AN ALGORITHMIC-BASED SOFTWARE CHANGE EFFORT PREDICTION MODEL USING CHANGE IMPACT ANALYSIS FOR SOFTWARE DEVELOPMENT

MUHAMMAD SUFYAN BIN BASRI

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Doctor of Philosophy (Software Engineering)

Advanced Informatics School
Universiti Teknologi Malaysia

JULY 2016

DEDICATION

ALHAMDULILLAH…

To Allah (SWT)

For my beloved mother and father

My dearest wife, Faizura Haneem Mohamed Ali

and children…

Luqman, Sakina, Hannah and Faheem

Thank you for the love and continuous support…

ACKNOWLEDGEMENT

Firstly, I would like to express my sincere gratitude and appreciation to my supervisors, Dr Mohd Nazri Kama and Dr Roslina Ibrahim, for the continuous support of my PhD study and related research, their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisors and mentors for my PhD study. I would like to thank them for encouraging my research and for allowing me to grow as a researcher.

Secondly, a special thanks to both my parents and my parents-in-law for all the sacrifices that you've made on my behalf. They have taught me the value of perseverance and always prayed for my success, in my study and my career. I would also like to thank my beloved wife and kids for their patience during the course of my PhD. Without my wife's support, I would not have a dearest companion who spent sleepless nights with and was always my support in the moments when there was no one to answer my queries.

A great gratitude goes to the Ministry of Education for sponsoring my three years PhD study. My sincere thanks also goes to all individuals and staffs of Advanced Informatics School, Universiti Teknologi Malaysia, Kuala Lumpur for the advice, critical and insightful feedback in improving my quality of work and presentation during my PhD study. Finally, my appreciation goes to those who have been involved directly or indirectly in this research.

ABSTRACT

Software changes are inevitable due to the dynamic nature of the software development project itself. Some software development projects practice their own customised methodology but mostly adopt two kinds of methodologies; Traditional and Agile. Traditional methodology emphasizes on detailed planning, comprehensive documentation and extensive design that resulted a low rate of changes acceptance. In contrast, Agile methodology gives high priority on accepting changes at any point of time throughout the development process as compared to the Traditional methodology. Among the primary factor that has direct impact on the effectiveness of the change acceptance decision is the accuracy of the change effort prediction. There are two current models that have been widely used to estimate change effort which are algorithmic and non-algorithmic models. The algorithmic model is known for its formal and structural way of estimation and best suited for Traditional methodology. While non-algorithmic model is widely adopted for Agile methodology of software projects due to its easiness and requiring less work in term of effort predictability. The main issue is that none of the existing change effort prediction models is proven to suits for both, Traditional and Agile methodology. Additionally, there is as yet no clear evidence of the most accurate change effort prediction model for software development phase. One of the method to overcome these challenges is the inclusion of change impact analysis in the estimation process. The aim of the research is to overcome the challenges of change effort prediction for software development phase: inconsistent states of software artifacts, repeatability using algorithmic approach and applicability for both Traditional and Agile methodologies. This research proposed an algorithmic change effort prediction model that used change impact analysis method to improve the accuracy of the effort estimation. The proposed model used a current selected change impact analysis method for software development phase which is the SDP-CIAF (Software Development Phase-Change Impact Analysis Framework). A software prototype was also developed to support the implementation of the model. The proposed model was evaluated through an extensive experimental validation using case scenarios of six real Traditional and Agile methodologies software projects. A comparative study was also conducted for further validation and verification of the proposed model. The analysis result showed an accuracy improvement of 13.44% average mean difference for change effort prediction over the current selected change effort prediction model. The evaluation results also confirmed the applicability for both Traditional and Agile methodologies.

ABSTRAK

Perubahan perisian tidak dapat dielakkan kerana sifat dinamik projek pembangunan perisian itu sendiri. Sesetengah projek pembangunan perisian mengamalkan metodologi mereka sendiri yang telah disesuaikan tetapi kebanyakannya mengamalkan dua jenis metodologi; Tradisional dan Agil. Metodologi Tradisional memberi penekanan kepada perancangan terperinci, dokumentasi menyeluruh dan reka bentuk yang terperinci yang menyebabkan kadar penerimaan perubahan yang rendah. Sebaliknya, metodologi Agil memberi keutamaan yang tinggi pada penerimaan perubahan pada bila-bila masa sepanjang proses pembangunan berbanding metodologi Tradisional. Antara faktor utama yang mempunyai kesan langsung kepada keberkesanan keputusan penerimaan perubahan adalah ketepatan anggaran usaha perubahan. Terdapat dua model semasa yang telah digunakan secara meluas untuk menganggarkan perubahan usaha iaitu model algoritma dan bukan algoritma. Model algoritma dikenali dengan cara anggaran formal dan berstruktur dan paling sesuai untuk metodologi Tradisional. Sementara model bukan algoritma diterima pakai secara meluas bagi projek perisian metodologi Agil kerana ia mudah dan memerlukan kerja yang sedikit dari sudut anggaran usaha. Isu utama adalah kerana tiada model anggaran usaha perubahan sedia ada yang terbukti sesuai untuk kedua-dua metodologi Tradisional dan Agil. Selain itu, tiada lagi bukti yang jelas berkenaan model anggaran usaha perubahan yang paling tepat untuk fasa pembangunan perisian. Salah satu kaedah untuk mengatasi cabaran-cabaran ini adalah dengan memasukkan analisis kesan perubahan di dalam proses anggaran. Tujuan kajian ini adalah untuk mengatasi cabaran anggaran usaha perubahan untuk fasa pembangunan perisian: keadaan artifak perisian yang tidak konsisten, kebolehulangan menggunakan pendekatan algoritma dan kebolehgunaan untuk kedua-dua metodologi Tradisional dan Agil. Kajian ini mencadangkan satu model anggaran usaha perubahan berasaskan algoritma yang menggunakan kaedah analisis kesan perubahan untuk meningkatkan ketepatan anggaran usaha. Model yang dicadangkan menggunakan kaedah analisis kesan perubahan semasa yang terpilih untuk fasa pembangunan perisian iaitu SDP- CIAF (Rangka Kerja Fasa Pembangunan Perisian - Analisis Impak Perubahan). Satu perisian prototaip juga telah dibangunkan untuk menyokong pelaksanaan model. Model yang dibangunkan dinilai melalui pengesahan eksperimen yang luas menggunakan kes senario daripada enam projek-projek perisian metodologi Tradisional dan Agil sebenar. Satu kajian perbandingan juga telah dijalankan untuk pengesahsahihan dan pengesahan lanjut model yang dicadangkan. Keputusan analisis menunjukkan peningkatan ketepatan sebanyak 13.44% perbezaan min purata bagi anggaran usaha perubahan berbanding model anggaran usaha perubahan semasa yang terpilih. Hasil penilaian juga mengesahkan kebolehgunaan dalam kedua-dua metodologi Tradisional dan Agil.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AUP	-	Agile Unified Process
CASE	-	Computer-aided Software Engineering
CEPM	-	Change Effort Prediction Model
CDF	-	Class Dependency Filtration
CIA	-	Change Impact Analysis
CIP	-	Class Interaction Prediction
CIP-IPF	-	Class Interaction Prediction – Impact Prediction Filter
CISE	-	Change Impact Size Estimation
COCOMO II	-	Constructive Cost Model II
DAG	-	Directed Acyclic Graph
FP	-	Function Point
FPA	-	Function Point Analysis
ICP	-	Impacted Class Purification
JAD	-	Joint Application Development
LOC	-	Line of Codes
MDA	-	Method Dependency Addition
MDF	-	Method Dependency Filtration
MMRE	-	Mean Magnitude Relative Error
MRE	-	Magnitude Relative Error
PRED	-	Percentage of Prediction
RAD	-	Rapid Application Development
RE	-	Relative Error
RUP	-	Rational Unified Process
SCM	-	Software Change Management
SDLC	-	Software Development Life Cycle
SDP-CIAF	-	Software Development Phase Change - Change Impact Analysis Framework

