NAIST-IS-DD1301021

Doctoral Dissertation

Conclusion Stability on Performance of Analogy-Based Software Effort Estimation

Passakorn Phannachitta

February 4, 2016

Department of Information Science
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Passakorn Phannachitta

Conclusion Stability on Performance of Analogy-Based Software Effort Estimation*

Passakorn Phannachitta

Abstract

Analogy-based estimation (ABE) is one of the most successful methods to estimate the required amount of effort for a new software development project. According to our literature review, the excellent estimation accuracy of ABE is strongly associated with approaches adopted in its 5 essential components: normalization of software project features, feature subset selection, similarity measures, solution adaptation, and the number of analogues. Being a very successful effort estimation method, researchers have continually proposed new approaches to these 5 essential components, combined the approaches, and tailored them to the ABE models mainly for performance improvements. To date, it has been reported that over thousands of combinations of approaches are existed; however, to the best of our knowledge, no one could successfully determine the best one. The problem persists mainly due to conflicting research conclusions frequently shown in past studies of ABE, where different studies often produced divergent performance results. On the contrary, being able to solve this problem is important for a wide range of scientific and industrial processes, that is, unless a stable conclusion on the performance can be drawn, it would be difficult for industrial practitioners to be able to access a sufficiently accurate model, resulting in increasing risk of failed software project. Also for the research community, lack of sufficiently effective models may limit the potential improvement of future model proposals, since any new approaches maybe evaluated with inadequate standard. This brings to mind that being able to determine a stable conclusion on the

*Doctoral Dissertation, Department of Information Science, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD1301021, February 4, 2016.

performance of the estimation models is a key instrument of the software effort estimation activities.

This thesis comprises multiple studies to conclude the performance of ABE. The series of studies began with an attempt to improve a stable ranking method to produce a more trustworthy assessment of the performance. Then, this Improved stable ranking method was adopted to evaluate 2,304 combinations of approaches generated from those commonly selected from the 5 essential components. Leveraged by this Improved stable ranking method, we were able to draw the performance conclusion of these approaches and recommend those superior ones. Then, as a continuation study, the results were further analyzed for suggesting general hypotheses. This study is where we could successfully introduce a very simple and easy-to-use solution adaptation technique, whose performance was shown to be outstanding. Taken all the findings together, we coined ABE-Best as the ABE model tailoring with the best approaches in each of its 5 essential components determined in the studies of this thesis, and compared it with 7 other common machine-learning effort models (NNet, LReg, SWReg, PCReg, PLSReg, CART(yes), and CART(no)). The results of this comparison were conclusive that this ABE-Best model outperformed the other 7 commonly adopted effort models by all means of overall performance, generalized performance, stability, and robustness. Hence, we strongly recommend ABE-Best to be the standard benchmark effort estimation model for research, and to be the model of choice for the software industry.

Keywords:

software development effort estimation, analogy-based estimation, model-based software effort estimation, empirical software engineering, robust statistical method

Acknowledgements

First and foremost, I would like to thank my Ph.D. committee chair and my supervisor Professor Kenichi Matsumoto, who had continually supported me wholeheartedly, not only on my studies but also on my life during my stay in Japan. His advice he gave me on the day I started my life in Ph.D. has become one of my most prized pieces of advice towards the way I think and the way I approach any kind of problems in my life since then. I am also thankful to him for all his advice about my researches in both the big picture as well as the littlest details, and for all the precious opportunities he has continually gave me to allowing me to have great experience such as meeting with many great people. Those opportunities are priceless and are of the most important factors in completion of my Ph.D. I am heartily happy to be his student and be part of his laboratory.

This thesis would not have been completed without my advisor, Professor Akito Monden. It was a blessing to be one of his students and be able to conduct many research studies under his advice. His kindness and patience has put me at ease and made me more confident in carrying out all my works. I am also grateful to him for his encouragement, support, and patience to me all these years, and for allowing me to conduct my experiments as part of his research as an initial project of the research series of this thesis. Without the opportunity he gave me on that day, this thesis research might not have been existed.

To my thesis committee, I would like to thank Professor Hajimu Iida, who have seen my work from my undergraduate internship until my final defense. His advise during all of my seminar presentations inspired me a lot of new idea and drive on my new research. I really appreciated that.

To Assistant Professor Jacky Keung, my co-advisor, who introduced me to the world of empirical software engineering. His never-ending support and encouragement are the important factors towards a completion of my Ph.D. I am heartily appreciate his patient on me when I completely had almost zero knowledge to justify the difference between high quality and poor quality research, and about appropriate scientific writing. I would also thankful to him for his kindness in helping me reviewed my poorly written manuscripts and suggesting how to make them correct, and for his very kind words whenever I got very negative feedbacks from the reviewers of my papers. All of these have encouraged me a lot throughout my Ph.D. journey. In addition, not only limited to my research activities has he tremendous effect on, his wise advice on me as a person has completely changed me upon the way to better communicate with other people and to make connections. Further than the outputs from my research activities, to me, I am in his debt for making me improve my social skills and the way I see the world outside. I would say that my successful of my Ph.D. is greatly influenced by this change.

I am grateful to Associate Professor Arnon Rungsawang, my advisor during my undergraduate's study, who was very instrumental for me to begin master's study and continuing to my Ph.D.'s study. His intensive training in research methodologies since my undergraduate's study brought me to the research communities. He has a considerable influence on me to prefer a working life in academia.

I would like to thank Professor Barbara Kitchenham, a great opportunity to meet her in person in the 2015 EASE conference is absolutely priceless. Her wise advice has improved me a lot on the way to conduct a my research. I can say that this thesis would not have been completed with this entire research series without her advice in regard to the importance of robust statistical methods and the validity of research studies.

To Assistant Professor Akinori Ihara, he was my kind tutor when he was a doctoral student when I firstly visited Software Engineering Laboratory as an internship student six years ago. I appreciate all of his kind continuous support since then up to my graduation. To Assistant Professor Hideaki Hata, even if I did not have much opportunity to work with him, one of a few opportunities to talk with him had greatly affect on the decision for my future to continue working in academia. That talk reminded me upon how much I love to share my knowledge and experience, and to inspire other people. I am also grateful to him to allowing me to take care of some students on their research studies. I am really happy to be able to do that. To Associate Professor Masao Ohira, my advisor during my master's study, his advices and questions broaden a lot of my perspective on the way to conduct research. To Dr. Pattara Leelaprute, who introduced me to NAIST, and gave me an opportunity to come to Japan as an internship student when I was in my undergraduate. To Dr. Bundit Manaskasemsak, who taught me a lot of necessary principal knowledge in Computer Engineering. To Professor David Sell and Professor Mike Barker, their advise on advance scientific writing help me improve my literature a lot. They alway give me their valuable time to revise my research papers. I am greatly appreciated all their kindness and help.

Also I would like to thank Ministry of Education, Culture, Sports, Science and Technology (MEXT) for granting me the greatest scholarship, which supported my allowance through my student life here in Japan. Also to the Mobile Fund Communication (NTT DOCOMO) Scholarship Program, who supported me for 2 years during my master.

To all members of SE Lab in NAIST, all my friends in Japan and in Thailand, thank you all for all their friendship and support. Special thanks for SE Lab members go for Takagishi-san for her continually supports and helping. Special friendship thank for my brother, Papon-san, we spent a lot of great time during my life in Japan. I will never forget our countless moments. Many thanks go to Hamada-san, Ohta-san, Nagano-San, Yurie-san, Sumitani-san, Hatoyama-san, and Kamiya-san from International student section support.

Finally and most importantly, I would like to thank my mum and dad, who always and never ending encourage and guide me in many aspects of my life. I appreciate their constant source of continual supports in all forms throughout my life. Their unconditional love provided my inspiration and was my driving force that has pulled me this far. Special thanks to my dad on that he always teaches me how the vision is the most important thing to procure, and to my mum who always takes care of me. I am grateful to them for allowing me to chase my dreams and be as ambitious as I would always wanted to be. To them. I dedicate this thesis, which would certainly not have been existed without them, as a small present I could express them my gratitude for everything I owe them.

To my family,
for their love, endless support and encouragement

List of Major Publications

Peer review journal paper

1. Passakorn Phannachitta, Akito Monden, Jacky Keung, and Kenichi Matsumoto, “LSA-X: Exploiting productivity factors in linear size adaptation for analogy-based software effort estimation”, IEICE Transactions on Information and Systems, Vol.E99-D, No.1, pp.-, 2016.

Peer review international conference

1. Passakorn Phannachitta, Akito Monden, Jacky W. Keung, Kenichi Matsumoto, “Case consistency: a necessary data quality property for software engineering data sets,” In Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering (EASE), pp.19-28, 2015.

2. Passakorn Phannachitta, Jacky Keung, Akito Monden, Kenichi Matsumoto, “Scaling up analogy-based software effort estimation: a comparison of multiple Hadoop implementation schemes,” In Proceedings of the International Workshop on Innovative Software Development Methodologies and Practices (InnoSWDev), pp.65-72, 2014.

3. Passakorn Phannachitta, Jacky Keung, Akito Monden, Kenichi Matsumoto, “Improving analogy-based software cost estimation through probabilistic-based similarity measures,” In Proceedings of the 20th Asia-Pacific Software Engineering Conference (APSEC), pp.541-546, 2013.

4. Passakorn Phannachitta, Jacky Keung, Kenichi Matsumoto, “An empirical experiment on analogy-based software cost estimation with CUDA framework,” In Proceedings of the 22nd Australian Software Engineering Conference (ASWEC), pp.165-174, 2013.

Other Publications

Peer review journal paper

1. Passakorn Phannachitta, Akinori Ihara, Pijak Jirapiwong, Masao Ohira, Kenichi Matsumoto, "An algorithm for gradual patch acceptance detection in open source software repository mining", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E95-A, No.9, pp.1478-1489, 2012.

2. Yosuke Yamatani, Masao Ohira, Passakorn Phannachitta, Akinori Ihara, "A data mining method for understanding co-evolution of OSS systems and communities", IPSJ Journal, Vol.56, No.1, pp.59-71, 2015.

Peer review international conference

1. Passakorn Phannachitta, Pijak Jirapiwong, Akinori Ihara, Masao Ohira, and Kenichi Matsumoto, "An analysis of gradual patch application: a better explanation of patch acceptance," In Proceedings of the Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement (IWSM-MENSURA), pp.106-115, 2011.

2. Passakorn Phannachitta, Pijak Jirapiwong, Akinori Ihara, Masao Ohira, and Kenichi Matsumoto, "Understanding OSS openness through relationship between patch acceptance and evolution pattern," In Proceedings of the International Workshop on Empirical Software Engineering in Practice (IWESEP), pp.37-42, 2011.

3. Papon Yongpisanpop, Passakorn Phannachitta, Masao Ohira, and Kenichi Matsumoto,"An adaptive search framework for supporting cooperative work in organizations," In Proceedings of the 10th Asia Pacific Conference on Computer Human Interaction (APCHI), pp.603-610, 2012.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Overview

Estimation of software development effort is one of the most important processes in software development because an accurate estimation is non-trivial strongly associated with the success rate of software development projects [1]. For example, either underestimation or overestimation of the required effort is an underlying cause of unsuccessful projects [2]. Consequently, new methodologies to obtain accurate effort estimates have been continually proposed and research studies based on them have been frequently revisited [3–5].

Analogy-based software effort estimation (ABE) is one of the most widely adopted effort estimation method in both the industry and research communities. It derives an estimated effort value from the total amounts of effort used on completed similar software projects, following a hypothesis: *projects with similar characteristics will require similar amounts of effort to complete development* [6, 7]. Despite being an estimation method developed from an intuitively easy to understand hypothesis, an estimation accuracy based on ABE has been ranked among the best model-based software effort estimation methods [8, 9]. In other word, ABE is an effort estimation method being excellent in both terms of estimation performance and the intuitive ability in practice [7].

Improving the accuracy of ABE is an active research topic in the field of empirical software engineering [7, 10–17]. For over two decades, researchers have continually proposed new approaches as an extension to ABE models to improve either their maximum achievable accuracy or generalized accuracy. Our literature review provided in Chapter 2 reported that (1) normalization of software project features, (2) feature subset selection, (3) similarity measures, (4) solution adaptation to the effort, and (5) the number of analogues are approaches being continually proposed to improve the accuracy of ABE during the past decade. Elsewhere, Kocaguneli et al. [1] reported that more than thousands of combinations of these approaches have been accumulated since the proposal of ABE; however, a single study always focuses on single specific components of ABE, and an evaluation of the proposed approaches often configured other components with very simple approaches. For example, a research study aims to improve the process that retrieves similar projects to the new case often paid no attention on the solution adaptation of the estimated effort values [18], even though both of which are imperative processes of ABE [15, 16, 19]. Furthermore, studies that focused on the same essential components of ABE tend to evaluate the proposed approaches with different performance measures [17, 20–22], and thus the performance conclusions were often contradictory in different studies. Hence, to the best of our knowledge, no one has successfully defined a standard procedure which is sufficiently reliable to systematically assess the estimation accuracy of ABE models extended with thousands of the existing combinations of approaches commonly adopted to improve the process of ABE. Without this review and assessment available, it seems to be impossible to know the actual performance of ABE models, also an apparent answer to the question *which combinations of approaches would be best in general situations?* is yet available.

Speaking of evaluation and assessment, divergent performance results commonly appeared in different studies indicate clearly that conclusion instability [9, 23] is one major problem in software effort estimation research studies. According to our survey of the literature on software effort estimation, only single study focused mainly on this issue and proposed a method to overcome it. A method named *Stable ranking method* was proposed by Keung et al. [9] based on the suggestions by Menzies et al. [23] that the precise and stable results require the well-controlled experimental conditions of (1) the method to generate training/test instances, (2) the performance evaluation criteria used, and (3) the datasets. Leveraged by the *Stable ranking method* tailoring with well-controlled evaluation procedures, results from that study show that this method can overcome the conclusion instability problem in model-based effort estimation methods, where precise ranking of 9 simple effort models, such as linear regression, in combination with 10 data preprocessing techniques were successfully determined.

However, several components of the Keung et al.'s *Stable ranking method*, such as the selected performance measures and the power of the selected statistical test method, are still questionable upon their validity. Elsewhere, Foss et al. [24] concerned the validity of many performance measures based on relative error, such as *MMRE* and *MMER*, the measures being used in part of the *Stable ranking method*. Also, Mittas and Angelis [25] concerned a simple pairwise comparison, the essence of *Stable ranking method*, as it can easily mislead the comparison results due to lack of statistical power when applied in general situations. This motivated us to improve the validity of the Keung et al.'s *Stable ranking method* and use the improved method to perform performance assessment of thousands of combinations of approaches that commonly adopted to improve the accuracy of ABE models.

Our studies for the past 3 years composing into this thesis has been aimed to search for the maximum achievable performance of ABE models over the configuration variants of the models that commonly adopted in practice. We first discussed the *Stable ranking method* with an aim to improve the validity of the method. Our improvements to the method were empirically evaluated by finding the stable performance conclusion of approaches in the 5 essential components of ABE, being known as components strongly associated with its performance. In totals of 2,304 combinations of approaches were examined, it is conclusive to suggest approaches for all the 5 components, and the ABE model adopted based on the suggested combination of approach has proved to be consistently better than many other common software effort models.

Furthermore, a study in this thesis successfully proposed a new approach following a more comprehensive understanding of theoretical concepts underpinned by an analysis over the conclusive performance results with stable ranking. In that study, we proposed a very simple solution adaptation technique. It is a technique being very easy to compute and use, as well as being excellent in terms of estimation performance. Up to this point, we can suggest the conclusively best combination of approaches of choice for the ABE models, selected from the most commonly discussed approaches in the literature. In addition, a study of this thesis evaluated 6 data quality improvement techniques (**LTS**, **CD**, **Kmeans**, **BP**, **LM**, and **TEAK** [14, 26]), which are techniques originally proposed for ABE or commonly adopted with it in the literature. Our results strongly suggested to apply any of the these techniques to datasets before estimating effort using them. However, our results could not suggest the one single technique significantly outperformed the others, due to their different theoretical and practical behaviors. Hence, given a local dataset, we suggest to fit ABE models with that dataset prior to its application as long as there is no selection guideline of the data quality improvement techniques available. In the final study of Chapter 7, we reviewed the best ABE model we discovered in the study of the earlier chapter, with 7 other commonly adopted model-based effort estimation methods (**NNet**, **LReg**, **SWReg**, **PCReg**, **PLSReg**, **CART(yes)**, and **CART(no)** [2]), the results showed that this ABE model yielded the best performance in all terms of overall performance, generalized performance, stability, and robustness.

1.2. Contributions

The contributions of the studies composing into this thesis are mainly in three-fold:

- The main contribution of this thesis is to provide empirical evidence on the success of our *Improved stable ranking method* to draw stable conclusions for many studies of the thesis. We hope that this will encourage researchers or industrial practitioners to seek for a robust and consistent methodology when carrying out an assessment of any problems.
- The performance factors of ABE and the list of approaches compared in all the studies of the thesis are derived from a limited systematic literature review, carried out to cover the recent 10 years of journals, magazines, and conference papers from two common digital libraries: IEEE Xplore digital library and the ACM digital library. Summary of the approaches selected in our studies are provided in a review manner with their theoretical concepts,

along with their mathematical formulas. These can be considered as one of the most complete reviews of approaches in different components of ABE so far. We believe that this systematic literature review and the review of approach will be useful for the readers towards a future replication of our studies.

- The last but not least is all the findings from all the studies of the thesis, the experimental results of the earliest study in Chapter 4 answered the question *which combinations of approaches would be best in general situations.* Next, a continual study of Chapter 4 suggested a new and very easy to use solution adaptation technique in Chapter 5. More importantly, proposing this new technique can be considered as an evidence to suggest that an analysis of results obtained from a more robust and comprehensive evaluation methodology can greatly contribute to future direction of researches in the field of empirical software engineering. In Chapter 6, our results suggested that poor data quality is still a major and terrifying problem in our communities, where are still a long ways away from being able to appropriately handle it. Finally, the study of Chapter 7 concluded that the ABE model tailoring with the best combination of approaches indicated in the earlier chapters of this thesis significantly outperformed 7 other machine learning-based effort models, which were also commonly appeared in the literature on empirical software engineering.

1.3. Structure of this Thesis

This thesis is structured as follows: Chapter 2 shares common materials on background notes of software effort estimation and ABE. This chapter explains the 5 essential components of ABE models, commonly referred to as performance factors of the model, as well as the approaches commonly extended to each of the 5 components of ABE to increase its maximum achievable estimation performance. Chapter 3 explains the Keung et al.'s *Stable ranking method* and our improvements to the components of the method which are still being questionable, such as its performance measures and the statistical test method. Then, the detailed explanation on the application of our *Improved stable ranking method* adopted in all the experiments of this thesis is provided along with other common materials regarding the evaluations, such as datasets. The research studies of this thesis begin in Chapter 4, where we evaluated the validity of the *Improved stable ranking method* and the performance of 2,304 combinations of approaches adopted in each of the 5 essential components of ABE models. Analyses of the results of this chapter allowed us to propose a new solution adaptation technique, one of the most imperative components of ABE models in Chapter 5. Next, we examined 6 data quality improvement techniques with the superior ABE model concluded from earlier chapters in Chapter 6. In this chapter, we also defined **ABE-Best** as the ABE model built from the conclusively best combination of approaches in the 5 essential components of the model, concluded in earlier chapters. The final chapter for the research studies of this thesis is Chapter 7. It is where we compared this **ABE-Best** with 7 other commonly adopted model-based software effort estimation methods. From Chapter 4 to Chapter 7, where research studies were presented, the structure of these chapters are as given below:

These chapters begin with the introduction of the study corresponding to the chapter, followed by motivation and the hypothesis. If a chapter has its specific background knowledge required, materials for the background will be provided here. Next, a small section is presented to explain the evaluation procedure of each chapter. This small chapter only briefly explains how the study of each chapter adopted and utilized the *Improved stable ranking method*, where its the detailed explanations were provided earlier in Chapter 3. Then, experimental results are presented in the result section, followed by the discussion, which are presented along with interesting future works. The final section of these 4 chapters are the conclusion of each individual study.

