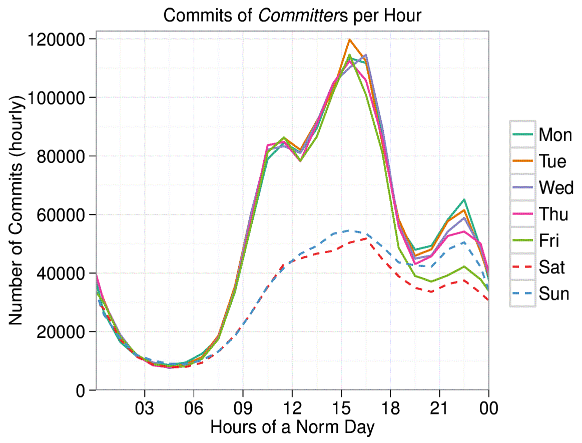

Figure 6. Number of commits by extended committers per hour counted over all weeks 2000-2007 for the Ohloh projects

of insights already. The most obvious insights are that

- there is a clear difference between work days and weekend days: about twice as much work is being done on a work day as is on a weekend day; and that
- most work is being performed during regular waking and working hours, i.e., from 9am to 5pm, even on the weekends and that
- developers take lunch and dinner breaks with work picking up again for a few hours after dinner before quieting down for the night.

With a working time definition of Mon-Fri, 9am-5pm, Table 1 shows the percentages of all code contributions respectively all commits made during working time for the Linux kernel and the Ohloh projects:

About 50% of all work contributed to open source software projects has been provided Monday to Friday, between 9am and 5pm.

3.2 Trends in Work during Working Time

Next, we look at how the working time work spent on open source has changed over the years.

Figures 7-10 show how the percentage of commits made during working time changed over the years. In Figures 7-10, each data point is the percentage of working time work for the given week. The moving average is a LOESS curve, and the grayed-out space around it indicates the 95% confidence interval for a data point. The widening of that space at the boundaries of the graph is an artifact of not using additional data beyond those boundaries.

Table 1. Percentage of work performed during working time (9am-5pm, Mon-Fri) for Linux (2005-2011) and the Ohloh projects (2000-2007)

		Percentage of total commits made during working time
Linux Kernel	author	45.00%
	committer	51.36%
Ohloh Projects	known committer	47.3% (min. 28,2%, max. 58,8%)
	extended committers	55.4% (min. 36.5%, max. 59.5%)

The Linux kernel data in Figures 7-8 shows significant fluctuations, which reflect the rapid release cycle of the project. A release is performed about every 80 days [5]. The process is highly regulated with defined time periods of increasing or decreasing activity. The activity is highest during the two week merge window, after which stabilization kicks in and activity decreases rapidly. Thus, there is no apparent annual schedule.

The Ohloh data, a more diverse data set, also shows some fluctuations, but much less so. The main annual dips from 2002 on onwards occur during Christmas week, where it seems naturally to have a drop in working time work relative to spare time work (cf. Section 2 for the dominance in open source activity by Western cultures).

Looking at the Linux kernel data in Fig. 7-8 again, we can see a clear upward trend for both authors and committers. Thus, from around 2007 through to 2010 increasing amounts of working time work in relation to spare time work was being spent on the Linux kernel. Starting 2010, this growth largely plateaued.

In contrast to the Linux kernel data, the Ohloh data set shows a straight line. Using a likelihood ratio test (F-Test), with a straight line as the null hypothesis, we have to reject any other hypothesis (at a confidence level of 98%), and conclude that no growth occurred. The percentage of working time work spent on the Ohloh projects has remained flat.

However, during these time frames, the total underlying data sets have grown substantially. The Linux kernel is growing at a polynomial rate [13] [23] while the combined Ohloh project data, and presumably all of open source, is growing at a near-exponential rate [8]. Thus, for the Ohloh project data, the year 2000 data is much more sparse than the year 2007 data. Still, as we just showed, no working time growth occurred in our open source project data, and the working time percentage of total work performed stayed flat.