SLOC	-	Single Line of Codes
UCM	-	Use Case Maps
UCP	-	Use Case Point
UML	-	Unified Modelling Language
XP	-	Extreme Programming

LIST OF SYMBOLS

CIS_{IC}	-	Change impact size of the impact class
r_{IC}	-	Any related requirement that map to the impact class
R	-	Requirement's quantity that related to the impact class
AT	-	Affection type of the change impact, either direct or indirect
CT	-	Change type of the change request
DS	-	Development status of the project
PM	-	Original estimated effort using COCOMO II in man per month
IE	-	Initial effort prediction using COCOMO II. It is equivalent to PM in man per month
UE	-	Updated effort prediction after change implementation in man per month
CP	-	Change priority multiplier or change request priority
E	-	Relative effort value
S	-	Original size prediction of the code
CS	-	Predicted code size after change implementation
A	-	COCOMO II multiplicative constant
B	-	COCOMO II constant variables
SF	-	COCOMO II scale driver values
S_{IC}	-	Size of the impacted class *IC*
ND	-	Not developed class constant multiplier
TND	-	Quantity of the not developed affected classes
PD	-	Partially developed classes constant multiplier
TPD	-	Quantity of partially developed affected classes
FD	-	Fully developed classes constant multiplier
TFD	-	Quantity of fully developed affected classes
T_{IC}	-	Total quantity of the impacted classes
EM	-	Effort multiplier

TDEV	-	Calendar time in months
C	-	COCOMO II constant
SCED	-	COCOMO II scheduling factor
SE	-	COCOMO II schedule equation
D	-	COCOMO II constant
Size	-	COCOMO II software size (KSLOC)

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

Software process consists of several defined activities which separated into distinct stages during software development project in order to deliver a software product with better quality and management. This process is also known as Software Development Life Cycle (SDLC) which usually starts from planning, requirement gathering, analysis, design, implementation, testing and ends with deployment. Foundational to this, there are two types of SDLC methodology mostly adopted; Traditional methodology and Agile methodology. Traditional methodology practices emphasize on detailed planning, comprehensive documentation and extensive design (Awad, 2005). On the contrary, Agile methodology practices customer collaboration over detailed planning, emphasizes on the working software over the comprehensive documentation and values individual interactions over extensive processes and design (Beck *et al.*, 2001). Regardless of any methodology adopted, the software project management is required with the intent of better planning, monitoring and control for the software development efforts. Software development efforts planning or estimation in a software process is one of the important criteria to deliver a successful software development project (Lehtinen *et al.*, 2014).

Software development effort estimation is a process predicting how much work required to develop a software in a software project, and normally will be described in man-days or man-hours unit. Studies of software development effort estimation has started since 1960s (Farr and Zagorski, 1964; Nelson, 1967) and it has been a continuous research because there are still a lot of arguments and discussions

in achieving an accurate software effort estimation result (Bardsiri *et al.*, 2013; Lehtinen *et al.*, 2014). Therefore, researchers have proposed various types of software effort estimation techniques to date. However, most of the techniques were proposed to estimate the development work at the beginning of software development phase based on pre-defined requirements. However, during development phase, the requirement might change due to the dynamic nature of any software projects. These changes will give an impact to the software project management in controlling the software development effort. Therefore, effort estimation for the requirement changes is critical to software project management in providing the final deliverables of software project. At present, there is lack of evidence of the current effort estimation model especially for requirement changes during software development phase.

This thesis presents a new change effort prediction model that can be used in both Traditional and Agile methodologies software projects. The new model identifies and considers the related factors that contribute to the effort estimation for requirement changes during software development phase.

This chapter describes the background of the research, problem statement, research questions, objectives, scope of research, significance of the study and thesis organization.

1.1 Background of the Research

Although effort estimation has existed for decades, it still remains a great challenge for software project management to produce an accurate estimation and eventually completed the software project successfully. Several studies highlighted the importance of managing the changes in the software projects by the software project manager to ensure the project success (Agarwal and Rathod, 2006; Drew Procaccino *et al.*, 2002; Lehtinen *et al.*, 2014; Verner *et al.*, 2007). Lehtinen *et al.* (2014) defines a software project failure means a recognizable failure to succeed in the cost, schedule, scope, or quality goals of the project.

Kaur and Sengupta (2013) states that the most common reasons for project failure are rooted in the project management process itself which include identified estimation mistakes, unclear project goals and objectives, and project requirement changing during the project. In any software development project, the software project manager is the main role who is responsible towards the software project success or failure. One of the main criteria of a successful software project manager is, responsible in managing the software requirement changes and hence justifies the change acceptance decisions made.

