Programming Language Trends in Open Source Development: An Evaluation Using Data from All Production Phase SourceForge Projects

Daniel P. Delorey
Brigham Young University
TMCB 2230
Provo, UT 84602
pierce@cs.byu.edu

Charles D. Knutson
Brigham Young University
TMCB 2214
Provo, UT 84602
knutson@cs.byu.edu

C. Giraud-Carrier
Brigham Young University
TMCB 3326
Provo, UT 84602
cgc@cs.byu.edu

ABSTRACT

In this work, we analyze data collected from the CVS repositories of 9,997 Open Source projects hosted on SourceForge in an effort to understand trends in programming language usage in the Open Source community between 2000 and 2005. The trends we consider include: 1) the relative popularity of the ten most popular programming languages over time, 2) the use of multiple programming languages by individual programmers and by individual projects, and 3) the programming languages most often used in combination.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*Performance Measures, Process Metrics, Product Metrics*

Keywords

Software Engineering, Metrics, Data Mining, Software Repositories, Programming Language Popularity

1. INTRODUCTION

Few subjects are debated with as much passion among software developers as the relative merits of programming languages. Conventional wisdom is that programmers tend to rely on one or two languages almost exclusively and are often averse to learning new languages until a clear advantage is recognized.

An aversion to switching programming languages is understandable in light of the purpose of a programming language. Programming languages are tools that allow programmers to translate their thoughts into commands a computer can execute. The noted linguist Edward Sapir long ago concluded that, "Language and our thought grooves are inextricably interrelated, are, in a sense, one and the same." [7] It is reasonable to expect that this conclusion holds true not only for natural language like those studied by Sapir, but for programming languages as well.

A tendency to hesitate in adopting new programming languages, however, may stifle creativity as the limited features of an out-dated language prevent developers from conceiving of novel solutions. To quote a well-worn cliche, "When all you have is a hammer, everything looks like a nail." Nielson [5] has demonstrated that developers hesitate to use multiparadigm features even after migrating to a new language opting instead to program as if they were using their old language.

In this paper, we use data gathered from the CVS repositories of Open Source projects hosted on SourceForge to evaluate trends in programming language usage among Open Source developers. Specifically, we investigate: 1) changes in the distributions of projects, authors, files, and lines of code across the ten most popular programming languages used in the SourceForge community between 2000 and 2005, 2) changes in the number of programming languages used in individual projects and by individual developers over the same time period, and 3) the most commonly used combinations of programming languages used in individual projects and by individual developers.

2. RELATED WORK

While anecdotal evidence and opinions about the relative popularity of particular programming languages abound both on the Internet and in print, much of it is published in the form of blog postings and trade magazine editorials. Even those authors whose work is published in scholarly venues tend to write as proponents of their favorite programming language or paradigm. Goth [3], for example, extols the virtues of Java and defends its position as the current most widely used programming language in commercial development. Paulson [6], on the other hand, emphasizes the gains in market share that are being made by dynamic programming languages.

Many authors, both academic and otherwise, who have written about programming language popularity, including the two mentioned above, cite the TIOBE Programming Community Index [8] to support their position. The TIOBE Software Company compiles this list of popular programming languages based mainly on search engine keyword frequency calculated from Google, MSN, and Yahoo. For a more detailed description of their methodology, we direct the reader to the TIOBE website [8]. For our purposes, the salient bits of information about the TIOBE index are that it is an indirect, proprietary metric and the data set used to calculate the metric is only available willing to pay the $1500 fee.

In contrast to the TIOBE Index, the methods we use to calculate programming language popularity are direct measurements extracted from the version control repositories of Open Source projects hosted on SourceForge and our data set is publicly available without a fee.

3. DESCRIPTION OF OUR DATA

Our data were gathered from the SourceForge Research Archive (SFRA) [4] and the CVS repositories Open Source projects hosted on SourceForge. We used `cvs2mysql` and SFRA$^+$ to collect the data. `cvs2mysql` gathers data from CVS repositories and writes them to SQL scripts for import into a MySQL 5.0 database. The data collected by `cvs2mysql` are the name of the file, the location of the file in the repository, the type and state of the file, as well as the author, date, number of lines added and removed, and the author's message for each revision to the file. SFRA$^+$ provides an improved interface to the SFRA which allows the user to not only more efficiently query the SFRA database but to export result sets to SQL scripts that can be imported into a MySQL 5.0 database. By combining the `cvs2mysql` data and the SFRA$^+$ data into a single database we are able to perform more comprehensive analyses. These tools and the methods we used to collect the data are described further in [2].

