SEQUENTIAL STRATEGY FOR SOFTWARE PROCESS MEASUREMENT USING STATISTICAL PROCESS CONTROL

MUHAMMAD ABUBAKAR ALHASSAN

A dissertation submitted in partial fulfillment of the
requirements for the award of the degree of
Master of Science (Computer Science)

Faculty of Computing
Universiti Teknologi Malaysia

JANUARY 2014

I strongly dedicated this dissertation to my beloved parents for their prayers and to my governor Dr. Rabi’u Musa Kwankwaso for sponsoring my study.

ACKNOWLEDGEMENT

First of all, I would like to express my utmost gratitude to Allah S.W.T for His endless blessings and guidance throughout my entire research process and stay in Malaysia, Alhamdlillah for everything. Then, sincere appreciation goes to my supervisor **Associate Professor, Dr. Dayang Norhayati Abang Jawawi** for her continued support, guidance, and patience throughout my research. Despite her tight schedule, she always tried to make herself available. I've never seen anyone as committed in nurturing their students like she is. I will always look up to her as my academic role model.

Also, my endless gratitude goes to my parents for their strong support and prayers. I would also like to express my sincere gratitude to my governor, Dr. Rabi'u Musa Kwankwaso, for always providing me with sufficient financial support, may Allah (S.W.T) rewards him abundantly. I will forever be grateful to my family, for their undulating support, encouragement and prayers.

ABSTRACT

Software development process (SDP) and Software products are like two sides of a coin. We cannot achieve one without another. Today, in our software industries, monitoring software process is very challenging. Many problems of software process monitoring are hampering the quality of our software products. Several researchers in this area contributed their quota on addressing process monitoring issues using quantitative techniques. In this study, we address the problem of detecting software process deviations as a result of variations, investigating the causes of variations in software process, and the problem of process measurement. In addition, the study focus on code peer review process (CPRP). The first two problems can be addressed using one of the powerful quantitative techniques known as statistical process control (SPC). Also, control charts would be used in this study as it has been proved to be one of the suitable tools of SPC in monitoring process issues. As we know, the more defects we found during SDP, the less quality of the software product. Therefore, this study considers defect density as the metric to be use due to its significance in determining product quality. In order to have good analysis, this study conduct a case study on both Capability Maturity Model (CMM), lower and higher maturity levels software industries. On the other hand, to handle the problem of process measurement, a Sequential Strategy for Process Measure (SSPM) is proposed. This strategy is evaluated by Instrument for Evaluating Software Measurement Repository (IESMR) and Normative Information Model-based System Analysis and Design (NIMSAD) framework. Based on its evaluation, the strategy is similar to IESMR but differ in selecting measures, therefore it can be use for process measurement.

ABSTRAK

Proses pembangunan perisian (SDP) dan produk perisian adalah seperti dua belah duit syiling. Kita tidak boleh mencapai satu tanpa yang lain. Kini, dalam industri perisian, pemantauan proses perisian adalah sangat mencabar. Banyak masalah pemantauan perisian proses yang menghalang kualiti produk perisian. Beberapa penyelidik dalam bidang ini menyumbang kuota mereka untuk menangani isu-isu pemantauan proses dengan menggunakan teknik kuantitatif. Dalam kajian ini, kami menangani masalah ketidakstabilan proses akibat daripada variasi, menyiasat punca variasi pada proses, dan masalah pengukuran proses. Di samping itu, tumpuan kajian adalah tertumpu kepada proses kajian kod rakan sebaya (CPRP). Dua masalah pertama boleh diatasi dengan menggunakan salah satu teknik kuantitatif yang berkuasa dikenali sebagai kawalan proses statistik (SPC). Juga, carta kawalan akan digunakan dalam kajian ini kerana ia telah terbukti menjadi salah satu alat SPC yang sesuai dalam memantau isu-isu proses. Seperti yang kita tahu, lebih banyak kecacatan didapati dalam SDP, kualiti produk perisian menjadi berkurangan. Oleh itu, kajian ini menganggap ketumpatan kecacatan sebagai metrik yang digunakan kerana kepentingannya dalam menentukan kualiti produk. Dalam usaha untuk mempunyai analisis yang baik, kajian ini menjalankan satu kajian kes di kedua-dua Model Kematangan Keupayaan (CMM), yang lebih rendah dan lebih tinggi tahap kematangan industri perisian. Sebaliknya, untuk mengendalikan masalah pengukuran proses, Strategi Jujukan Proses Langkah (SSPM) dicadangkan. Strategi ini dinilai dengan Instrumen untuk Menilai Repositori Pengukuran Perisian (IESMR) dan rangka kerja Analisis dan Reka bentuk sistem berasaskan Model Maklumat Normatif (NIMSAD). Berdasarkan penilaiannya, strategi ini adalah sama dengan IESMR tetapi berbeza dalam memilih langkah-langkah, oleh itu ia boleh digunakan untuk pengukuran proses.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