For the final part of this thesis, the validity threats of all the 4 studies of this thesis are elaborated in Section 8. Finally, we discuss the worth highlighting future directions of ABE and conclude the thesis in Chapter 9.

Chapter 2

Background of Software Development Effort Estimation

Chapter 2 explains the common materials that are shared by later chapters. In this chapter, we mainly focus on the general concept of software effort estimation, and analogy-based effort estimation and its components. For materials related to the evaluation procedures, such as datasets and performance measures, they will be explained in the next chapter along with the explanation of our *Improved stable ranking methods*, the evaluation methods mainly used across studies of this thesis. All these materials are presented upfront to avoid repeating of the information as many later chapters frequently refer to these materials. For a more specific background that may be only required by an individual chapter, we will provide them later in those chapters.

2.1. Software Effort Estimation (SEE)

Software effort estimation (SEE) is a process to acquire the total amount of effort required by a new software project to complete development. Estimating the necessary effort to develop software projects is one of the most vital software process because either underestimation or overestimation of the required effort is an underlying cause of unsuccessful projects [2]. Therefore, software project managers always seek for accurate and reliable estimation methodologies for allowing them to control the development project to completion of desired objectives and goals. The challenge towards facilitating an accurate estimation is that the information sources available in the early stage of development is frequently limited in amount and being uncertain, where as inaccurate estimation involving high risk that possibly affects the project quality and may end up as worst as project cancellation due to resource shortage.

Several previous studies suggested a strong association established between successful software project and accurate software development effort estimation [2, 6, 14]. Consequently, a large number of methodologies to obtain accurate estimates have been continually proposed in software engineering research communities [3, 4, 27, 28]. If a researcher or an industrial practitioner reads the literature on SEE, many taxonomies to classify the SEE methods are possibly encounters. Kocaguneli et al. [2] reported several taxonomies commonly referred to in the literature. One of the most comprehensive taxonomy divided the SEE methods into 3 classes, suggested by Shepperd et al. [29].

Based on this taxonomy, methods are classified into expert-based methods, model-based methods, and analogy-based methods. Expert-based estimation methods seek for the consensus of human experts; thus, the estimation accuracy of this the method may rely heavily on explicit domain experts. Model-based estimation methods involves the application of mathematical models [30] or machine learning models [31] to summarize historical project data as references to estimate the required effort for a new software project. Analogy-based estimation methods also make prediction based on historical project data; however, their procedure are divergent. Analogy-based estimation is mainly a data-driven method that finds similar historical projects, and then reuses the effort values of these projects for the new project cases.

Suggested by Nagpal et al. [32], expert judgements are prominent as long as it is clear that the estimation is made by the real expert. However, in practice, it is difficult to ensure that every single software company can procure the real expert. This possibly becomes more challenging for small-size enterprises where the SEE processes are being responsible by only a few employees because knowledge based on individual employee is difficult to completely transfer to others [29]. Regarding this concern, model-based and analogy-based methods are generally more preferable in practice. In addition, according to a systematic literature review by Jorgensen and Shepperd [3], machine learning model-based estimation and analogy-based estimation were found to be the most commonly applied estimation methods in both the software industry and research communities.

2.2. Analogy-Based Software Effort Estimation (ABE)

An analogy-based estimation (ABE) is a widely used approach to estimate the amount of software development effort required for a software project development. This approach is more preferable mainly because of its excellent estimation performance and its intuitive ability in practice [7]. ABE derives an estimated effort value for a new software project from the total amount of effort used on already completed similar software projects, following a hypothesis: *projects with similar characteristics will require similar amounts of effort to complete development* [6, 7]. Following this essential hypothesis, an effort estimation framework based on ABE commonly consists of two main processes: a retrieval of the past similar projects to the new case, and an adaptation to effort of the retrieved projects to the new case.

Using ABE to estimate the software development effort involves a 4-stage case-based reasoning process [33], consisting of:

Retrieve the project cases most similar to the new case;

Reuse the information from the retrieved past cases to propose a solution to the new case;

Revise the proposed solution to better adapt to the new case;

Retain the solved case for future problem solving.

In practice, this 4-stage process is commonly viewed as an incremental process. Fig. 2.1 graphically explains the process.

Adopting this 4-stage process as an effort estimation framework, the computing process considers software project cases as having two parts [6]: the description and the solution. The solution part holds information of required effort

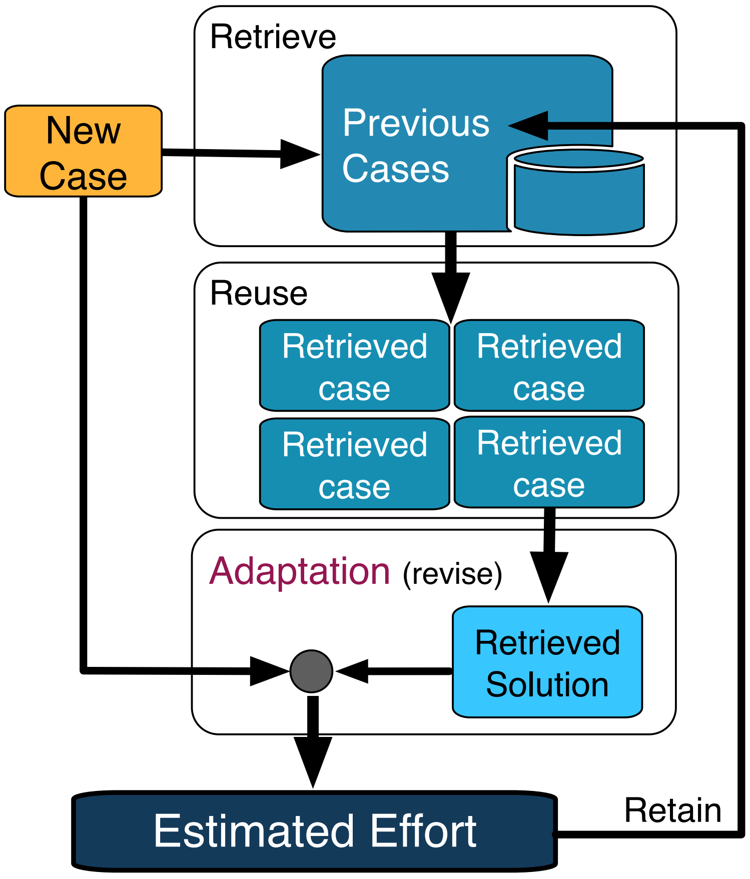

Figure 2.1. The effort estimation process based on ABE

for developing each project, and the description part consists of features that describe the project. Following the 4-stage cycle, similar cases are measured by similarity of the description parts between project case pairs. The solution part retrieving from the similar cases are then reused and revised to calculate the estimated effort of the new software project. Finally, the entire description and solution part of the new case will be recorded for future problem solving.

2.2.1 Influential Factors on the Estimation Performances of ABE

As we mentioned earlier in this thesis that there are more than thousands of combinations of approaches commonly tailored the basic ABE model (i.e., ABE0) [9] to improve either its maximum achievable accuracy or its generalized accuracy, the beginning of this section is devoted to a limited systematic literature review to show which are approaches commonly referred to or discussed as the essences of ABE by means of being its influential factors on its estimation performance.

The review covered journals, magazines, and conference proceedings published between January 2006 and August 2015 (a timeframe of 10 years) in the IEEE Xplore digital library and the ACM digital library. The review was carried out with the procedures used by Kitchenham and Mendes in one of their prominent studies [34], where we decided to follow up only the references related to the performance factors of ABE. If the references are obviously being relevance, we followed them up even if they were published before 2006. However, we did not extend the search to cover either more digital libraries nor any offline libraries. The search was performed in August 2015 using the query: "(Software) AND (cost OR effort) AND (analogy OR reason*) AND (estimat* OR predict*)".

The search retrieved 302 relevant papers. Of which, 10 papers reviewed the performance factors of ABE [7, 10–17, 19]. Summarized from these 10 papers, 7 distinct performance factors are listed in Table 2.1 along with the frequency counts of the appearance of each factor in these 10 papers.

Table 2.1 shows that similarity measure was the factor most frequently referred to as performance factors of ABE models, followed closely by feature subset selection, number of analogues, and solution adaptation. The bottom-end of this table reported that data quality, feature weighting, and case subset selection were also the performance factors of ABE, but were less frequently considered than the 5 other factors. Based on this summary, we consider the 5 factors located on the top ranks of Table 2.1 as essential components of ABE, which will be later discussed throughout this thesis.

Table 2.1. Range and diversity of the performance factors of the ABE method.

Performance factors	Studies	Counts
1 Similarity measures	[7, 10–17, 19]	10
2 Feature subset selection	[7, 10, 11, 13–17, 19]	9
3 Number of analogues	[7, 10, 11, 14–17, 19]	8
4 Solution adaptation	[7, 10, 11, 13–16, 19]	8
5 Normalization or scaling	[11, 12, 14, 15, 17, 19]	6
6 Data quality	[12–14, 19]	4
7 Feature weighting	[10, 16, 17]	3
8 Case subset selection	[14]	1

2.2.2 The Common Baseline ABE Model: ABE0

The name *ABE0* was coined by Keung et al. [9] as a basic and standard form of an ABE model. *ABE0* adopts Euclidean distance function [35] to calculate the level of similarity between project cases, and uses k nearest neighbor (kNN) technique to select the k software project cases that are most similar to the new case. The default solution adaptation technique of *ABE0* is an unweighted mean of the effort values of the k software projects selected by kNN. The k parameter value is commonly fixed as a static value in the literature. For example, Walkerden and Jeffery [13] set the number statically to 1 in their experiment. Kirsopp and Shepperd observed $k = 2$ in [36]. Mendes et al. examined the number in a range from 1 to 3 in [37] to be appropriate. And Baker [38] tuned the k parameter value by a method called wrapping and used the *best-k* value that fits the best to the training set.

2.2.3 Tailoring ABE0 with Combinations of the Five Essential Components of ABE models

This section describes the approaches that were proposed to improve performance of ABE models and were commonly discussed in the literature. The combinations of these approaches generated numerous variants of ABE models. The following in this section explains 5 components of ABE models, commonly referred to as their performance factors:

- 3 Normalization approaches
- 4 Feature subset selection methods
- 6 Similarity measures
- 8 Solution adaptation techniques
- 4 Approaches to determine the number of analogies

Combining all the 5 components being listed above, there are 2,304 combinations of approaches of ABE models to be examined.

Three Normalization Approaches

Feature normalization is commonly applied to continuous features (quantitative variables) of effort estimation datasets before their applications. Two main purposes of the feature normalization are (1) to assure the equal influence of all the individual continuous features, and (2) to transform each feature to a more closely approximated normal distribution. Our experiments explore three schemes to interpret the continuous features as given below.

1) None leaves all data values unadjusted.

2) Interval0-1 normalizes each feature attribute value x_i of a continuous feature x by:

$$Interval0\text{-}1(x_i) = \frac{x_i - min(x)}{max(x) - min(x)}, \tag{2.1}$$

where x_i is the i^{th} value of x. This procedure commonly applied in the software effort estimation research studies, such as in [1, 9, 39, 40], to assure the equal influence of all the features.

3) Log transforms all continuous features in a dataset to a natural logarithmic scale. This transformation procedure is suggested in a study by Kitchenham and Mendes [34] as a simple procedure to approximate a normal distribution.

Four Feature Subset Selection Methods

1) All selects all the features in a dataset.

2) Sfs (Sequential forward selection [31]) is a greedy algorithm that continually adds features one by one into an initially empty candidate subset, until there is no further improvement from the remaining features. The criteria we used to evaluate the improvement of the **Sfs** method is the MATLAB's default *objective* function of **Sfs**. This function reports the mean-square error of a simple linear regression on the training set.

3) Swvs (Stepwise variable selection - configured with bidirectional elimination [31]) iteratively adds or removes features from the feature set based on the statistical significance of a multilinear regression model in explaining the objective variable. In each iteration it builds two models that include and exclude one selected feature. The process is repeated until there is no additional improvement by either adding or removing a feature from the list of selected features. The estimated target value from **Swvs** is the regression result subject to the list of features, selected by the last step of the feature subset selection process.

4) Pca (Principal component analysis [40]) is frequently used as a feature dimension reduction method in a field of machine learning. The procedure of **Pca** applies a transformation function to the entire dataset and map it to a new lower-dimensional space, where remaining values are uncorrelated and retain the essential variance of the original dataset.

Six Similarity measures

1) Euclidean distance:
The function of the original *Euclidean* distance [35] is defined as:

$$\delta_{original\ Euclidean}(x, y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}, \tag{2.2}$$

where m is the total number of project features, and x and y are vectors representing two input cases e.g., a pair of two software project cases with m features. A lower value of *Euclidean* distance yields the lower degree of difference between x and y, and thus indicates a higher level of similarity. In software development effort estimation, input data often contain both continuous (e.g., software size) and categorical (e.g., development language) feature types; however, the original *Euclidean* distance as depicted in Eq. 2.2 can handle only continuous-type features. To compensate for cases where categorical-type features are comprised in the input cases, the *Hamming* distance is commonly combined with the original Euclidean distance [7,35]. The combined distance function between the Euclidean and the Hamming distances is defined as:

$$\delta_{Euclidean}(x, y) = \begin{cases} \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} & i \text{ is continuous} \\ 0 & i \text{ is nominal and } x_i = y_i \\ 1 & \text{otherwise} \end{cases} \tag{2.3}$$

This combined function is formally called Heterogenous Euclidean-overlap metic (*HEOM*) in other fields [35, 41], while it is commonly referred to only as Euclidean distance in the field of software development effort estimation [7]. This notion also applies to the other distance functions discussed in area of study. For example, the combined distance function between *Manhattan* and *Hamming* distances is only called as *Manhattan* in software effort estimation study [20, 22].

The following 4 similarity measures commonly discussed in the literature on ABE are also based on the Euclidean space [10, 18].

2) Manhattan distance:

$$\delta_{Manhattan}(x,y) = \sum_{i=1}^{m} |x_i - y_i| \tag{2.4}$$

3) Minkowski distance:

$$\delta_{Minkowski}(x,y) = (\sum_{i=1}^{m} |x_i - y_i|^3)^{1/3} \tag{2.5}$$

4) Maximum distance:

$$\delta_{Maximum}(x,y) = \max_{i=1}^{m} |x_i - y_i| \tag{2.6}$$

5) Mahalanobis:

$$\delta_{Mahalanobis}(x,y) = \sqrt{(x-y)'S^-1(x-y)}, \tag{2.7}$$

where S is the covariance matrix of the vector x and y.

6) Grey rational analysis (GRA):

Grey rational analysis (GRA) [42] is one of the essences of Grey system theory [43] which is a system accounted for interpreting incomplete information. The purpose of GRA is aimed for a better similarity measurement of uncertain or incomplete small datasets, which are difficult to be analyzed or modeled. By using the GRA method, a considerable accurate model can be effectively built when the available datasets can provide only a small number of project cases, which may be insufficient for some other types of model to produce a robust estimate.

The method of GRA enables a more comprehensive interpretation of incomplete data through its specific normalization procedure, which considers the difference of values across all cases and features of a dataset, instead of normalizing one feature at one time as done by many other similarity measurements. To calculate the level of similarity between a pair of input project cases x and y, GRA specifically denotes x as a new case and considers y as a small component y_a of the entire training set $Y = y_0, y_1, y_2, ..., y_n$, where $x \nexists Y$. Then, GRA normalizes the level of difference between x and y following the formula of Eq. 2.8.

$$\delta_{GRA}(x,y) = \sum_{i=1}^{m} |w_i \cdot \frac{\Delta_{min} + \xi \cdot \Delta_{max}}{\Delta(i) + \xi \cdot \Delta_{max}}|, \text{where} \tag{2.8}$$

$$\Delta(i) = \begin{cases} \delta_{Manhattan}(x_i, y_{a_i}) & i \quad \text{is continuous;} \\ \delta_{Hamming}(x_i, y_{a_i}) & i \quad \text{is nominal,} \end{cases}$$

$$\Delta_{min} = \min_{x} \min_{Y_{argmin(x)}} \Delta(i),$$

$$\Delta_{max} = \max_{x} \max_{Y_{argmin(x)}} \Delta(i)$$

Eight Solution Adaptation Techniques

Compared with an analogy-based estimation process without any adaptation applied, an effort value estimated by a process in which the estimated effort value was adjusted by any adaptation technique is on average more robust and reliable as shown by a number of studies, such as in [13, 40, 44, 45]. This is because the effort consumed in similar retrieved projects often deviate from the new project. In this study we replicate 8 solution adaptation techniques, often appeared in analogy-based estimation studies [13, 21, 36, 40, 44–46]. Note that there are other techniques that are not included in this study. The list of 8 techniques are the choices of our study because they were recently replicated in the study by Azzeh [45], in which several inconsistent behaviors among the adaptation techniques were observed. In the study of Azzeh [45], these 8 techniques were evaluated based on 7 datasets, and the author concluded that solution adaptation techniques for ABE are still a long way from reaching the optimal solutions, because their behaviors are inconsistent with those techniques that performed well in certain selected studies, but did not consistently performed well in other studies. A detailed explanation of the 8 techniques is as follows:

1) Unweighted average of the effort (*UAVG*) is a primitive adaption technique to adjust the effort value for the basic *ABE0-kNN* [6,9]. *UAVG* aggregates effort values of the selected k analogues (neighbors) by using an unweighted mean:

$$Effort(P_{new}) = Mean(\ Effort(P_{analogs})\ ), \tag{2.9}$$

where $Effort\ (P_{new})$ is the estimated effort for the new case.

2) Inverse-rank weighted mean (*IRWM*) applies different weight values to different project analogues. The weight used in *IRWM* is based on the similarity ranking between each project analogue and the new case, as the technique assumes that project analogues that are closer to the new case are more important [37]. The *IRWM* formula is depicted in Eq.2.10.

$$Effort(P_{new}) = \frac{1}{\sum_{i=1}^{k} i} \sum_{i=1}^{k} \big((k - i + 1) \times Effort(P_{analog_i})\big) \tag{2.10}$$

3) Linear size adaption (*LSA*) is based on the linear extrapolation between a software size variable of the new case and of its retrieved similar projects [13]. The software size variable is, for example, Adjusted Function Points, Raw Function Points, and Lines of code. Walkerden and Jeffery [13], based on their observations, suggested that adjusting the effort using a size variable can provide a robust estimate because size always has a strong correlation with the effort variable. Furthermore, adjusting the effort value using size allows the estimation to scale the retrieved effort value up or down to the expected size of the new case, if the new case is planned to be developed in a different size from that of its similar projects. Eq.2.11 depicts the formula of the *LSA* technique.

$$Effort(P_{new}) = \frac{SS(P_{new}) \times Mean(\ Effort(P_{analogs})\ )}{Mean(\ SS(P_{analogs})\ )}, \tag{2.11}$$

where SS indicates a single software size variable, and $Effort$ indicates an effort value. Note that in the case where a dataset consists of multiple features that indicate size, we selected the one single feature which has the strongest correlation coefficient value between it and the effort.