In all we used data from 9,997 projects and 23,838 authors comprising records of more that 7.5 million individual files and more than 25 million distinct changes to those files in this analysis. In particular, we used: 1) the author, date, and lines added fields of the `cvs_revision` table produced by `cvs2mysql`, 2) file type data calculated from the file extensions of the files in the `cvs_file` table produced by `cvs2mysql`, 3) the project creation date field of the `groups` table from the SFRA, and 4) the user creation date field of the `users` table from the SFRA.

4. LANGUAGE POPULARITY

We have previously used our data set to investigate the effects of programming languages on annual programmer lines-of-code productivity. [1] As part of that research we determined the ten most popular programming languages in the seven year history of SourceForge by ranking the languages by number of project, the number of authors, the number of files, the number of revisions, and the number of lines of code and then taking the average ranking as shown in Table 1.

While the ordering shown in Table 1 gives the all-time popularity rankings for the programming languages listed, it is reasonable to question whether the rankings have remained constant as older programming languages have fallen out of use and newer programming languages have been developed. Figures 1, 2, 3, and 4 show the changes in the distributions of projects, authors, files, and lines of code across the ten programming languages in Table 1 between 2000 and 2005. The values on the y-axis of each graph range from 0% to 100% and the height of each language's vein in the graph above a specific year tick on the x-axis gives the percentage of that graph's metric belonging to the programming langauge for the year.

Table 1: Top ten programming languages on SourceForge by popularity rankings

	Project Rank	Author Rank	File Rank	Revision Rank	LOC Rank	Final Rank
C	1	1	2	2	1	1
Java	2	2	1	1	2	2
C++	4	3	4	4	3	3
PHP	5	4	3	3	4	4
Python	7	7	5	5	5	5
Perl	3	5	9	9	6	6
JavaScript	6	6	6	8	10	7
C#	9	9	7	6	7	8
Pascal	8	10	8	7	8	9
Tcl	11	8	10	10	9	10

Many interesting trends in language popularity can be observed in these figures. For example, there is a strong similarity between Figure 1 and Figure 2 which confirms observations made in our previous work [2] of an approximate one-to-one correspondence between projects and authors for the majority of SourceForge projects. We also observe that the percentage shares of some languages, particularly C++, Python, and Pascal, have remained approximately constant, while others, such as Java, PHP, and JavaScript are increasing, and still others, including C, Perl, and Tcl, are decreasing.

Figures 3 and 4 provide interesting langauge contrasts. For instance, the percentage of files written in C is decreasing at a much faster rate than the number of lines of code written in C, while the opposite is true for Java. This indicates that Java source files tend to contain fewer lines of code than C source files, an observation that seems reasonable given the restrictions Java places on the contents of individual files. In addition, we can see that Perl, Python, and Tcl have much smaller percentages of the files and the lines of code than they do of the projects and authors. This may reinforce the claims of adherents of these languages that their high levels of abstraction allow developers to achieve comparable levels of functionality in far fewer lines of code or it may indicate that these are niche languages which are not used at the core of large systems but rather at the peripheries.

5. MULTIPLE LANGUAGES

The proliferation of programming languages shown in the figures in Section 4 begs the question of who is using the new programming languages. Possible answers to this question are: 1) authors are beginning to use more languages over time, 2) new authors entering the community are using the newer languages while existing authors continue to use the older languages, or 3) existing authors are migrating to the new languages while new authors are moving into the community to take their places using the older languages. To investigate this question, we chart the distributions of programming languages used per author and per project. Figures 5 and 6 show the percentage of projects and authors using between one and ten of the languages listed in