With growing market share [25] the economic significance of the Linux kernel has only been increasing, so it is not surprising to see growth in working time work being spent on it. What is surprising is that open source in total (using the Ohloh data as a proxy), which has been growing near-exponentially, has maintained a constant ratio of working time to spare time work. Thus, for every project with increasing economic significance that received more paid development work, new projects have been started with less working time engagement, but possibly growing into it. It is too early to speculate about a stable state of open source in terms of a stable ratio of paid working time work to volunteer spare time work contributions, but open source appears to have reached at least an intermediate stable state. Thus, even with underlying near-exponential growth, we expect this ratio to remain stable for now.

3.3 Developer Classification

While overall weekly working times and working time trends are interesting, we also would like to know how many developers are earning their living by performing open source software development. Thus, we now look at individual developers and how much of their code is written during working time, i.e., to what extent they are being paid for their work.

Fig. 11-12 show the distribution of developers over the percentage of work that is paid work (working time work) for the Linux kernel and Ohloh projects, respectively. It has been counted over all years. Please note that the y-axis is log-scale and that we are talking about contributors now, not just contributions. Only the extended committer data is shown, because the known set was too small to provide meaningful data for this particular discussion.

Both the Linux kernel and Ohloh projects are dominated by the extremes: Developers doing all their work during spare time and developers doing all their work during working time. Table 2 shows the dominance of these extremes. Here, we define paid developers to be those who performed 95% or more of their commits during working time, and volunteer developers to be those who performed 95% and more of their commits during spare time, outside the weekdays 9am-5pm time frame.

Thus, at least 23.15% of all authors working on the Linux kernel, totaling 1,807 developers, are paid for their work. 11.28% of all committers working on the Linux kernel, totaling 37 developers, are paid for their work as well. Given the economic significance of the Linux kernel, one would expect more committers (maintainers) to be paid for their work than authors. A possible explanation for not confirming this assumption is the long-term engagement of committers that

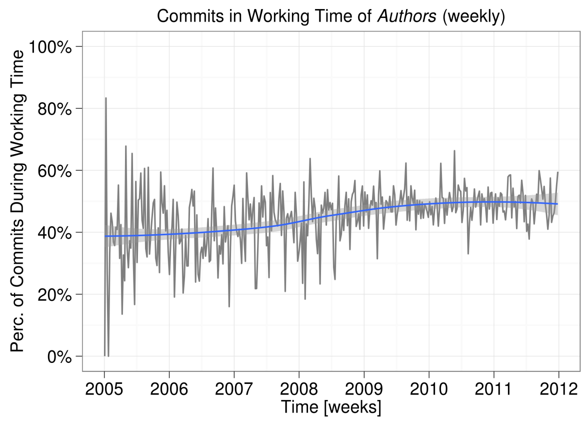

Figure 7. Data and trend line for percentage of commits made by authors to the Linux Kernel during working time for a given week

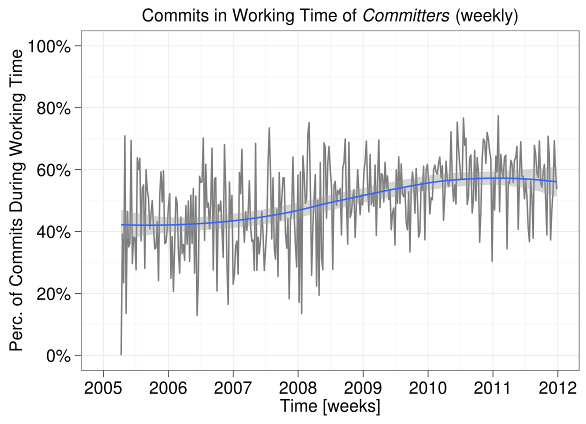

Figure 8. Data and trend line for percentage of commits made by committers to the Linux Kernel during working time for a given week

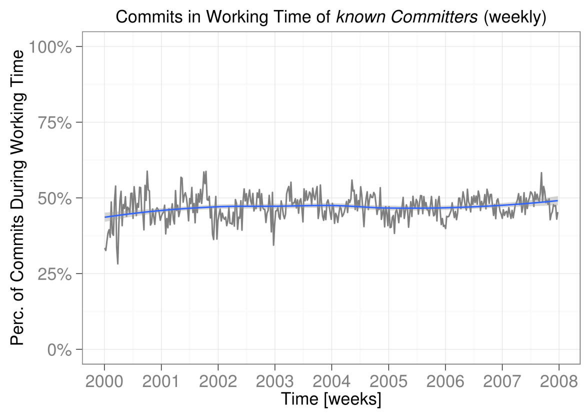

Figure 9. Data and trend line for percentage of commits made to the Ohloh projects during working time for a given week, known committers