Change request may occur at any point in SDLC (Chen and Chen, 2009; Nurmuliani *et al.*, 2006). It is important to manage the changes in the software to meet the evolving needs of the customer and hence, satisfy them (Bennett and Rajlich, 2000; Brooks Jr, 1956; Finkelsteiin and Kramer, 2000; Kotonya and Sommerville, 1998; Pfleeger and Bohner, 1990). Introducing software changes during software development phase may need to identify the impacts to the software artifacts and consequences to the efforts due to the software change. Accepting too many changes might lead to project cost overrun and delay. Rejecting too many changes may cause customer dissatisfaction.

While this is the case of Traditional methodology, where software project manager has the option to accept or reject the change request, it is the opposite in the case of modern SDLC such as Agile methodology. In view of the change request, Agile methodology gives high priority on accepting changes at any point of time throughout the software development process compared to the Traditional methodology (Beck *et al.*, 2001). Henceforth, an efficient software project management and change management in Agile methodology are more crucial, and accurate effort estimation are not the second option in ensuring a software project success.

Generally, two types of information that could assist the software project manager in managing the software change management are change impact analysis and change effort prediction (Stammel and Trifu, 2011). Change impact analysis is a procedure of identifying the possible effect of a change, or predicting the process required to undertake a change (Bennett and Rajlich, 2000; Brooks Jr, 1956;

Finkelsteiin and Kramer, 2000; Kotonya and Sommerville, 1998; Pfleeger and Bohner, 1990). Change effort estimation, on the other hand, is a procedure of predicting the processes and activities required in terms of work, resources and time in implementing the changes (Asl and Kama, 2013; Bee Bee, 2010; Chua and Verner, 2010).

Verner *et al.* (2007) highlighted it is important to software project manager to obtain enough information during estimation process in order to ensure the project success. In the context of requirement changes, the impacts to the software artifacts is one of the required information. Software artifacts include documents, data and source code or class are subjected to impact due to the changes. During software development phase, some documents may subject to update and review process which requires resources effort. In case of source code or class, some classes may still be under development state or not developed at all. Software project manager has the difficulties to make the decision whether to implement or discard the changes due to inconsistent states of software artifacts during software development phase.

Another essential point, the change effort prediction for software development phase also need to consider is the effort distribution of SDLC methodology adopted for a software project. Few earlier studies highlighted the importance of phase wise effort estimation to achieve more accurate results (Chatzipetrou *et al.*, 2015; Choudhari and Suman, 2012; Yang *et al.*, 2008). For instance, during requirement phase in the Traditional methodology i.e. Waterfall model, the effort allocation for coding might be zero, but in the Agile methodology, coding effort must be allocated accordingly. Additionally, effort estimation for requirement changes also needs to consider the inconsistent states of the artifacts during software development phase.

1.2 Statement of the Problem

Software changes may occur at any stages during software development process. Current study stated that the Traditional methodology software projects

usually recorded a low rate of changes acceptance due its detailed planning, comprehensive documentation and extensive design (Awad, 2005). Meanwhile, Agile methodology gives high priority on accepting changes at any point of time throughout the development process due to its environment of incremental elaboration to fulfil the customer satisfaction (Awad, 2005; Stålhane *et al.*, 2014). Regardless the SDLC models adopted by the software development projects, either Traditional or Agile methodology, it is crucial in managing the changes during software development phase in order to meet and satisfy the requirements volatility of the customer (Bennett and Rajlich, 2000; Brooks Jr, 1956; Finkelsteiin and Kramer, 2000; Kotonya and Sommerville, 1998; Pfleeger and Bohner, 1990). Nevertheless, accepting too many changes can drag the project timeline and increase project cost while declining the change request from the customer may trigger dissapointment. Hence, it is a very crucial needs for a software project manager to manage the ever changing requirements as well as make the best decision for the software projects success. One of the input that can assist and support the software project manager to make the best decision is the change effort prediction during software development phase.

However, very little has been written on change effort prediction for software development phase. During this phase, two most related concepts in estimating the required effort for the change request are the change impact analysis and the software effort estimation. Change impact analysis is a procedure of identifying the possible effect of a change, or predicting the process required to undertake a change (Bennett and Rajlich, 2000; Brooks Jr, 1956; Finkelsteiin and Kramer, 2000; Kotonya and Sommerville, 1998; Pfleeger and Bohner, 1990). The objective of the change impact analysis is to detect the potential affected software artifacts (i.e., requirement, design, class and test artifacts) due to the change. Whereas, the objective of the change effort prediction is to estimate the amount of work and time required in implementing the particular changes (Asl and Kama, 2013; Bee Bee, 2010; Chua and Verner, 2010). There are two current models that have been widely used to estimate effort which are the Algorithmic and Non-algorithmic models. Algorithmic models that are commonly used in estimating effort estimation for Traditional methodology include the well-known COCOMO II (Boehm, 2000), Function Point Analysis (Lubashevsky, 1996; Yinhuan *et al.*, 2009) and Use-Case Points (Ochodek *et al.*,

2011). On the other hand, earlier researchers highlighted that Non-Algorithmic model such as expert estimation is preferable in estimating efforts in most of Agile methodology software projects (Keaveney and Conboy, 2006; Popli and Chauhan, 2014) due to the easiness and simplicity in producing effort estimation result without the need of specific tools or techniques (Huang *et al.*, 2008). Although several extensions have been developed based on the current effort estimation models (Ahmed *et al.*, 2012; Lazić and Mastorakis, 2009; Merlo–Schett *et al.*; Yang *et al.*, 2006), but those extensions are still lacking in considering the change effort prediction during software development phase.

The integration of change impact analysis and effort estimation may improve the accuracy of change effort prediction. According to Nurmuliani *et al.* (2006), some change request attributes such as change request type and change requirements have direct effect on the predicted effort to implement that change. Furthermore, Nurmuliani *et al.* (2006) stated that his biggest challenges in his study were that there is no formal impact analysis method to support the change effort prediction, and there are no traceability models for the relations between requirements and classes. Nevertheless, there is at present, no satisfactory explanation of change impact analysis and software effort estimation integration has been provided. Furthermore, most of the current researches only focus on the change impact analysis for software maintenance phase and less attention had been given in software development phase (Kama, 2013a). Hence, it also implied that little attention has been paid to change effort prediction during software development phase.

