Finding Code on the World Wide Web: A Preliminary Investigation

James M. Bieman Vanessa Murdock
Computer Science Department
Colorado State University
Fort Collins
Colorado 80523 USA
970-491-7096
bieman@cs.colostate.edu

Abstract

To find out what kind of design structures programmers really use, we need to examine a wide variety of programs. Unfortunately most program source code is proprietary and is unavailable for analysis. The World Wide Web (Web) potentially can provide a rich source of programs for study. The freely available code on the Web, if in sufficient quality and quantity, can provide a window into software design as it is practiced today. In a preliminary study of source code availability on the Web, we estimate that 4% of URLs contain object-oriented source code, and 9% of URLs contain executable code — either binary or class files. This represents an enormous resource for program analysis. We can, with some risk of inaccuracy, conservatively project our sampling results to the entire Web. Our estimate is that the Web contains at least 3.4 million files containing either Java, C++, or Perl source code, 20.3 million files containing C source code, and 8.7 million files containing executable code.

Keywords: Design, source code analysis, World Wide Web estimation, code on the World Wide Web.

1. Introduction

One reason to analyze program source code is to learn more about the kinds of programs that programmers actually write. We can learn how they actually structure their code, to see what design structures are actually used in practice. For example, object-oriented software can be characterized by design properties that include the distribution of functionality across classes, the distribution of the size of classes, methods, and interfaces, and even the number and quality of comments. We can determine the extent that programmers actually apply current recommended design practices such as the use of design patterns [5] or the use of short methods with minimal comments [4].

Studies of the design structure of actual programs can potentially lead to the discovery of some observation based "natural laws" of programming. For example, empirical studies may show that certain design structures are more or less adaptable or error prone. We may find that programs in one application domain have different structures that those in other application domains. Only observational studies can discover relationships that are not expected.

Observational studies require data. Ideally, we would like to sample a wide-range of applications in different domains. And the samples should be representative of the universe of all programs. Unfortunately, defining the notion of a universe of application domains is an elusive goal. We can list many application domains such as graphical user interfaces (GUI's), language processing (i.e., compilers), telecommunications software, embedded systems, etc. However, any classification is likely to be incomplete and the relative amount and distribution of software in the different domains is unknown.

Even if we can define a universe of application domains, obtaining a reasonably random sample of actual applications is just about impossible. Most companies are not willing or able to share their source code for the purpose of study due to proprietary concerns. Companies occasionally provide source code for studies, but such source code is usually of just one or two systems. This data is very useful, but its usefulness is limited to case studies that do not easily give generalizable results [3].

Although it may never be possible to characterize all programs, we can study one large universe of readily available programs: the program code that is posted to the World Wide Web (Web). Our conjecture is that there is a large quantity of analyzable program code on the Web. In this paper, we provide an initial characterization of this code. We use sampling techniques to estimate the relative amount of program code on the Web, and generate initial descriptive metrics to characterize this sample.

Prior studies have estimated the amount of specific kinds of information on the Web. Lawrence and Giles [8] used randomized internet addresses to estimate the number of Web servers, and content of various general classifications: scientific/educational, pornography, government, health, personal, community, religion, and societies. The mainstream media frequently quotes this study's estimate of the distribution of pornographic sites on the Web. Grefenstette and Nioche [6] used the frequency of language-specific words to estimate the growth of content of non-English based natural language sites on the Web. Lawrence and Giles [7] also used real queries, queries of employees of the NEC Research Institute, to estimate the coverage (of the Web) of 11 major search engines, and the overlap between them. They have also studied the organization of information access on the Web [9], and mechanisms for automating Web-based citation indices [10].

We did not find any prior work that focused on estimating the amount of program code, whether source or executable code, that is available on the Web. In this study, we make an initial attempt to determine how much code is posted to the Web. Our general approach is to generate randomly selected URLs, and then examine the referenced Web pages to determine which ones contain downloadable code. We develop a rough characterization of the code and estimate the relative amount available on the entire Web.