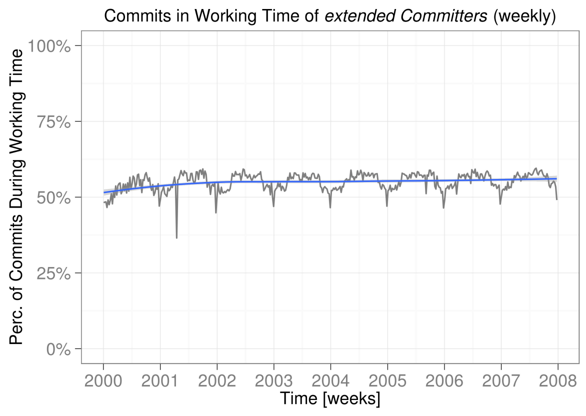

Figure 10. Data and trend line for percentage of commits made to the Ohloh projects during working time for a given week, extended committers

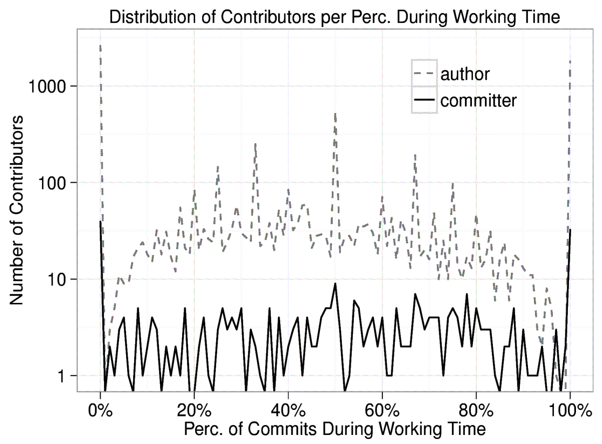

Figure 11. Number of authors and committers with a given average percentage of paid work for the years 2005-2011 for the Linux Kernel

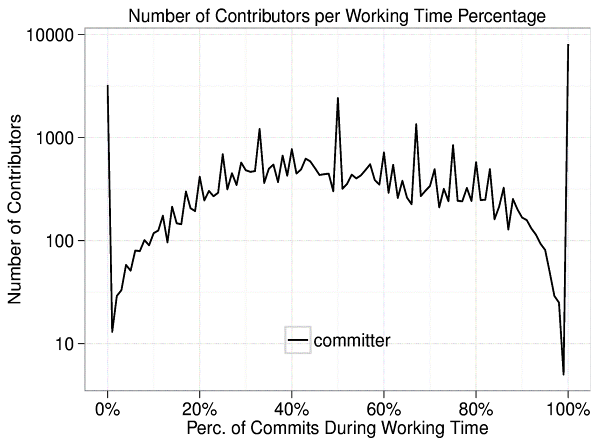

Figure 12. Number of committers with a given average percentage of paid work for the years 2000-2007 for the Ohloh projects, extended committers

Table 2. Distribution of volunteer (spare time) to paid (working time) developers, binned, over all years

		Volunteer (Spare Time) Work		Mixed	Paid (Working Time) Work	
Working Time Work %		***0%***	***0.01%-5%***	***5.01%-94.99%***	***95%-99.99%***	***100%***
Linux Kernel	author	33.06%	0.35%	43.45%	0.17%	22.98%
	committer	11.59%	3.05%	74.09%	1.52%	9.76%
Ohloh Projects	known committers	2.41%	1.21%	95.69%	0.00%	0.69%
	extended committers	7.04%	0.4%	74.58%	0.41%	17.56%

motivates many to keep working outside traditional working time boundaries, which makes them fall outside our conservative definition of paid work.

As to the Ohloh projects, 17.97% of all extended committers, totaling 8,244 developers, are being paid for their work.

Common to these paid developers is that they do not work on open source projects in their spare time, i.e., fall outside the boundaries of the traditional open source enthusiast and volunteer categories.

