Measuring Similarity of Large Software Systems Based on Source Code Correspondence

Tetsuo Yamamoto[1], Makoto Matsushita[2], Toshihiro Kamiya[3], and Katsuro Inoue[2]

[1] College of Information Science and Engineering, Ritsumeikan University,
Kusatsu, Shiga 525-8577, Japan
Phone: +81-77-561-5265, Fax:+81-77-561-5265
tetsuo@cs.ritsumei.ac.jp
[2] Graduate School of Information Science and Technology, Osaka University,
Toyonaka, Osaka 560-8531, Japan
Phone: +81-6-6850-6571, Fax:+81-6-6850-6574
[3] Presto,Japan Science and Technology Agency,
Current Address:Graduate School of Information Science and Technology, Osaka University,
Toyonaka, Osaka 560-8531, Japan
Phone: +81-6-6850-6571, Fax:+81-6-6850-6574
{matusita, inoue, kamiya}@ist.osaka-u.ac.jp

Abstract. It is an important and intriguing issue to know the quantitative similarity of large software systems. In this paper, a similarity metric between two sets of source code files based on the correspondence of overall source code lines is proposed. A Software similarity MeAsurement Tool SMAT was developed and applied to various versions of an operating system(BSD UNIX). The resulting similarity valuations clearly revealed the evolutionary history characteristics of the BSD UNIX Operating System.

1 Introduction

Long-lived software systems evolve through multiple modifications. Many different versions are created and delivered. The evolution is not simple and straightforward. It is common that one original system creates several distinct successor branches during evolution. Several distinct versions may be unified later and merged into another version. To manage the many versions correctly and efficiently, it is very important to know objectively their relationships. There has been various kinds of research on software evolution[1, 2, 3, 4], most of which focused on changes of metric values for size, quality, delivery time or process, etc.

Closely related software systems usually are identified as product lines, so development and management of product lines are actively discussed[5]. Knowing development relations and architectural similarity among such systems is a key to efficient development of new systems and to well-organized maintenance of existing systems[6].

We have been interested in measuring the similarity between two large software systems.This was motivated by our scientific curiosity such as what is the quantitative similarity of two software systems. We would like to quantify the similarity with a solid

F. Bomarius and S. Komi-Sirviö (Eds.): PROFES 2005, LNCS 3547, pp. 530–544, 2005.

and objective measure.The quantitative measure for similarity is an important vehicle to observe the evolution of software systems, as is done in the Bioinformatics field. In Bioinformatics, distance metrics are based on the alignment of DNA sequences. Phylogenetic trees using this distance are built to illustrate relations among species[7]. There are huge numbers of software systems already developed in the world and it should be possible to identify the evolution history of software assets in a manner like that done in Bioinformatics.

Various research on finding software similarities has been performed, most of which focused on detecting program plagiarism[8, 9, 10]. The usual approach extracts several metric values (or attributes) characterizing the target programs and then compares those values.

There also has been some research on identifying similarity in large collections of plain-text or HTML documents[11, 12]. These works use sampled information such as keyword sequences or"fingerprints". Similarity is determined by comparing the sampled information.

We have been interested in comparing all the files. It is important that the software similarity metric is not based on sampled information as the attribute value (or fingerprint), but rather reflect the overall system characteristics. We are afraid that using sampled information may lose some important information. A collection of all source code files used to build a system contains all the essential information of the system. Thus, we analyze and compare overall source code files of the system. This approach requires more computation power and memory space than using sampled information, but the current computing hardware environment allows this overall source code comparison approach.

In this paper, a similarity metric called S_{line}, is used, which is defined as the ratio of shared source code lines to the total source code lines of two software systems being evaluated.

S_{line} requires computing matches between source code lines in the two systems, beyond the boundaries of files and directories. A naive approach for this would be to compare all source file pairs in both systems, with a file matching program such as *diff*[13], but the comparison of all file pairs does not scale so that it would be impractical to apply to large systems with thousands of files.

Instead, an approach is proposed that improves efficiency and precision. First, a fast, code clone (duplicated code portion) detection algorithm is applied to all files in the two systems and then *diff* is applied to the file pairs where code clones are found.

Using this concept, a similarity metric evaluation tool called *SMAT*(Software similarity MeAsurement Tool) was developed and applied to various software system targets. We have evaluated the similarity between various versions of BSD UNIX, and have performed cluster analysis of the similarity values to create a dendrogram that correctly shows evolution history of BSD UNIX.

Section 2 presents a formal definition of similarity and its metric S_{line}. Section 3 describes a practical method for computing S_{line} and shows the implementation tool *SMAT*. Section 4 shows applications of *SMAT* to versions of BSD UNIX. Results of our work and comparison with related research are given in Section 5. Concluding remarks are given in Section 6.