4) Multiple size adaptation (*MSA*) extends the *LSA* technique to handle the case in which size is described by multiple arbitrary attributes. These cases are commonly seen in web application development [36]. *MSA* aggregates multiple size variables using a mathematical mean, and then applies the mean of the size variables to the SS terms in Eq.2.11.

5) Regression toward the mean (*RTM*) calibrates the productivity of project cases in which the productivity value may not consistent with each of the other project cases [46] before applying the calibrated productivity to adjust the effort. The essential hypothesis for the productivity calibration follows the statistical phenomenon, known as *Regression toward the mean*, stating that, any extreme instance on its first measurement will tend to move toward the population mean on its second measurement. In the estimation process, projects are divided into groups by a single categorical variable, selected prior to the estimation. Then, the productivity of the similar retrieved projects are adjusted towards the average productivity among projects in the same group. The adjusted productivity is then calculated into the estimated effort by following the triangular relation of the $Effort = Size \times Productivity$ [13]. The entire formula for the RTM technique is depicted in Eq.2.12.

$$Effort(P_{new}) = \quad SS(P_{new}) \times \left(Mean(\ Pr(P_{analogs})\ ) +\ (M - Mean(\ Pr(P_{analogs}))\ ) \times (1-r) \right), \tag{2.12}$$

where M is the mean productivity of the projects in the same coherence group, and r is the correlation between the productivity of the project analogues and the actual productivity of the projects in the same group.

6) Similarity-based adaptation (*AQUA*) [40] adjusts the estimated effort value by the aggregated degree of similarity between pairs of project features of the new case and its retrieved similar projects. The degree of similarity is an inverse of distance, which is used in identifying similar projects of the new case. The aggregation formula involves the sum of the product of the normalized *Global similarity* degree as depicted in Eq.2.13. Note that the term *Global similarity* refers to the similarity between the paring of the new case and its analogues, measured based on all project features. In addition, *Global similarity* is the product of the sum of the *Local similarity*, which is referred to as the similarity between paring of the new case and its similar projects, measured based on only a single project feature.

$$Gsim(P_{new}, P_{analog_i}) = \sum_{j=1}^{f} Similarity\big(Feature_j(P_{new}), Feature_j(P_{analog_i})\big)$$

$$Effort(P_{new}) = \frac{1}{k} \times \frac{\sum_{i=1}^{k} \big(Gsim(P_{new}, P_{analog_i}) \times Effort(P_{analog_i})\big)}{\sum_{i=1}^{k} Gsim(P_{new}, P_{analog_i})}, \quad (2.13)$$

where $Gsim$ is the *Global similarity*, $Feature_j(P_{new})$ is the value of feature j of the new case, and k is the number of project analogues of the new case.

7) Adaptation based on Genetic algorithm (*GA*) proposed by Chiu and Huang [21] adopts a Genetic algorithm method to adjust the effort values retrieved from project analogues. The adaptation mechanism is similar to $AQUA$, which is based on the similarity degrees between the new case and its similar past projects. One genetic algorithm model is adopted for one single project feature to derive a suitable linear model that approximates the effort difference by *Local similarity* (i.e. the same notion as described in $AQUA$). Then GA aggregates each optimized linear models by the sum of its products into a single linear equation for adjustment, and applies the aggregated model to effort in additive form as shown in Eq.2.14.

$$e(P_{new}, P_{analog_i}) = \sum_{j=1}^{f} \beta_j \times Similarity\big(Feature_j(P_{new}), Feature_j(P_{analog_i})\big)$$

$$Effort(P_{new}) = \frac{1}{k} \sum_{i=1}^{k} \big(Effort(P_{analog_i}) + e(P_{new}, P_{analog_i})\big), \quad (2.14)$$

where $e(P_{new}, P_{analog_i})$ is an aggregation of the optimized linear models, each of which is optimized by a Genetic algorithm based on one single project feature, and β_j is the coefficient of $Feature_j$ and $Effort$ which is learned by the Genetic algorithm. According to the replicated study by Azzeh [45], this adaptation techniques is on average the most accurate.

8) Non-linear Adaptation based on neural networks (*NNet*) proposed by Li et al. [44], trains a neural network to capture the degree of difference of project features between pairs of project cases. Then the degree of difference is converted to the amount of effort difference in the adjustment process. The main benefit of adopting a neural network is that it is considerably an optimal solution for datasets which do not have underlying Gaussian distribution. The adaptation mechanism is also in an additive form the same as in the *GA* technique. The formula of the *NNet* technique is depicted as:

$$Effort(P_{new}) = \frac{1}{k}\sum_{i=1}^{k}\big(Effort(P_{analog_i}) + f(P_{new}, P_{analog_i})\big), \tag{2.15}$$

where $f(P_{new}, P_{analog_i})$ is an output value from the neural network model, trained to tell the effort difference of a pair of project cases, given their difference in project features. The main difficulty in adopting this adaptation technique is due to a very large configuration possibility for its optimization. This possibly requires many parameter optimizations such as a number of hidden layers and learning procedures. Therefore, neural network approaches in many studies were not seen to be simply replicable with the same conclusion. For example, while many replicated results from the Azzeh study [45] were in agreement with the original proposed studies, the results based on neural networks were contradicted with the original proposed by Li et al. [44].

Four Approaches to Select the Number of Analogues

The number of analogues, commonly referred to as a parameter k of ABE models, is commonly assigned as a static value or a range of static values [47, 48]. For example, Walkerden and Jeffery [13] fixed *k=1* in their experiments. Kirsopp and Shepperd observed *k=2* in [36]. Mendes et al. examined the number in a range from 1 to 3 in [37] to find the appropriate value of k. More recently, several studies such as the study by Keung et al. [8] suggested a use of a dynamic selection method to determine the appropriate k value for a given dataset. One of the accepted approaches is the Baker's [38] Best-k method, which tuned the k parameter value to fit the best with the training set and use that k value on the test set. The experiments in all studies of this thesis examine *k=1*, *k=2*, *k=3*, and *Best-k*.

2.3. Summary

This chapter explains the overview of the process of software development effort estimation and why successful estimations of software effort is very important. Then, it introduces ABE as one of the most successful effort estimation methods being increasingly used in both the industry and research communities. Following the explanations of the processes of ABE, explanations of the performance factors of ABE are presented in detail through a systematic literature reviews undertaken with over 300 papers published between January 2006 and August 2015. A summary of these papers suggests 25 approaches classified into 5 dimensions as the approaches commonly adopted with ABE model to improve its maximum achievable estimation performance. ABE models tailored with these approaches can be cross generated up to more than 2,000 combinations of model variants, where the performance of all these models will be evaluated in the later chapters of this thesis.

Chapter 3

Improvements to the Stable Ranking Method

3.1. Conclusion Instability Issues in the Literature on ABE

Conflict results regarding comparisons of multiple effort estimation methods (a.k.a. ranking instability) were an issue subject to debate in SEE research studies [2]. This debate existed because if a researcher or an industrial practitioner read the literature of available estimators, divergent conclusions regarding the performance or superiority of a selected method are commonly found. For example, Shepperd et al. [6, 29] and Mendes et al. [49] suggested that analogy-based method consistency provided higher estimation accuracy; however, Myrtveit and Stensrud [50] and Braind et al. [51] found that analogy-based method was not significantly better than other model-based methods.

If one investigates further in the literature on the selection of approaches to tailor ABE, such as similarity measures and approaches to perform solution adaptations, conflict results regarding the comparison of multiple approaches were also commonly seen. For example, our survey for the literature review explained in Chapter 2 show controversial conclusions of the choice of similarity measures. In our review, 8 out of the 302 retrieved papers provide comparison results of multiple similarity measures adopted in the case retrieval process of ABE [10,

Table 3.1. Examples of conflict results in previous studies on ABE

Studies	Results	Performance measures
Compare between ABE and other models		
Shepperd et al. [29]	ABE performed better than Linear regression and Stepwise regression.	MMRE
Mendes et al. [49]	ABE performed better than Stepwise regression.	MMRE and Pred(25)
Myrtveit and Stensrud [50]	Multiple regression models performed better than ABE.	MMRE, MdMRE, SD, and MAX_{MRE}
Braind et al. [51]	CART performed the best among ABE, Stepwise regression, Stepwise anova models.	MMRE, MdMRE, and Pred(25)
Chiu and Huang [21]	ABE was superior to the CART model.	MMRE, MdMRE, and Pred(25)
The choice of similarity measures		
Rashid et al. [52]	Euclidean distance performed slightly more accurate than Manhattan distance.	MMRE
Paikari et al. [16]	Manhattan distance performed slightly more accurate than Euclidean distance.	MMRE and Pred(25)
Liu et al. [22]	Conflict results produced among Manhattan, Euclidean, Minkowski, Maximum, and Mahalanobis distances when using different performance measures.	MMRE and Pred(25)
Khoshgoftaar et al. [18]	Manhattan distance performed better than Euclidean and Mahalanobis when feature subsets were selected by Principle component analysis.	AAE and ARE
The choice of solution adaptation techniques		
Azzeh [45]	Solution adaptation based on Neural networks were consistently the worse among 8 common techniques.	MMRE, AR
Li et al. [44]	Solution adaptation based on Neural networks outperformed 5 other common techniques, all of which were later replicated by the study by Azzeh. [45].	MMRE, MdMRE, and Pred(25)

16–18, 20–22, 52]. Rashid et al. [52] reported that the most commonly-adopted similarity measure, the Euclidean distance performed slightly more accurate than Manhattan distance when the performance was measured in terms of MMRE [24]; however, this finding was not fully in agreement with other studies where performance was measured in terms of Pred(0.25), such as a study by Paikari et al. [16]. Conflicting results as a consequence of using different performance measures were also often produced in the comparison between Euclidean and Minkowski, Maximum, and Mahalanobis distance functions such as in [17, 18, 20–22].

Among these 8 selected studies, only the study by Khoshgoftaar et al. [18] performed statistical significant tests. However, their results indicated that there was no significant performance difference between the Euclidean, Manhattan, and Minkowski distance functions. However, this study can be criticized upon inappropriate use of performance measures [24]. Therefore, a lack of empirical evidence and conclusion instability make it still be inconclusive to state which similarity measure is more appropriate for the process of ABE.

Also, in the literature on the solution adaptation, the performance of many techniques were reported variably in different studies. For example, in a recent replicated study by Azzeh [45] covering 8 most commonly adopted techniques in practice, the replicated evaluation of a technique based on neural networks produced contradictory results with another proposed study by Li et al. [44]. Azzeh also suggested based on his experimental results that there is no single best technique; most of the techniques are still a long way from reaching real optimal solutions.

The summary of these conflict results in previous studies on ABE are shown in Table 3.1. Based on these conflict results and inconsistent behaviors observed in the recent studies, we believe that a more comprehensive experimental design than that of the existing studies may be required to reach a more stable conclusion of the overall performance of ABE models and of all extensional approaches commonly adopted to improve the performance of the models

3.2. The Original Stable Ranking Methods

A recent study by Keung et al. [9] revisited the issue of conclusion instability in model-based SEE, and proposed a *Stable ranking method* as an evaluation framework to provide stable performance conclusion through a stable ranking of multiple methods. Evaluating multiple methods by using ranking, the output ranks indicate the estimation performance of an effort estimators among many other methods by means of relative performance, where higher ranks indicate higher expectable performance in general situations.

An additional important feature of the *Stable ranking method* is that it can be used to assess the stability in performance of the methods being evaluated. This can be done by analyzing the agreements between multiple rankings of methods generated by different performance measures subject to a statistical significant test. The importance of the statistical test is to determine whether the performance results are sufficiently significant to draw stable performance conclusion.

The *Stable ranking method* was developed following a guideline suggested in a study by Menzies et al. [23], where 3 conditions were suggested to be carefully controlled for a stable ranking result when undertaking experiments to compare multiple estimation methods. The 3 conditions are as given below:

- Variants of dataset instances are sufficient to draw a stable conclusion;
- The procedure to sample the training/test instances is logical and replicable;
- The performance evaluation criteria are sufficient in amount and are valid.

The use of large variants of dataset mainly contributes for obtaining a statistically significant results. In the past, an absent of the statistical test over the performance conclusion was one of the main reasons for the issue of ranking instability. Without statistical evidence, it is difficult to justify whether one method does significantly outperforms the others. The procedure to sample the training/test instances is an issue often addressed as a validity threat in empirical software engineering studies [1]. This is because, different sample methods based on different theoretical hypotheses can produce largely different results. Fortunately, even if there seems to be no single best method to fit all kinds of studies, a recent study by Kocaguneli et al. [53] was able to conclude that the leave one out is the sampling methods most suitable with SEE studies. For the last condition, performance evaluation criteria is non-trivial an important factor associated with stable conclusion. The use of biased or unreliable performance measures would made the results difficult to interpret, potentially resulting in an instability performance conclusion.

Following these 3 conditions, this evaluation framework adopted from the *Stable ranking method* is proceeded in 5 steps as depicted in Fig.3.1:

In the proposed study of the *Stable ranking method*, Keung et al. selected 11 industrial datasets from the tera-PROMISE repositories and decomposed 3 of which into 9 homogenized datasets [6]. Thus, the *Stable ranking method* was originally evaluated using 20 datasets. The next step was to sample the training/test instances. Keung et al. selected the leave-one-out approach as the sampling method mainly because that it does not rely on a random selection of training/test split [54]. The third step was to generate effort estimation methods to be evaluated. Keung et al. [9] evaluated the evaluation framework adopted following the proposed *Stable ranking method* using 90 variants of software effort predictors commonly appeared in the literature on SEE. Next, Keung et al. selected 7 error measures to be the performance measures of the proposed *Stable ranking method*. These 7 error measures were also frequently appeared in the literature on SEE [2, 6, 14, 45]. At the final step, the performance in terms of estimation error recorded from the previous step were then used as sources to perform statistical test. Keung et al. selected the win-tie-loss statistics subject to Wilcoxon rank-sum test as the statistical test method of choice. The main reason to select this method was that it is a non-parametric test that does not

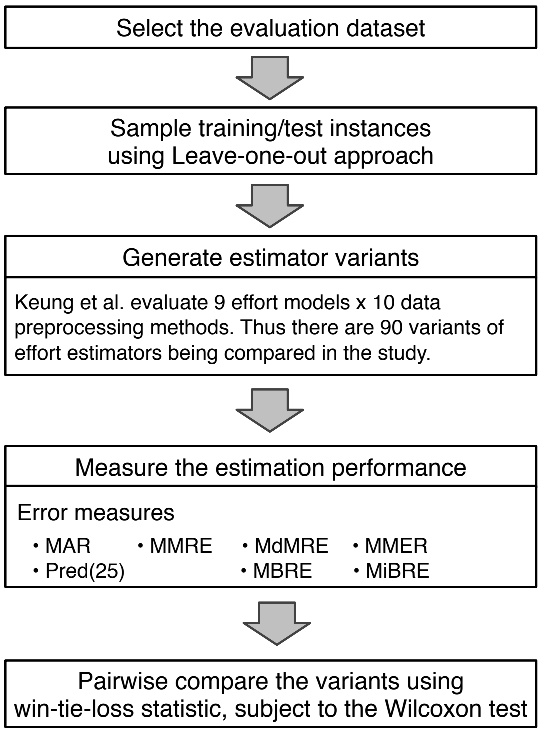
Select the evaluation dataset
Sample training/test instances using Leave-one-out approach
Generate estimator variants
Keung et al. evaluate 9 effort models x 10 data preprocessing methods. Thus there are 90 variants of effort estimators being compared in the study.
Measure the estimation performance
Error measures
• MAR • MMRE • MdMRE • MMER
• Pred(25) • MBRE • MiBRE
Pairwise compare the variants using win-tie-loss statistic, subject to the Wilcoxon test

Figure 3.1. The Evaluation Framework Adopted from the Original Stable Ranking Method

require the test input to be normally distributed.

The successful of the study by Keung et al. [9] provided an empirical evidence to that these 3 conditions are necessary to consider when a stable performance conclusions are the main goal of a study.

3.3. The Improved Stable Ranking Method

Even though there exists empirical evidence showing the success of the *Stable ranking method*, we see 2 main areas of concern being existed: the reliability of the performance measures and the power of the statistical test method selected in proposed study of the *Stable ranking method*. Elsewhere, Foss et al. [24] concerned the validity of many performance measures based on relative error, such as *MMRE* and *MMER*, the measures being used in part of the original *Stable ranking method*. Also, Mittas and Angelis [25] concerned a simple pairwise comparison, the essence of *Stable ranking method*, as it may not have sufficiently high statistical power and may have high chances of committing the statistical type I error in general situations. A more detail on theoretical discussion of the inappropriate used of this type of statistical test methods is available in a study by Zimmerman [55].

Our improvements on the performance measures are to replace the list of error measures being used in the *Stable ranking method* with more stable measures suggested by Foss et al. [24], who reviewed and evaluated many commonly-used error measures based on statistical methods. In the study of Foss et al., many well-known measures that are based on relative difference between the actual and the estimated efforts such as *MMRE* and *Pred(25)* were proofed as biased and suggested to be no longer used. We therefore dropped those error measures from our choices of performance measures, and rather used the measures guaranteed by Foss et al. [24] in multiple studies of this thesis. Fig.3.2 provides a detailed explanation of the 5 error measures selected in this study.

For the component related to statistical test methods, we replaced the test method from the Wilcoxon rank-sum test to be based on the Brunner test [56]. Wilcox [57] and Kitchenham [58] recommended the Brunner test over the Wilcoxon rank-sum test to be the non-parametric test of choice in general situations. For the main concern of the Wilcoxon rank-sum test, Zimmerman [55] pointed out

Figure 3.2. A summary of the 5 error measures selected in this study

Mean Absolute Residual *(MAR)*:

$MAR = mean(all|E_i - \hat{E}_i|)$

Standard Deviation *(SD)*:

$SD = \sqrt{\frac{\sum_{i=1}^{n}(E_i-\hat{E}_i)^2}{n-1}}$

Median Absolute Residual *(MdAR)*:

$MdAR = median(all|E_i - \hat{E}_i|)$

Relative Standard Deviation *(RSD)*:

$RSD = \sqrt{\frac{\sum_{i=1}^{n}\left(\frac{E_i-\hat{E}_i}{SS_i}\right)^2}{n-1}}$

Logarithmic Standard Deviation *(LSD)*:

$LSD = \sqrt{\frac{\sum_{i=1}^{n}\left(e_i-\left(-\frac{s^2}{2}\right)\right)^2}{n-1}}$

where E_i is the actual effort, $\hat{E}_i$ is the estimated effort, n is the total number of project cases, SS is a single software size variable, and s the estimated variance of $ln(E_i/\hat{E}_i))$

that unequal variance can greatly reduce the power of any test based on the Wilcoxon test, even if the two sample groups being tested have equal sample sizes. To avoid such invalid statistical inferences, we followed the guidelines provided by Wilcox [57] and applied the Brunner test, which can be considered as the Wilcoxon test based on the confidence interval of the *p-hat* metric. This *p-hat* metric considers a probability of random observations instead of a direct ranking comparison as done by the conventional Wilcoxon test, where the direct ranking comparison was considered as the main causes of problem suggested by Zimmerman [55]. Overall, the use of the *p-hat* metric contributes to the Brunner test as to significantly improve the statistical power in testing and reduce the chance of committing the statistical type I error, compared with the other common pairwise test methods such as the Wilcoxon rank-sum test, the main points being criticized by Mittas and Angelis [25]. In addition, Kitchenham [58] also suggested that this Brunner test is more suitable than the convention Wilcoxon test when the samples' size is small e.g., less than 300 data points, where the datasets commonly appeared in the literature on SEE are this size [2, 6, 14, 45].