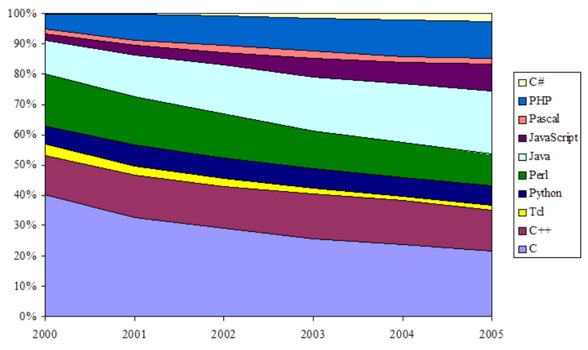

Figure 1: Distribution of projects

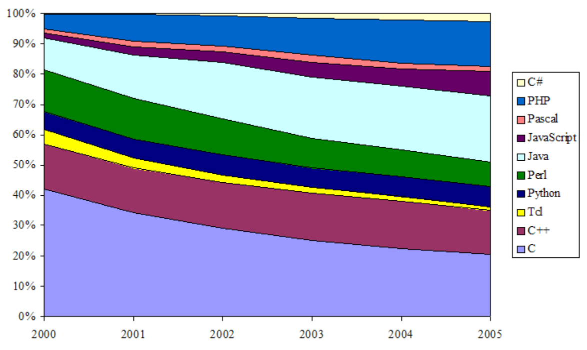

Figure 2: Distribution of authors

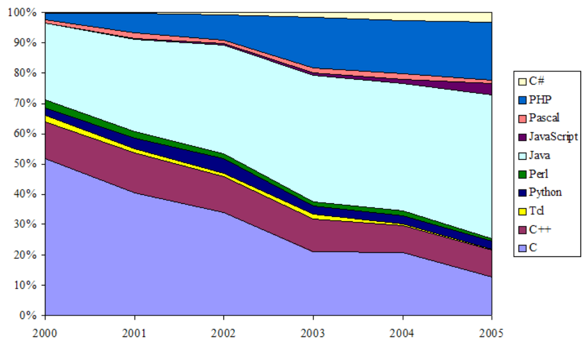

Figure 3: Distribution of files

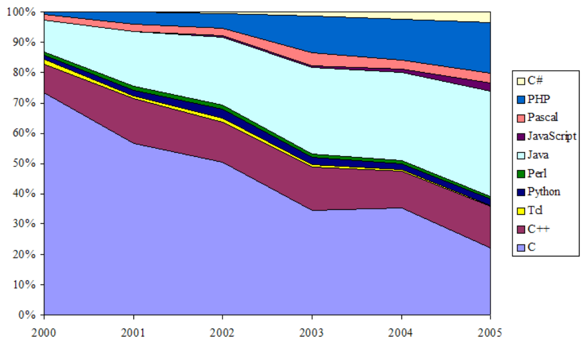

Figure 4: Distribution of lines of code

Table 1 between 2000 and 2005.

If it were the case that more authors were beginning to use multiple languages over time, we would expect to see a flattening of the surfaces in these graphs along the z-axis. However, the distributions in these figures are remarkably consistent with just over 65% of projects and just over 70% of authors using only a single language every year. This provides evidence that either the newer languages are being used by new authors entering the community or that existing authors are migrating to new languages and being replaced by new users entering the community and using the older languages at a comparable rate.

We do, however, see from the figures that there are significant portions of authors and projects that use two and three languages per year (approximately 25% and 10% for authors and 20% and 12% for projects). Comparing the surfaces in Figures 5 and 6 we can see that while there are slightly higher percentages of authors using one and three languages than there are projects using one and three languages, there is a slightly higher percentage of projects using two languages than there are authors using two languages. We do not expect, however, that these differences are significant.

6. LANGUAGE PROFILES

The fact that substantial portions of the authors and projects are using two or three languages as shown in Section 5 leads us to wonder if there are certain language profiles, that is combinations of programming languages that are commonly used. Certainly, anyone familiar with software develpment is aware that there are some languages that complement each other well while there are others that do not. However, our goal here is to empirically investigate not only what language profiles are in use, but how commonly they are used together compared alone. To investigate this question, Table 2 lists the most common programming language profiles for authors and projects by year along with the percentage of authors and projects that used each profile. In the interest of space and readability, all profiles that represent less than 2% of the authors or projects for a given year are excluded from the table.

As we expect based on the figures in Section 5 it is most common for programmers to use a single language. In fact, the top three most common profiles for each year between 2000 and 2005 are single language profiles. However, as the number of languages in use in the community increases we see a clear narrowing of the gap between the top profile and its nearest competitors. Through 2002, the most commonly used programming language for authors and projects is C. However, in 2003, Java becomes the most commonly used language for authors; and, in 2004, Java becomes the most commonly used language for projects.

