International Journal of Engineering Inventions
ISSN: 2278-7461, www.ijeijournal.com
Volume 1, Issue 1(August 2012) PP: 38-41

Source Code Reusability Metric for Enhanced Legacy Software

Shashank Dhananjaya[1], Yogesha T[2], Syeda Misba[3]
[1]*Assistant professor, Dept. of Information Science &Engg, Vidya Vardhaka College of Engineering, Mysuru, Karnataka INDIA*
[2]*Lecturer, Center for PG Studies, VTU, Bengaluru, Karnataka INDIA*
[3]*Assistant professor, Dept. of Information Science & Engg, New Horizon College of Engineering, Bengaluru, Karnataka INDIA*

***Abstract*—Most of the application software's are developed by reusing the source code in order to reduce the development effort, delivery time, to improve the productivity and quality. Software reuse has been a solution factor to acquire the existing knowledge from software repository. To add new functionalities source code of the older version are reused. It is difficult keep track the source code lines that are being reused i.e. changed and added. There is no indicator of the productivity of the project, nor does it give an insight to the number of expected defects. Both of these are related to the actual number of new and changed source code lines.In this paper, we have proposed novel algorithm for software reusability metric using pattern matching. We have implemented using C# language to evaluate. Result determines the reused files added and changed.**

***Keywords*—software development, software reuse, pattern matching, software metrics, knowledge reuse.**

I. INTRODUCTION

Software is continuously growing in its importance for application development. Currently at Hewlett-Packard Company, approximately 70% of our engineers do some software development. SystematicSoftware reuse is becoming a key factor in reducing the development time and effort in the software development process and also to improve the software quality and productivity. New horizons are opened since the idea of using the existing knowledge for software reuse appeared in 1968 [1]. The main idea behind the software reuse is domain engineering (aka product line engineering). Reusability is the degree to which a thing can be reused [2]. Software reusability represents the ability to use part or the whole system in other systems [3,4] which are related to the packaging and scope of the functions that programs perform [5]. According to [6], the US department of defense alone could save 300 million $ annually by increasing its level of reuse as little as 1%. Moreover, software reusability aimed to improve productivity, maintainability, portability and therefore the overall quality of the end product [7]. Ramamoorthy et. al.[8] mentions that the reusability is not limited to the source code, but it has to include the requirements, design, documents, and test cases besides the source code. New technologies andtechniques are required to reuse the existing knowledge from software historical data such as code bases, execution traces, historical code changes, contains a wealth of information about a software project's status, progress, and evolution. Basically software reuse process consists of four steps such as identifying the software components, understanding the context, applying software reuse techniques, integration and evaluating.

II. RELATED WORKS

Several research works has been carried out on software reuse by many authors. Morisio et. al [9] has identified some of the key factors such as adapting or running a companywide software reuse. Prediction of reusability of object oriented software systems using clustering approach have been made by [10]. G. Boetticher et.al proposed a neural network approach that could serve as an economical, automatic tool to generate reusability ranking of software [11]. Morisio et. al [9] TSE article success and failure factors in software reuse sought key factors that predicted for successful software reuse. Tim Menzies et. al [12] has identified numerous discrepancies which exist between expert opinion and empirical data reported by Morisioet.al.'s in TSE article.Hironori Washizaki[13]found that metrics can effectively iden-tify black-box components with high reusability.ShrddhaSagar[14]proposed aapproach using Fuzzy logic to select highlyreusable components in the systems will eventuallyhelp in maintaining the system in a better way.G.Singaravel[15] told Reusability is motivated to reduce time and cost insoftware development.YunwenYe [16]proposedCodeBroker model that helps Java programmers in learningreusable software components in the context of their normaldevelopment environments and practice by proactivelydelivering task-relevant and personalized information.Mogilensky[17]written about the importance ofsoftware component reuse as a key means of improving software productivity and quality. Young Lee and Kai H. Chang[18]In his paper he proposed a quality model for object-oriented software and an automated metric tool.Octavian Paul[19] proposed metrics and a mathematical model for the of reusability, adaptability, composeabilityand flexibility of software components.