We named our modified *Stable ranking method* as *Improved stable ranking* *method*. We believe that using more stable error measures subject to a more robust statistical test method will increase the validity of the experimental results, thus leading to a valid performance's conclusion of approaches being reviewed and evaluated throughout this thesis.

3.4. Application of the Improved Stable Ranking Method in the Studies of this Thesis

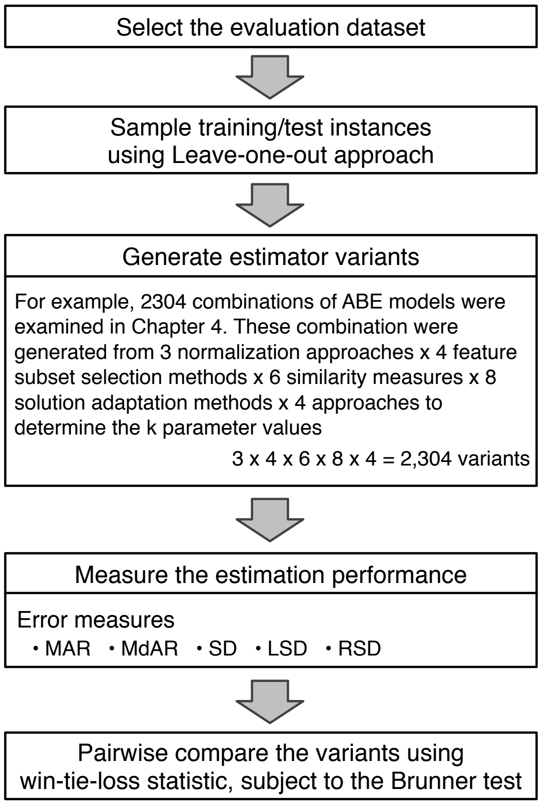
Select the evaluation dataset
Sample training/test instances using Leave-one-out approach
Generate estimator variants
For example, 2304 combinations of ABE models were examined in Chapter 4. These combination were generated from 3 normalization approaches x 4 feature subset selection methods x 6 similarity measures x 8 solution adaptation methods x 4 approaches to determine the k parameter values
3 x 4 x 6 x 8 x 4 = 2,304 variants
Measure the estimation performance
Error measures
• MAR • MdAR • SD • LSD • RSD
Pairwise compare the variants using win-tie-loss statistic, subject to the Brunner test

Figure 3.3. The evaluation procedure used in all the experiments of this thesis.

Since the original *Stable ranking method* could overcome the long standing debates of the choices of effort models, we positively believe that its 5-step evaluation procedure depicted in Fig.3.1 is sufficiently excellent. Hence, we decided to retain the 5-step procedure in our *Improved stable ranking method* and used this evaluation procedure as the main procedure for all the experiments carried out in this thesis. Fig. 3.3 shows the overview of the 5-step procedure after the core evaluation methodology is changed to *Improved stable ranking method*. The detailed explanation of each step is given as below.

3.4.1 Select the Evaluation Datasets

Table 3.2 lists 12 industrial software development project datasets used in all the studies of this thesis. Of which, 9 datasets are available in the tera-Promise software engineering repository (Accessible through [67]), where datasets are made available and commonly used in software development effort estimation studies [2,6,9,14]. The 3 other datasets are subsets of the standard benchmark ISBSG dataset (Release 9) [62], which is a large and divergent collection of industrial software projects. These 3 datasets namely ISBSG-banking, ISBSG-insurance, and ISBSG-communication were selected as subsets from the ISBSG following the policy used to generate the ISBSG-banking datasets in a recent prominent study of Kocaguneli et al. [14] and the guideline to interpret the ISBSG dataset suggested by Mendes and Lokan [68]. Specifically, the criteria to select the ISBSG-banking are as follows:

- Project cases indicated both organization type and business type as banking.
- Project cases were being rated A in both terms of data quality and UFP integrity.
- Project features that had more than 40% of their values missing were excluded.
- Project features contained redundant information were excluded.

For the ISBSG-insurance, we used the criteria as described above to select project cases having both organization type and business type described as insurance projects. For ISBSG-communication, we selected project cases having organization type and business type indicated as communication and telecommunication, respectively.

Table 3.2. The 12 industrial datasets consisting of 582 software project cases (F = number of features, and N = number of project cases)

	Datasets	F	N	Description	The effort variables				
					Unit	Min	Mean	Median	Max
1	Albrecht	7	24	Completed projects from IBM in 70s [59]	months	1	22	12	105
2	Cocomo-sdr	22	24	Projects from various companies in Turkey [60]	months	2	32	12	342
3	Cocomo81	18	63	Nasa software projects [30]	months	6	684	98	11400
4	Desharnais	12	81	Well-recognized Canadian software project [6]	hours	546	4834	3542	23940
5	Finnish	8	38	Data from different companies in Finland [61]	hours	460	7678	5430	26670
6	ISBSG-Banking	14	29	Banking projects of ISBSG datasets [62]	hours	690	7030	3034	36046
7	ISBSG-Communication	17	69	Communication projects of ISBSG datasets [62]	hours	31	4561	2412	52172
8	ISBSG-Insurance	17	36	Insurance projects of ISBSG datasets [62]	hours	354	5174	2368	51527
9	Kemerer	7	15	A small data set from large business [63]	months	23	219	13	1107
10	Maxwell	27	62	Projects from commercial banks in Finland [64]	hours	583	8223	5190	63694
11	Miyazaki94	8	48	Software project developed in COBOL [65]	months	7	88	38	1586
12	Nasa93	18	93	Nasa software projects [66]	months	8	624	252	8211
		Total	582						

If one reads the literature on SEE, the use of homogenized datasets (a.k.a., decomposed dataset) from the tera-Promise source are often encountered [1, 2, 6, 9, 14, 53, 69]. However, we decided to use only full datasets from this source in all the experiments of this thesis because, in the majority of our experiments, we summarized the overall results by aggregating the performance results across all datasets. In this way, the use of both full and homogenized datasets at the same time will mislead the performance conclusion based on the win-tie-loss statistic. To explain in short, the win-tie-loss statistics used throughout this thesis is based on counting. Its procedure counts the number of times a test variant significantly outperforms the other test variants regrading all performance measures and all datasets used in the experiments. Interpreting this statistic with both full and homogenized datasets would make them counts the same samples twice, and resulting in wrong performance conclusion produced.

3.4.2 Sample the Training/Test Instances

All the studies of this thesis used leave-one-out approach [53, 54] to sample the training/test instances for the performance evaluation. For a dataset with n project cases, one project at a time becomes a test instance of a model built from all the remaining *n-1* cases. The estimated effort values for n different training/test instances will produce n estimated effort outcomes. In contrast to its alternative choice of N-way cross-validation approach, the estimation outcomes produced by the leave-one-out will be the same in every single run because it does not depend on any randomization. A recently study by Kocaguneli et al. [53] systematically compared the leave-one-out and the N-way cross-validation approaches for SEE studies and recommended the leave-one-out. The superiority of the leave-one-out for SEE studies is that it often generates the lower estimation bias and is more robust when experimenting on small and medium-sized datasets [53, 70] (i.e., less than 300 samples), where most of the available effort datasets are within this range [67].

3.4.3 Generate Estimator Variants of ABE Models

As suggested by Keung et al. [9] and Kocaguneli et al. [69], a performance evaluation based on an aggregation of multiple configuration variants of an estimation model will offer a better generalization of results. In Chapter 4, we generated 2,304 variants of ABE models based on a combination of approaches commonly adopted in 5 essential components of ABE (i.e., feature normalization, feature subset selection, similarity measures, solution adaptation, and number of analogues), and had them evaluated with the *Improved stable ranking method*. In Chapter 5, we selected only more successful approaches determined in Chapter 4 and had them reviewed with our proposed new approach. In short, we generated and reviewed 36 variants of ABE models in Chapter 5. Next, in Chapter 6, we evaluated the single best variants of ABE models determined in Chapter 4 and Chapter 5 (we named it ABE-Best) with 6 additional data quality improvement techniques. Thus, we generated and reviewed 7 models in Chapter 6. In the final chapter, we evaluated the ABE-Best model with 7 other variants of machine learning model-based effort estimators cross generated with 4 feature subset selection methods $\times$ 3 normalization approaches. Thus, in Chapter 7 we generated and reviewed 85 variants of SEE models in total.

3.4.4 Measure the Estimation Performance

Estimation errors are the commonly used performance measures in software development effort estimation studies [9,14,24,69,71]. Suggested by Keung et al. [9], an agreement established among many error measures would greatly contribute towards a more intuitive and more trustworthy performance's conclusion. Hence, for performance evaluation of the configuration variants of ABE models examined throughout this thesis, we selected 5 error measures from a list of more stable measures suggested by Foss et al. [24], as described earlier in Fig.3.2 of this chapter.

3.4.5 Pairwise Comparisons of the ABE Variants using Win-tie-loss Statistics, Subject to the Brunner Test

In our experiments, the overall performance of a single combination of approaches commonly tailored ABE models is determined by the number of times it statistically outperforms the other variants or being outperformed by the other variants, aggregated from all of its possible pairwise comparisons. This performance evaluation approach is known as *win-tie-loss* statistics [2,9,69]. The common criteria to determine whether a combination statistically outperforms or being outperformed other combinations is the Brunner test [56]. Fig. 3.4 describes the procedure of the *win-tie-loss* statistics based on the Brunner test.

Figure 3.4. Comparing the performance between two configuration variant i and j of an ABE model, on their single error measure Err_i and Err_j. Lower values indicate better performance for all the 5 error measures selected in this study.

```
1: $win_i = tie_i = loss_i = 0$
2: $win_j = tie_j = loss_j = 0$
3: if Brunner test($Err_i, Err_j, 0.95$) says they are the same then
4:     $tie_i = tie_i + 1$
5:     $tie_j = tie_j + 1$
6: else
7:     if better($Err_i$,$Err_j$) then
8:         $win_i = win_i + 1$
9:         $loss_j = loss_j + 1$
10:    else
11:        $win_j = win_j + 1$
12:        $loss_i = loss_i + 1$
13:    end if
14: end if
```

Using the statistics, the results are measured through three counters, *wins*, *ties*, and *losses*. The statistic counts the results of pairwise comparisons of all pairs of ABE variants for each error measure. The procedure shown in Fig. 3.4 tests one pair of techniques for one error measure, and for example, it will test all pairs of 2,304 combinations and all error measures at the test completion in the study of Chapter 4. When the Brunner test shows that the estimation performance of a pair of ABE variants are not significantly different i.e., p-value > 0.05, the *tie* counters of these two variants will increment. Otherwise, the *win* counter of the variant with lower error will increment, and the *loss* counter of the other variant will increment. If a variant i totally outperforms a variant j in terms of all 5 error measures, its total wins will increase by 5, while the total losses of variant j will increase by 5.

Interpretation of the Win-Tie-Loss Statistics

The overall estimation performance based on the *win-tie-loss* statistics is commonly presented as the total counts of *wins*, *losses*, or *wins-losses* (i.e., wins minus losses). According to Demšar [72] and Kocaguneli et al. [2], there is no generally accepted method among these three counters. In this thesis, our comparison results interpreted *wins - losses* as the main results because their values can obviously distinguish high-performed and poor-performed variants of estimation models. That is the variants having the value of *wins-losses* less than 0, i.e., *losses* more than *wins* are non-trivial a poor-performed ABE variants. For the other two counters, we interpreted them as measures to observe robustness of the ABE variants being observed. Specifically, robustness was observed by that ABE variants with high *wins* and low *losses* are more robust than other variants with high *wins*, but their *losses* are also high. Finally, we summarized results based on all the three counters to observe the overall ranking stability. This is done by calculating the average ranks of the ABE variants being observed across the three ranking lists in regards to the three different counters.

3.5. Summary

Conclusion stability has continually been one of the major problems in SEE research studies [2], where divergent results were frequently generated from different replications of the same methods. The studies of this thesis revisit this problem and suggest improvements to a method named *Stable ranking method*, originally proposed by Keung et al. [9], to be the evaluation method of choice for assessments of estimation models used in the activities in software engineering processes, such as SEE. Two main suggestions for improvements of the original *Stable ranking method* are in two parts: the performance measures and the statistical test method, where the measures and the test method adopted in the original *Stable ranking method* can be questionable upon their validity. Explained in details, we selected 5 performance measures from the list of more robust and reliable measures evaluated by Foss et al. [24] and selected the statistical test method that is assumed to be most suitable for the distribution and the characteristics of the commonly used effort estimation datasets, following the recommendations by Wilcox [57] and Kitchenham [58]. The successful evidence of adopting this evaluation method will be presented from the next chapter.

Chapter 4

Identifying Best Combinations of Approaches in the 5 Essential Components of ABE

The motivation of this study is that we frequently encountered unstable conclusions and conflicting results in many existing research studies in ABE [18,44,45]. We speculated that a various use of evaluation methodologies undertaken among existing works were the major causes of conclusion instability. For example, we found that biased performance measures such as *MMRE* [24] were chosen in many existing studies. Furthermore, a massive number of studies that relied on such bias performance measures rarely carried out statistical significant test to validate their results. Therefore, without a more robust and reliable evaluation method being able to draw a stable performance conclusion exists, the problem related to conclusion instability and conflicting results will be continually reproduced in this area of study.

In this chapter, we evaluated our *Improved stable ranking method* to show that the method is very reliable and should be used in an evaluation of any future studies in this area of research. At the same time, we sought for the superior approaches from the existing approaches in the 5 essential components of ABE.

The 2,304 combination of approaches evaluated in this study are all the possible combinations being cross generated from:

- 3 Normalization approaches: **None**, **Interval0-1**, **Log**;
- 4 Feature subset selection methods: **All**, **Sfs**, **Swvs**, and **Pca**;
- 6 Similarity measures: **Euclidean**, **Manhattan**, **Minkowski**, **Maximum**, **Mahalanobis**, and **GRA**;
- 8 Solution adaptation techniques: **UAVG**, **IWRM**, **LSA**, **MSA**, **RTM**, **AQUA**, **GA**, and **NNet**;
- 4 Approaches to determine the number of analogies: **k=1**, **k=2**, **k=3**, **Best-k**.

4.1. Research Questions and Hypothesis

In this study, experiments and analyses of results are organized to answer the following research questions:

RQ1 How stable is the ranking list of 2,304 combinations of ABE variants identified by the *Improved stable ranking method*?

RQ2 Which approaches could be concluded as being superior than other approaches in the same components of ABE models?

For RQ1, even if we improved the Keung et al.'s *Stable ranking method* [9] with a more robust and reliable statistical test method, we may still need empirical evidences to show that our *Improved stable ranking method* can provide stable results and conclusions. For RQ2, after we validated the *Improved stable ranking method*, the results and analyses based on it would provide a better comprehension of the approaches selection to tailor ABE models in practice. An available of this knowledge will be an important step towards better facilitate an effort estimation framework based on the concept of ABE.

The primary hypotheses of this study are given below.

H1: With thorough comparisons of the approaches commonly tailored to ABE models to improve the overall performance, undertaken by a comprehensive evaluation method, we will be able to assess a stable ranking list of ABE variants being examined.

H2: If we are able to rank the combinations of approaches for ABE correctly, we would be able to observe similar behavior between approaches being ranked closely. For example, solution adaptation techniques with similar adaptation mechanisms should produce similar estimation performances.

4.2. Evaluation Methodologies

The evaluation procedures of the study of this chapter adopted the procedure explained in Chapter 3 to evaluate the 2,304 variants of ABE models, aforementioned earlier in this chapter. After all the 2,304 variants of ABE models were executed using the leave-one-out approach, their performance were recorded with 5 error measures. Then, pairwise comparisons were carried out between each of the 2,304 variants × 2,303 others, using the *win-tie-loss* statistics. Finally, the overall performance of each variant was aggregated by the total summations of *wins*, *losses*, and *wins-losses* across all the selected datasets and the 5 robust error measures.

To justify the validity of an issue possibly introduced when any pairs of the approaches are being dependent with each other (i.e., not being fully orthogonal), we design the methodology of this study, which is the fundamental part of this thesis, to exhaustively examine all the possible combinations of the selected approaches. Adopting the evaluation procedure based on use of multiple rankings followed by an analysis based on them in a manner of ranks agreement, we strongly believe that the any validity issue regarding approaches in different dimensions are being dependent will not be introduced.