A^2 QPM	Assessment Approach for Quantitative Process Management
c-Chart	Constant Chart
CL	Control or Centre Line
$CL\ \bar{x}$	Centre Line for Xbar chart
CMM	Capability Maturity Model
CPRP	Code Peer Review Process
DBMS	Database Management System
DOGO	Defining Organizational Goals
ERP	Enterprise Resource Planning
HLD	Half Logistic Distribution
HSE	Hitachi Software Engineering
IESMR	Instrument for Evaluating Software Measurement Repository
IMSP	Integrating Measures with Software Process
IPI	Investigating Process Issues
ISBSG	International Software Benchmarking Standards Group
ISTQB	International Software Testing Qualifications board
LCL	lower control Limit
$LNPL\ \bar{x}$	Lower Natural Process Limit for Xbar chart
MMLE	Modified Maximum Likelihood Estimation
NHPP	Non-Homogeneous Poisson Process
NIMSAD	Normative Information Model-based System Analysis and Design
p-Chart	Percentage Chart
SDLC	Software Development Life Cycle
SDM	Selecting and Defining Measures
SDP	Software Development Process

SLOC	Source Lines of Code
SSPM	Sequential Strategy for Process Measurement
u-Chart	Unit chart
UCL	Upper Control Limit
UNPL $\bar{x}$	Upper Natural Process Limit for Xbar chart
XmR-Chart	Individual Moving Range Chart

CHAPTER 1

INTRODUCTION

1.1 Overview

Software development life cycle (SDLC) can be simply defined as the sequence of stages or phases of developing software. These phases are arranged in cycle process in which the output of one phase is input to another in a cyclic manner. Similarly, there are many standard SDLC models that are used by software developers or engineers for developing software systems (Fuggetta, 2000). Waterfall model, spiral model, V-shape model are few examples of SDLC models. In addition to these models, new models of SDLC such as iterative evolutionary and agile exist in order to improve software quality products.

However, these models of software development describe the software development process in terms of requirement analysis, design, coding and testing phases of SDLC. Each model has its strengths as well as weaknesses. Therefore, it depends largely on the software organizations to select or choose the model that is suitable for them to develop the appropriate and quality software product. But, producing quality software is very challenging. In other words, it is not a very simple activity.

Moreover, in order to develop good quality software product, we need to investigate the process of developing the software (Fuggetta, 2000). This is because; software process and software products are like two sides of a coin. We cannot achieve quality software product without quality software process. Therefore, there is a need to monitor and improve software development process. Based on the studies of (Florac and Carleton, 1999), variations may be present during the activities of software development process. This is because software process may have one or more inputs as well as outputs, and these outputs have measurable entities or attributes (Humphrey, 1989).

In the field of software engineering, we cannot control what is not measured. As a result of this, many statistical techniques such statistical process control (SPC) plays a very important role in managing and controlling these attributes. In other words, control charts of SPC can help us to determine whether a process is under control or not by calculating the control limits so as to visualize the process behaviour over time. As a result of this, many researchers shared their experience on implementing this quantitative technique within software domain.

Even though SPC is used in manufacturing process; for example, (Mahesh and Prabhuswamy, 2010) used SPC to reduce process variability in manufacturing process. But, according to (Shewhart, 1930), SPC can be used in many other fields. Since that time until today, there is increase interest by many researchers in using SPC within software domain in order to improve software development process. Recently, a study on the use of control chart to improve software development process was conducted by (Pandain *et al.*, 2013). Also, (SrinivasaRao *et al.*, 2012) conducted a study on assessing software reliability using SPC. However, SPC has proven to be effective statistical method in not only software engineering area but also, in many areas such as engineering and medicine.

However, this study focus on code peer review process (CPRP). This is because; the process is one of the backbone software processes that play a vital role in ensuring good quality software products. Stabilizing the CPRP to behave consistently would enhance the quality of the software product. Also, the selected metric is relevant to the process because they are all geared towards achieving quality products.

1.2 Problem Background

As the use of software in our daily life is increasing day by day, the problem of software quality is also increasing. According to (Dupuis, 2004), the Software Engineering Body of Knowledge guide emphasizes that software quality is one of the challenging issues in the field of software engineering. That is to say, developing good quality software products that will meet business goals is very challenging. Therefore, to achieve software quality, the process used to develop or produce the software products should be considered (Olson *et al.*, 1989). This implies that, the quality of software depends largely on the quality of the process used to develop the software. In line with this, effective monitoring and controlling software development process is one of the successful paths for producing quality software products.

