Journal of
Engineering, Computing and Architecture ISSN 1934-7197
Volume 2, Issue 2, 2008

Assessing Software Process Hotspots via Analysis and Visualiation of Software Repository Data

Shihong Huang, Assistant Professor, Florida Atlantic University, shihong@cse.fau.edu

Abstract

The software development process is an incremental and iterative activity. Source code is constantly changed to reflect changing requirements, to respond to testing results, and to address problem reports. This paper presents a methodology and a toolkit (VITA) for applying source code analysis techniques to configuration management repository data with the aim of identifying the impact on file changes due to change requests and problem reports. The repository data are clustered, analyzed, and visualized in a semi-automated manner according to user-selectable criteria. The approach is illustrated with a real-world model problem concerning software process improvement of an embedded software system.

Keywords: knowledge management, impact analysis, repository data, software maintenance, visualization, process improvement

1. Introduction

Large-scale software systems must be continuously maintained and evolved to respond to shifting business requirements. Business decisions are often made based on the measurements of the software quality and priorities of business goals. The measurements should be able to check if the software had reached the required quality threshold and highlight areas of the software development where special foci are needed. Software product measurements can be used to make general predictions about a system's attributes, such as the number of faults in the system, to identify anomalous components that are likely to have more errors that management can concentrate on these components during quality review process.

A number of large companies such as Hewlett-Packard [8], AT&T [2], and Nokia [13] have introduced metrics programs and are using collected metrics in their quality management processes. Most of the focus has been on collecting metrics on program defects and the verification and validation processes. The measurement used could be control measurements used by software process or predictor measurements used by software products [22], both of which could influence management to make decisions about the software systems.

However, some of software process external quality attributes often cannot be measured directly. Questions such as, "Are the developers spread efficiently?", "Do the components require refactoring?", and "Where do we spend most of our development resources?" are affected by many factors; there is no simple way to answer them. One of the possible solutions to measure these external quality attributes is to use internal attributes that can be derived from entities of software development process, such as configuration management repository.

A configuration management repository includes abundant data of not only configuration items, but also the use of software components, proposed changes, components that could be affected by these changes, and number of changes per entry. However, some of the knowledge is implicitly hidden in the repository data and cannot be obtained by doing simple queries.

This paper presents an integrated approach to process improvement for identifying production problems areas and potential bottlenecks in development. As the first step of this project, software analysis techniques were used on configuration management repository data with the aim of identifying and visualizing the impact on file changes due to change of requests and problem reports. The approach realizes the goal of providing objective assessments of current software production processes and makes implicit information that are hidden in the configuration management repository explicit so that informed decision can be made by project managements.

More specifically, the methodology presented in this paper provides a means for analysts to ask questions of the data that were not previously made available to them. Such questions are qualitative in nature and thus do not possess any direct quantitative correspondence in typical metrics found in repository data [12][15]. The benefit of this methodology is to provide analysts with a means to locate potential bottlenecks in development as well as aid in their dissolution [9]. Goal-Question-(Indicator)-Metrics and Six Sigma are used to as guidelines to measure the software project, process, and product to visualize the integrated information from configuration management repository to raise the overall quality of the product [5][24].

The next section discusses some of the fundamental issues related to software process improvement and measurement techniques. Section 3 outlines an integrated approach to analyze and visualize configuration management repository and quantify qualitative data with the aims of helping identify the impacts of particular changes of requests and problems reports. Section 4 provides a real-world case study involving a large IT company’s non-proprietary data to illustrate the approach and highlight the key steps of this methodology. Finally, Section 5 summarizes the paper and outlines possible avenues for future research.

2. Related Work

A software process is a set of activities that leads to the production of a software product [22]. Process improvement is based on the assumption that the quality of the engineering process is critical to product quality. Normally, dramatic changes to existing process are not a viable solution; it can only generate turbulence within the organization. Therefore, sometimes a quiescent, non-invasive, and adoption-centric approach to process improvement is needed [11]. To identify software process problems areas and bottlenecks, objective assessment of the process and quantifying qualitative data is need. Software metrics, specially looking at configuration management repository data can provide valuable feedback of the system.

There have been many studies conducted on software metrics. Metrics from repositories of configuration management are able to provide developers and analysts with large quantities data regarding developers’ work habits that might help analysts deduce which modules provide the most amount of effort. Studies have been conducted that utilize metrics for effort estimation in several capacities. More often than not, the metrics that are acquired from previous efforts made to the system can help in estimating how much effort is required for similar changes in a related feature, module of similar size and nature, or even the same module that requires more work [9][12][15]. When one requires information about how to proceed into the future, it is useful to look to the past for guidance. In addition to metrics, the requirements documents of a project are also treasure troves of information regarding logical and natural design that occurs from requested features. The interaction between certain requirements provides an understanding of how to organize the large features and modules of the system [15].