2. Study Method

The first step is to estimate the required sample size, the number of web pages that must be examined to accurately determine the proportion containing program code. The estimates are based on the relative number of pages that contain the following keywords: "download", "source code", or "ftp". The initial rough estimate determines the required sample size.

We query four search engines to generate the required number of web pages. That is, we generate a number of web pages equal to the required sample size from each of the four search engines — HotBot, Lycos, Altavista, NorthernLight. We identify pages containing the same keywords ("download", "source code", or "ftp"). We conduct an in depth hand analysis of a relatively small subset of the pages with the key words to help us to design an automated search of the full sample. Then we conduct a search of the full sample and extract any downloadable files and analyze what they contain. We count the number of pages that actually contain code rather than just mentioning the word "download" or containing non-code downloadable files.

We identify the language, the number of lines of code, and the number of classes. We did not specifically target any of the commonly known sources of code, such as the Perl modules or the Netscape source code, as our purpose was to explore less obvious sources of code. However, we did not exclude these sources, and if they appeared in our sample they were counted.

2.1. Estimating the Required Sample Size

In 1999, Lawrence and Giles [8] estimated that the Web contained 800 million indexable web pages. Bharat and Broder [1] estimate that the Web is growing at a rate of 20 million pages a month. The total number of pages indexed by any of the search engines in this study so greatly outnumbers pages containing source code that for the purposes of sampling, we can assume the total population is infinite.

To obtain an initial estimate of how many pages contain source code, we generate 10 groups of 100 random URLs, and counted the pages that contained the words "download" "source code" or "ftp". We compute the required sample size using the proportion of source code web pages in the group with the largest proportion, because that value will require the largest sample size. Using a larger sample than needed will produce more accurate results.

The following commonly used formula expresses the relationship between the sample size, N, and the desired accuracy, d:

$$d = z_c \cdot \sqrt{\frac{pq}{N}} \cdot \sqrt{\frac{N_p - N}{N_p - 1}}$$

where z_c is the confidence level found on a z-table, p is the estimated proportion of pages in the sample that contain program code, q is $1 - p$, and N_p is the total number of web pages in the universe.

We can assume that N_p is very large, for our purposes it approaches infinity. Then the last subexpression approaches 1 — the sample size N is negligible compared to N_p; Thus we can remove the subexpression $\sqrt{\frac{N_p - N}{N_p - 1}}$ from the equation. Solving for N, we get the following:

$$N = \frac{z_c^2 pq}{d^2} \qquad (1)$$

From a z-table, setting $z_c = 1.96$ gives a 95% confidence level. Setting $d = .01$ gives a 1% accuracy.

2.2. Generating the Sample

Search engines use deterministic algorithms to index and rank pages, and each engine uses a different algorithm. We must ensure the randomness of any sample of search engine results.

The approach used by Lawrence and Giles [7] was to gather queries from their colleagues used in the course of their daily work. This approach is practical and appropriate because Lawrence and Giles were studying the accessibility of scientific information on the web. To account for the different ranking systems used by search engines, Lawrence and Giles gathered the entire result of each query from the search engine. We did not use this approach in our study, since it does not produce a random sample. It is likely that research scientists use similar queries.

The approach used by Bharat and Broder [2] was to use a lexicon of common internet terms as queries, and then to randomly sample from the results. We do not have a lexicon of common internet terms, thus we took 1.5 million queries from the peek page of the meta-search engine AskJeeves, and the meta-spy page of MetaCrawler. AskJeeves is a natural language meta-search engine, while MetaCrawler is a typical meta-search engine. Using queries from both AskJeeves and MetaCrawler include a wider variety of queries than only using queries from our colleagues, and provide more realistic "real world" queries than using a list of common internet keywords. To compensate for the different ranking systems of the search engines, we took the entire set of results returned from each query. This approach is consistent with the work done by Lawrence and Giles [7].

To randomize the query set, the queries are read into a hash table (since hash tables do not preserve order) and then into an array so that we could access them by index. The sample is generated from the array using a random number generator to generate the array indices to be selected. The queries are posed to AltaVista, Lycos, HotBot and NorthernLight separately over the course of several days, since search engines are known to return different results depending on internet traffic, and traffic varies at different times of day. To generate the sample, the resulting URLs are randomized and selected in the same way as the queries.