However, monitoring and controlling software process is not a very simple activity. Today, many researchers are working on software process improvements. Recently, (Pandain *et al.*, 2013), used control charts of SPC to improve software process performance. During their study, they conducted a case study on Capability Maturity Model (CMM) level 4 software industry in which they investigate the industry's software process behaviour. Similarly, (Satya Prasad *et al.*, 2011), proposed control mechanism (SPC) based on time between failures observations using Half Logistic Distribution (HLD) and Modified Maximum Likelihood Estimation (MMLE) to asses' software reliability. They used SPC in inspection and

testing process. Also, they used HLD and MMLE to predict time between failures and SPC as a method to control the predicted time of the process so as to produce reliable quality software within budget and specified time. But, as we know, a software process for one project may not be appropriate to another project. These authors did not clearly specify the project that is suitable to use MMLE together with SPC.

Similarly, additional effort was done by these researchers; (Nguyen *et al.*, 2012). In their study, they used SPC technique to detect software performance regression. As we know, performance regression simply means that a new version of software has worse performance than the previous version. To address this issue, the authors proposed an approach to analyze performance counts across test runs using control charts of SPC. Also, they used SPC in test and inspection process in which their result shows that control charts can be used to identify performance regressions in software systems.

However, these authors; (Baldassarre *et al.*, 2009) study the use of SPC for software as systematic approach. They used SPC in specification and inspection process. Also, they set many issues of software process monitoring and addressed the issues using SPC. These authors suggested that software engineers can implement SPC during process monitoring. At the end of their studies, they concluded that SPC is a suitable statistical technique that can be used to address many problems of process monitoring. But, there are many issues related to software process monitoring such as the difficulty of identifying process deviations as well as their causes.

In the studies of (Talib *et al.*, 2010), they used different quantitative techniques such as SPC in different phases of software development life cycle. These authors did not consider the critical software processes that are suitable to use statistical techniques. In addition, (Caivano, 2005) studied the use of SPC for continues software process improvements. In his work, it is now necessary to

measure, and control software process for the purpose of finding process variations and eliminating them.

Moreover, (Tuan *et al.*, 2006) proposed a new model (ABC model) for improving software development process. As we know, defects can be found during requirement analysis or design phase of SDLC and sometimes during testing, one of the major strengths of this important model is defects prevention. But, this model is focused on process prediction. We can only predict what is under control. More work can be done on monitoring software development process for good quality product.

However, based on the studies of (Baldassarre *et al.*, 2009), software process monitoring is still a challenging issue. Additional effort is required on monitoring and controlling software process deviations and their causes. But, software process monitoring is a complex activity and is still one of the issues that currently affects software quality product. Nevertheless, the utilization of SPC in software process monitoring is still an open issue. The problem of process measurement affects the successful implementations of SPC. Difficulty of selecting appropriate metric; its reliability as well as its selection for selected monitoring process characteristics. Also, the high intensive human nature in software process can have a great impact on SPC and monitoring effectiveness.

1.3 Problem Statement

Based on the work of (Baldassarre *et al.*, 2009), monitoring software process stability is still an open issue in the field of software engineering. There are some issues such as; process performance deviation that affect software process, by carefully monitoring these issues; we can improve the stability and capability of the process. Therefore, in this section, we describe some problems which we plan to address at the end of our study. These are as follows;

1. Problem of detecting software process deviations as a result of variations.
2. Problem of investigating the causes of variations in software process.
3. Problem of software process measurement for successful SPC implementations

Historically, SPC was used by many of researchers within software engineering domain and proved to be effective quantitative technique of process monitoring and control. For example, (Lantzy, 1992) and (Burr and Owen, 1996), demonstrate practical application of SPC in software setting in order to improve product quality. However, SPC can help us to address the above mentioned problems and evaluate whether the process is under control or not by using control charts. When the process is affected with either special or assignable causes of variations, control charts will play a vital role of identifying these causes through the use control limits. On the other hand, when these causes exceed control limits, the process is said to be unstable and the causes must be identify and eliminated. However, controlling this process will improve software development process and contribute for the production of good quality software products that will meet customer's satisfaction or requirements.

1.4 Research Aim

The main aim of this research is to justify and analyse the use of SPC in software process monitoring and control. Also, to propose a strategy that would support process measurement so as to achieve successful SPC implementations.