There have been many doubts regarding the necessity and the actual benefits of implementing a process improvement strategy in a company [5]. Studies have been performed on a wide array of projects, both successful and failures, in order to determine the effectiveness of implementing process improvement schemes such as Capability Maturity Model Integration (CMMI) [4]. It is shown that among the collection of reports studied, on average success rates of improved productivity, higher returns on investment (ROI), improved cycle times, and improved schedule fidelity. This suggests that although cumbersome in appearance, programs such as CMM maturity can aid in all of a development team’s efforts. An earlier study mentions the possibility of combining the use of CMM, GQM, and Six Sigma in order to locate bottlenecks [24]. Such a concept, which was very similar in scope to another paper by Wilson [26], was intended to combine the similar nature of two supposedly competing process improvement schemes into a synergistic approach to removing bottlenecks during development.

To identifying development process bottleneck an problem areas, a set of metrics of the process need to be collected. Basili *et al* introduced the Goal Question Metric (GQM) approach for collecting data with a purpose [3]. By spending the proper amount of time on defining the business goals, questions can then be derived from these goals. Finally, the appropriate metrics can be collected. The three levels that GQM address correlates with the three parts of the approach are the conceptual, operational, and quantitative. These three levels relate accordingly to the goal, question, and metrics that can be obtained for improving new or existing processes.

The Goal Question Indicator Metric Approach (GQ(I)M) is an extension of the GQM approach to measuring and improving a system [19]. The indicators can be anything that provides a gauge for the metrics. Even with detailed understanding of a system, it is often difficult for an analyst to understand the significance of certain gathered metrics without having to compare them to previously recorded data [7][16]. Indicators can be composed of multiple metrics that provide a subjective understanding of the status of a designated area of the system.

Six Sigma is a disciplined and data-driven approach and metholody for eliminating defects in process. It was originally developed at Motorola in the 1980s in an effort to reduce the number of defects in the final products and is still providing benefits [1]. Six Sigma focuses on continuously improving the products and the processes used to develop these products as a means to provide higher levels of customer satisfaction, reduce costs, and increase profits. The fundamental theory as well as namesake of the methodology is derived from the statistical idea of having only 3.4 or fewer defect parts per million (ppm) [21]. DMADV (Define, Measure, Analyze, Design, Verfy) [21] is a statistical process control method for Six Sigma.

A large set of metrics without direction or proper prioritization is hard to manage. Saaty developed the Analytic Hierarchy Process (AHP) [18], which is a multi-tiered decision-making process that effectively utilizes both qualitative and quantitative information to aid decision makers make a mathematically based choice dependant on the weights that are chosen for the attributes in consideration. The rankings generated are the subjective rankings given to these attributes by the analyst. This allows for some level of personal experience and preference to enter the decision-making process. From experience or previous studies, it is shown that there is a quantitative preference in these attributes; this can also be reflected as well [17]. The final rankings are quantitative weights of the benefit of choosing each alternative through analyzing both qualitative and quantitative data. In addition, these benefits can be further calculated with the cost of each alternative. Thus, a cost-benefit analysis can be performed alongside the AHP benefits decision-making approach.

3. VITA Methodology

The **V**isualization of **I**mpac**T** **A**nalysis (VITA) methodology incorporates the theories from AHP, GQ(I)M, and DMADV/DFSS in order to analyze and visualize configuration management repository data. Analysts first determine the goals and questions that are to be asked of the system and the data. Next, these goals are classified as alternatives and are prioritized according AHP. From this step, an analyst is able to derive the most pressing goals and focus attention on gathering the metrics necessary to answer these questions. These metrics combined form indicators that will be used as gauges throughout the rest of the process. Thresholds will be determined in order to see if the indicators chosen have remained static or have breeched the threshold. The timeframe can be arbitrarily set to any range depending on the approach that is taken or at the analyst's discretion. The impacts of change of requests and problem reports are visualized in static and dynamic graphs. The entire process can be performed semi-automatically and indefinitely in a recursive fashion or until focus is shifted to another goal. The following sections detailes the steps taken by VITA methodology.

3.1 Goals, Questions and Metrics

GQM is used to guide the creation of different types of metrics that could be used to measure the current status of the development process. The goals are determined by two parties. The first is that of the data provider that acts as the customer in this situation. The customer can make suggestions as to their intentions as well as areas of focus. The second source of goals is derived from the researcher. The researcher should choose a set of goals which will satisfy their research [10]. After both parties have defined their goals, priorities should be assigned to these goals. In this particular project, there are several goals in mind. The first is that of determining the grouping of change requests in an effort to locate virtual clustering of files. The second is to identify and improve developer effort distribution. These two goals can be mapped into a set of questions that map to metrics. Samples of the mapping of the questions to be answered and metircs related to these questions are listed in the following tables. These metrics are generated by using GQM and AHP methodologies mentioned in Section 2.

