In Proceedings of the *19th International Conference on Software Engineering and Knowledge Engineering* (SEKE 2007: July 9-11, 2007; Boston, MA) Page 67-70

Analyzing Configuration Management Repository Data for Software Process Improvement

Shihong Huang
Computer Science & Engineering
Florida Atlantic University
shihong@cse.fau.edu

Christopher Lo
Computer Science & Engineering
Florida Atlantic University
clo1@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. Proper software measurement that derives meaningful numeric values for some attributes of a software product or process can help in identifying problem areas and development bottlenecks. Objective assessment is needed of the current status of software development so that informed project managers can make decisions. This paper presents a methodology called VITA for applying software 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 can be analyzed and visualized in a semi-automated manner according to user-selectable criteria. The approach is illustrated with a model problem concerning software process improvement of an embedded software system in the context of performing high-quality software maintenance.

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

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 [4], AT&T [1], and Nokia [9] 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 [13], 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 distributed 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.

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 change 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 [13]. 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 [7]. 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 [8][10]. Metrics from repositories of configuration management are able to provide developers and analysts with large quantities of data regarding developers' work habits that might help analysts deduce which modules require 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 and module of similar size [5][8][10]. 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 [10].

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 [2]. 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.

3. VITA METHODOLOGY

The **V**isualization of **I**mpac**T** **A**nalysis (VITA) methodology incorporates the theories from Anaylitic hierarchy Process (AHP) [11], 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 thusly 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 breached the threshold. The time frame 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-automatedly and indefinitely in a recursive fashion or until focus is shifted to another goal. The following sections details 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 to be 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 to be derived from the researcher. The researcher should choose a set of goals which will satisfy their research [6]. 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.

Based on the GQM methodology, three metrics tables were generated to help managers to make informed decisions about the software development process. The first table is Developer Effort Distribution Focused. This table could be used to answer question such as "Are the Developers distributed Efficiently?" It includes 19 entrances of matrics such as Time spent per file, Time spent per branch etc. The second table is Problem Report Focuses. It could be used to answer question such as "Do the Components Require Reorganization?". It includes 14 entrances of matrics such as Number of files per branch, Branches with the most test hours etc. The third table is Change Request Focused. It coul be used to answer question such as "Where are the Clusters Reated to CRs and PRs?". It includes 12 entrances of metrics such as Time spent per file in change request (CR), Time spent per branch in CR etc.

3.2 Threshold

The threshold is the maximum imposed level a given metric can achieve before passing into a warning level that warrants further investigation. A threshold does not usually exist naturally unless a governing body such as the managerial staff of a project or a development team specifies it. At 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 VITA method takes three-step approach [14]: 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

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 is collected as a developer checks out a file as well as relating it to the appropriate work order. Each working set entry contains information such as filename, working directory, branch directory, date, time, action taken, work order number, as well as developers.

The second set of data, denoted by *Working Order Table*, is comprised of a unique work order number, a work order type (problem report or change request), date submitted, date resolved, fix hours, and 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.

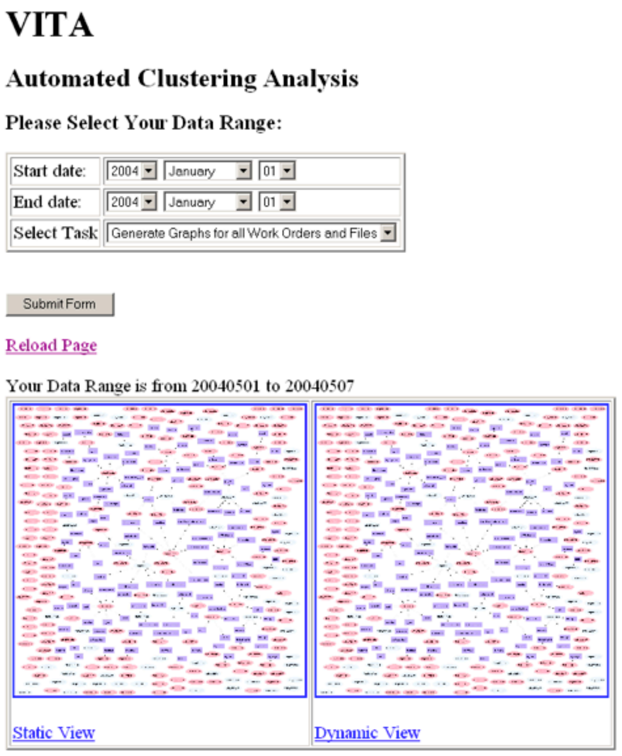
As an illustrative example of this approach, this paper uses the first two weeks of data provided by industrial partner to show the feasibility of the VITA methodology. The earliest two weeks were taken into analysis so that any hypotheses developed for these two weeks can then be applied to following two week sections of time in the data. It was taken into consideration the size of the organization 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. The visualization can display and interactively explore data that might be abstract and may not be intuitively comprehensible [3]. 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. 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 are shown in Figure 1.

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 2 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 ovals represent a relationship between a change request and a file or a problem report and a file.


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

Figure 1: VITA Clustering Analysis Toolkit

4.3.2 Result Analysis

Preliminary results from these two weeks of data suggest that there is 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 time frame. 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 time frame. Another possibility is that a change request or a problem report was made during the time frame 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 towards 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 ratios could be deemed high risk and therefore warrant extra attention in the next stage of development.

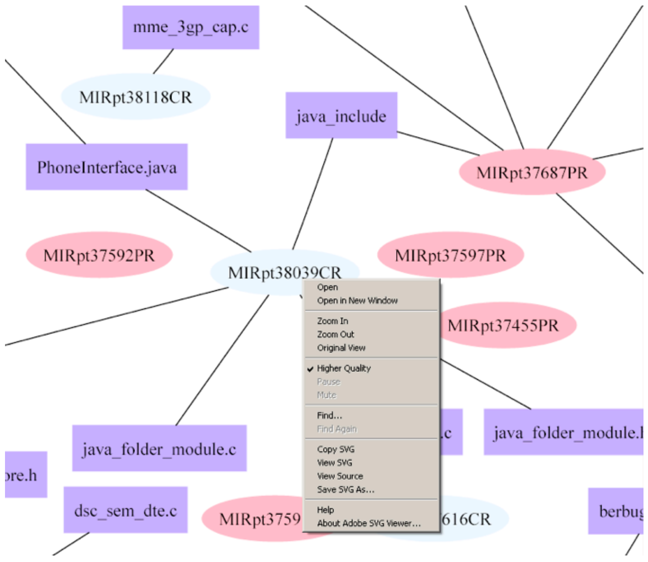


Figure 2: Clustering Analysis for Two Weeks Data (Partial Graph)

5. SUMMARY

This paper presented a methodology for software process improvement, specifically visualize the impact analysis 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 at 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 – specially 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.

Future work will include applying statistical methods, such as classification and regression tree algorithms (C&RT), to the repository data. Identifying and predicting continues variables or categorical variables for development process. For example, to identify the clusters of files that are likely to be changed together over the development process, 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 as to redesigning [12]. In addition, statistical analysis of the files, problem reports, and change requests should be made in order to help focus the study.

REFERENCES

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

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

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

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

[5] 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.

[6] 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

[7] 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

[8] 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

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

[10] 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

[11] 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

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

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

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