III. PROPOSED ALGORITHM

Various approaches for source level comparison of different software releases are available but they all have a serious shortcoming that makes them unsuited for source code metrics with current software development practices they do not consider files that are being moved or renamed during the course of development. Also, they do not take into account

that developers create a significant amount of code by a copy-and-modify approach. This is where this novel approach is different. This approach compares two source code trees one is called the current source code tree, the other the previous source code tree and reports which files are changed, added, deleted or unmodified. For each of the changed files, it is reported how many lines are added or deleted. However, it does not assume a file that is not present in the previous source code tree is a completely new file. It uses powerful heuristics to determine if another file in the previous source code tree is highly similar. If so, that file is considered the previous version and the number of added and deleted lines for the current file is determined. It turns out that a significant number of files are derived from other files in your development organization. This novel approach helps especially in multi-site project teams it gives more control of and confidence in the metrics collected on the source code being developed.

A. Determining the difference between files

For the difference between file A and file B, the following algorithm is used.

1. First comments (in C or C++ syntax) are removed.
2. Any whitespace is removed for each line in the file.
3. If the line is empty, it is removed.
4. If a file is considered to be 'generated' (see heuristics above) it is ignored; other files are processed further.
5. A hash code is calculated for each line (the Hash algorithm of .Net for strings is used). This reduces a file to an array of 32bit integers.
6. This array is sorted. From this, the number of new entries (= new lines) and the number of removed entries (= deleted lines) is determined in a straightforward way.

B. Algorithm to find matching files

Finding matching files between the older source code tree and the newer sourced code tree is done in two steps: first potentially matching files are coupled using the Locators second the content of the potentially matching files is examined to determine whether the newer one is indeed derived from the older one.

The Locator defines, given a file from the older source code tree, how to find a match in the older source code tree:

- SameParentSameName:for a file <new>/mix/sol2/mes.c, only the file <old>/mix/sol2/mes.c is a possible match. The content of the file mes.c is not examined.
- SameParentDiffName: for a file <new>/mix/sol2/mes.c, any file in the directory <old>/mix/sol2/ is a possible match. Whether a file actually matches is determined by comparing the contents (see below).
- DiffParentSameName: for a file <new>/mix/sol2/mes.c, the file mes.c in any directory in <old> is a possible match. The content of all files with the name mes.c are examined to find the one with the best match.
- DiffParentDiffName: for a file <new>/mix/sol2/mes.c, any file at any location in <old> is a possible match. Whether a file actually matches is determined by comparing the contents.

To determine if the file A (in the old source code tree) that is matched by the algorithm using the Locators as described above is indeed a previous version of file B (in the new source code tree), the following algorithm is used for each pair of potentially matching files A and B:

Algorithm 1. Pattern Matching for Software Reuse Metric

1. Let A be the old version file and B be the new version file and following parameters are calculated:

- The number of new lines in B that are not in A: $N(B)$
- The number of deleted lines in A that are no longer in B: $D(A)$
- The total number of lines in A: $LC(A)$
- The total number of lines in B: $LC(B)$

2. The matching criteria are tested. The files are considered to match if one of the following conditions hold:

- $N((B) < \frac{LC(B)}{10}$, or less than 10% of file B is new.
- $D(A) < \frac{LC(A)}{4} \wedge N(B) < \frac{LC(B)}{2}$, or less than a quarter of file A is deleted and less than half of file B is new. There is no scientific reasoning for the number used in those formulas; they are determined by experimentations and guesstimates and built into the tool.

3. If they fulfil the matching criteria (M(A,B) the match value is determined as:

$$M(A,B) = \frac{D(A)}{LC(A)} + \frac{N(B)}{LC(B)}.$$

The pair of files A and B with the lowest match value is considered to be the pair where file B is derived from file A.