3.4 Project Classification

Finally, not only are we interested in what percentage of developers are being paid to work on open source, we also would like to know how they are allocated to projects. It is fair to assume that some projects get more commercial attention than others. Thus, we investigate which projects receive this attention. The Linux kernel project is a single (albeit large) project, so in this Section we are looking at the Ohloh projects only.

We find that there is a large number of small (1-2 developers) projects with a long-tail distribution of size that are fully paid for by companies: All developers, frequently just one, are making their contributions only during working time. The top 5 smallest projects in our sample, fully paid for, are called subtle, WebPA, ShARPE, gst-openmax, and phpESP.

These smallest projects have low commit numbers (in the 100's only, sometimes less). Inspection by hand shows that many times, code is being committed in large chunks. This is uncommon in traditional open source software development, where the most frequent commit size is one line of code [16]. Thus, it is safe to assume that these small but still active projects are being developed in-house and are being provided in a snapshot-style to the public at appropriate times.

The largest projects in our sample maintain a paid-for developer percentage in the 10-20% range. Five example projects of this size are GNOME, Netbeans IDE, Eclipse Platform, KDE, and KVM. These are well-known open source projects that are being developed in an open collaborative style, and the paid developer population of these projects can serve as an indicator of healthy public open source projects.

4. Discussion of Findings

In this work, we are making the assumption that work performed during working time hours (Mon-Fri, 9am-5pm, in the resp. local time zone) is paid work. This time frame has been defined as working time by most Western countries and thus we feel justified in considering it paid work. Even students typically have to go to class during that time and spending it on open source development implies economic self-sponsorship as it delays graduation and hence borrows against future income.

The time of a commit is not the actual time the work was performed; it is the point of time when it is committed (made public). Thus, the actual work is performed right up to that point in time. In other work we show that the median time between two commits of the same open source developer is about 100min [15]. When we ran the analyses with shifted working time frames, we found little difference to the numbers from the 9am-5pm time frame and decided to ignore this imprecision. We believe it has no effect on the results.

There is a cultural bias implied by these working hours, as some countries work on other days than Monday to Friday. Our analysis of contributions by time zone (Section 2) demonstrates that open source software development is strongly dominated by Western societies, as witnessed by a sharp drop in activities around Christian holidays like Easter or Christmas.

One might argue that the Ohloh data is getting old. One advantage of the Ohloh data is that it draws broadly on the total population of available open source projects. It was seeded by the original providers of the Ohloh service with the most popular open source projects (by Yahoo search engine ranking) and has since been maintained by hand by the respective providers of open source projects. Unlike other data sources, the Ohloh data it is much less biased to any particular subgroup of open source projects. If there is a bias, it is a bias towards active well-working projects, which happen to be those we are interested in.

Still, it would be desirable to have new data. Unfortunately, there is no alternative at present: No public access to the new Ohloh data is available on the level of detail required, other newer data sources are substantially more biased towards particular subgroups of open source, and it is prohibitively expensive for a research group to build a comprehensive and representative data set for all of open source (which is why nobody has done it yet). Consequently, this is the best research we can do for now.

Our definition of "paid developer" is highly conservative. It is a person who does 95% or more of their work during working time hours only. It represents a regular developer with a regular life-style and presumably no interest in open source software development beyond their work. There are important and common exceptions to this type of person:

- Many paid developers are open source enthusiasts and keep working outside regular working hours.
- The software industry is by and large not unionized and tends to ignore regular working hours.

Consequently, our estimate presents a lower boundary for the number of paid developers.

5. Related Work

Since 2008, Corbet et al. have been providing statistics about the Linux kernel development annually [6]. They investigate topics like evolution of the release frequency as well as number of changes introduced per release. In addition, they provide a list of the most-active companies supporting the development of the Linux kernel and list the percentage of commits performed by each of them. Similar to the work presented in this paper, the reports distinguish between authors and committers. Corbet et al. consider a contribution commercial, if it is made using a company's email address to identify the contributor. They also maintain a separate mapping list for regular contributors that allows tracking a person even if he or she changes the employer. They find that at least 75% of all contributions since 2005 can be assigned to company employees.