Software development phase includes an important factor that need to be considered in estimating the change effort which is the inconsistent states of software artifacts in estimating the change implementation effort. The attention of this factor is important as during the software development phase consists of: (1) the existence of partially developed artifacts; (2) the existence of developed artifacts that some of them have been developed conceptually but not technically (or have yet been implemented), and (3) the existence of fully developed artifacts. Although earlier researcher, Sharafat and Tahvildari (2008) had proposed change impact prediction approach in object oriented software projects which uses the UML diagram that representing design of the class artifacts to estimate the propagation posibilities from

one class to another class, yet the approach still did not consider the inconsistence states of the class artifacts. The failure to acknowledge the existence of these type of artifacts will lead to inaccurate estimate and hence, contribute to either project failure or customer dissapointment.

Thus, this research was inspired by the research works of Asl and Kama (2013); Kama and Azli (2012) in which they consider the existence of the inconsistence states of the software artifacts in their change impact analysis approach. This research presents a new algorithmic change effort prediction model by including the principal of the change impact analysis approach that consider the inconsistence states of software artifacts to one of the established effort estimation model for software development phase. This new algorithmic change effort prediction is expected to be applicable in Traditional and Agile methodology software projects and may improve the accuracy of change effort prediction as compared to current effort estimation models.

1.3 Research Questions

This research deals with the main question of "How to improve the accuracy of software change effort prediction for software development by including a change impact analysis into an effort estimation model that applicable for both Traditional and Agile methodologies?"

To provide an effective solution for the main research question, several sub-questions are constructed:

i. What are the existing software change effort prediction and change impact analysis techniques used for software development?
ii. How to calculate the estimated effort required for requirement changes for software development?

iii. How effective the new change effort prediction model as compared to the existing effort estimation model for Traditional and Agile methodology software development?

1.4 Research Objectives

The aim of this research is to propose a new algorithmic-based software change effort prediction model using change impact analysis which could be used to improve the accuracy of the change effort prediction in the Traditional and Agile methodology software projects and to evaluate the applicability and accuracy improvement of the proposed model. Hence to achieve the aim, three objectives are identified as follow:

i. To propose an algorithmic-based software change effort prediction model using a change impact analysis technique for software development.
ii. To build a software change effort prediction prototype that implements the algorithmic-based software change effort prediction model for software development.
iii. To evaluate the applicability and accuracy improvement of the algorithmic-based software change effort prediction model for Traditional and Agile methodology software development as compared to the existing effort prediction model.

1.5 Scopes of Research

The main reason of defining a research scope is to focus the research area and emphasize the boundaries and constraints of the research. Limitation of the research scopes are as following:

1.5.1 Research Context

The research aims to produce a software change effort prediction model using existing change impact analysis for software development phase as in Figure 1.1.

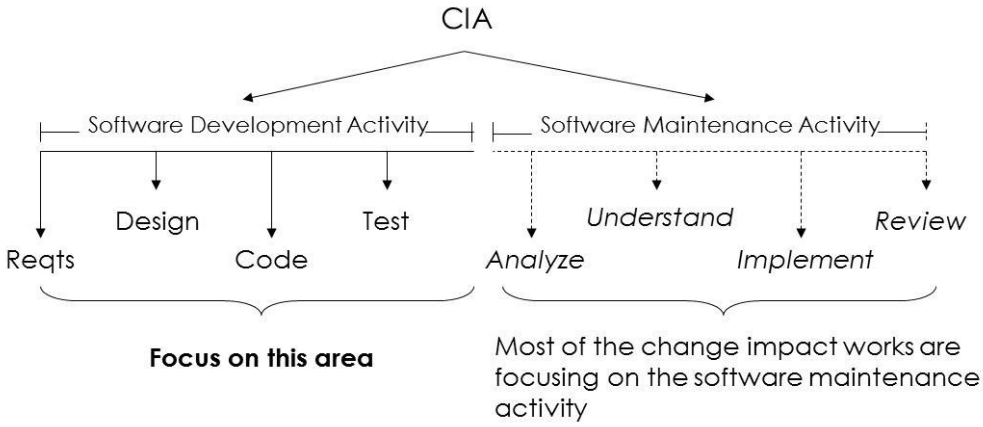

Figure 1.1 Research Context

Most of change impact analysis techniques were developed to support software evolution or requirement changes during maintenance phase. However, this research only focuses on impact analysis techniques that suitable for software development phase. In general, software development phase differs from software maintenance phase due to the existence of inconsistent states of software artifacts such as partially developed classes.

1.5.2 Research Challenges

Since this research focuses on software development phase, there were challenges in capturing the actual information in real software projects during software development process. Although the intention of the research might focus on the software artifacts and its related components, real software project in the industry are constraint with other factors; for example, confidentiality, commercial obligations, politics, complex organization structures, among others. These factors

might affect the research timeline and milestones. Thus, the author outlines following criteria of the software projects:

- Real software projects – Real software projects require participation from business organization or industry. Since this research might not be able to benefit them directly, it is difficult to capture and collect relevant data for this study. However, based on experiences of the author, who had involved in the software industry for more than 10 years, access to real software projects is unpretentious.

- Sufficient documentation – Generally, documentation involves include the change request form, software requirements and software design documentation, source code and progress report. Meanwhile, in the case for Agile methodology the documentation involves may include the product backlogs and sprint backlogs.

- Platform / Language – Software development might be created using certain programming platform and language. Since this research involves dynamic analysis process which involves the dynamic artifacts at class level, it is difficult to develop a prototype that will be able to handle all types of programming platform and language. Thus this research only focuses on a single or two programming platform and language that the author familiar with, to develop a prototype and produce the evaluation results to demonstrate the proposed model.

1.6 Significance of the Research

Main contribution of this research is significant in two perspectives. First, the new effort estimation model will provide crucial information in predicting the amount of work and time required to implement a requirement changes. The new model shall be applicable for two well-known software development process group:

(1) Traditional methodology; and (2) Agile methodology. The effort to implement the requirement changes need to be assessed precisely in order to support the change acceptance decision during software development phase. Additionally, it will support for better planning and prioritization of the requirements implementation during the Traditional and Agile methodology software development.

Next, most change acceptance decision assessment during software development phase is based on change impact analysis techniques. The change impact analysis examines the potential impacts by assessing current state of software artifacts such as requirement specifications and source code during software development phase. By realizing the significance of the change impact analysis, the effectiveness of the development work prediction will be expected to be improved by including the current change impact analysis into the new change effort prediction model during software development phase.

1.7 Operational Definition

The operational definitions of terminologies used in this research are presented below:

Traditional methodology	:	Describe one of the process to develop a software that practices detailed planning, comprehensive documentation and extensive design
Agile methodology	:	More recent technique in developing a software that practices customer collaboration over detailed planning, emphasizes on the working software over the comprehensive documentation and values individual interactions over extensive processes and design.