2.3. Examining A Portion of The Sample By Hand

We conduct a hand-analysis of a subset of the sample URLs to better understand how downloadable source code is referenced and distributed among web pages. The objective is to determine how many of the sample URL's that contain the keywords "download", "source code", or "ftp" actually contain source code. This exercise gives us further insights into how to design an automated search of the full sample.

2.4. Finding Source Code in the Full Sample

From each of the samples of queries from the four search engines, we downloaded any pages that contained the words "download", "source code", or "ftp". We examine these files and download any links that had the following file extensions: .tar, .jar, .gz, .zip, .pl, .java, .tgz, .z. We also downloaded any publicly accessible files from any URL contained in the pages that began with "ftp://".

To ftp all the publicly available files, we follow the directory structure indicated by the ftp address, and store any file listed in that directory. In the case of URLs and links that used http, we download only those files available from that specific link — we do not follow any links that redirect a browser to another location. The limitation of the search to the top layer in the sample of web sites provides a conservative estimate, and demonstrates what is practically available.

In order to process the large number of pages in a sample, the search must be automated. We encode the search process using Perl scripts.

3. Results

3.1. Required Sample Size

Our first estimate was taken from ten samples of 100 random URLs generated from AltaVista. Our objective here is to get a rough estimate of the required sample size, not the estimate of the amount of code. We need to determine how many URLs to generate from each search engine for the purpose of estimating the amount of code to ensure a 95% confidence level and $\pm 1\%$ accuracy. We need to use only one search engine here and we chose AltaVista because of claims that it indexes the most pages. Using the search engine that indexes the greatest number of pages should produce the highest estimate of required sample size.

Between 6% and 19% of the samples contain downloadable programs. We used the 19% estimate because it requires a larger sample which should give us the most accurate estimate. Plugging this estimate into Equation (1) gives a sample size of 5920. That is, a sample of 5920 URLs is large enough to ensure that our estimate is within 1% with a 95% confidence level.

3.2. The Sample

We took samples of 5920 URLs from each of four different search engines. Thus the sample included 23,680 URLs, although they were not all distinct, since the search engines have a small overlap. Only three URLs produced in the four samples from AltaVista (685 URLs), Lycos (867 URLs), HotBot (865 URLs), and NorthernLight (789 URLs) were exact matches. Bharat and Broder [2], in estimating the size and overlap of search engine coverage, estimated average overlap of any two search engines to be 8%, the estimated overlap of any three search engines to be 4% and the estimated overlap of any four search engines to be 1.4%.

AltaVista produced 685 pages, or 11% that contain the keywords "download", "source code", or "ftp". Northern-

Table 1. Hand analysis of 225 randomly generated links from AltaVista that match the keywords “download”, “source code”, or “ftp”. Some links refer to more than one downloadable file, or a file that matches more than one classification.

File Type	Number of Links Referencing Downloadable Files
Executable files	53
Source code	9
Sound, video, or image	46
Freeware	5
Shareware	11
Commercial demos	9
Commercial repositories	18
Text or no download files	123

Light produced 789 pages, or 13%, HotBot produced 865 or 14.6%, and Lycos produced 867 or 14.6%.

AltaVista, HotBot, Lycos and NorthernLight produced 3203 unique pages containing the significant keywords. Although the number of pages containing source code is a small fraction of the total Web, most of these pages contain multiple files. Thus a relatively small number of sites can produce many downloadable files.

3.3. Hand Analysis Results

Of the 5920 AltaVista URLs in the sample, 685 URLs reference pages contain the keywords “download”, “source code”, or “ftp’. We examined 225 of these pages by hand to see how many contained source code. Table 1 displays the results of this analysis.

The web sites contained content in the following categories: executable files, source code, images, video, sound, and games. In addition, we noted software that is identified as “freeware”, “shareware” or “demo”. We identified any download link that led to larger commercial repositories (such as zdnet or cnet), because a number of pages, although they claim to contain downloadable files, actually refer to common download points.