A "Report on the International Status of Open Source Software 2010" finds that the U.S.A., Australia, and the West European countries lead the development and adoption of open source software [19]. This is in line with our observation that weekly work as well as holiday drops line up well with Western cultural work patterns.

Godfrey and Tu studied the Linux kernel growth in 2000 [13], and Robles et al., following up on Godrey and Tu, studied the Linux kernel growth in 2005 [23]. Robles et al. provide a good summary about what analyses were made in the area of evolution research of open source software projects. They study 123 stable and 457 development releases up to April 2005 (the point in time where the data for our analysis starts) and, by also counting the number of uncommented lines of code, confirm a super-linear growth rate, that is even more significant than already shown in the preceding paper. At the same time, the authors point out that not all work in a project is programming, but that also many tasks, such as testing, are done outside of the code repository and thus are hard to measure. Independently of that paper, a study by Succi et al. about the growth in "libre" (open source) software systems, confirms this super-linearity for the Linux kernel [24].

Both the proceedings of ICSE (the international conference on software engineering) and MSR (a conference on mining software repositories) as well as other conferences and journals by now provide extensive literature on empirical analyses of open source and closed source projects. An example classic open source studies is [17] by Mockus et al. Topics of interest range from bug prediction [2] [4] [18] [26] through engineering practices [1] [21] [22], social structures and community management [3] [14], software evolution [11] [12], all the way to issues of global collaboration and distributed development [2]. Research methods itself, mostly on data quality issues, are also analyzed [9] [27]. A few papers compare open source with commercial software development [1]. However, to the best of our knowledge none of this work addresses the issue of paid vs. volunteer work as discussed in this paper.

We did not find any research that analyzes the commercialization of open source software projects by investigating when what work was done. A reason might be that modern version control systems, such as Git and Mercurial, have allowed us to access commit history data in detail, including time zone information, only recently. Older systems, such as CVS or svn only store a single UTC time stamp per commit.

6. Conclusions

This paper analyzes to what extent open source software development has become commercial paid-for software development. A paid contribution is defined as having been contributed during regular (Western) working hours, Mon-Fri, 9am-5pm. By studying the Linux kernel from 2005 to 2011 and the Ohloh projects, a large set of more than 5,000 active open source projects, from 2000 to 2007, we find that about 50% of all contributions to projects in our sample population have been paid work. Moreover, no change in this percentage has occurred for the Ohloh projects, suggesting that the ratio of paid-to-volunteer work is stable in open source for now.

Going one step further, we find that 10-20% of the developers engaged in our sample projects perform development work only during working hours, suggesting that they are fully paid for their work. Unlike traditional volunteers, they perform no work on our sample projects outside this time-frame, making our estimate a conservative one. We also find that many small projects are fully paid for, and that larger projects have a healthy mixture of paid and volunteer work in the 10-20% range as well. In future work, we intend to analyze the relationship between these categories of developers, company engagement, and project success.

References

[1] C. Bird, A. Gourley, P. Devanbu "Detecting Patch Submission and Acceptance in OSS Projects," in Proceedings of the Fourth International Workshop on Mining Software Repositories (MSR '07), pp. 26.

[2] C. Bird, N. Nagappan, P. Devanbu, H. Gall, B. Murphy, "Does distributed development affect software quality?: An empirical case study of Windows Vista," in Communications of the ACM, vol. 52, no. 8, pp. 85-93.

[3] C. Bird, D. Pattison, R. D'Souza, V. Filkov, P. Devanbu, "Latent social structure in open source projects," in SIGSOFT '08/FSE-16 Proceedings, 2008, pp. 24-35.

[4] E. Capra, "An Empirical Study on the Relationship Between Software Design Quality, Development Effort and Governance in Open Source Projects," in IEEE Transactions on Software Engineering, vol. 34, no. 6 (2008), pp. 765-782.

[5] J. Corbet, "How to participate in the Linux community," 2008, at http://www.linuxfoundation.org/content/how-participate-linux-community.

[6] J. Corbet, G. Kroah-Hartman, and A. McPherson, "Linux Kernel Development – How fast it is going, who is doing it, what they are doing, and who is sponsoring it?", 2012, from http://go.linuxfoundation.org/who-writes-linux-2012.