Software development	:	A software engineering process in developing a software or in short software process. Sometimes also known as Software Development Lifecycle (SDLC)
Software development phase	:	Identify the stages of the software process in developing a software. The stages may start from requirement, analysis, design, implementation, testing until deployment.
Algorithmic model	:	A formal technique that apply algorithms and formulas in order to derive a result of the estimation calculation.
Non-algorithmic model	:	An informal technique that are not using any algorithms or formal methods and / or formulas in deriving the estimation result.
Change	:	The modification or adjustment that occurs during software development phase, which may involve the requirement or the software being developed.
Effort prediction or estimation	:	A process of predicting the amount of work and task required to develop a software that usually described in the form of man/days or man/hours.
Change impact analysis	:	A process of identifying potential consequences of change, or estimating what needs to be modified to accomplish a change.

Change effort prediction or estimation	:	A process of predicting the amount of work or task required in implementing the modification that occurred.
Magnitude Relative Error (MRE)	:	An absolute value that was derived from the difference between the estimated value as compared to the actual value.
Applicability	:	The degree of how much the new model is relevant to the Traditional and Agile methodology
Accuracy	:	The degree of precision of the estimated effort as compared to the actual effort

1.8 Organisation of the Thesis

This thesis comprises of six chapters. This chapter gives an overview of the research area. It also includes the research background, problem statement, research questions, objectives of the research, and the scope of the research. Then it is followed by the significance of the research and finally it outlines the organisation of this thesis.

Chapter Two discusses the comprehensive review of the literature.

Chapter Three describes the research methodology used in conducting this research.

Chapter Four introduces the proposed Change Effort Prediction Model and the development of the prototype.

REFERENCES

Agarwal, N., and Rathod, U. (2006). Defining 'success' for software projects: An exploratory revelation. *International Journal of Project Management, 24*(4), 358-370.

Ahmed, N., Asim, M. R., and Qureshi, M. (2012). A step forward to component-based software cost estimation in object-oriented environment. *arXiv preprint arXiv:1202.2511*.

Ambler, S. W. (2006, 13/5/2006). The Agile Unified Process (AUP). v1.1. Retrieved 25/1/2016, 2016, from http://www.ambysoft.com/unifiedprocess/agileUP.html

Anandhi, V., and Chezian, R. M. (2014, 6-7 March 2014). *Regression Techniques in Software Effort Estimation Using COCOMO Dataset.* Paper presented at the Intelligent Computing Applications (ICICA), 2014 International Conference on, 353-357.

Ani, Z. C., and Basri, S. (2013). A Case Study of Effort Estimation in Agile Software Development Using Use Case Points. *Science International*, *25*(4).

Arnold, R. S., and Bohner, S. A. (1993, 27-30 Sep 1993). *Impact analysis-Towards a framework for comparison.* Paper presented at the Software Maintenance ,1993. CSM-93, Proceedings., Conference on, 292-301.

Asl, M. H., and Kama, N. (2013, 4-7 June 2013). *A Change Impact Size Estimation Approach during the Software Development.* Paper presented at the Software Engineering Conference (ASWEC), 2013 22nd Australian, 68-77.

Attarzadeh, I., Mehranzadeh, A., and Barati, A. (2012, 24-26 July 2012). *Proposing an Enhanced Artificial Neural Network Prediction Model to Improve the Accuracy in Software Effort Estimation.* Paper presented at the Computational Intelligence, Communication Systems and Networks (CICSyN), 2012 Fourth International Conference on, 167-172.

Awad, M. (2005). A comparison between agile and traditional software development methodologies. *University of Western Australia*.

Bardsiri, V. K., Jawawi, D. N. A., Bardsiri, A. K., and Khatibi, E. (2013). LMES: A localized multi-estimator model to estimate software development effort. *Engineering Applications of Artificial Intelligence*(0).

Beck, K. (2000). *Extreme programming explained: embrace change*: Addison-Wesley Longman Publishing Co., Inc.

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., et al. (2001). Manifesto for Agile Software Development. from http://www.agilemanifesto.org/

Bee Bee, C. (2010, 22-27 Aug. 2010). *Rework Requirement Changes in Software Maintenance.* Paper presented at the Software Engineering Advances (ICSEA), 2010 Fifth International Conference on, 252-258.

Benington, H. D. (1987). *Production of large computer programs*. Paper presented at the Proceedings of the 9th international conference on Software Engineering.

Bennett, K. H., and Rajlich, T. (2000). *Software maintenance and evolution: a roadmap*. Paper presented at the Proceedings of the Conference on The Future of Software Engineering.

Bhalerao, S., and Ingle, M. (2009). *Agile estimation using CAEA: A comparative study of agile projects.* Paper presented at the Proceedings of International Conference on Computer Engineering and Applications (ICCEA 2009).

Boehm, B. (1986). A spiral model of software development and enhancement. *SIGSOFT Softw. Eng. Notes, 11*(4), 14-24.

Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., and Selby, R. (1995). Cost models for future software life cycle processes: COCOMO 2.0. *Annals of Software Engineering, 1*(1), 57-94.

Boehm, B. W. (1981). *Software Engineering Economics*: Prentice Hall PTR.

Boehm, B. W. (2000). *Software Cost Estimation with Cocomo II*: Prentice Hall.

Bohner, S. A. (2002a, 5-6 Dec. 2002). *Extending software change impact analysis into COTS components.* Paper presented at the Software Engineering Workshop, 2002. Proceedings. 27th Annual NASA Goddard/IEEE, 175-182.

Bohner, S. A. (2002b, 2002). *Software change impacts-an evolving perspective.* Paper presented at the Software Maintenance, 2002. Proceedings. International Conference on, 263-272.

Bolar, K., and Dastidar, S. G. (2008). Estimating Effort in Agile Software Development Using FPA and COCOMO II. *ICFAI Journal of Systems Management, 6*(4), 17.

Breech, B., Tegtmeyer, M., and Pollock, L. (2006, 24-27 Sept. 2006). *Integrating Influence Mechanisms into Impact Analysis for Increased Precision.* Paper presented at the Software Maintenance, 2006. ICSM '06. 22nd IEEE International Conference on, 55-65.

Brooks Jr, F. (1956). No Silver Bullet - Essence and Accident in Software Engineering.