4.3. Result

Fig.4.1 shows box plots of the ranking based on *wins-losses* of the approaches compared in this study. Each plot shows the approaches belong to the same essential components of ABE models. From this figure, we can indicate superior approaches belong to feature subset selection, number of analogues, and solution adaptation. On the other hand, we cannot see any single approach performed differently than others in the other 2 components. In our view, their are two

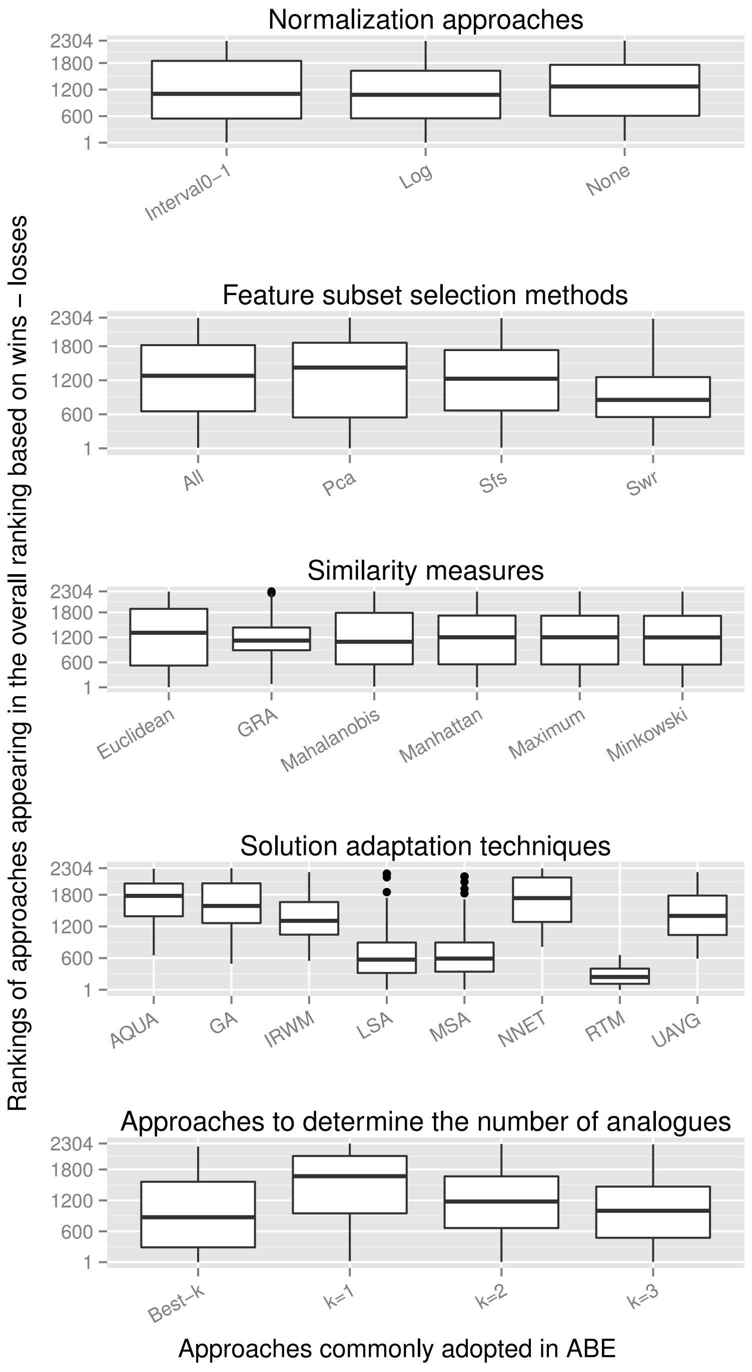

Figure 4.1. Box plots of the ranking of the approaches selected in this study, appearing in the overall ranking of the 2,304 combinations in terms of *wins-losses*

possible causes of this situation: (1) the *Improved stable ranking method* may not sufficiently robust to draw stable conclusion, or (2) there are no actually best approaches in these 2 components such that all the existing approaches belong to these components are sensitive to shape, size, and distribution of local datasets.

4.3.1 Investigating Research Question 1

The first research question examines how robust is the *Improved stable ranking method*. To answer this research question, we selected the approaches belong to solution adaptation techniques and feature subset selection methods to perform a more detailed analysis of the ranking results. Solution adaptation was selected as a component with strongly conclusive ranking results, and feature subset selection was selected as a component with somewhat uncertain performance conclusion. Regarding this selection of components, if the performance ranking of this analysis was conclusive, the results would be a strong evidence to support the stability and robustness of the *Improved stable ranking method*.

Our analyses for this research question are in two-fold. The first is to observe the ranking agreement across the three ranking lists based on *wins*, *losses*, and *wins-losses*. Then, we discussed whether solution adaptation techniques with similar theoretical concepts tended to be in very close ranks.

Ensure the stability of the extended ranking method based on analyses of ranking agreement

Table 4.1 shows the ranking of solution adaptation techniques in combinations with feature subset selection methods based on all the three terms of *wins*, *losses*, and *wins-losses*, along with the average ranks summarized by all these three ranking lists.

We divided the ranking of 32 combinations of solution adaptation techniques $\times$ feature subset selection methods into 4 bands based on mean ranks and the values of *wins-losses*. We decided to define the cut off between band #1 and band #2, and between band #2 and band #3 based on the significantly different values of *wins-losses*. For the cut off between band #3 and band #4, we decided it based on the values of mean rank. Observing the 4 bands of solution adapta-

Table 4.1. Win-tie-loss ranking results from the experiments. Results for each combination of configuration variants of feature selection methods and solution adaption techniques are sorted by *wins - losses*.

Rank	wins-losses	Adapt	FSS	wins	loss	wins-losses	mean rank
1	2482286	RTM	Pca	1	1	1	1.0
2	2173850	RTM	Swvs	2	2	2	2.0
3	1931752	RTM	All	3	3	3	3.0
4	1870382	RTM	Sfs	4	4	4	4.0
5	1213196	LSA	Pca	5	5	5	5.0
6	1182229	MSA	Pca	6	6	6	6.0
7	997004	MSA	Swvs	8	10	8	8.7
8	991503	LSA	Swvs	7	11	7	8.3
9	974632	LSA	All	9	7	9	8.3
10	925706	MSA	All	11	8	11	10.0
11	887245	LSA	Sfs	10	12	10	10.7
12	826047	MSA	Sfs	12	13	12	12.3
13	156078	IRWM	Swvs	14	9	14	12.3
14	-60361	UAVG	Swvs	15	14	15	14.7
15	-139083	GA	Swvs	13	18	13	14.7
16	-580587	AQUA	Swvs	21	17	21	19.7
17	-581496	IRWM	Sfs	16	21	16	17.7
18	-596759	NNET	Swvs	24	15	24	21.0
19	-688553	UAVG	Sfs	17	22	17	18.7
20	-741886	IRWM	All	18	23	18	19.7
21	-808048	UAVG	All	19	27	19	21.7
22	-839483	IRWM	Pca	20	25	20	21.7
23	-889918	UAVG	Pca	23	24	23	23.3
24	-999817	GA	Sfs	22	30	22	24.7
25	-1094188	GA	All	25	31	25	27.0
26	-1165733	AQUA	Sfs	28	26	28	27.3
27	-1176339	AQUA	Pca	27	28	27	27.3
28	-1183944	NNET	Sfs	31	16	31	26.0
29	-1215889	NNET	All	30	20	30	26.7
30	-1251847	AQUA	All	29	29	29	29.0
31	-1269609	GA	Pca	26	32	26	28.0
32	-1328370	NNET	Pca	32	19	32	27.7

tion techniques × feature subset selection methods, we could see that the ranks were more likely changed within the same bands, while they were very consistent across different bands. For example, the top 4 combinations of Table 4.1 were consistently ranked between rank #1 and rank #4 in all terms of *wins*, *losses*, and *wins-losses*. Also for the second band, only the *losses* value of (**MSA**, **Sfs**) was exceeded rank #12, and all the combinations in this band consistently had their mean ranks between rank #5 and rank #12.

For remaining two bands i.e, rank #13 to rank #32, we found only 3 out of 60 ranking values of *wins*, *losses*, and *wins-losses* were not ranked in the within their own bands. Based on this preliminary finding, we can suggest that the ranking list of Table 4.1 was very stable and a further analysis based on it would be able to draw performance conclusion of solution adaptation techniques.

Observe whether approaches with similar theoretical concepts tended to be ranked closely

Solution adaptation techniques are the component ABE being most appropriate for this analysis because many techniques compared in this study were clearly based on similar theoretical concepts. Table 4.2 shows a summary of the properties of the 8 solution adaptation techniques, examined in this study.

From Table 4.2, it is clear that the theoretical concepts of **UAVG** and **IRWM** are more similar than that of the other methods. Also, **LSA**, **MSA**, and **RTM** were based on linear adaptation and made use of software size (and productivity) in their adaptation procedures. For **GA** and **NNet**, they are the only two techniques that adopt machine learning models for the adaptation. For **AQUA**, even if it does not apply any machine leaning method in its adaptation procedure, the concept behind **AQUA** are similar to techniques that based on applying machine learning models such as **NNet**. Based on this brief summary of the theoretical concepts of the 8 solution adaptation techniques, results in Table 4.1 clearly indicated that adaptation techniques with similar theoretical concepts were ranked very closes to each others in terms of *wins-losses* as well as of the average ranks summarized across *wins*, *losses*, and *wins-losses*. Important and interesting results for here are that the conclusive ranks within the same bands shown in Table 4.1 strongly suggested that similarity measures with similar theoretical concepts

Table 4.2. The 8 adaptation techniques selected in this study.

Abbrev.	Adaptation Techniques	Adjustment function	Adjustment feature
1 UAVG [73]	Unweighted mean of the k analogues	Mean	Effort
2 IRWM [37]	Inverse-rank weighted mean of the k analogues	Mean	Effort
3 LSA [13]	Linear size adaptation	Linear	Software size
4 MSA [36]	Multiple size adaptation	Linear	Software size(s)
5 RTM [46]	Regression towards the mean	Linear	Software size
6 AQUA [40]	Similarity-based adaptation	Linear	All features
7 GA [21]	Adaptation based on Genetic algorithm	Linear	All features
8 NNet [44]	Non-linear adaptation based on Neural network	Non-linear	All features

were more likely to be in the same bands, where the bands were assigned based on both performance ranking and ranking stability.

In summary the results of two analyses carried out to answer this research question were sufficient to allow us to say that the *Improved stable ranking method* is sufficiently robust to draw a stable conclusion on performance of multiple test samples, if it does exist.

4.3.2 Investigating Research Question 2

The second research question examines which approaches commonly adopted in ABE models are more superior than others. From the experiments and analyses of the previous research question, **Best-k** and **RTM** were the conclusively superior approaches for ABE, whereas the choices for approaches belong to other components were remaining inconclusive. To suggest the approaches of choice for those inconclusive dimensions, we further investigated the 2,304 combinations of approaches by focusing only conclusively more superior combinations. The rationale behind the investigation in this way is that, if we can correctly identify which combinations of the approaches are significantly superior than others, in practice, only those superior configurations of approaches would be more likely configured and used for the effort estimation process. Since we had been able to ensure correctness and robustness of the *Improved stable ranking method*, we can therefore say that ranking of these 2,304 combinations were also being accurate and valid.

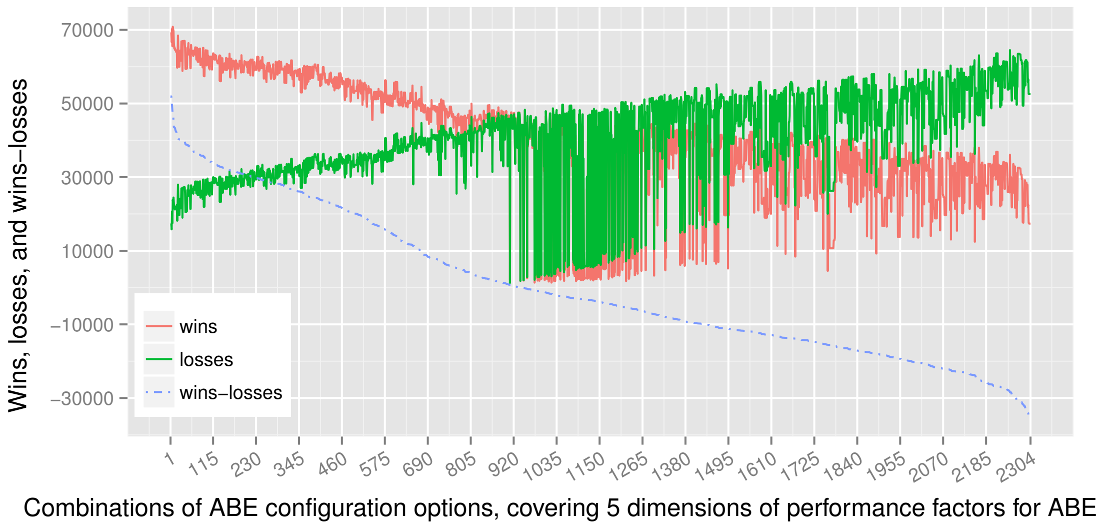

Figure 4.2. The sum of *wins*, *loss*, and *wins-losses* values of 2,304 combinations of common approaches selected in this study (over all error measures and all datasets).

.

The experiment for this investigation began with an identification of the significantly superior configurations of approaches. Fig.4.2 shows line plots of the 2,304 combinations. The x axis of this figure indicates the ranking of combinations of approaches sorted by *wins-losses* in ascending order. The y axis indicates the counts of *wins*, *losses*, and *wins-losses*. Based on Fig.4.2, we used the following criteria to consider the combinations falling on the top 5% i.e, ranked #1 to ranked #115, as high performers and the bottom 5% i.e, ranked #2,185 to ranked #2,304, combinations as low performers:

- All the combinations ranked top 5% have *wins* twice as higher than *losses*.

- All the combinations ranked bottom 5% consistently have *losses* higher than *wins*.

To ensure whether the approaches falling the top 5% of the ranking list of Fig.4.2 are definitely superior than others, we observed them with the bottom 5% combinations. Table 4.3 shows frequency counts per approach when they appeared in the top 5% and the bottom 5% of the ranking based on *wins-losses*. The frequency counts strongly suggested **Best-k** and **RTM** as approaches of choices for solution adaption methods and approaches to determine the number of analogue. These results are in broad agreement with the results of Fig.4.1. Furthermore, the frequency counts could suggest that **Log** would be an approach of choice for the feature normalization. This is because the counts clearly indicated that **Log** was the most frequently appeared normalization approach on the top 5% combinations, and it is also the least frequently appeared approach at the bottom 5% combinations.

For feature subset selection methods and similarity measures, the results of Table 4.3 indicate that **Pca** and **Euclidean** performed somewhat better than other approaches. **Pca** performed better than others with the top 5 % combinations; however, it also frequently appeared in the bottom 5% combinations. For **Euclidean**, its performance was not clearly different than others for both the top 5% combinations not the bottom 5% combinations. In our opinion, we would suggested **Pca** to be approaches of choice when the ABE framework being used was configured with all the other conclusively superior approaches suggested by our earlier experiment, such as **RTM** and **Best-k**. For other situations, such as for a study that proposes any new approach, we suggest to evaluate the new proposed approach with the entire list of feature subset selection methods.

One the other hand, to suggest a similarity measures of choice, the results of this experiment suggested that there was actually no single best similarity measure out of the 6 commonly adopted measures. Hence, we would suggested to continually adopt the **Euclidean** distance to be the similarity measure for ABE as long as there is no new measure based on better theoretical concepts available. In summary, our suggestion for the approaches of choice to be adopted practical ABE models are as follows:

- Normalization approach: **Log** [34];

Table 4.3. Frequency counts of approaches belong to the 5 essential components of ABE, appearing at the top 5% ranks and the bottom 5% ranks of Fig.4.2

Approaches	# appeared in top 5% ranks	# appeared in bottom 5% ranks
Normalization		
Log	60	18
Interval0-1	45	72
None	9	20
Feature subset selection		
Pca	51	24
Swvs	30	8
Sfs	7	41
All	26	43
Similarity measures		
Euclidean	32	24
Mahalanobis	25	16
Minkowski	18	15
Maximum	18	15
Manhattan	18	15
GRA	3	6
Solution adaptation		
RTM	73	0
MSA	22	0
LSA	19	2
UAVG	0	2
IRWM	0	2
Nnet	0	44
GA	0	29
AQUA	0	26
Number of analogues		
Best-k	68	4
k=2	18	33
k=3	16	10
k=1	12	50

- Feature subset selection:
    - **Pca** [40], if all the approaches suggested in this list were selected;
    - otherwise, fit a local dataset with all the commonly adopted approaches;
- Similarity measure: **Euclidean** [7, 28];
- Solution adaptation technique: **RTM** [46];
- Approach to determine number of analogues: **Best-k** [38].

4.3.3 Sanity Check

Since this study is the fundamental part of the thesis, we carried out a sanity check as an extended experiment to evaluate the rational of the results. In this extended experiment, we repeated the experiment of our second research question but fixed the best approaches in the dimensions of the parameter k and the solution adaptation technique as **RTM** and **Best-k**, which had been earlier concluded as the best approaches in preliminary experiment of this study. Hence, combinations of approaches examined in this extended experiment are in total of $1 \times 1 \times 3 \times 4 \times 6 = 72$ combinations, being cross generated from:

- 1 Solution adaptation techniques: **RTM**;
- 1 Approaches to determine the number of analogies: **Best-k**;
- 3 Normalization approaches: **None**, **Interval0-1**, **Log**;
- 4 Feature subset selection methods: **All**, **Sfs**, **Swvs**, and **Pca**;
- 6 Similarity measures: **Euclidean**, **Manhattan**, **Minkowski**, **Maximum**, **Mahalanobis**, and **GRA**;

Using the same procedure to generate Fig.4.2, Fig.4.3 shows line plots of the 72 combinations of approaches as described above. The x axis of this figure indicates the ranking of combinations sorted by *wins-losses* in ascending order. The y axis indicates the counts of *wins*, *losses*, and *wins-losses*. Since the distributions of the *wins*, *losses*, and *wins-losses* are different from that of Fig.4.2 due to different

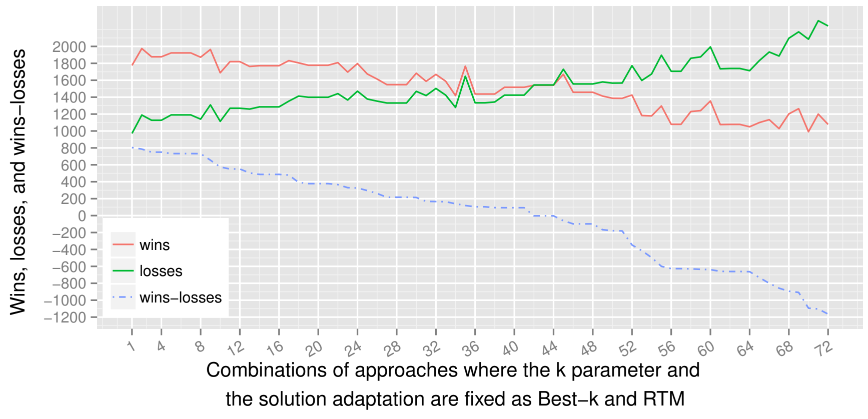

Figure 4.3. The sum of *wins*, *loss*, and *wins-losses* values of 72 combinations of approaches (over all error measures and all datasets).

.

sample sizes, the use of percentage to define the cutoff between the top and the bottom does not seem to be fully appropriate. Hence, we decided to change the criteria for this cutoff at the top ranks to be the point where both *wins* and *losses* suddenly significantly increases after they have continually dropped, and for the bottom ranks, we decide that point to be where both *wins* and *losses* suddenly significantly drops after they have continually increased. In this way, the sudden changes of the *wins*, *losses* may imply these points to be where the rankings in terms of *wins*, *losses*, and *wins-losses* start to exhibit less agreement (i.e., less stable ranking). Specifically, to perform the top-bottom analysis, we defined the 9 combinations having the highest values of *wins-losses* as the top ranks, and the 12 combinations having the lowest values of *wins-losses* as the bottom ranks.

Based on this criteria, the frequency counts of approaches classified into the top and the bottom ranks, respectively, are presented in Table 4.4. Interestingly, the results obtained from this table appear to be in broad agreement with that of all the other experiments presented earlier in this chapter. That is, **Log** was the non-trivial approach of choice to perform feature normalization. Both **Pca** and **Swvs** were endorsed by the *Improved Stable Ranking Method* but they did not have a significantly different performance. Also, all the similarity measures compared in this study shown very similar performance, where **Euclidean** can be suggested as the one performed slightly better than the others. In summary, we found no conflict between the earlier experiments and this extended experiment, as **Log** and **Euclidean** are suggested for SEE based on ABE. In additional to the somewhat uncertain conclusion on the performance of the feature subset selection methods, we suggest to follow the results from the earlier experiments, where **Pca** was recommended.