3.1.1 Developer Effort Distribution Focused

One of the indicators for managers to make informed decisions about the software development process is to have the right people with the right skills in the right place. Managers must question whether developers are working on the tasks they are good at and if they work efficiently on particular tasks. Table 1 is a sample of the external metrics that can be used to answer developer effort-related questions.

Table 1: Are the Developers Distributed Efficiently?
(CR: Change Request; PR: Problem Report**)**

01. Time spent per file
02. Time spent per branch
03. Number of developers per file.
04. Number of developers per branch
05. Number of developers per PR
06. Number of developers per CR
07. Time spent per PR
08. Time spent per CR
09. Time spent for each phase
10. Time spent per file per phase
13. Time spent per developer per file.
14. Average time spent per file per branch
15. Average number of file versions
16. Top X amount of file versions
27. Actions that take the most time.
28. Actions associated with the highest version numbers.
29. Distribution of developers among a PR or CR.
30. Percentage of time spent fixing
31. Percentage of time spent development testing

3.1.2 Problem Report Focused

Table 2 lists some of the data from the problem report that can be used to answer the question “Do the components require reorganization?”

Table 2: Do the Components Require Reorganization?

01. Time spent per file in PR
02. Time spent per branch in PR
03. Number of developers per file in PR.
04. Number of developers per branch in PR
05. Number of developers per PR
07. Time spent per PR
18. Number of files per PR
29. Distribution of developers among a PR.
30. Percentage of time spent fixing
31. Percentage of time spent development testing
34. Branches with the most fix hours
35. Branches with the most PR

37. Number of files per branch
38. Branches with the most test hours

3.1.3 Change Request Focused

Table 3 lists some of the data that can be used to answer the question “Where are the clusters related to the change reports and the problem reports?”

Table 3: Where are the Clusters Related to CRs and PRs?

01. Time spent per file in CR
02. Time spent per branch in CR
03. Number of developers per file in CR.
04. Number of developers per branch in CR
06. Number of developers per CR
08. Time spent per CR
17. Number of files per
29. Distribution of developers among a CR.
31. Percentage of time spent development testing
36. Branches with the most CR
39. Time spent in Requirements Phase
40. Number of branches per CR

3.2 Threshold

The threshold is the maximum imposed level a given metric can reach before passing into a warning level that warrants further investigation. Usually a threshold does not exist unless a governing body such as the managerial staff of a project or a development team specifies it. In the initial stages, there is not a set of threshold gauges to compare to all measurements in each smaller iteration or phase. Therefore, the initial thresholds can be obtained by deriving the average and median from the whole of the data set. However, like any statistical analysis, there will be outliers. Hopefully, the difference between the average and median will be small enough to take the halfway point between the two numbers.

4. Case study

The previous section outlined an integrated approach (VITA) that uses configuration management repository data to identify software development problems areas and bottlenecks. This section illustrates this approach by using real-world data from a large industrial partner’s non-proprietary data.

The case study is based on our collaboration with a large multinational corporation. The data includes a four-year record of configuration management and task tracking data that covering several major projects, and involved hundreds of developers. All the departments represented in the data are at CMMI level 3 or higher.

The VITA method takes three-step approach [23]: gathering data from repository, analyzing and integrating data from different sources (i.e., knowledge management), and visualizing clustering information according to end-users’ selection.

4.1 Data Gathering

In our work, we use metrics from the two most common tools: bug/task tracking, and configuration management. Specifically, the data we used in the case study are from ClearQuest and ClearCase. In the future study, we are planning to expend the sources of information to include requirements, source code and testing.

A collection of non-proprietary repository data from a large industrial partner was used for this research. There are two sets of data that were acquired for this case study. The first set, denoted by *Repository Activity Table*, is a collection of day-to-day concurrency versioning system data that was collected as a developer checks out a file as well as relating it to the appropriate work order. In this case study, we analyzed total 275,000 entries among 10 tables. Each entry includes the following data that are related to the development:

- Date & time
- Developer ID
- Action taken
- File path
- File name
- Branch
- Work order ID
- Branch version number

The second set of data, denoted by *Working Order Table*, is comprised of the following information:

- Working Order ID (unique ID to each work order, and automatically generated by the system)
- Working order type: Problem Report (PR) or Change Request (CR)
- Date submitted
- Date resolved
- Phase found
- Fix hours
- Development test hours

4.2 Knowledge Management

The information gathered from the first step was then parsed into a database with the logical organization of the pieces of the data in order to provide meaning to each row of data. A random section of data was selected for analysis. The approach for analyzing these data is as follows:

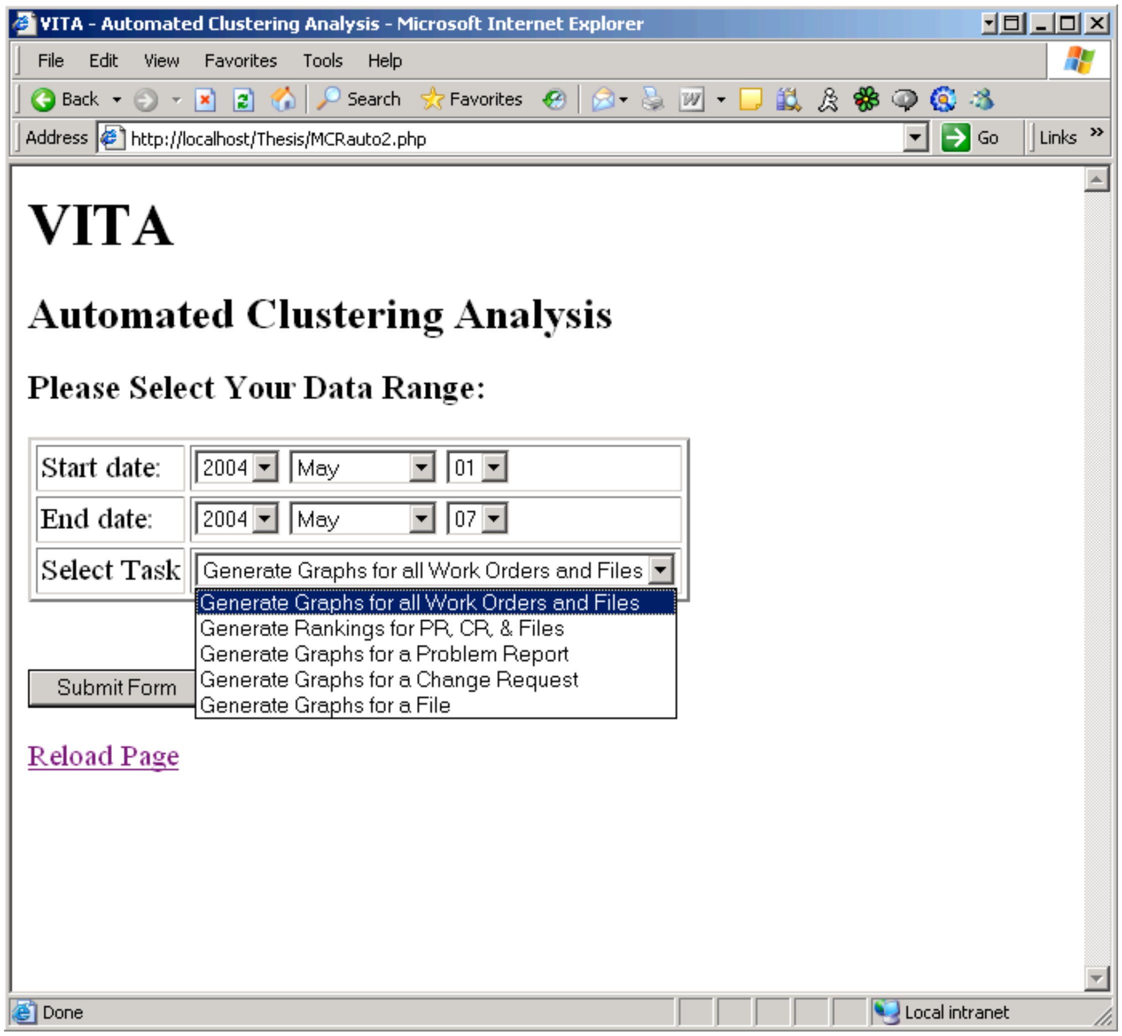
1. Select a timeframe of interests by end-user.
2. Perform clustering analysis on Change Requests (CRs), Problem Reports (PRs), and files that are affected by CRs and PRs.
3. Determine which files had the highest Problem Reports to Change Requests ratio.
4. Apply table of attributes (metrics) analysis to this section of data in order to determine the most problematic files and Change Request areas.
5. Compare these results with subsequent sections of time in the data.

As an illustrative example of this approach, this paper uses the first weeks of data provided by industrial partner to show the feasibility of the VITA methodology. The earliest two weeks were taken into analysis so that any hypotheses developed for these two weeks can then be applied to following two weeks sections of time in the data. The size of the organization was taken into consideration from which this information was derived, therefore, two weeks represents a meaningful section of time in which requests can be made as well as significant bugs have been remedied for large pieces of the code.