[7] C. Daffara, "Estimating the number of active and stable FLOSS projects", 2007, from http://robertogaloppini.net/2007/08/23/estimating-the-number-of-active-and-stable-floss-projects (Archived at http://www.webcitation.org/69t8UM0lX).

[8] A. Deshpande and D. Riehle, "The total growth of open source," in Proceedings of the fourth Conference on Open Source Systems (OSS 2008), Springer Verlag, 2008, pp.197–209.

[9] M. Fischer, M. Pinzger, H. Gall, "Populating a release history database from version control and bug tracking systems," in Proceedings of the International Conference on Software Maintenance (ICSM 2003), pp. 23-32.

[10] FLOSSmole. Collaborative collection and analysis of free/libre/open source project data, 2012, from http://flossmole.org/ (Archived at http://www.webcitation.org/69t9UhDSX).

[11] B. Fluri, M. Wursch, H. C. Gall, "Do Code and Comments Co-Evolve? On the Relation between Source Code and Comment Changes," in Proceedings of the 14th Working Conference on Reverse Engineering (WCRE 2007), pp. 70-79.

[12] H. C. Gall, M. Lanza, "Software evolution: analysis and visualization," in Proceedings of the 28th International Conference on Software Engineering (ICSE 2006), pp. 1055-1056.

[13] M. W. Godfrey and Q. Tu, "Evolution in open source software: a case study," in Proceedings of the International Conference on Software Maintenance (ICSM), 2000, pp.131–142.

[14] V. K. Gurbani, A. Garvert, J. D. Herbsleb, "A case study of open source tools and practices in a commercial setting," in Proceedings of the Fifth Workshop on Open Source Software Engineering, pp. 1-6.

[15] C. Kolassa, D. Riehle, M.A. Salim. "The empirical commit frequency distribution of open source projects." In Proceedings of the 2013 International Symposium on Open Collaboration (WikiSym + OpenSym 2013), ACM, 2013, paper C4.

[16] C. Kolassa, D. Riehle, M.A. Salim. "A Model of the Commit Size Distribution of Open Source." In Proceedings of the 39th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2013), LNCS 7741. Springer Verlag, 2013, pp52-66.

[17] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "A case study of open source software development: The Apache server," in ICSE 2000 Proceedings, 2000, pp. 263–272.

[18] N. Nagappan, T. Ball, "Using Software Dependencies and Churn Metrics to Predict Field Failures: An Empirical Case Study," in First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), pp. 364-373.

[19] National Open Source Software Observatory, "Report on the International Status of Open Source Software," 2010 (Archived at http://www.webcitation.org/69t93TcPq).

[20] Ohloh, the open source network, 2012, online at http://www.ohloh.net/ (Archived at http://www.webcitation.org/69t9byLCw).

[21] J. W. Paulson, G. Succi, A. Eberlein, "An empirical study of open-source and closed-source software products," in Transactions on Software Engineering, vol. 30, no. 4 (April 2004), pp. 246-256.

[22] P. C. Rigby, D. M. German, M.-A. Storey, "Open source software peer review practices: a case study of the apache server," in Proceedings of the 30th

International Conference on Software Engineering (ICSE 2008), IEEE, pp. 541-550.

[23] G. Robles, J. J. Amor, J. M. Gonzalez-Barahona, and I. Herraiz, "Evolution and growth in Large libre software projects," in Proceedings of the eigth international workshop on Principles of Software Evolution, 2005, pp. 165–174.

[24] G. Succi, J. Paulson, and A. Eberlein, "Preliminary results from an empirical study on the growth of open source and commercial software products," in EDSER-3 Workshop, co-located with ICSE, 2001.

[25] S. J. Vaughan-Nichols, Linux servers keep growing, Windows and Unix keep shrinking. ZDnet. (Archived at http://www.webcitation.org/69wwiYWT9)

[26] T. Zimmermann, N. Nagappan, "Predicting defects using network analysis on dependency graphs," in Proceedings of the 30th International Conference on Software Engineering (ICSE 2008), pp. 531-540.

[27] T. Zimmermann, P. Weißgerber, A. Zeller, "Mining version histories to guide software changes," in Proceedings of the 26th International Conference on Software Engineering (ICSE 2004), pp. 563-572.