Table 4.4. Frequency counts of approaches appearing at the top ranks and the bottom ranks of Fig.4.3

Approaches	# in the top ranks	# the in the bottom ranks
Normalization		
Log	6	2
Interval0-1	3	3
None	0	7
Feature subset selection		
Pca	4	3
Swvs	4	3
Sfs	0	4
All	1	2
Similarity mesures		
Euclidean	2	0
Maximum	2	1
Manhattan	2	1
Minkowski	2	2
Mahalanobis	1	0
GRA	0	8

4.4. Discussion

4.4.1 The Improved Stable Ranking Method

The results achieved from the comprehensive experimental procedure used in our study show that we can overcome the ranking instability issues [9] in ABE research studies. The successful of this study allows us to identify the superior approaches in 4 out of the 5 essential components of ABE models, which are (1) normalization techniques, (2) feature subset selection methods, (3) solution adaptation techniques, and (4) number of analogues. The stability of the ranking result indicated by the *Improved stable ranking method* was ensured by a broad agreement established across multiple stable error measures that were used to assess all the 2,304 combinations of the common approaches adopted belong to these 4 essential components of ABE models.

According to the study by Keung et al. [9] that overcame ranking instability issues in model-based effort predictions, Keung et al. suggested many factors that would make changes to ranking of the effort estimators. The key factors are (1) the use of different evaluation methodologies among different literatures (e.g. using different sampling methods to generate the training/test set), and (2) an inappropriate experimental design (e.g. the commonly seen misuse of a biased *MMRE* error measure [24] and the absence of the statistical significant test).

From the analyses of the results in this study, we agree with the suggestion by Keung et al. [9] that the use of a more superior evaluation method that mainly considered the amount and diversity of datasets, amount and robustness of performance measures, the robustness of statistical test method, and the agreement among all measurements being used are the key success factors of this study.

The following in this section, we discuss successful approaches along with opportunities of their future direction for the 5 essential components of ABE models.

Feature Normalization Approaches

Since both **Interval0-1** and **Log** showed higher performance than the **None** method in Fig.4.1, this result suggested that feature normalizations are an important preprocess of ABE. Between these 2 approaches, the results of 4.3 suggested that **Log** performed better when all the other superior approaches were being adopted. Regarding this finding, we recommend **Log** over the other approaches in practice.

In our opinions, **Log** is considerably theoretically superior than the other two approaches because it can also approximate a normal distribution in its transformed space. However among the approaches examined in the experiment, the use of **Log** normalization was much less discussed in contexts of ABE than all the other approaches. We hope that our results of this study would encourage future studies to explore more on the application of **Log** transformation in the process of ABE.

Other interesting future research direction for ABE with regard to data normalization and data transformation is to further explore on the adoption of data quality improvement techniques. In our opinion, research studies in this direction are as important as a search for the superior effort models because it is difficult to accurately model noisy and inconsistent data.

Similarity measures

This study evaluated 6 similarity measures appeared in more than one research papers in IEEE xPlore and ACM digital library during this 10 years. The results from the experiments and the analysis of results suggested that there was no significantly difference between **Euclidean**, **Minkowski**, **Maximum**, **Manhattan**, and **Mahalanobis** measures. We further investigated for the reasons to explain why our robust evaluation methods showed that these measures were no significantly different. In a study by Wilson and Martinez [35], the authors noted that the theoretical concepts of these 5 similarity measures were almost the same, as all of which are corresponding to the Euclidean space. Since our results suggest no significant difference between these measures, and the **Euclidean** has been continually adopted since ABE was introduced [6, 7], we therefore suggest to continual use the **Euclidean** for future studies and in practice, as long as there is no totally new approaches adopted from a new theoretical concept available with a consistent empirical evidence.

For **GRA**, despite no review study being available at the time of writing, we see that its functions can also be considered as an **Euclidean** distance with a normalization function applied to its formula. However, it performed consistently less accurate than the combination between **Euclidean** and **Log**. Other important note in regards to **GRA** is that, even if **GRA** performed accurately in other area of studies, it did not performed successfully with ABE. One of the possible reasons is related to the somewhat unique characteristics of SEE datasets. Kocaguneli et al. [1] commented on SEE datasets that sparse data is one of the main reasons that successful approaches in other fields did not necessarily perform successfully with SEE datasets. This points out the same direction of future study to consider more on the data attributes, the same direction that we discussed earlier with the future directions for the normalization approaches.

Solution adaptation techniques

Our results clearly endorsed **LSA**, **MSA**, and **RTM** as superior solution adaptation techniques than other techniques. The common theoretical concept of these 3 techniques are that they make use of the software size variable in their adaption procedures following the triangular relationship between size, productivity, and effort: $Effort = Size \times Productivity$ [30]. This mean that these 3 successful solution adaptation techniques exploit the productivity in their adaption procedure by calibrating the productivity of the new case before adjust the effort. For a future research direction of solution adaptation techniques, our results are encouraging to suggest a more detail analysis and further exploration of productivity variable in solution adaptation process as well as in any other processes of ABE.

Methods to determine the number of analogues and Feature subset selection methods

The **Best-k** method was also strongly recommended by several studies by Kocaguneli et al. such as [2,14] to be an approach of choice. Compared with methods that statically assign the number of analogues to local datasets such as $k = 1$ or $k = 3$, $Best-k$ is far more theoretical superior. Our results of this study provided an empirical evidence to conclude that an analysis of local datasets to find the appropriate number of k using the $Best-k$ method will produce a better estimation performed ABE model. In our view, one concern of adopting the $Best - k$ method in practice is that it is considerably more compute intensive than any method that heuristically assigns the number of analogues to local datasets. In a very large dataset, the search space of the best k value will become very large, and will demand powerful technologies to facilitate the search.

A demand of more powerful technology may also be addressed in future to conduct a search for the most fit feature subset selection method to a local dataset.

The results of Fig.4.1, Table 4.1, and Fig.4.2 indicated that there was no single best feature subset selection method and a search of methods that would performed best for a given local dataset should be performed in every single run. Hence, we see that the future direction for these two dimensions are considerably in common. That is they both demand more powerful technologies to facilitate the search for the best possible approaches of choice in every single run. We therefore suggest a future direction for these two components of ABE models to explore the possibility of harnessing high performance computing technologies to facilitate the exhaustive search when the search space become very large.

4.5. Conclusion

Motivated by instability performance results frequently encountered across various research studies on ABE, in this study, we revisited approaches being commonly adopted with the ABE models in practice to improve the achievable performance. The selected approaches being evaluated in this study were cross-generated from the 5 essential components of ABE models to be in total of 2,304 variants of ABE models.

Compared with the existing assessments of many variants of ABE models, our experiments of this study were performed on a more comprehensive experimental setup by using a greater number (12) of datasets coming from 2 distinct sources, using a greater number (5) of performance measures (*MAR*, *MdAR*, *SD*, *LSD*, and *RSD* [24]) which were all proved as stable measures, and more robust statistical methods (i.e., the Brunner test [56]). Based on this comprehensive evaluation study, the following findings have been concluded:

- A more comprehensive evaluation method consisting of stable error measures and a robust statistical test method enabled us to assess the stable rankings of approaches commonly adopted to improve the performance of ABE models.

- Solution adaptation techniques that linearly adapted the effort by using a few relevant features based on size and productivity, such as Linear size adaption (**MSA**) and Regression toward the mean (**RTM**) were both high performers and have stable ranks consistently.

- A method that dynamically assigned number of analogues such as the Baker's **Best-k** are more likely produce a better overall estimation performance for ABE models.
- Feature normalization is an important preprocess of ABE. Based on our results, we endorsed **Log** transformation as a method being both theoretical superior supported by empirical evidences.
- There seems to be no single best similarity measure exist, we therefore suggested to practitioners to continue using the commonly-adopt **Euclidean** distance in the context of ABE.
- For feature subset selection, we suggested **Pca** as the method of choice if ABE models were configured using all the approaches being listed above; otherwise, we rather suggested to fit a local dataset with all the common feature subset selection methods before performing an effort estimation.

Chapter 5

Exploiting Productivity Factors in Linear Size Adaptation for ABE

The study in the previous chapter concluded that solution adaptation techniques that exploit software size and productivity in their adaption procedures, such as **LSA** and **RTM**, consistently produced higher accuracy than any techniques based on other theoretical concepts. This finding motivated us to further exploit productivity and its factor variable in the solution adaptation process of ABE.

The study of this chapter proposed a new technique named e*X*tended *L*inear *S*ize *A*daptation (*LSA-X*). *LSA-X* extends one adaptation phase of the **LSA** technique by exploiting productivity factors, project variables having the highest value of correlation (r) with productivity, in its adaptation procedure. Specifically, *LSA-X* first determines the productivity factor for a given dataset. It then calibrates the productivity value based on a linear extrapolation between productivity and the productivity factor. Finally, it applies the productivity function to the calibrated productivity and software size to produce the estimated effort value. Our results show that in circumstances where productivity factors are being useful ($r \geq 0.30$), the maximum achievable estimation accuracy can be further improved than adopting any existing solution adaptation techniques. In other circumstances, our results suggest using the **RTM** technique to compensate for the limitations of the *LSA-X* technique.

5.1. Essential Hypothesis

The essential hypotheses behind *LSA-X* are that:

H1: If a dataset exhibits a high correlation between productivity and its factor variable, we will be able to precisely refine the estimated productivity of the new case prior to adjusting its effort.

H2: Calibrating the productivity prior to adjusting the effort will produce a more robust estimate, as shown by the *RTM* technique in the previous chapter and also in previous a study by Azzeh [45].

H3: Calibrating the productivity using linear extrapolation with its factor variable will generate a more significant estimate, in the same way that linear extrapolation was successfully performed between software size and effort in the *LSA* technique [45].

5.2. The Proposed LSA-X Technique

Recall the equation of **LSA**:

$$Effort(P_{new}) = \frac{SS(P_{new}) \times Mean(\ Effort(P_{analogs})\ )}{Mean(\ SS(P_{analogs})\ )}, \tag{5.1}$$

where SS indicates a single software size variable, and $Effort$ indicates an effort value. This equation can be explained as a function of $Productivity = Effort/Size$, where $Pr(P_{analogs})$ is the productivity values of the analogue projects:

$$Effort(P_{new}) = SS(P_{new}) \times Mean(\ Pr(P_{analogs})\ ), \tag{5.2}$$

In other word, **LSA** estimates the term $Effort/Size$ as the objective variable of the ABE model in its procedure, instead of the effort, which is commonly done by simple ABE model such as ABE0-1NN [2, 9]. Then **LSA** applies a linear adjustment based on a software size variable of the new case to produce its final estimated effort value.

As an extension to the **LSA** technique, the proposed *LSA-X* technique introduces a procedure to utilize productivity (Pr) and an influential factor variable ($PrFactor$) in the solution adaptation stage of ABE. *LSA-X* adds one more adaptation step to **LSA** by calibrating the productivity of the retrieved similar project cases before adjusting the effort using a software size. Specifically, *LSA-X* applies a basic analogy-based estimation framework to estimate the term $Pr/PrFactor$ as well as $Effort/Software\ size$ for the new case using the $PrFactor$ and software $size$ of the new case to produce the required effort value.

5.2.1 The Procedure of LSA-X

LSA-X is proceeded in three adaptation steps as follows:

Step 1 is a search for the $PrFactor$. We define the $PrFactor$ as any single continuous project variable of a dataset that has the highest positive degree of Pearson correlation with the productivity (Pr). This single criteria is extended from the essence of the *LSA* technique, where software size is its adaptation variable of choice because software size commonly exhibits the highest correlation to the effort, as empirically observed in the proposed study by Walkerden and Jeffery [13]. In addition, since the Pearson correlation assumes a normal distribution, we apply a logarithmic transformation to all the continuous variables prior to perform the correlation test, as suggested by Kitchenham and Mendes in [34].

In **Step 2**, after the $PrFactor$ is identified, the Pr of the new case is calibrated using linear adjustment:

$$Pr'(P_{new}) = PrFactor(P_{new}) \times \left(\frac{Mean(\ (Pr(P_{analog})\ )}{Mean(\ PrFactor(P_{analog})\ )} \right) \tag{5.3}$$

where Pr' is the calibrated productivity.

In **Step 3**, *LSA-X* adjusts the term Pr' using a software size variable of the new case. This is done by replacing the term Pr in Eq.5.2 of the **LSA** technique with the Pr', calculated in Step 2. Eq.5.4 depicts the third adaptation step of *LSA-X*.

$$Effort(P_{new}) = SS(P_{new}) \times Pr'(P_{new}) \tag{5.4}$$

5.2.2 Example

This section provides an example of the 3-step procedure of *LSA-X*, by showing an estimation process for the effort required by the first project case (*Case #1*) of the Cocomo-sdr dataset. This case was completed with 3,000 LOC and consumed 1.2 man-months effort, thus its productivity is equal to 0.0004 man-months per LOC. Note that we configured *k=2* to in this example.

Step 1 Table 5.1 shows the correlation coefficient values between all the continuous variables of the Cocomo-sdr dataset and its productivity. This table shows that the *Sced* feature has the highest positive value of correlation coefficient with the productivity of this dataset.

Step 2 Without any feature subset selection method applied i.e., use all features, *Case #2* is the closest analogue of *Case #1*, followed by *Case #3*. The values of $PrFactor$ (the *Sced* feature), $Software\ size$, and $Effort$ of the *Case #1* and its three closest analogues are shown in Table 5.1.

Software project cases	Software size	Sced	Actual effort
$P_{new}(Case1)$	3,000	3	**1.2**
$P_{analog1}(Case2)$	2,000	4	2
$P_{analog2}(Case3)$	4,250	4	4.5
Average	3,125	4	3.25

Estimated Effort		
UAVG	*LSA*	*LSA-X*
3.25	$3.25 \times \frac{3,000}{3,125} = 3.12$	$3.25 \times \frac{3,000}{3,125} \times \frac{3}{4} = 2.34$

Figure 5.1. Estimating Case #1 of the Cocomo-sdr dataset using UAVG, LSA, and LSA-X

Following Eq.5.2.1, Pr'_{Case1} is equal to $PrFactor_{Case1} \times (Pr_{Case1}/ Mean(PrFactor_{Analogues}) = 3 \times (3.25/3,125)/4 = 0.00078$ man-months per LOC.

Table 5.1. Correlation coefficient between continuous project features and oroductivity of Cocomo-sdr dataset

Features	Description	$r(Pr, PrFactor)$
Prec	Precedentedness	0.198
Flex	Development Flexibility	0.460
Resl	Arch/Risk Resolution	-0.320
Team	Team Cohesion	0.516
Pmat	Process Maturity	-0.206
Rely	Required Software Reliability	0.463
Data	Database Size	-0.275
Cplx	Product Complexity	0.085
Ruse	Required Reusability	-0.108
Docu	Documentation match to life-cycle needs	-0.061
Time	Execution Time Constraint	-0.051
Stor	Main Storage Constraint	-0.047
Pvol	Platform Volatility	0.033
Acap	Analyst Capability	0.180
Pcap	Programmer Capability	-0.276
Pcon	Personal Continuity	0.240
Aexp	Applications Experience	0.369
Pexp	Platform Experience	-0.009
Ltex	Language and Tool Experience	0.547
Tool	Use of Software Tools	0.388
Site	Multisite Development	-0.343
Sced	Development Schedule	0.582
Loc	Lines of Code	-0.733

Step 3 At this final step, *LSA-X* adjusts $Pr'_{Case\#1}$ using its software size and estimates the effort value for $Case\#1$ as $0.00078 \times 3,000 = 2.34$ man-months. In this example, the estimated effort value using *LSA-X* produces a more accurate estimate than that of **UAVG** and **LSA**, which are equal to 3.25 and 3.125 man-months, respectively.

5.2.3 The Rationale behind the LSA-X Technique

Compared with the **UAVG** technique, **LSA** generally produces higher accuracy [13, 36, 37] mainly because it can capture and exploit the degrees of difference between project cases with similar features, but different in software sizes. **LSA** applies the information captured with degree of the difference in software size to the estimated effort value by scaling the estimated effort to match the expected software size of the new case.

Following the adaptation mechanism of **LSA** that the effort value will be more realistic after being adjusted by other variables to match the characteristics of the new case, the proposed *LSA-X* technique further adjusts the estimated effort value by exploiting one additional variable from the software size. In other word, the proposed *LSA-X* technique can further identify and exploit the productivity differences between a new project case and its analogues by first adjusting the productivity, followed by the software size. As showed in the example case of the Cocomo-sdr dataset, this extension to the adaptation procedure makes *LSA-X* better adapt the effort value, and thus produces more accurate estimation.

5.3. Evaluation

The experiments of this study also fully utilized the experimental procedure explained in Chapter 3. Since the study in the previous chapter had already concluded the superior approaches for the 5 essential components of ABE models, the experiments of this study therefore only adopted the ABE models with all those superior approaches, excepted the dimension of solution adaptation. We included all the 8 solution adaption techniques examined in the previous chapter in this study to fully evaluate the *LSA-X* technique. Specifically, the ABE models adopted in this study are based on the combinations of the following approaches:

- Similarity measure: **Euclidean**;
- Approach to identify number of analogues: **Best-k**;
- Normalization method: **Log**;
- Feature subset selection: **All**, **Sfs**, **Swvs**, **Pca**.

Best-k, **Log**, and **Euclidean** were chosen as being the approaches recommended by our study of Chapter 4. However, we decided to examine all the 4 common feature subset selection in this study because the experiments of the previous chapter suggested to examine all of which when evaluating any now approaches deviated from the best approaches of choices concluded in the previous chapter. In short, this study evaluated ABE models built with 4 feature selection methods $\times$ 9 solution adaptation techniques = 36 variants in total.

After all the 36 combinations of ABE models were executed using the leave-one-out approach, their performances were recorded with 5 error measures. Then, pairwise comparisons were carried out between each of the 36 variants $\times$ 35 others, using the *win-tie-loss* statistics. Evaluated the performance in terms of *wins*, *losses*, and *wins-losses*, the maximum *wins* or *losses* of (solution adaptation technique, feature selection method) per single dataset is 35 $\times$ 5 error measures = 175. Finally, we determined the overall performance of each single solution adaptation technique by calculating the median values of *wins*, *losses*, and *wins-losses* of feature selection methods over the same solution adaptation techniques. For a more detailed result showing the ranking of feature subset selection methods $\times$ solution adaptation techniques, we will show the *wins*, *losses*, and *wins-losses* values without having any aggregation applied.