4.3 Information Visualization

The last part of the methodology is information visualization. Information visualization often deals with data that are normally large, semi-structured, or multivariate. Software and information visualization techniques have proven to be effective ways to improve comprehension of the system underdeveloped. Much work has been done in mining software repository data [25] [14]. The visualization can display and interactively explore data that might be abstract and may not be intuitively comprehendible [6]. To make the VITA methodology easy to use by end-user and to show the results of clusters in an intuitive way, a configuration management repository visualization tool called VITA Toolkit has been developed.

Users could select of the development artifacts that interest them. Figure 1 shows the VITA interface that provides five choices for user to select.


VITA - Automated Clustering Analysis - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/Thesis/MCRauto2.php Go Links
Submit Form
Reload Page
Done
Local intranet

VITA

Automated Clustering Analysis

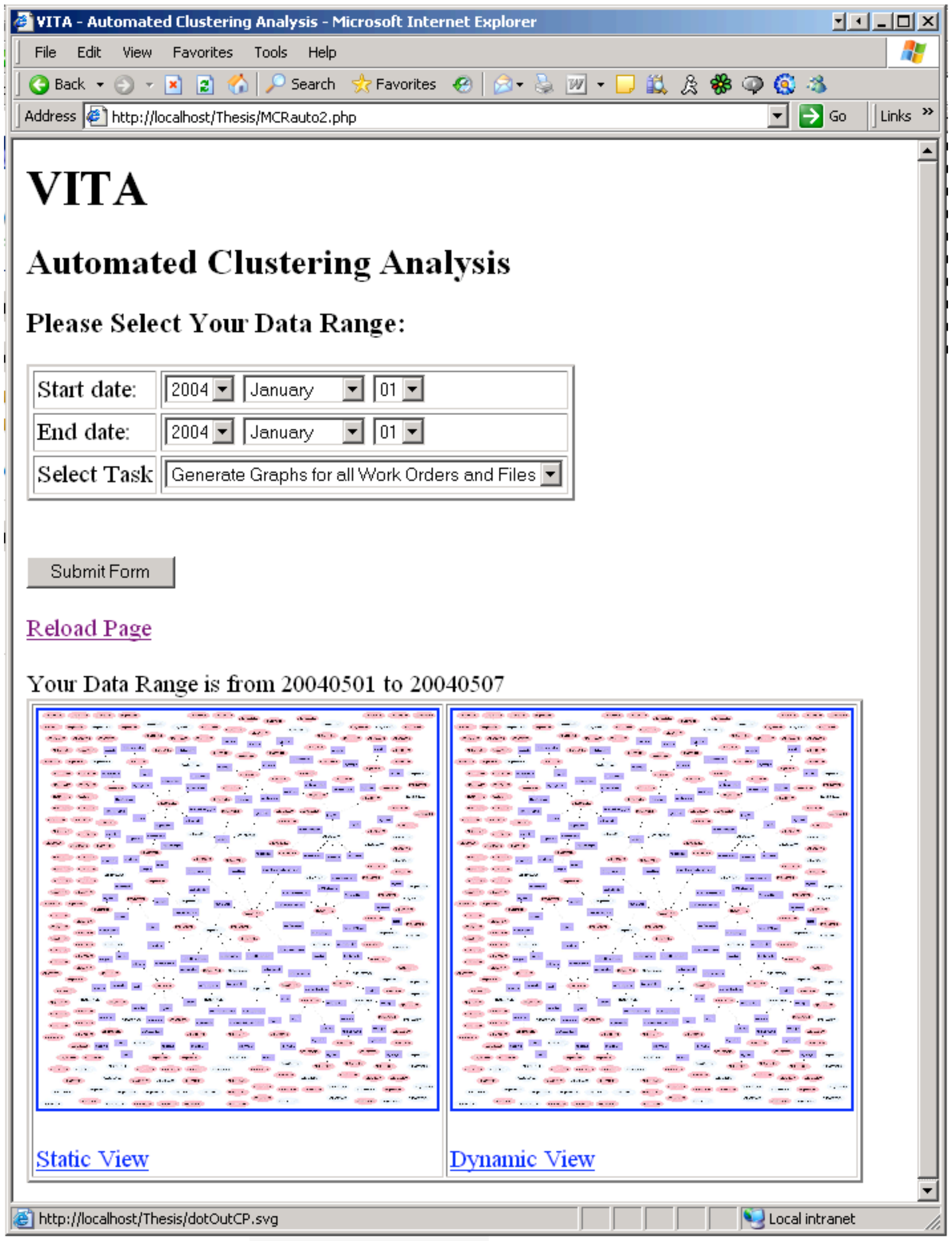
Please Select Your Data Range:

Start date:	2004 May 01
End date:	2004 May 07
Select Task	Generate Graphs for all Work Orders and Files

Generate Graphs for all Work Orders and Files
Generate Rankings for PR, CR, & Files
Generate Graphs for a Problem Report
Generate Graphs for a Change Request
Generate Graphs for a File

Figure 1: VITA Clustering Analysis Toolkit User Interface

From the preliminary prototype, a user can select timeframe that they are interested and two types of graphic views, static view in JPEG, PNG, or GIF and dynamic view in SVG format. Examples of these two types of graphical view from the user’s input in Figure 1 are shown in Figure 2.