Cai, H., Santelices, R., and Xu, T. (2014, June 30 2014-July 2 2014). *Estimating the Accuracy of Dynamic Change-Impact Analysis Using Sensitivity Analysis.* Paper presented at the Software Security and Reliability (SERE), 2014 Eighth International Conference on, 48-57.

Chatzipetrou, P., Papatheocharous, E., Angelis, L., and Andreou, A. S. (2015). A multivariate statistical framework for the analysis of software effort phase distribution. *Information and Software Technology, 59*, 149-169.

Chen, C.-Y., and Chen, P.-C. (2009). A holistic approach to managing software change impact. *Journal of Systems and Software, 82*(12), 2051-2067.

Choudhari, J., and Suman, U. (2012). *Phase wise effort estimation for software maintenance: an extended SMEEM model*. Paper presented at the Proceedings of the CUBE International Information Technology Conference.

Chua, B. B., and Verner, J. (2010). Examining Requirements Change Rework Effort: A Study. *arXiv preprint arXiv:1007.5126*.

Chua, Y. P. (2006). *Asas Statistik Penyelidikan* (2 ed.): McGrraw-Hill (Malaysia).

Cleland-Huang, J. (2012). Traceability in Agile Projects. In J. Cleland-Huang, O. Gotel and A. Zisman (Eds.), *Software and Systems Traceability* (pp. 265-275): Springer London.

Cockburn, A. (2002). *Agile software development*: Addison-Wesley Longman Publishing Co., Inc.

Conte, S., Dunsmore, H. E., Shen, V., and Zage, W. (1987). A Software Metrics Survey.

Creswell, J. W. (2003). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*: SAGE Publications.

Creswell, J. W. (2011). *Educational Research: Planning, Conducting, and Evaluating Quantitative and Qualitative Research*: Pearson.

Creswell, J. W. (2013). *Research design: Qualitative, quantitative, and mixed methods approaches*: Sage.

Danhua, S., Khurshid, S., and Perry, D. E. (2009, 14-17 April 2009). *Semantic Impact and Faults in Source Code Changes: An Empirical Study*. Paper presented at the Software Engineering Conference, 2009. ASWEC '09. Australian, 131-141.

Dieste, O., Juristo, N., Moreno, A. M., Pazos, J., and Sierra, A. (2001). Conceptual Modeling in Software Engineering and Knowledge Engineering: Concepts, Technqiues and Trends. In *Handbook of Software Engineering and Knowledge Engineering* (pp. 733-766).

Drew Procaccino, J., Verner, J. M., Overmyer, S. P., and Darter, M. E. (2002). Case study: factors for early prediction of software development success. *Information and software technology, 44*(1), 53-62.

Faria, P., and Miranda, E. (2012, 17-19 Oct. 2012). *Expert Judgment in Software Estimation During the Bid Phase of a Project -- An Exploratory Survey.* Paper presented at the Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2012 Joint Conference of the 22nd International Workshop on, 126-131.

Farr, L., and Zagorski, H. J. (1964). *Factors that Affect the Cost of Computer Programming. Volume Ii. a Quantitative Analysis*: DTIC Documento. Document Number)

Fasolino, A. R., and Visaggio, G. (1999). *Improving software comprehension through an automated dependency tracer.* Paper presented at the Program Comprehension, 1999. Proceedings. Seventh International Workshop on, 58-65.

Febbraro, N., and Rajlich, V. (2007, 13-17 Aug. 2007). *The Role of Incremental Change in Agile Software Processes.* Paper presented at the Agile Conference (AGILE), 2007, 92-103.

Fedotova, O., Teixeira, L., and Alvelos, H. (2013). Software Effort Estimation with Multiple Linear Regression: Review and Practical Application. *J. Inf. Sci. Eng., 29*(5), 925-945.

Finkelsteiin, A., and Kramer, J. (2000). *Software engineering: a roadmap.* Paper presented at the Proceedings of the conference on The future of Software Engineering, 3-22.

Fojtik, R. (2011). Extreme Programming in development of specific software. *Procedia Computer Science, 3*(0), 1464-1468.

Garcia, C. A. L., and Hirata, C. M. (2008). *Integrating functional metrics, COCOMO II and earned value analysis for software projects using PMBoK*. Paper presented at the Proceedings of the 2008 ACM symposium on Applied computing.

Ghazarian, A. (2008). *Traceability patterns: an approach to requirement-component traceability in agile software development*. Paper presented at the Proceedings of the 8th conference on Applied computer scince.

Hall, N. (2007). R. A. Fisher and his advocacy of randomization. *Journal of the History of Biology, 40*(2), 295-325.

Hassine, J. (2015). Early modeling and validation of timed system requirements using Timed Use Case Maps. *Requirements Engineering, 20*(2), 181-211.

Hassine, J., Rilling, J., Hewitt, J., and Dssouli, R. (2005, 5-6 Sept. 2005). *Change impact analysis for requirement evolution using use case maps.* Paper presented at the Principles of Software Evolution, Eighth International Workshop on, 81-90.

Haugen, N. C. (2006, 23-28 July 2006). *An empirical study of using planning poker for user story estimation.* Paper presented at the Agile Conference, 2006, 9 pp.-34.

Huang, S.-J., Chiu, N.-H., and Chen, L.-W. (2008). Integration of the grey relational analysis with genetic algorithm for software effort estimation. *European Journal of Operational Research, 188*(3), 898-909.

Humayun, M., and Gang, C. (2012). Estimating Effort in Global Software Development Projects Using Machine Learning Techniques *International Journal of Information and Education Technology, Vol. 2, No. 3*.

Ibrahim, S., Idris, N. B., Munro, M., and Deraman, A. (2005a, 15-17 February). *Implementing a Document-based Requirements Traceability: A Case Study.*

Paper presented at the IASTED Conf. on Software Engineering, Austria, 124-131.

Ibrahim, S., Idris, N. B., Munro, M., and Deraman, A. (2005b). Integrating Software Traceability for Change Impact Analysis. *Arab International Journal of International Technology, 2*(4), 301-308.

Ibrahim, S., Idris, N. B., Munro, M., and Deraman, A. (2006, June 26-29). *A Software Traceability Validation For Change Impact Analysis of Object Oriented Software.* Paper presented at the Software Engineering Research and Practice, Las Vegas, 453-459.

IEEE Standard for Software Configuration Management Plans. (2005). *IEEE Std 828-2005 (Revision of IEEE Std 828-1998)*, 1-30.