Note that extensions do not play a role in the matching of files. I.e. it is possible that mes.c is considered to match with include.h. This does (infrequently) occur when code is moved from header files to a source code file (or the other way around).

IV. RESULTS AND DISCUSSIONS

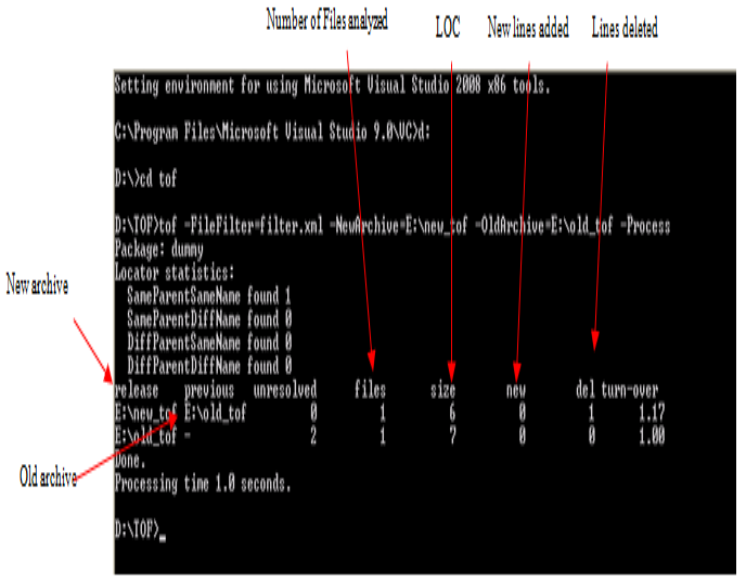

Fig.1. Identification added, changed and deleted source code in old and new version.

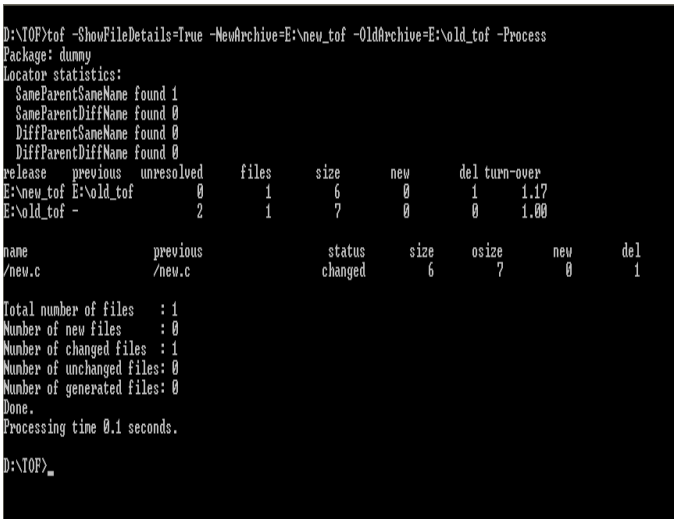

Fig.2. Calculation of total number of new files, changed files, unchanged, generated files and processing time.

This then results in the following output information::
Package: dummy
Locator statistics:
SameParentSameName found 306
SameParentDiffName found 0
DiffParentSameName found 25
DiffParentDiffName found 20
release previous unresolved files sizenew del turn-over
c_new.zip c_old.zip 169 469 58442 33548 8890 1.31
c_old.zip - 408 355 42858 0 0 1.00
name previous status sizeosize newdel
/…/CCeTvPca9554.cd /…/CCeTvPca9554.cd changed 91 93 1 3
/…/cetv9554_m.c /…/cetv9554_m.c unchanged69 69 0 0

/…/cetv9554_mpow.c /…/cetv9554_mpow.c changed 61 50 15 4

…
/ … /cetvscreen_msplit.c new 135 0 135 0
/ … /cetvdef_mconst.cgenerated 0 0 0 0
/ … /ICetvBkedSouWireIds.id new 13 0 13 0
/ … /DCetvPictureProfile.dd new 1250 125 0
Total number of files : 469
Number of new files : 159
Number of changed files : 166
Number of unchanged files: 141
Number of generated files: 3
Done.
Processing time 1.6 seconds

The following explains some of the relevant output lines:

- Lines 3-6 show a summary of how many files were tracked to a previous version although the name or the position in the source code tree was different in the new release. In this case 25 files were moved to a different directory and 20 files were renamed as well as moved to a different directory.
- Lines 7-9 show a summary of the line counting statistics over the entire release.