VITA - Automated Clustering Analysis - Microsoft Internet Explorer

Address http://localhost/Thesis/MCRauto2.php

VITA

Automated Clustering Analysis

Please Select Your Data Range:

Start date:	2004	January	01
End date:	2004	January	01
Select Task	Generate Graphs for all Work Orders and Files		

Submit Form

Reload Page

Your Data Range is from 20040501 to 20040507

Static View

Dynamic View

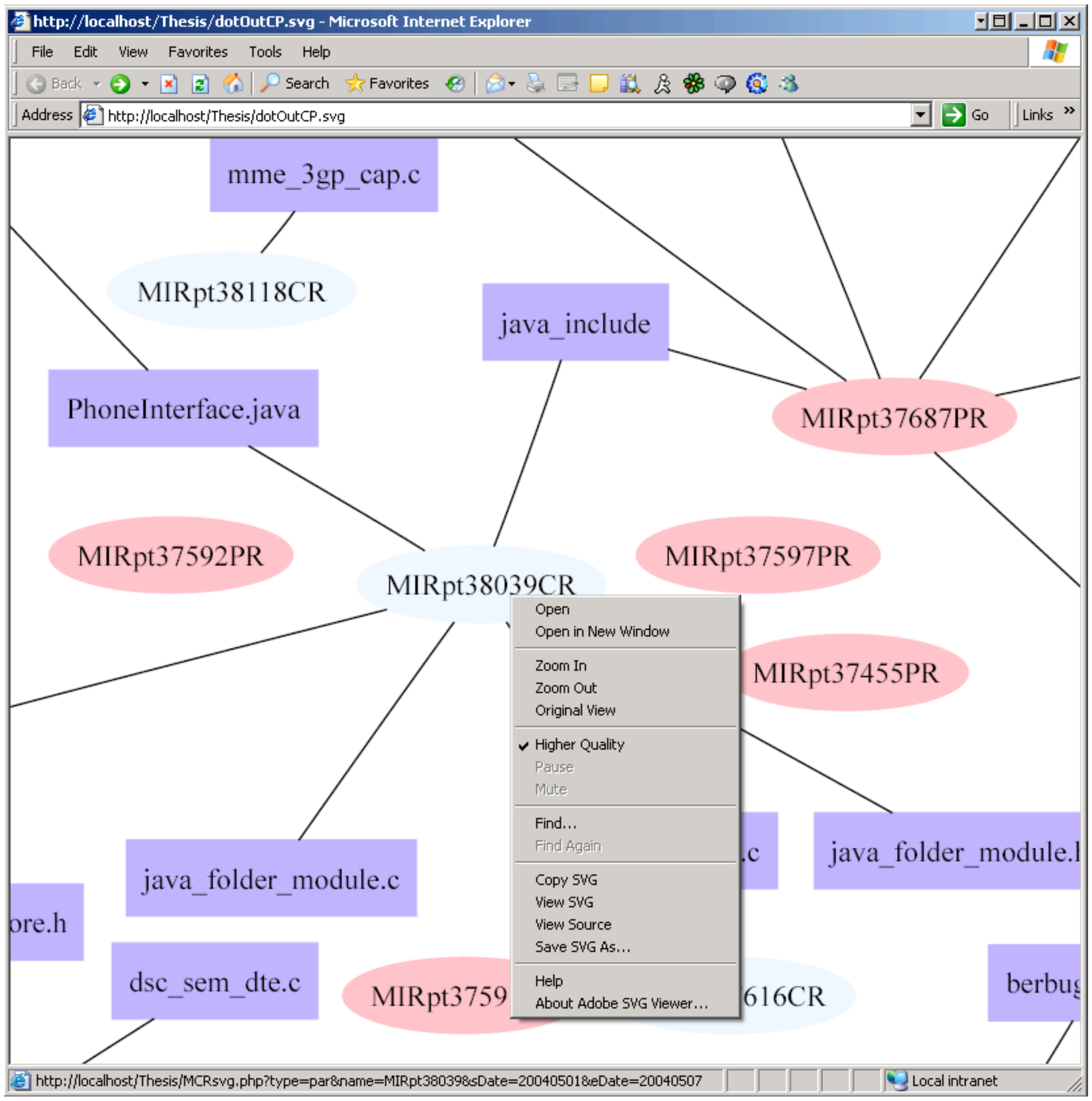
http://localhost/Thesis/dotOutCP.svg

Local intranet

Figure 2: Sample Results of VITA Query

4.3.1 Sample Clustering Visualization

By using the VITA Toolkit, the first two weeks' data from the industrial partner was analyzed and visualized. The first week of this two-week time chunk showed that there are many files that gravitate towards each other due to particular change requests and problem reports. The diagram in Figure 3 shows the excerpts generated by VITA Toolkit from the total clustering of the total of two weeks (1st and 2nd week) of analysis. Color codes are used to distinguish different artifacts in the analysis. For example, pink ovals represent problem reports (CRs), light blue ovals represent change requests (PRs), and slate blue boxes represent files. The edges that connect the boxes to the ovals represent a relationship between a change request and a file or a problem report and a file.



http://localhost/Thesis/dotOutCP.svg

http://localhost/Thesis/MCRsvg.php?type=par&name=MIRpt38039&sDate=20040501&eDate=20040507

Figure 3: Two Weeks of Clustering Analysis (Partial Graph)

4.3.2 Result Analysis

Preliminary results from these queries of two weeks of data suggest that there are certain clustering of files that gravitate to each other due to a change request. It is likely these files show a natural affinity due to the design. Certain change requests and problem reports had multiple files associated with them. In addition, the reverse was true where certain files saw multiple change requests and problem reports during the specified timeframe. In addition to these, there are many files that exist without the attachment to any change requests or problem reports as well as change requests and problem reports that do not have any nodes connected to them. This could be due to the fact that the change request or problem report that required the work of these specific files exists outside of the desired timeframe. Another possibility is that a change request or a problem report was made during the timeframe but no action was taken yet.

By acquiring the knowledge of the files associated with a particular change request or problem report, an analyst would be able to make assumptions at a high level and keep track of all other files that were manipulated together so that future work on a particular file might raise a flag for the developer so that considerations should be taken towards other files that were previously associated with the file in question. In addition, a file with the highest problem report to change request ratio could be deemed high risk and therefore warrant extra attention in the next stage of development. It should be noted that these diagrams can be utilized for analysis of evolved design and architecture as well as requirements and impact analysis.

5. Summary