5.3.1 $PrFactor$ of the Experimental Datasets

The evaluation of *LSA-X* was performed over the 12 datasets used in the study of the previous chapter. Table 5.2 presents the $PrFactor$ along with its description and the value of $r(Pr, PrFactor)$ for each of the selected datasets. Even though the characteristics of the 12 datasets were greatly diversified, we can categorize the variables selected as their $PrFactor$ in two groups, all of which are known to be influential factors of Pr:

- Size, complexity and constraints of the software project, e.g., Envergure, DevelpmentType, ResourceLevel, Sced, Time, T08;
- Developers team and experience, e.g., AverageTeamSize;

In the case where a $PrFactor$ was selected with a very low value of $r(Pr, PrFactor)$, e.g. $r(Pr, PrFactor) < 0.30$, such as the top 4 datasets of Table 5.2, we specu-

Table 5.2. Productivity factors, their descriptions, and the values of $r(Pr, PrFactor)$ of the 12 selected datasets

Dataset	PrFactor	PrFactor Description	r(Pr, PrFactor)
Desharnais	Envergure	Complexity adjustment factors	0.10
ISBSG-insurance	DevelopmentType	Development type	0.24
Miyazaki94	File	Number of different record formats	0.28
Finnish	Co	N/A	0.30
ISBSG-Banking	ResourceLevel	Resource level	0.34
Maxwell	T08	Requirement volatility	0.38
Kemerer	Hardware	Type of hardware	0.44
ISBSG-communication	AverageTeamSize	Average team size	0.49
Cocomo-sdr	Sced	Schedule constraint	0.59
Cocomo81	Time	Time constraint for cpu	0.62
Nasa93	Time	Time constraint for cpu	0.67
Albrecht	Output	Number of external outputs	0.68

late that these datasets may not contain any real $PrFactors$. This may be due to their data collection or data processing prior to their publicly available. We therefore hypothesize that the proposed *LSA-X* technique will not perform well for these datasets since the $PrFactor$ seems not to be useful.

5.4. Results

Table 5.3 shows the comparison results of between the 9 adaptation techniques. In this table the win-tie-loss statistic are summarized by the number of *wins-losses*. The highlighted entries in this table indicate the solution adaptation technique with the highest number of *wins-losses* for the dataset in their corresponding row. Rows of this table are sorted by the Pearson correlation strength between productivity and productivity factor ($r(Pr, PrFactor)$). The results show that the **LSA-X** technique performed best in terms of *wins-losses* for 6 out of the 13 datasets. **LSA** was next with 4 datasets, followed closely by **RTM** with 3 datasets. According to the results of this experiment, **LSA-X** performed the best by means of general performance.

When we look closely at the value of $r(Pr,\ PrFactor)$ in Table 5.3, we can divide the datasets into 2 bands based on $r(Pr,\ PrFactor)$ with regard to the

Table 5.3. Win-tie-loss results from (4 feature selection methods x 5 error measures) of Leave-one-out experiments. The highlighted entires indicate the estimators with best performance in terms of $wins - losses$ for the corresponding dataset in the same rows.

Datasets	wins-losses									$r(Pr, PrFactor)$
	UAVG	IRWM	LSA	MSA	LSA-X	RTM	AQUA	GA	Nnet	
Desharnais	-14	-70	121.5	80	19	36	-107	-15.5	-106	0.10
ISBSG-insurance	-34	-10	96	78	-96.5	-8	24	-29.5	-16	0.24
Miyazaki94	-20	-61.5	20	20	35	89	-61.5	0	-34	0.28
Finnish	-14.5	-37	16.5	16.5	84.5	56	-51	-12	-57	0.30
ISBSG-Banking	-33	-36	27	27	63	14.5	-49.5	-33.5	-4	0.34
Maxwell	-12.5	-50	27	27	58.5	39	-58.5	-5.5	-40	0.38
Kemerer	-42	-11	37.5	31	40	33.5	-32	-12	-23.5	0.44
ISBSG-communication	12.5	6.5	54	48	0	10	-77	24.5	-52	0.49
Cocomo-sdr	-44	-64.5	43	43	90	-14	-39.5	-15	-35.5	0.59
Cocomo81	-8	-24.5	5	5	21.5	40.5	-16	5	-4.5	0.62
Nasa93	-23	-13.5	-4	-4	18.5	21	9.5	-16	10	0.67
Albrecht	3	8	69.5	51	109.5	3	-78	-27	-120	0.68

accuracy achieved by the *LSA-X* technique:

Band 1: datasets with $r(Pr, PrFactor) < 0.30$;

Band 2: datasets with $r(Pr, PrFactor) \geq 0.30$.

It seems to be clear that **LSA-X** was the method of choice for datasets with $r(Pr, PrFactor)$ equal to or greater than 0.30. In this band, **LSA-X** had the best performance for 6 out of the 9 datasets. Also for 1 of the other 3 datasets where **LSA-X** did not show the best performance, **LSA-X** performed almost well as the best performers. On the other hand, **LSA-X** performed poorly for the datasets with $r(Pr, PrFactor)$ less than 0.30. In these cases, the **LSA** technique performed well for 3 out of the 4 datasets, while **RTM** performed best for the other one dataset.

As suggested by Keung et al. [9] and the study of our previous chapter, ranking agreement between results based on *wins*, *losses*, and *wins-losses* are necessary to draw performance conclusion. Hence, we aggregated the *wins*, *losses*, and *wins - losses* for each band using the mathematical mean and presented them in Table 5.4. The highlighted entries in Table 5.4 show the best techniques in each band in terms of *wins*, *losses*, and *wins - losses*.

Table 5.4. A summary of the results of Table5.3 in the 2 bands of datasets. The highlighted entries show the best-performing techniques.

Datasets	Measures	UAVG	IRWM	LSA	MSA	LSA-X	RTM	AQUA	GA	NNet
Band 1 ($r < 0.30$)	wins	64.00	51.67	107.00	99.67	60.00	86.17	49.33	69.33	40.33
	losses	86.67	98.83	27.83	40.33	74.17	47.17	97.50	84.33	92.33
	wins-losses	-22.67	-47.17	79.17	59.33	-14.17	39.00	-48.17	-15.00	-52.00
Band 2 ($r \geq 0.30$)	wins	43.22	42.72	72.11	68.56	82.44	70.44	32.11	39.33	30.94
	losses	61.17	67.39	41.50	41.39	28.50	47.83	75.67	49.50	67.22
	wins-losses	-17.94	-24.67	30.61	27.17	53.94	22.61	-43.56	-10.17	-36.28

Table 5.4 provides strong evidence that the *LSA-X* consistently performed best for datasets with $r(Pr, PrFactor)$ equal to or greater than 0.30, as its *wins*, *losses*, and *wins-losses* in this table are better than all other techniques examined in this study. On the other hand, the datasets with $r(Pr, PrFactor)$ less than 0.30 are clearly a limitation of the **LSA-X** technique. For this band of datasets,

Table 5.4 shows that **LSA** performed better than all other techniques. Hence, for this band of data sets, we suggest **LSA** to compensate for the limitation of **LSA-X**.

Table 5.5. A comparison of the 3 most accurate adaptation techniques in terms of the 4 feature subset selection methods

Datasets	Measures	LSA				LSA-X			
		All	Pca	Sfs	Swvs	All	Pca	Sfs	Swvs
Band 1 ($r < 0.30$)	wins	109.00	105.67	109.00	85.00	58.33	87.33	58.33	59.33
	losses	28.00	35.67	28.00	63.00	75.67	50.33	75.67	73.00
	wins-losses	81.00	70.00	81.00	22.00	-17.33	37.00	-17.33	-13.67
Band 2 ($r \geq 0.30$)	wins	68.78	77.44	68.44	68.78	83.44	77.33	82.67	75.67
	losses	41.33	26.00	41.33	46.78	27.00	31.44	28.44	35.11
	wins-losses	27.44	51.44	27.11	22.00	56.44	45.89	54.22	40.56

The *win-tie-loss* results of Table 5.3 and Table 5.4 calculated the mean of the accuracy results to demonstrate the generalized performance of each adaptation technique. As we discussed in the previous chapter that the choices of feature subset selection methods were remaining inconclusive, a search for the best fit method given different setup and datasets may be required in every single estimation. To examine more specific results regarding the use of different feature subset selection methods, Table 5.5 shows the performance in regard to all 4 common feature selection methods. To focus only on the superior techniques, Table 5.5 only presents the best 2 solution adaptation techniques ranked by performance as in Table 5.4. These more specific results indicate that for the datasets with $r(Pr, PrFactor)$ equal to or greater than 0.30, **LSA-X** performed best when the feature subset selection was selected by the **All** or **Sfs** method. Also for the datasets with $r(Pr, PrFactor)$ less than 0.30, these two feature subsets selection methods were also suggested to be adopted with **LSA** method to produce a more accurate estimate.

5.5. Discussion

This study introduces a solution adaptation technique named **LSA-X** for ABE. **LSA-X** adds one more adaptation step to the Linear Size Adaptation (*LSA*) technique by exploiting the productivity factor variables (*PrFactor*) in its effort adjustment procedure. The highlighted benefits of the **LSA-X** technique are described below.

Benefit 1: We can see from the highlighted entries in Table 5.3 that **LSA-X** achieved the best generalized performance. Even though we later found that the *LSA-X* was consistently less accurate than other techniques with datasets that exhibit a value of $r(Pr, PrFactor)$ below 0.30, such case are a minority of datasets and may be unlikely to arise in practice (i.e., only found in 3 out of the 12 datasets used in this study).

Benefit 2: **LSA-X** does not require a high computational power. Compared with more complex adaptation techniques such as the **GA** and **NNet** techniques, **LSA-X** is significantly more straightforward, such that even a spreadsheet application can handle the necessary computation.

Benefit 3: It is easy to decide when to use **LSA-X**. As we observed, a Pearson correlation coefficient between *Pr* and *PrFactor* can simply determine the accuracy of the *LSA-X* technique:

- *LSA-X* is a very robust and accurate technique for datasets with $r(Pr, PrFactor)$ equal to or greater than 0.30;
- *LSA-X* is consistently inaccurate for datasets with $r(Pr, PrFactor)$ less than 0.30.

Hence, we recommend the *LSA-X* technique as the adaptation technique of choice in circumstances where a dataset exhibits $r(Pr, PrFactor)$ equal or greater than 0.30. For the other less common cases, we suggest **LSA** to compensate for the limitation of *LSA-X*.

5.5.1 And Why does it Work?

To provide a clear justification as to why exploiting the productivity factors using **LSA-X** is an improvement, further quantitative analysis may be required. However, this section discusses some possible reasons.

The first reason concerns the exploitation of productivity. The benefit of productivity exploitation in predictive modeling is well recognized in various research areas such as Economics, because productivity is one of the important factors to analyze to improve the throughput of a work process [74, 75]. Among the existing solution adaptation techniques that consider the degrees of difference of both software size and productivity between the new case and its analogues, only the **LSA-X**, **LSA** and **RTM** techniques also calibrate the productivity values in their adaptation procedures. Our results showed the **LSA** and **RTM** techniques were also high performance techniques.

The second reason concerns the use of the $PrFactors$ in addition to productivity. This is possibly the main reason that **LSA-X** outperforms all the other techniques when $PrFactors$ are useful, i.e., $r(Pr, PrFactors) \geq 0.30$. Based on our observations, productivity is considered the second most influential factor on the effort next to the software size. That is, when the influence of the software size is removed from the dataset, productivity then becomes the most dominant factor of the effort. Therefore, when considering adaptation, we first need to consider the difference in software size among analogues, and next, we need to consider the difference in productivity among analogues. The use of $PrFactors$ plays the main role in better measuring and adapting productivity differences among analogues.

Specifically, the use of $PrFactors$ allows the **LSA-X** technique to focus the productivity adjustment based only on the difference in $PrFactors$ between the new case and a few of its analogues, rather than requiring the adjustment to be based on the entire coherence group of projects as performed in **RTM**. A recent study by Kocaguneli et al. [14] concluded that an estimation that relies on a smaller number of more relevant analogues will result in better estimation performance than relying on a larger number of less relevant analogues. Our results are in agreement with Kocaguneli et al. as **LSA-X** achieved significantly improved estimation performance. This finding thus allows us to suggest interesting future studies to consider exploitation of the degree of relevance between new cases and their analogue as well as to further exploit productivity.

5.5.2 Findings

We further discuss ways to compensate for the limitation of **LSA-X** when a given dataset has $r(Pr, PrFactor)$ less than 0.30. This summarizes our findings based on all tables and results in this paper.

Finding 1: Results regarding the **RTM** techniques were deviated from the study of the previous chapter, where **RTM** were consistently the best techniques among all the 8 techniques selected in this study. We speculated that the reduced performance were mainly because of the possibly high number of tie results achieved between **LSA-X** and **RTM**. Indicated in Table 5.3 where the performance of *LSA-X* stood out with datasets $r(Pr, PrFactor)$ equal to or greater than 0.30, **RTM** also performed well in several datasets.

Finding 2: According to Table 5.4 and Table 5.5, for datasets with $r(Pr, PrFactor)$ less than 0.30, the **LSA** technique performed the best. Table 5.3 showed that **LSA** performed as well as **LSA-X** for some specific datasets with $r(Pr, PrFactor)$ greater than 0.30. This finding shows that in SEE, exploitation of $r(Pr, PrFactor)$ contributed to accuracy.

Finding 3: Results from all tables show that the solution adaptation techniques utilizing linear adjustment functions, e.g. **LSA-X**, are consistently more accurate than those using non-linear functions e.g. **NNet**. The results also show that and adaptation techniques that exploit productivity are more accurate than any other techniques. This finding supports the usefulness of exploiting productivity in software development effort estimation.

5.6. Conclusion

The study of this chapter explores the possibility of exploiting productivity (Pr) and its influential factor ($PrFactor$) in the solution adaptation stage of ABE. Productivity is one of the essential factors to analyze to estimate software development effort, and it is widely studied in many areas of expertise [74, 75]. Our results showed strong evidence that the proposed *LSA-X* adaptation technique, which exploits both Pr and $PrFactor$ in its adaptation procedure, is a successful technique to generate a robust and accurate effort estimate. In an evaluation subject to a robust statistical test method using the Brunner test (95% confidence), the proposed *LSA-X* technique outperformed 8 techniques commonly adopted in the literature and in practice [45], for datasets that exhibit a high correlation

($r \geq 0.30$) between Pr and $PrFactor$. For the other less common cases, our results showed that *LSA-X* has limited accuracy, and we suggest using another adaptation technique that also based on software size and productivity such as the **LSA** and the **RTM** techniques to compensate for this limitation.

Based on our detailed experiment evaluated both adaptation techniques and feature selection methods in this study, we suggest an estimation with ABE model to use all features (i.e. no feature subset selection is performed) and adapted to the effort by:

- For a dataset with $r(Pr, PrFactor) \geq 0.30$, using the **LSA-X** technique;
- For a dataset with $r(Pr, PrFactor) < 0.30$, using the **LSA** technique.

With solution adaption technique and feature subset selection method selected by the suggestion mentioned above, other components that performed the best with this configured became:

- Similarity measure: **Euclidean**;
- Approach to identify number of analogues: **Best-k**;
- Normalization method: **Log**;
- Feature subset selection: **All**.

Chapter 6

Review Data Quality Improvement Techniques for ABE

The results of the study of Chapter 4 suggested that data normalization appears to be an important process of ABE. Despite no empirical evidence showing that data normalization is strongly associated with estimation performance improvement for all the situations, our experimental results shown in Chapter 4 suggested that a normalization using **Log** consistently performed better than not performing normalization at all or using the most commonly adopted **Interval0-1** method. The benefit of applying **Log** transformation to a dataset is not only that it assures the equal influence of all the project features of the datasets, but it also approximates normal distribution of the data. The more successful of applying the **Log** normalization motivated us to question whether the attributes or quality of data are actually associated with estimation accuracy. Therefore, the study of this chapter revisits 6 data quality improvement techniques, 5 of which are techniques to improve data purity recently empirically reviewed by Seo et al. [26]. The other one is a technique to filter inconsistent data from software effort datasets, proposed by Kocaguneli et al. [14].

6.1. Hypothesis, Motivations, and Research Questions

A recent empirical study Seo et al. [26] reviewed 5 common outlier elimination techniques with Linear regression model and ABE0, and suggested that applying outliers elimination before an application of a dataset did not consistently produce a better accurate estimation in every situation. However, the study in the previous chapter of this thesis revealed that some successful configuration such as **Log** normalization only performed outstandingly with some combinations of other standout approaches such as **LSA-X**. Hence, we hypothesize that applying outliers elimination to a dataset before performing an estimation with the ABE model built the combinations of only more successful approaches may result differently than that of based on ABE0, discussed in the study by Seo et al. [26].

In this study, experiments and analyses of results are organized to answer these following research questions:

RQ1 In terms of maximum performance, is there any single best data quality improvement techniques among the commonly adopted techniques?

RQ2 In terms of generalized performance, which are the techniques most frequently improved the estimation performance from applying no techniques?

RQ3 Which are the techniques provide less desirable performance than others?

For the first research question, we speculate that there may be no single best technique that fit all the dataset. This is because the procedure of each data quality improvement technique are stemmed from different theoretical concepts and each of which seems to fit for different purposes. For example, **LTS** is based on a theoretical concept related to linear regression, while **Kmeans** is based on the concepts of exploiting the level of similarity between project cases. The second research question is motivated by that if there is no single best technique, at least, it would be helpful if we can find a technique that generally improves the estimation accuracy in most situations, compared with estimating using full datasets.

6.2. Background of Data Quality

Many empirical studies and reviews suggest a strong association between data quality improvement and estimation accuracy improvement of software effort models [26, 76–79]. Based on a taxonomy of data quality issues in empirical software engineering study proposed by Bosu and MacDenell [77], where issues were summarized from prominent empirical software engineering research papers published between 2004 and 2013, data quality issues were classified into three main issues: accuracy, relevance, and provenance. These three main issues can be further divided into sub-issues as follows:

Accuracy: outliers, noise, inconsistency, incompleteness, redundancy;

Relevance: amount of data, heterogeneity, timeliness;

Provenance: commercial sensitivity, accessibility, trustworthiness.

As we mainly focus on accuracy, this study selects 6 common methods to handle outliers and data inconsistency for the evaluation. The following are the detail explanation and the parameters we configured for all the 6 method selected in this study:

6.2.1 Five Outlier Elimination Methods

These 5 methods (**LTS**, **CD**, **Kmeans**, **BoxPlot**, and **LM**) were selected as being method most commonly studied and adopted in practice, suggested in the recently empirical study by Seo et al. [26].

1) Least trimmed squares (**LTS**) [80, 81], minimizes the sum of squared residuals of the simple **LReg** model given:

$$min \sum_{i=1}^{h} AR_i^2 \quad | \quad \frac{n}{2} < h \leq n \tag{6.1}$$

where i is the order of project cases sort in ascending order, n is the number of cases, h is the number of data points in the subset of n, and AR is the absolute difference between E_i and $\hat{E}_i$. **LTS** can be considered as a subset search method, where the final subset yield the minimum AR. In our experiment, h was configured to $0.75n$ based on the suggestion in many previous studies [26, 80, 81].

2) Cook's distance (**CD**) [68], measures if the changes as results of a deletion of each single project case from the training set will result the changes of the overall residual of the entire training set. The Cook's distance D_i of a $case_i$ is calculated by:

$$D_i = \frac{\sum_{j=1}^{n}(\hat{E}_j - \hat{E}_{j(i)})^2}{p \times MSE} \quad | \quad MSE = \frac{1}{n}\sum_{k=1}^{n}(E_k - \hat{E}_k)^2 \tag{6.2}$$

where $\hat{E}_{j(i)}$ is the estimated effort of $case_j$ using the simple **LReg** model trained without $case_j$, and p is the number of parameters in the model. In this experiment we consider a $case_j$ as an outlier if its D_i is greater than $3 \times (\frac{4}{n})$, as suggested by a study by Mendes and Lokan [68].