Pages containing downloadable copies of Internet Explorer, Linux, Netscape, winamp, realtime player and similar widely available free commercial executables were not counted in the list of downloadable files to avoid unfairly skewing the estimate. Many pages that contain no downloadable files have links to download Netscape or Adobe Acrobat so that their material can be viewed or heard correctly. These links do not indicate that the body of code available is larger, since they all point to the same product.

Of the 225 links containing the keywords “download”, “source code” or “ftp” from AltaVista, 53 contained executable files, 9 contained source code, 46 contained sound, video, or image files, 19 contained games, 18 pointed to commercial repositories. Five of the links contained code explicitly labeled “freeware”, 11 contained code explicitly labeled “shareware” and 9 contained commercial demos. Most sites (123) contained either no downloads at all, or contained text downloads, or widely available commercial downloads as discussed above. Based on the number of sites containing at least one source code file (9 in 225), 4% of the web sites contain at least one source code file.

Examining a number of web pages by hand made it obvious that many pages returned by the search engine do not contain downloadable files themselves, rather they contain links, in some cases several pages deep, to downloadable files. Many of the links are not identified by keywords that could be used to automate the parsing of the page. In order to get every downloadable file, an exhaustive search of all links would be necessary. Such a search would be intractable. Even limiting the search to follow only those links identified by keywords as containing downloadable files is impractical.

We are interested in a rough, but conservative estimate of available source code, and we want to obtain results in a timely fashion. Thus, the full search can be safely limited to one level — search the parent page returned in the full sample, but not its children or grandchildren. The results of such a search will be conservative; many more pages actually contain source code than our estimate.

3.4. What We Found in the Full Sample

To distinguish between an individual file, and the collection of files contained in one download, we refer to the collection of files contained in one download as a *download bundle* and an individual file as a *file*. We downloaded 5767 bundles containing 125,011 files. The average number of files per bundle was 28.22. The bundles represent 426 or 13% of the URLs. The vast majority of source code files were c files, also represented were java source files and java class files, perl files, tcl files and c++ files. Table 2 shows the number of files with each extension in this sample.

Because of the time it takes to process all of the files, and because the files can contain absolutely anything we find on the Web, we were not able to process all of the files. We had a total of 5767 bundles, and of those we were able to examine 4430 bundles, around 77%. From these URLs we obtained 38,124 source code files in C, C++, Java, Tcl, and Perl, mdl and xml.

Of the total set of files in the sample, the average number of lines per file was 469, with a range from 1 line to

Table 2. Types of program and program design files found in the sample of 5,767 bundles containing 125,011 files.

File Type	Number Found	% of Files Dowloaded
C++ (.cc, .c++, .cpp)	1,560	1.25%
C (.c)	25,836	21%
java source (.java)	1,633	1.3%
java class (.class)	9,134	7.3%
binary executables (.o or .exe)	2,675	2.1%
C/C++ headers (.h)	7,210	5.8%
Perl (.perl, .pl, .pm, .pml)	1,169	.9%
Tcl (.tcl)	659	.5%
XML (.xml)	22	–
Object model (.mdl)	17	–

Table 3. Unique classes found in the program files contained in the sample.

Language	Number of Classes
Java	6,112
C++	5,639
Perl	1,867

2,781,535 lines. The average number of bytes was 37701 with a range from 0 to 189,531,985 bytes.

We identified the unique classes written in the object-oriented languages. For Java bundles the unique file names were counted without the .java or .class extensions. Thus, all public classes and interfaces are counted as classes. For C++ or Perl, we counted the occurrences of the keyword “class” in the .h files or “package” in Perl files. Table 3 shows the number of classes found in each language.

4. Code on the Web

We can conservatively project the results from the sample to the entire Web.

The 4430 bundles that we examined produced a total of 125,011 files, of which 38,124 were source code files. This means that 30% of the files downloaded from the sample contained source code. We can be 95% sure that 30% $\pm$ 1% of files containing the words “download”, “source code”, or “ftp” will contain actual source code.