This paper presented a methodology for software process improvement, specifically identifying problem areas and process bottlenecks by analyzing configuration management repository. As the first step in identifying the development bottleneck and problem areas, an impact analysis and visualization toolkit VITA of files changes due to change request and problem report was developed. A case study that applies this methodology by using real-world data from a large industrial partner illustrates the feasibility of the methodology.

Although this project does not involve analysis on the code level, it does provide meaningful input for further code level analysis because each artifact in the generated graphic view is clickable to the source related to it. Goal Question Indicator Metrics methodology combined with Six Sigma’s DMADV methodology can prove to be a rather successful combination – especially if priorities are set using the Analytic Hierarchy Process. The advantage of a statistically-based prioritization and decision-making process is that it allows managers and individuals in the position to make decisions regarding a project to possess increased credibility regarding their reasons for certain choices. The GQ(I)M approach places emphasis on the goals and questions that should be asked of the process improvement initiatives. In addition to these two solid theories is Six Sigma’s iterative process of reducing defects in both the product and process in order to improve overall quality. It is the hope of this research that bottlenecks in development can be alleviated through the use of the concepts and theories proposed.

Future work will include applying statistical methods, such as classification and regression tree algorithms, to the repository data. The future work will also include identifying and predicting continues variables or categorical variables for development process. For example, a reasonable next step is to find the files with the highest Problem Report to Change Request ratio that might suggest that these files are prone to errors and bugs and warrant consideration for redesign[20]. Subsequently, the other metrics associated with analyzing groupings will be applied in order to supply more meaningful information to the analyst. In addition, statistical analysis of the files, problem reports, and change requests should be made in order to help focus the study.

Acknowledgements

The author is grateful for Christopher Lo’s contributions to the project and early draft of the paper. The author is also grateful to anonymous referees for careful reading and comments.

References

[1] “Six Sigma Still Pays Off at Motorola” *BusinessWeek*, December 4, 2006.

[2] Barnard, J. Price, A. “Managing Code Inspection Information.” *IEEE Software*, 11(2), 59-69, 1994.

[3] Basili, V. R., Caldiera, G., Rombach, H. D. “The Goal Question Metric Approach.” *Encyclopedia of Software Engineering,* Wiley&Sons Inc., 1994.

[4] Capability Maturity Model Integration (CMMI), online at http://www.sei.cmu.edu/cmmi/index.html

[5] Galin, D., Avrahami, M. “Are CMM Program Investments Beneficial? Analyzing Past Studies.” *IEEE Software* November-December 2006. pp. 81-87

[6] Gansner, E., North, S. “An Open Graph Visualization System and Its Applications to Software Engineering.” *Software: Practice and Experience*. 1999.