The most common two-language profiles are C/Perl, C/C++, and JavaScript/PHP. Both the C/Perl and the C/C++ suggest extensions of an old but powerful imperative language to incorporate additional features not supported by the core language. C/Tcl and C/Python were also popular prior to 2003, but seem to have fallen out of use since, apparently for different reasons. It appears from the table that Python has continued to be used, but it is now used primarily on its own; while TCL has fallen out of general use among the projects in our study. The JavaScript/PHP profile is common in web development where PHP is used to provide function-

Table 2: Most common langauge profiles by year for authors and projects

2000				2001			
Author Languages		Project Languages		Author Languages		Project Languages	
C	35%	C	32%	C	27%	C	25%
C++	9%	Java	10%	Java	14%	Java	13%
Java	9%	Perl	8%	C++	9%	Perl	7%
Perl	6%	C, Perl	7%	PHP	7%	C++	6%
C, Perl	5%	C++	7%	Perl	6%	C, C++	6%
C, C++	5%	C, C++	5%	C, C++	5%	PHP	5%
PHP	4%	Python	3%	C, Perl	4%	C, Perl	5%
Python	3%	PHP	3%	Python	3%	Python	3%
Tcl	2%	C, Tcl	2%	C, Python	2%	C, C++, Perl	2%
C, Tcl	2%	C, Python	2%			C, Python	2%
C, Python	2%	C, C++, Perl	2%				
C, C++, Perl	2%						

2002				2003			
Author Languages		Project Languages		Author Languages		Project Languages	
C	22%	C	22%	Java	21%	C	19%
Java	19%	Java	16%	C	19%	Java	18%
C++	9%	C++	7%	C++	10%	C++	7%
PHP	7%	PHP	6%	PHP	9%	PHP	6%
Perl	5%	Perl	6%	C, C++	5%	C, C++	5%
C, C++	5%	C, C++	5%	Perl	4%	Perl	5%
Python	4%	C, Perl	4%	Python	4%	Python	3%
C, Perl	3%	Python	3%	C, Perl	2%	C, Perl	3%
Pascal	2%	Pascal	2%	JavaScript, PHP	2%	JavaScript, PHP	3%
		JavaScript, PHP	2%	Pascal	2%	Pascal	2%
		C, Python	2%			C++, Perl	2%
						Java, JavaScript	2%

2004				2005			
Author Languages		Project Languages		Author Languages		Project Languages	
Java	22%	Java	20%	Java	23%	Java	21%
C	16%	C	17%	C	15%	C	16%
PHP	11%	C++	8%	PHP	11%	C++	8%
C++	10%	PHP	8%	C++	10%	PHP	7%
C, C++	4%	Perl	5%	Python	4%	C, C++	4%
Python	4%	C, C++	5%	JavaScript, PHP	4%	JavaScript, PHP	4%
Perl	3%	JavaScript, PHP	3%	C, C++	4%	Perl	3%
JavaScript, PHP	3%	Python	3%	Perl	3%	Java, JavaScript	3%
C, Perl	2%	C, Perl	3%	Java, JavaScript	2%	Python	3%
Java, JavaScript	2%	Java, JavaScript	2%	C#	2%	C, Perl	2%
Pascal	2%	Pascal	2%	C, Perl	2%	Pascal	2%
C#	2%			Pascal	2%	C#	2%