Our original sample included 5920 URLs from each search engine. It yielded between 11% (AltaVista) and 14.6% (HotBot and Lycos) web pages that contained the keywords “download”, “source code” or “ftp”, and 30% of files downloaded from those pages ($\pm$1%) contain source code. The Web is estimated to be 800 million pages, and a conservative estimate is that 11% of those contain the download keywords that corresponds to 88 million pages. In our sample, 13% of the URLs, or 3,203 of the pages that contain the keywords representing 426 URLs, actually had something to download. If we assume the same proportions in the entire Web, then 88 million pages out of 800 million have the keywords. Of those 88 million pages, 11.4 million pages actually have something to download. Since the average download yields 28.2 files, the 11.4 million pages will yield on average 323 million files. Of the 323 million files, 30% $\pm$ 1% will contain source code, giving an estimate of $\sim$97 million source code files that are available for download on the Web.

The total representation of source code written in object oriented languages (Java, C++, and Perl) includes 4362 or 3.5% of the files downloaded. Note that we did not include C++ header files since we did not distinguish between C++ and C headers (both have .h suffixes). Assuming that the same proportions hold over the entire Web, we can estimate that 3.4 million files (3.5% of 97 million) on the Web contain source code in C++, java, or perl.

The total number of executable files (including .exe, .class, .o files) represent 11,809 files or 9% of our sample files. Again, assuming that the same proportions apply to to the entire Web, 9% of the downloadable files on the Web (9% of 97 million) gives an estimate of 8.7 million files on the Web containing executable files.

5. Conclusions and Future Work

We sampled the Web to estimate the amount of code that is available for analysis. Our random sample of 23,680 URLs, we were able to download 4430 bundles of information. The bundles contain 125,011 files. In these files, we found 38,124 source code files in various languages including C, C++, Java, Perl, and others. This sample represents a statistically significant, but tiny fraction of the estimated 800 million URLs in the entire Web. Clearly, there is a large amount of program code that is available on the Web.

We are planning further analyses of the content and structure of the programs in our sample. This analyses will include studies of the class dependencies and use of design patterns in object oriented code. We also plan to conduct a more exhaustive search for program by following links to files with downloadable code more than two pages deep in any given site. Future work will include evaluations of other search engines. Larger search engines that index more of the Web, or search engines known for indexing scientific or technical information may prove worthwhile.

The estimates of available program code are conservative. The search aimed for a limited set of file extensions and a limited set of keywords to identify potential source code.

We did not include known sources of code, and code repositories. Thus, there may be much more code on the Web than indicated by our study.

One of our long term research objectives is to build a repository of actual software designs, code, and other artifacts. We are encouraged to find that so much code is available from the Web to populate the repository.

6. Acknowledgement

This work is partially supported by U.S. National Science Foundation grant CCR-0098202, and by a grant from the Colorado Advanced Software Institute (CASI). CASI is sponsored in part by the Colorado Commission on Higher Education (CCHE), an agency of the State of Colorado. We thank the anonymous reviewers whose suggestions greatly improved the presentation.

References

[1] K. Bharat and A. Broder. Measuring the web. *7th International WWW Conference*, 1998.

[2] K. Bharat and A. Broder. A technique for measuring the relative size and overlap of web search engines. *7th International WWW Conference*, 1998.

[3] J. Bieman, D. Jain, and H. Yang. OO design patterns, design structure, and program changes: An industrial case study. *Proc. Int. Conf. Software Maintenance*, 2001. To Appear.

[4] M. Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, Reading MA, 1999.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading MA, 1995.

[6] S. Grefenstette and J. Nioche. Estimation of english and non-english language use on the www. *Proc. RIAO 2000, Content-Based Multimedia Information Access*, pages 237–246, 2000.

[7] S. Lawrence and C. L. Giles. Searching the world wide web. *Science*, 280(5360):98–100, 1998.

[8] S. Lawrence and C. L. Giles. Information on the web. *Nature*, 400:107–109, 1999.

[9] S. Lawrence and C. L. Giles. Searching the web: General and scientific information access. *IEEE Communications*, 37(1):116–122, 1999.

[10] S. Lawrence, C. L. Giles, and K. Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67–71, 1999.