[7] German, D.M. “Mining CVS repositories, the softChange experience” (2004). *1st International Workshop on Mining Software Repositories*. May 2004. Pages 17-21.

[8] Grady, B. “Practical Results from Measuring Software Quality.” *Communication of the ACM*, 36(11), 62-68, 1993.

[9] Graves, T.L., Mockus, A. “Inferring change effort from configuration management data.” *Metrics 98: Fifth International Symposium on Software Metrics*, pages 267-273, Bethesda, Maryland, November 1998.

[10] Hayes, J.H., Dekhtyar, A., Sundaram, S.K., Howard, S. “Helping analysts trace requirements: an objective look’” *Requirements Engineering Conference, 2004. Proceedings*. 12th IEEE International 2004. pp. 249 – 259

[11] Huang, S.; Tilley, S.; VanHilst, M.; Distante, D. “Adoption-Centric Software Maintenance Process Improvement via Information Integration.” *Proceedings of the 13th IEEE International Conference on Software Technology and Engineering Practice* (STEP 2005: September 24-25, 2005; Budapest, Hungary). Los Alamitos, CA: IEEE Computer Society Press, 2006

[12] Huffman Hayes, J., Patel, S., and Zhao, L. “A Metrics-Based Software Maintenance Effort Model.” *Proceedings of the 8th European Conference on Software Maintenance and Reengineering,* Tampere, Finland, March 2004. pp. 254-258. http://selab.netlab.uky.edu/Homepage/csmr_ameffmo_hayes_2004%5Eas_published.doc

[13] Kilpi, T. “Implementing a Software Metrics Program at Nokia.” *IEEE Software*, 18(6), 72-77, 2001.

[14] Kim, S., Zimmermann, T., Kim, M., Hasssan, A., Mochus, A. “TA-RE: An Exchange Language for Mining Software Repository.” *Proceedings of the International Workshop on Mining Software Repositories* (MSR 2006: May 22-23, 2006; Shanghai, China). Co-located with IEEE International Conference on Software Engineering (ICSE 2006).

[15] Lehman, M.M., Perry, D.E., and Ramil, J.F. “Implications of Evolution Metrics on Software Maintenance.” ICSM'98, November 1998. http://www.ece.utexas.edu/~perry/work/papers/feast2.pdf

[16] Mierle, K., Laven, K., Roweis, S., Wilson, G. “Mining Student CVS Repositories for Performance Indicators”. ACM Publications. *MSR 2005*.

[17] Morasca, S., Briand, L.C., Basili, V.R., Weyuker, E.J. and Zelkowitz, M.V. “Comments on ‘Towards a Framework for Software Measurement Validation’.” *IEEE Transactions on Software Engineering*, 23(3), March 1997, 187-188

[18] Mustafa, M.A., Al-Bahar, J.F. “Project risk assessment using the analytic hierarchy process.” *IEEE Transactions on Engineering Management*. Volume 38, Issue 1, Feb. 1991 Page(s):46 – 52

[19] Park, R. E., Goethert, W. B., Florac, W. A. *Goal-Driven Software Measurement – A Guidebook*. Software Engineering Institute of Carnegie Melon University. August 1996. http://www.sei.cmu.edu/publications/documents/96.reports/96.hb.002.html

[20] Sandusky, R.J., Gasser, L., Ripoche, G. “Bug Report Networks: Varieties, Strategies, and Impacts in a F/OSS Development Community.” 2004.

[21] Siviy, J. Six Sigma. Software Engineering Institute, 2001. http://www.sei.cmu.edu/str/descriptions/sigma6_body.html

[22] Sommerville, I. *Software Engineering* (8th Edition). Addison-Wesley, 2006.

[23] Tilley, S. *A Reverse-Engineering Environment Framework*. Software Engineering Institute, Carnegie Mellon University. Technical Report CMU/SEI-98-TR-005, 1998.

[24] VanHilst, M., Garg, P., Lo, C. Repository Mining and Six Sigma for process improvement. *Proceedings of the International Workshop on Mining Software Repositories*. 2005.

[25] Voinea, L., Telea, A. “An Open Framework for CVS Repository Querying, Analysis and Visualization.” *Proceedings of the International Workshop on Mining Software Repositories* (MSR 2006: May 22-23, 2006; Shanghai, China). Co-located with IEEE International Conference on Software Engineering (ICSE 2006).

[26] Wilson, D. “CMMI and Six Sigma Synergy.” Software Engineering Institute of Carnegie Melon University. 2005. http://www.sei.cmu.edu/cmmi/adoption/pdf/wilson-sixsigma.pdf