Jacobson, I., Booch, G., Rumbaugh, J., Rumbaugh, J., and Booch, G. (1999). *The unified software development process* (Vol. 1): Addison-Wesley Reading.

Jorgensen, M. (2005). Practical guidelines for expert-judgment-based software effort estimation. *Software, IEEE, 22*(3), 57-63.

Jørgensen, M. (2013). The influence of selection bias on effort overruns in software development projects. *Information and Software Technology, 55*(9), 1640-1650.

Jorgensen, M., and Molokken-Ostvold, K. (2004). Reasons for software effort estimation error: impact of respondent role, information collection approach, and data analysis method. *Software Engineering, IEEE Transactions on, 30*(12), 993-1007.

Jorgensen, M., and Shepperd, M. (2007). A Systematic Review of Software Development Cost Estimation Studies. *Software Engineering, IEEE Transactions on, 33*(1), 33-53.

Jørgensen, M., and Sjøberg, D. I. K. (2004). The impact of customer expectation on software development effort estimates. *International Journal of Project Management, 22*(4), 317-325.

Kama, N. (2013a). Change Impact Analysis for the Software Development Phase: State-of-the-art. *International Journal of Software Engineering and Its Applications, 7*(2), 10.

Kama, N. (2013b). Integrated Change Impact Analysis Approach for the Software Development Phase. *International Journal of Software Engineering & Its Applications, 7*(2), 9.

Kama, N., and Azli, F. (2012). *A Change Impact Analysis Approach for the Software Development Phase*. Paper presented at the Proceedings of the 2012 19th Asia-Pacific Software Engineering Conference - Volume 01.

Kama, N., French, T., and Reynolds, M. (2010). *Impact Analysis using Class Interaction Prediction Approach*. Paper presented at the Proceedings of the 2010 conference on New Trends in Software Methodologies, Tools and Techniques: Proceedings of the 9th SoMeT_10.

Kama, N., and Halimi, M. (2013). *Extending Change Impact Analysis Approach for Change Effort Estimation in the Software Development Phase.* Paper presented at the WSEAS International Conference. Proceedings. Recent Advances in Computer Engineering Series.

Kama, N., and Ridzab, F. A. A. (2012). *Requirement level impact analysis with impact prediction filter.* Paper presented at the International Conference on Software Technology and Engineering (ICSTE 2012).

Kaur, R., and Sengupta, J. (2013). Software Process Models and Analysis on Failure of Software Development Projects. *CoRR, abs/1306.1068*.

Keaveney, S., and Conboy, K. (2006). Cost estimation in agile development projects.

Khan, P. M., and Beg, M. M. S. S. (2013, 6-7 April 2013). *Extended Decision Support Matrix for Selection of SDLC-Models on Traditional and Agile Software Development Projects.* Paper presented at the Advanced Computing and Communication Technologies (ACCT), 2013 Third International Conference on, 8-15.

Khatibi, V., and Jawawi, D. N. (2011). Software cost estimation methods: A review. *Journal of Emerging Trends in Computing and Information Sciences, 2*(1), 21-29.

Khurana, P., Tripathi, A., and Kushwaha, D. S. (2013, 22-23 Feb. 2013). *Change impact analysis and its regression test effort estimation.* Paper presented at the Advance Computing Conference (IACC), 2013 IEEE 3rd International, 1420-1424.

Kotonya, G., and Sommerville, I. (1998). *Requirements engineering: processes and techniques*: J. Wiley.

Kumar, R. (2011). *Research methodology : a step-by-step guide for beginners*. Los Angeles: SAGE.

Kumar, R. (2014). *Research Methodology: A Step-by-Step Guide for Beginners*: SAGE Publications.

Law, J., and Rothermel, G. (2003, 3-10 May 2003). *Whole program path-based dynamic impact analysis.* Paper presented at the Software Engineering, 2003. Proceedings. 25th International Conference on, 308-318.

Lazić, L., and Mastorakis, N. (2009). The COTECOMO: COnstractive Test Effort COst MOdel. In N. Mastorakis, V. Mladenov and T. V. Kontargyri (Eds.), *Proceedings of the European Computing Conference: Volume 2* (pp. 89-110). Boston, MA: Springer US.

Lee, A. S. (1989). A scientific methodology for MIS case studies. *MIS Q., 13*(1), 33-50.

Lehtinen, T. O. A., Mäntylä, M. V., Vanhanen, J., Itkonen, J., and Lassenius, C. (2014). Perceived causes of software project failures – An analysis of their relationships. *Information and Software Technology, 56*(6), 623-643.

Li, B., Sun, X., Leung, H., and Zhang, S. (2013). A survey of code-based change impact analysis techniques. *Software Testing, Verification and Reliability, 23*(8), 613-646.

Li, J., Ruhe, G., Al-Emran, A., and Richter, M. (2007). A flexible method for software effort estimation by analogy. *Empirical Software Engineering, 12*(1), 65-106.

Litoriya, R., and Kothari, A. (2013). An Efficient Approach for Agile Web Based Project Estimation: AgileMOW. *Journal of Software Engineering and Applications, 6*(06), 297.

Lubashevsky, A. (1996, 15-19 Apr 1996). *Living with function points.* Paper presented at the Network Operations and Management Symposium, 1996., IEEE, 632-635 vol.632.

Martin, J. (1991). *Rapid application development*: Macmillan Publishing Co., Inc.

Mauro Gasparini, M. P. R. (2008). *7 Design of Experiments. Handbook of Probability: Theory and Applications. SAGE Publications, Inc*. Thousand Oaks, CA: SAGE Publications, Inc.

Merlo–Schett, N., Glinz, M., and Mukhija, A. Seminar on Software Cost Estimation WS 2002/2003. *Department of Computer Science*, 3-19.

Moløkken-Østvold, K., Haugen, N. C., and Benestad, H. C. (2008). Using planning poker for combining expert estimates in software projects. *Journal of Systems and Software, 81*(12), 2106-2117.

Nelson, E. A. (1967). *Management handbook for the estimation of computer programming costs*: DTIC Documento. Document Number)

Nguyen, V., Steece, B., and Boehm, B. (2008). *A constrained regression technique for cocomo calibration*. Paper presented at the Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement.

Nurmuliani, N., Zowghi, D., and Williams, S. P. (2006). *Requirements volatility and its impact on change effort: evidence-based research in software development projects.* Paper presented at the Proceedings of the Eleventh Australian Workshop on Requirements Engineering.