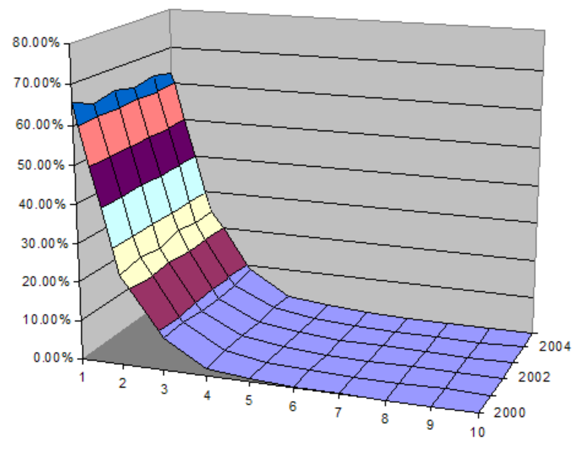

Figure 5: Languages per project by year

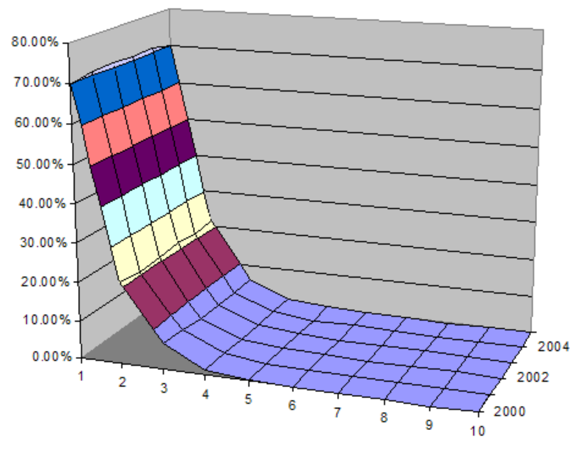

Figure 6: Languages per author by year

ality and JavaScript handles the presentation. In addition, Java/JavaScript has become some what popular in recent years. This profile represents another form of web development where Java Server Pages (JSP) provide functionality instead of PHP.

The only three-language profile evident in the table is C/C++/Perl which is a hybrid of the C/Perl and C/C++ two-language profiles. Another three-language profile which represented just under 2% of the authors and also just under 2% of the projects and does not appear in the table is JavaScript/PHP/Perl which is an extension of the JavaScript/PHP two-language profile.

7. CONCLUSIONS

In this paper we have investigated the trends in programming language use since 2000 among the Open Source projects hosted on SourceForge. While our sample size is substantial, consisting of data gathered from almost 10,000 projects, it is important to remember that this is an observational study. As such, generalization to the larger population of all Open Source projects is not necessarily appropriate. However, these results can be used to suggest additional studies to investigate further the patterns we have observed here and to provide compelling evidence of generalizable trends in the absence of additional contradictory results.

In observing the trends in language popularity over time, we have seen that the web development languages, like Java, JavaScript, and PHP are increasing in popularity; the scripting languages, like Python, Perl, and Tcl, are remaining small but consistent in their popularity; and the traditional desktop development languages, C and C++ are declining in popularity.

Our investigation of the use of multiple programming languages by individual authors and in individual projects revealed that the percentages of authors and projects using one, two, and three languages remained approximately constant from 2000 to 2005 with a little over two thirds using one langauge, a little under a quarter using two languages, and roughly one tenth using three languages.

Further evaluating the most common language profiles we found that the three most popular profiles each year were single language profiles. We also found that the common two-language profiles were web development profiles, like JavaScript/PHP and Java/JavaScript, and classical language extension profiles, like C/C++ and C/Perl.

Each of these results suggests additional studies. For example, would these results hold if we studied Open Source projects hosted on another systems, such as RubyForge or Tigris, or if we studied commercial or governmental software development organizations? Also, what new language profiles are beginning to evolve and is it possible to predict which languages will be or ought to be used in combination?

8. REFERENCES

[1] D. Delorey, C. Knutson, and S. Chun. Do programming languages affect productivity? a case study using data from open source projects. In *Proceedings of the First International Workshop on Emergin Trends in FLOSS Research and Development*, May 21, 2007.

[2] D. Delorey, C. Knutson, and A. MacLean. A comprehensive evaluation of production phases sourceforge projects: A case study using cvs2mysql and the sourceforge research archive, 2007.

[3] G. Goth. News: Not in the script–news of java's demise is premature. *IEEE Distributed Systems Online*, 7(2):4, 2006.

[4] G. Madey. Sourceforge research data archive. http://www.nd.edu/õss/Data/data.html, 2005.

[5] S. J. Nielson and C. D. Knutson. Design dysphasia and the pattern maintenance cycle. *Information and Software Technology*, 48(8):660–675, 2006/8.

[6] L. Paulson. Developers Shift to Dynamic Programming Languages. *Computer*, 40(2):12–15, 2007.

[7] E. Sapir. *Language: An Introduction to the Study of Speech.* Harcourt, Brace, New York, 1921.

[8] T. Software. Tiobe programming community index. http://www.tiobe.com/tpci.htm, 2007.