1.5 Objectives of the Study

In order to achieve the above mentioned aim, the objectives of this research are as follows:

1. To analyse the use of SPC in CMM lower maturity software industries for process monitoring and control.
2. To propose a strategy that would support software process measurements so as to ensure successful SPC implementations
3. To evaluate the proposed strategy using instrument for evaluating software measurement repository (IESMR) and NIMSAD framework.

1.6 Scope of the Study

In this research, we are going to see the effectiveness of using statistical process control to improve software development process based upon the use of the following:

- Control charts of SPC
- Code peer review process (CPRP)
- Defect density as the metric to be used
- A case study on CMM lower and higher maturity levels

1.7 Significance of the Study

The significance of this important research is to improve software development process, based upon the use of SPC technique in order to ensure quality software production for use in our industries or companies as well as our academic

environment. In other words, this research is very important in the sense that, it will support CMM standard model of software process improvement for use in the developing large and complex software systems.

Moreover, this study proposes a strategy that would play a vital role in process measurement activities through its sequence of stages. When this strategy is followed accordingly, we can achieve accurate process measurements for successful SPC implementations.

1.8 Dissertation organization

This research is made up of six chapters. In Chapter 1, we discussed about the research introduction, problem background, problem statement and objectives of the study. Similarly, Chapter 2 presents software development process, capability maturity model, statistical process control as well as the literature review of the study. In Chapter 3, we explained the research methodology in sequence of phases. Moreover, Chapter 4 presents the result we obtained by utilizing control chart of SPC (u-chart) in lower CMM maturity level (level 2) software industry.

Furthermore, in Chapter 5, we discussed about the software process investigation we carried out on CMM highest maturity level (optimization) using u-chart control charts of SPC. Also, the sequential strategy for process measurement is proposed and evaluated in chapter 5. However, Chapter 6 presents the study contributions and future work.

REFERENCES

Baldassarre *et al.* (2009). *Trustworthy Software Development Processes.* (11-23). Springer.

Barcellos *et al.* (2013). A strategy for preparing software organizations for statistical process control. *Journal of the Brazilian Computer Society.* 1-29.

Barcellos *et al.* (2010). Evaluating the suitability of a measurement repository for statistical process control. *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement.* 27.

Burnstein (2003).*Process Control and Optimization*, Springer.

Burr and Owen (1996).*Statistical methods for software quality: using metrics to control process and product quality*, Coriolis Group.

Caivano (2005). Continuous software process improvement through statistical process control. *Software Maintenance and Reengineering, 2005. CSMR 2005. Ninth European Conference on.* 288-293.

Card (1994). Statistical process control for software? *Software, IEEE.* 11 (3.), 95-97.

Carleton *et al.* (1999). Panel Discussion: Can Statistical Process Control Be Usefully Applied to Software? *The 11th Software Engineering Process Group (SEPG) Conference, Atlanta, Georgia.* 8-11.

Deming and Edwards (1982).*Quality, productivity, and competitive position*, Massachusetts Institute of Technology, Center for Advanced Engineering Study Cambridge, MA.

Dupuis (2004). Software Engineering Body of Knowledge.

Eickelmann and Anant (2003). Statistical process control: what you don't measure can hurt you! *Software, IEEE.* 20 (2.), 49-51.

Falbo and da Rocha (2012). Using a reference domain ontology for developing a software measurement strategy for high maturity organizations. *Enterprise*

Distributed Object Computing Conference Workshops (EDOCW), 2012 IEEE 16th International. 114-123.

Florac and Carleton (1999).*Measuring the software process: statistical process control for software process improvement*, Addison-Wesley Professional.

Florac *et al.* (2000). Statistical process control: analyzing space shuttle onboard software process. *Software, IEEE.* 17 (4.), 97-106.

Florak *et al.* (1997). Practical software measurement: Measuring for process management and improvement.

Fuggetta (2000). Software process: a roadmap. *Proceedings of the Conference on the Future of Software Engineering.* 25-34.

Humphrey (1988). Characterizing the software process: a maturity framework. *Software, IEEE.* 5 (2.), 73-79.

Humphrey (1989).*Managing the Software Process (Hardcover)*, Addison-Wesley Professional.

Jacob and Pillai (2003). Statistical process control to improve coding and code review. *Software, IEEE.* 20 (3.), 50-55.

Jayaratna (1994).*Understanding and evaluating methodologies: NIMSAD, a systematic framework*, McGraw-Hill, Inc.

Juran and Riley (1999).*The quality improvement process*, McGraw Hill New York, NY.

Kan (2002).*Metrics and models in software quality engineering*, Addison-Wesley Longman Publishing Co., Inc.