Ochodek, M., Nawrocki, J., and Kwarciak, K. (2011). Simplifying effort estimation based on Use Case Points. *Information and Software Technology, 53*(3), 200-213.

Oliver, P., and Jupp, V. (2006). Purposive sampling. In V. Jupp (Ed.), *The SAGE dictionary of social research methods* (pp. 244-245): Sage.

Ott, R. L., and Longnecker, M. T. (2008). *An Introduction to Statistical Methods and Data Analysis*: Cengage Learning.

Pfleeger, S. L. (1995). Experimental design and analysis in software engineering. *Annals of Software Engineering, 1*(1), 219-253.

Pfleeger, S. L., and Bohner, S. A. (1990). *A framework for software maintenance metrics.* Paper presented at the Software Maintenance, 1990, Proceedings., Conference on, 320-327.

Popli, R., and Chauhan, N. (2014, 6-8 Feb. 2014). *Cost and effort estimation in agile software development.* Paper presented at the Optimization, Reliabilty, and Information Technology (ICROIT), 2014 International Conference on, 57-61.

Popli, R., Chauhan, N., and Sharma, H. (2014, 7-8 Feb. 2014). *Prioritising user stories in agile environment.* Paper presented at the Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on, 515-519.

Popovic, J., and Bojic, D. (2012). A comparative evaluation of effort estimation methods in the software life cycle. *Computer Science and Information Systems, 9*(1), 455-484.

Royce, W. W. (1987). *Managing the development of large software systems: concepts and techniques*. Paper presented at the Proceedings of the 9th international conference on Software Engineering.

Runeson, P., and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering, 14*(2), 131-164.

Schwaber, K. (1997). Scrum development process. In *Business Object Design and Implementation* (pp. 117-134): Springer.

Schwaber, K., and Beedle, M. (2001). *Agile Software Development with Scrum*: Prentice Hall PTR.

Seo, Y.-S., Bae, D.-H., and Jeffery, R. (2013). AREION: Software effort estimation based on multiple regressions with adaptive recursive data partitioning. *Information and Software Technology, 55*(10), 1710-1725.

Shahid, M., and Ibrahim, S. (2016, 12-16 Jan. 2016). *Change impact analysis with a software traceability approach to support software maintenance.* Paper presented at the 2016 13th International Bhurban Conference on Applied Sciences and Technology (IBCAST), 391-396.

Shanks, G. G., and Parr, A. N. (2003). *Positivist single case study research in information systems: a critical analysis.* Paper presented at the ECIS, 1760-1774.

Sharafat, A. R., and Tahvildari, L. (2008). *Change Prediction in Object-Oriented Software Systems: A Probabilistic Approach* (Vol. 3).

Sharif, B., Khan, S. A., and Bhatti, M. W. (2012). Measuring the Impact of Changing Requirements on Software Project Cost: An Empirical Investigation. *International Journal of Computer Science Issues (IJCSI), 9*(3), 170-174.

Shepperd, M., Schofield, C., and Kitchenham, B. (1996). *Effort estimation using analogy*. Paper presented at the Proceedings of the 18th international conference on Software engineering, 170-178.

Shull, F., and Feldmann, R. (2008). Building Theories from Multiple Evidence Sources. In F. Shull, J. Singer and D. K. Sjøberg (Eds.), *Guide to Advanced Empirical Software Engineering* (pp. 337-364): Springer London.

Singh, M., and Vyas, R. (2012). Requirements Volatility in Software Development Process. *International Journal of Soft Computing, 2*.

Small, A. W., and Downey, E. A. (2001, 2001). *Managing change: some important aspects.* Paper presented at the Change Management and the New Industrial Revolution, 2001. IEMC '01 Proceedings., 50-57.

Sommerville, I. (2010). *Software Engineering* (9 ed.). Harlow, England: Addison-Wesley.

Stålhane, T., Katta, V., and Myklebust, T. (2014). Change Impact Analysis in Agile Development. *EHPG Røros*.

Stammel, J., and Trifu, M. (2011). *Tool-supported estimation of software evolution effort in service-oriented systems.* Paper presented at the First International Workshop on Model-Driven Software Migration (MDSM 2011), 56.

Suri, P. K., and Ranjan, P. (2012). Comparative Analysis of Software Effort Estimation Techniques. *International Journal of Computer Applications (0975–8887), 48*(21).

Usman, M., Mendes, E., Weidt, F., and Britto, R. (2014). *Effort estimation in agile software development: a systematic literature review*. Paper presented at the Proceedings of the 10th International Conference on Predictive Models in Software Engineering.

Valerdi, R. (2011). Convergence of expert opinion via the wideband delphi method: An application in cost estimation models.

Verner, J. M., Evanco, W. M., and Cerpa, N. (2007). State of the practice: An exploratory analysis of schedule estimation and software project success prediction. *Information and Software Technology, 49*(2), 181-193.

Wohlin, C., and Aurum, A. (2014). Towards a decision-making structure for selecting a research design in empirical software engineering. *Empirical Software Engineering*, 1-29.

Wood, S., Michaelides, G., and Thomson, C. (2013). Successful extreme programming: Fidelity to the methodology or good teamworking? *Information and Software Technology, 55*(4), 660-672.

Yang, D., Wan, Y., Tang, Z., Wu, S., He, M., and Li, M. (2006). COCOMO-U: An Extension of COCOMO II for Cost Estimation with Uncertainty. In Q. Wang, D. Pfahl, D. Raffo and P. Wernick (Eds.), *Software Process Change* (Vol. 3966, pp. 132-141): Springer Berlin Heidelberg.

Yang, Y., He, M., Li, M., Wang, Q., and Boehm, B. (2008). *Phase distribution of software development effort.* Paper presented at the Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement, 61-69.

Yin, R. K. (2003). *Case Study Research: Design and Methods*: SAGE Publications.

Yin, R. K. (2013). *Case Study Research: Design and Methods*: SAGE Publications.

Yinhuan, Z., Beizhan, W., Yilong, Z., and Liang, S. (2009, 25-28 July 2009). *Estimation of software projects effort based on function point.* Paper presented at the Computer Science & Education, 2009. ICCSE '09. 4th International Conference on, 941-943.

Zhou, R., and Hansen, E. A. (2006). Breadth-first heuristic search. *Artificial Intelligence, 170*(4–5), 385-408.