C. Implementation of the Algorithm

The development tools used to implement the Algorithm is Visual Studio IDE 2005 for C#.Net,Automation test scripts are designed for testing the Algorithm using the PERL. The programming language is C#.

V. CONCLUSIONS AND FUTURE WORK

Software reuse has become the solutions to reduce development time, improve productivity and quality. Data mining techniques can be effectively used in software reuse process. The main contributions for this paper are as follows:

- Identified the need of software reuse in software development practices
- Proposed novel algorithm for software reusability metric using pattern matching

REFERENCES

1. P. Gomes.and C. Bento. "A Case Similarity Metric for Software Reuse and Design," Artif. Intell. Eng. Des. Anal.Manuf., Vol. 15, pp. 21-35, 2001.
2. W.Frakesand C.Terry, Software Reuse: Metrics and Models, ACM Computing Surveys, Vol. 28, No. 2, June 1996
3. J.A. Mccall, et. al., "Factors in Software Quality," Griffiths Air Force Base, N.Y.RomeAirDevelopmentCenter Air Force Systems Command, 1977.
4. N.S Gill. "Reusability Issues in Component-Based Development," SigsoftSoftw. Eng. Notes, Vol. 28, pp. 4-4, 2003.
5. J.J.E Gaffney "Metrics in Software Quality Assurance," Presented at the proceedings of the ACM '81 Conference, 1981.
6. J.SPoulin 1997. Measuring Software Reuse–Principles, Practices and Economic Models, Addison-Wesley,
7. A.Sharma.et al., "Reusability assessment for software components," SigsoftSoftw. Eng. Notes, Vol. 34, pp. 1-6, 2009.
8. C.V.Ramamoorthy, et. al., "Support for Reusability in Genesis," Software Engineering, IEEE Transactions, Vol. 14, pp.1145-1154, 1988.
9. M.Morisio, and C. Tully, "Success and Failure Factors in Software Reuse," IEEE Transactions on Software Engineering, Vol. 28, No. 4, pp. 340–357, 2002
10. AnjuShri, Parvinder S. Sandhu, Vikas Gupta, SanyamAnand, Prediction of Reusability of Object Oriented Software Systems Using Clustering Approach, World Academy of Science, Engineering and Technology, pp. 853-858, 2010.
11. G.Boetticher. and D. Eichmann, "A Neural Network Paradigm for Characterising Reusable Software," Proceedings of The 1st Australian Conference on Software Metrics, 18-19 November 1993.
12. Tim Menzies, Justin S. Di Stefano. "More Success and Failure Factors in Software Reuse", IEEE Transactions on Software Engineering, May 2003
13. Hironori Washizaki,Yoshiaki Fukazawa and Hirokazu YamamotoA Metrics Suite for Measuring Reusability of Software Components
14. N.W. Nerurkar,Shrddha Sagar.A Soft Computing Based Approach to Estimate Reusability of SoftwareComponents
15. G.SingaravelDr.V.PalanisamyDr.A.Krishnanoverview analysis of reusability metrics in software development for risk reduction
16. YunwenYe.Information Delivery in Support of Learning Reusable Software Components on Demand
17. Applying reusability to somm process definition Judah Mogilensky and Dennis Stipe
18. Young Lee and Kai H. ChangReusability and. Maintainability Metrics for Object-Oriented Software
19. Octavian Paul ROTARU Marian DOBREReusability Metrics for Software Components