Komuro (2006). Experiences of applying SPC techniques to software development processes. *Proceedings of the 28th international conference on Software engineering.* 577-584.

Lantzy (1992). Application of statistical process control to the software process. *Proceedings of the ninth Washington Ada symposium on Ada: Empowering software users and developers.* 113-123.

Leau *et al.* (2012). Software Development Life Cycle AGILE vs Traditional Approaches. *2012 International Conference on Information and Network Technology (ICINT 2012) IPCSIT.*

Lynch and Cross (1992).*Measure up!: The essential guide to measuring business performance*, Mandarin.

Mahesh and Prabhuswamy (2010). Process variability reduction through statistical process control for quality improvement. *International Journal for Quality Research.* 4 (3.), 193-203.

Martin *et al.* (1996). Process performance monitoring using multivariate statistical process control. *Control Theory and Applications, IEE Proceedings-*. 132-144.

Nandyal (2004).*Cmmi*, Tata McGraw-Hill Education.

Nguyen *et al.* (2012). Automated detection of performance regressions using statistical process control techniques. *Proceedings of the third joint WOSP/SIPEW international conference on Performance Engineering.* 299-310.

Oakland (2008).*Statistical process control*, Routledge.

Olson *et al.* (1989). *Conducting SEI (Software Engineering Institute)-Assisted Software Process Assessments*, DTIC Document.

Pall (1987).*Quality Management*, Prentice Hall PTR.

Pandain *et al.* (2013). CONTROL CHARTS FOR IMPROVING THE PROCESS PERFORMANCE OF SOFTWARE DEVELOPMENT LIFE CYCLE. *International Journal of Research and Reviews in Applied Sciences.* 14 (2.),

Park (1992). *Software size measurement: A framework for counting source statements*, DTIC Document.

Paulk *et al.* (1993). Capability maturity model, version 1.1. *Software, IEEE.* 10 (4.), 18-27.

Paulk *et al.* (1995).*The capability maturity model: Guidelines for improving the software process*, Addison-wesley Reading.

Radice (1998). Statistical process control for software projects. *10th Software Engineering Process Group Conference. Chicago, Illinois.*

Rao *et al.* (2012). Monitoring Software Reliability using Statistical Process Control An Ordered Statistics Approach. *arXiv preprint arXiv:1205.6440.*

Ren *et al.* (2011). Research on software quality control method based on control chart. *Computing, Control and Industrial Engineering (CCIE), 2011 IEEE 2nd International Conference on.* 274-277.

Rifkin (2001). What makes measuring software so hard? *Software, IEEE.* 18 (3.), 41-45.

Sargut and Demirörs (2006). Utilization of statistical process control (SPC) in emergent software organizations: pitfalls and suggestions. *Software Quality Journal.* 14 (2.), 135-157.

Satya Prasad *et al.* (2011). Software Reliability Measuring using Modified Maximum Likelihood Estimation and SPC. *International Journal of Computer Applications.* 21 (7.), 1-5.

Senbergs and Misnevs (2006). Research of a Statistical Process Control in Software Engineering Project. *Transport and Telecommunication.* 7 (1.), 70-75.

Senge (1994).*The fifth discipline fieldbook,* Random House Digital, Inc.

Shewhart (1930). Economic Quality Control of Manufactured Product1. *Bell System Technical Journal.* 9 (2.), 364-389.

Spanos *et al.* (1992). Real-time statistical process control using tool data [semiconductor manufacturing]. *Semiconductor Manufacturing, IEEE Transactions on.* 5 (4.), 308-318.

SrinivasaRao *et al.* (2012). A Comparative Study of Assessing Software Reliability using SPC: An MMLE Approach. *International Journal of Computer Applications.* 50 (19.), 23-27.

Talib *et al.* (2010). Techniques for Quantitative Analysis of Software Quality throughout the SDLC: The SWEBOK Guide Coverage. *Software Engineering Research, Management and Applications (SERA), 2010 Eighth ACIS International Conference on.* 321-328.

Tarhan and Demirors (2012). Apply Quantitative Management Now. *Software, IEEE.* 29 (3.), 77-85.

Tuan *et al.* (2006). Using ABC Model for Software Process Improvement: A Balanced Perspective. *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on.* 229c-229c.

Weber *et al.* (1991). *Key practices of the capability maturity model,* DTIC Document.

Weller (2000). Practical applications of statistical process control [in software development projects]. *Software, IEEE.* 17 (3.), 48-55.

Wheeler and Chambers (1992). Understanding statistical process control.

Zhang and Kim (2010). Monitoring software quality evolution for defects. *Software, IEEE.* 27 (4.), 58-64.