3) K-Means Clustering (**Kmeans**) adopts an idea that outlier cases should be classified into same group of a very few number of cases. Seo et al. [26] adopted silhouette index in their reviewed study to identify the number of clusters being most fit with a local training set. The formula of the silhouette index is given below:

$$S_i = \frac{b(i) - a(i)}{max(a(i), b(i))} \tag{6.3}$$

where $a(i)$ is the mean distances between $case_i$ and all the other cases in the same cluster, and $b(i)$ is the average distance from $case_i$ to all other clusters where a case that is closest to $case_i$ represented that cluster. The higher silhouette value indicates more appropriate clustering results. Using this approach Seo et al. [26] suggested any case with Si less than 0 or being clustered into groups of less than 3 cases as being outliers.

4) Box plot (**BP**) is one of the most simple statistical-based method to identify outliers. Using **BP**, outliers are identified using the components of box plots, which are the lower quartile (Q1), the median, the upper quartile (Q3), and the inter-quartile range (i.e., IQR = Q3-Q1). The simple rule commonly adopted to detect outliers based on the components of **BP** is to consider any feature value

with a value higher than Q3 + 1.5 $\times$ IQR or lower than Q1 - 1.5 $\times$ IQR as outliers. Since **BP** examines each feature independently, we treated any $case_i$ as an outlier case if more than one features had been indicated by **BP** to be the outliers.

5) Mantel Leverage Metric (**LM**) [8] is based on generating the Mantel correlation on every single training instance, sampled by the leave-one-out approach. Each resulting sample $LM_i \in LM_{1\ to\ n}$ indicates how much does $case_i$ influence the Mantel correlation of the overall dataset. LM_i is calculated by:

$$LM_i = R_i - \hat{R}_i \quad | \quad \hat{R}_i = \sum_{i=1}^{n} R_i/n \tag{6.4}$$

where n is the total number of cases, R_i is the Mantel correlation of all cases excluding $case_i$, and $\hat{R}_i$ is the overall Mental estimation of the entire dataset. One important property of $\hat{R}_i$ is that it approximates normal distribution of overall resampled Mantel correlations, with S^2 approximates to $\sum_{i=1}^{n}(R_i - \hat{R}_i)^2/n - 1$. Exploiting this property, any $case_i$ with $|LM_i/S|$ greater than 2 can be validly considered as an outlier at significance level of 0.05.

6.2.2 Adopting TEAK as an Inconsistency Elimination Method

TEAK or Test Essential Assumption Knowledge is a method originally proposed by Kocaguneli et al. [14] to built ABE models based only on project cases that do not violate its essential assumption i.e., similar software project should require similar amount of effort for development. In other word, the procedure of **TEAK** is to filter project cases with similar characteristic with each other but were completed with diversified amount of efforts, which clearly indicates inconsistent cases.

The essential hypothesis behind **TEAK** is that *locality implies homogeneity.* Adopted by this essential hypothesis, **TEAK** is proceeded with the following procedures:

1. Generate a binary tree, which:

   (a) Leaf nodes represent the effort value;

   (b) Branches are formed by similarity values between pairing of project cases;

   (c) The higher-level subtrees merge the lower-level subtrees using their median of similarity values within the same subtrees.

2. Remove all height=1 subtrees in which the project cases in that subtree have σ^2 greater than 10% of the maximum σ^2 among all the height=1 subtrees [14].

Project cases in the same subtree are more likely to be relevant with each other than any project cases in any other disjoint subtrees, and project cases in the lower level of the same subtree should be more relevant with each other than being in its supertree. This is because subtrees are formed by level of similarity. Based on the *locality implies homogeneity* hypothesis, **TEAK** prunes all subtrees with height=1 where their σ^2 values are significantly higher than all the other subtrees with height=1. Project cases in these subtrees are assumed inconsistent because high σ^2 values detected in any subtree with height=1 means that those cases are very similar projects but they were completed with significantly diversified amount of effort.

For the technical detail, the generated binary tree is the greedy agglomerative clustering (GAC) tree. It is one of the most simple method to generate a data structure to represent hierarchical clusters [82]. GAC is built bottom up, in a manner of greedy algorithm. Its procedure is terminated when the all tree nodes are connected to at least one nodes and the full GAC tree is completely built. Kocaguneli et al. [14] suggested based on their experimentations to remove all the height=1 subtrees which have the value of σ^2 greater than 10% of the maximum σ^2 observed in all the height=1 subtrees of the full GAC tree. In their study, the suggestion of this 10% parameter value was confirmed the validity by 20 randomized trials across various performance measures and various experimental datasets.

6.3. Evaluation

Based on the results of the study of Chapter 5, we updated the superior approaches for the 5 essential components of ABE models as follows:

- Normalization approach: **Log**;
- Feature subset selection: **All**;
- Similarity measure: **Euclidean**;
- Solution adaptation techniques: **LSA-X** for datasets for datasets with $r(Pr, PrFactor)$ equal to or greater than 0.30, and **LSA** for datasets with $r(Pr, PrFactor)$ less than 0.30;
- Approach to determine number of analogues: **Best-k**.

Let **ABE-Best** denote this ABE model. In this study we evaluate this **ABE-Best** model with before and after having each of 6 data quality improvement technique (**LTS**, **CD**, **Kmeans**, **BP**, **LM**, and **TEAK**) applied. Thus there are 7 variants of ABE models compared in this study. The experiments for this study are in two-fold. First we performed the pairwise comparisons as done in all the studies of this thesis. The other part is to examine the effect of applying each of the data quality improvement method by comparing each of which directly with **ABE-Best**.

For the first part of the experiments, the comparison between these 7 variants utilized the procedure explained in Chapter 3 that begins with the leave-one-out approach experiments, followed by recording the performance using the 5 robust error measures. The pairwise comparisons were carried out between 7 variants $\times$ 6 others. Thus, the maximum values of *wins* and *losses* per single variant and single dataset were $6 \times 5 = 30$. Finally, the overall performance of each variant was determined in terms of overall *wins*, *losses*, and *wins-losses*, and the stability of the ranking results were determined by observing the ranking agreement across these 3 ranking lists.

For a second part of the experiments, where we compared each of the 6 data quality improvement techniques head to head with **ABE-Best**, we compared the raw MAR and $MdAR$ performance as the main results along with the *wins*, *losses*, and *wins-losses*. Note that since this comparison were not in terms of ranking, we did not examine whether stability between *wins*, *losses*, and *wins-losses* were established in this experiment.

6.4. Results

6.4.1 Investigating Research Question 1

The first research question examines whether there is any single best data quality improvement techniques among the commonly adopted techniques, in terms of maximum achievable performance.

Table 6.1. Total summations of wins, losses, and wins-losses results of the comparisons between adopting **ABE-Best** with full datasets and adopting it with each of the 6 selected data quality improvement techniques. The results were aggregated across the 12 datasets used throughout this thesis

Total	ABE-Best	LTS	CD	Kmeans	BP	LM	TEAK
wins	99	146	140	131	106	138	144
losses	90	126	143	144	154	120	139
wins-losses	9	20	-3	-13	-48	18	5

Table 6.1 shows the *wins*, *losses*, and *wins-losses* results of the comparisons between **ABE-Best** and other 6 variants of this model, each of which was built after applying each of the 6 common data quality improvement techniques to the datasets. The aggregated results of Table 6.1 show that **LTS**, **CD**, and **TEAK** were best in terms of wins. Our of these three techniques, **LTS** also performed best in terms of *wins-losses*. However, in terms of *losses*, **ABE-Best** yielded the best accuracy. Since we have assured the robustness and stability of the evaluation method being used, we speculated from lesson learned in the earlier chapters for this circumstance that each data quality improvement technique may performed accurately with only some limited situations, while not for all the datasets. Also, some datasets may not need any consideration of case eliminations.

To investigate further on whether different techniques performed well with only a few distinct datasets, Table 6.2 shows the results by datasets in terms

Table 6.2. Total counts of wins, losses, and wins-losses results per dataset, of the comparisons between adopting **ABE-Best** with full datasets and adopting it with each of the 6 selected data quality improvement techniques.

Datasets	Measures	ABE-Best	LTS	CD	Kmeans	BP	LM	TEAK
	wins	2	10	9	17	4	14	15
Albrecht	losses	8	11	16	3	14	10	11
	wins-losses	-6	-1	-7	14	-10	4	4
	wins	19	14	19	7	0	17	9
Cocomo-sdr	losses	6	6	6	20	20	8	19
	wins-losses	13	8	13	-13	-20	9	-10
	wins	0	2	6	6	8	11	1
Cocomo81	losses	1	1	10	10	8	5	1
	wins-losses	-1	1	-4	-4	0	6	0
	wins	1	14	16	8	16	8	7
Desharnais	losses	14	7	7	5	1	13	23
	wins-losses	-13	7	9	3	15	-5	-16
	wins	16	19	13	9	18	24	1
Finnish	losses	14	10	17	21	12	5	27
	wins-losses	2	9	-4	-12	6	19	-26
	wins	18	8	16	9	14	3	11
ISBSG-Banking	losses	4	21	14	12	8	7	13
	wins-losses	14	-13	2	-3	6	-4	-2
	wins	8	16	15	13	7	8	19
ISBSG-Communication	losses	16	10	11	10	15	17	7
	wins-losses	-8	6	4	3	-8	-9	12
	wins	0	14	12	9	5	11	13
ISBSG Insurance	losses	0	10	9	14	13	13	5
	wins-losses	0	4	3	-5	-8	-2	8
	wins	26	16	9	18	8	0	22
Kemerer	losses	0	13	21	12	22	28	5
	wins-losses	26	3	-12	6	-14	-28	17
	wins	0	16	3	13	5	21	13
Maxwell	losses	0	9	18	11	20	2	11
	wins-losses	0	7	-15	2	-15	19	2
	wins	2	2	11	9	19	15	19
Miyazaki94	losses	9	24	6	16	6	10	6
	wins-losses	-7	-22	5	-7	13	5	13
	wins	7	15	11	13	2	6	14
Nasa93	losses	18	4	8	10	15	2	11
	wins-losses	-11	11	3	3	-13	4	3
	wins	3	2	1	1	2	3	1
Total best ranks	losses	6	1	1	1	2	1	3
	wins-losses	3	3	1	1	2	3	3

of *wins*, *losses*, and *wins-losses*, along with the number of times each technique achieved the higest rank for each dataset. The highlighted entires of this table clearly indicated that there were no single techniques performed best with all datasets. Also, in quite a number of datasets, applying no technique yielded better estimation performance. In terms of highest achieved ranks, **ABE-Best** and **LM** performed best in terms of *wins*, **ABE-Best** also performed outstandinly in terms of *losses*. However, in terms of *wins-losses*, all the techniques almost showed no difference in performance.

In summary, the answer to the first research question is that there is no single best data quality improvement techniques among the commonly adopted techniques. Different datasets appeared to prefer different techniques, and some datasets do not need any data quality improvement techniques.

6.4.2 Investigating Research Question 2

The second research question searches for the techniques that most frequently improve the estimation performance than applying no techniques. The results and analyze for this study would suggested techniques with better generalized performance.

Table 6.3 show results in terms of MAR for all the 7 variants of ABE models examined in this study. Entries highlighted with bold text represent the datasets where applying a technique in its corresponding column has improved the estimation performance from applying no technique (i.e., **ABE-Best**). Entires highlighted in grey represent the technique that performed the best for the dataset in each of their corresponding rows. The total number of datasets where each technique did improve the performance from applying no single technique are summarized at the bottom of the table, along with the total number of the times that each technique has achieved the best performance.

The results showing in this table strongly indicate that, in general situations, applying any of data quality improvement techniques before an application of datasets do not necessarily produce a better estimation performance. That is, apart from **LTS** and **LM**, which seem not to not produce much improvement to **ABE-Best**, applying any other techniques more likely produce a fallback in performance. Furthermore, in terms of the number of datasets that a technique

Table 6.3. Raw results in terms of MAR from comparing the ABE-Best model and this model built after applying LTS, CD, Kmeans, BF, LM, or TEAK techniques to the datasets before performing an effort estimation

Datasets	ABE-Best	LTS	CD	Kmeans	BP	LM	TEAK
Albrecht	9.25	**7.66**	**7.61**	**5.04**	10.50	**6.54**	**5.31**
Cocomo-sdr	2.05	2.41	2.05	6.86	7.17	**1.78**	2.99
Cocomo81	807.17	**366.66**	**451.94**	**460.39**	**396.52**	**440.30**	920.60
Desharnais	1989.61	**1630.53**	**1712.13**	**1901.30**	**1511.38**	**1709.15**	**1863.40**
Finnish	3873.87	**3786.03**	3898.67	3910.89	3894.22	**3651.13**	4506.29
ISBSG-banking	1991.49	3007.31	2068.30	2911.90	2906.88	3395.05	3031.90
ISBSG-communication	2744.47	**2256.95**	2868.75	2829.96	3612.22	2870.87	**2493.20**
ISBSG-insurance	9526.99	**3223.14**	**3606.08**	**3833.43**	**4324.25**	**3659.01**	**2813.89**
Kemerer	92.23	109.59	122.51	110.04	115.28	124.78	110.46
Maxwell	2455.13	3494.24	3790.51	3652.44	4472.87	3405.14	3545.10
Miyazaki94	55.73	**49.54**	**44.48**	**50.35**	**45.01**	**47.18**	**47.25**
Nasa93	311.19	**214.68**	**232.22**	**241.87**	346.46	**250.43**	**227.72**
#improve from ABE-Best	-	8	6	6	4	8	6
#best	3	3	2	1	1	2	1

perform the best in *MAR*, applying no technique performed as good as **LTS**, which is considered to be better than the other selected techniques. This means that, for all the datasets in which **ABE-Best** performed the best, all the techniques compared in this study produced a fallback in performance after they are applied to these datasets.

In summary, the results of Table 6.3 suggest that applying any of the commonly adopted data quality improvement techniques before applications of datasets can improve the accuracy of ABE models, but not always and consistent. Among the 6 commonly adopted techniques selected in this study, there is no sufficient evidence to conclude that applications of these techniques would be beneficent to the SEE process based on ABE.

6.4.3 Investigating Research Question 3

Apart from seeking for successful techniques, being able to determine the less desirable techniques is hugely important for the real-world SEE processes. That is,

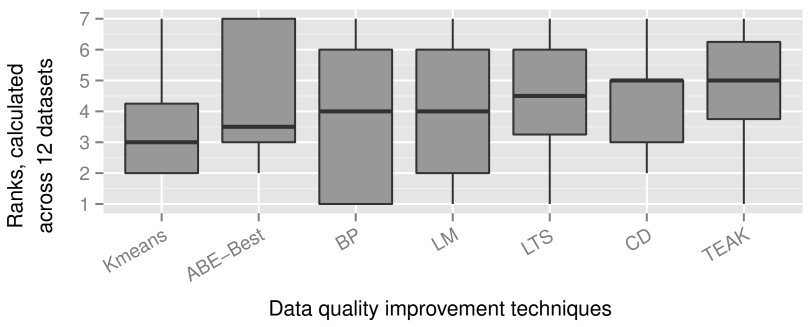

Figure 6.1. Box plots of the ranking of data quality improvement techniques

whether we are reluctant to definitely say whether there is a single best technique to recommend for the purpose of data quality improvements, it would helpful if at least we can suggest which are the commonly-adopted techniques that may not be needed to examine, since, on average, they produce lower performance than others.

Fig. 6.1 shows box plots of the rankings in terms of *wins-losses* summarized from the 12 selected datasets. A technique exhibits higher values of *wins-losses* achieves better ranks. For example, as shown in Table 6.1, **Kmeans** achieves rank #1 for the Albrecht and the Deshanais datasets, while it achieved very poor rank, i.e., rank #6 for the ISBSG-Insurance dataset. This plot is similar to the Dem̌sar's significance plot [72], with the main difference being that the ranking of this plot is based on the Brunner test rather than Nemenyi test. The modification of the test method is mainly because the Nemenyi test was strongly concerned by Mittas and Angelis [25] mainly because it can easily mislead the comparison results due to a lack of statistical power.

The results of Fig. 6.1 are in agreement with that of our investigation for the second research question, and more plausibly, the results of this figure appear to be more obvious to state that most of the commonly-adopted data quality improvement techniques are not fully suitable for ABE. Fig. 6.1 illustrates that only **Kmeans** more frequently produce better generalized performance than applying any other techniques nor applying no technique.

6.5. Discussion

The results based on the *Improved stable ranking method* suggested that applying the commonly adopted data quality improvement techniques to datasets before their applications did not consistently improve the estimation performance when estimating the software effort using ABE. However, in our opinion, the possibility to improve the performance is intuitive to suggest ones who are going to carry out an SEE with abundant computing resources and timescale to fit their data with **ABE-Best** and many data quality improvement techniques, such as those selected in this study. Even if an attempt to do that would be costly, we strongly feel that it is worthwhile because of the importance of successful SEE is clearly associated with the success rate of software development projects.

On the other hand, when the computing resources and timescale appear to be constraint, we rather suggest that it is not worthwhile to allocate more resources or budgets to facilitate a search for the most-fit data quality improvement technique for a given dataset. Based on the results of Fig. 6.1, fitting the data with **Kmeans** and seeing whether there will be any performance improvement seem to be sufficient for SEE in the situations when the computing resources and timescale very limited.

6.5.1 On the Quality Improvement of Software Effort Estimation Datasets

The results of all the tables belong to this study say in common that all the selected techniques produced almost the same performance. This may imply that both outliers and inconsistent cases are commonly contaminated in SEE datasets. For the reason that these issues are commonly seen, we strongly believed that it is mainly due to the data collection process. For example, even though the ISBSG datasets are commercially available, we found massive amount of project cases containing a very high amount of missing values even if they were labelled in ranked A in both terms of quality and UFP integrity. Also in other tera-PROMISE datasets, we found many datasets lacked the meta-information to describe themselves. These could cause much difficulty in practice. Without such meta-information available, it is very difficult to know the actual level of quality of a data set and to further interpret those data to improve the estimation process.

In a different direction, we expect software effort dataset to provide structure of metadata along with the attributes of dataset. This availability will greatly benefit to the more comprehensive and accurate estimation. For instance, the well-structured data such as protein sequences greatly contribute to the more comprehensive and accurate protein function prediction [83]; even though; the estimation technique commonly used in protein function prediction seems to be very similar to the ABE0 model. Furthermore, data sets commonly used in SEE as well as other empirical research studies were less likely to provide sufficient information to understand the numerous changes that occur over time (e.g. technologies and people). Since prediction models are based on historical data, any model would produce a more accurate estimate if the entire set of data being used is sufficiently relevant. We believe that an available of filtering technique or any approach that is able to maintain the level of relevance will greatly contribute to software industries.

6.6. Conclusion

It has been widely accepted that data quality is one of the important factors associated with the estimation accuracy of SEE methods [26, 76, 79]. However, according to our survey conducted in Chapter 2 and several previous studies in this area, the quality of data was much less focused than on developing effort estimation methods [77, 78]. The study of this chapter revisited the review and evaluation of 6 data quality improvement techniques. 5 out of the 6 techniques are the commonly adopted outlier elimination techniques, and the other one is an inconsistent data elimination technique.

Using the *Improved stable ranking method*, where its credibility was empirically proofed by the study of Chapter 4, we can rather conclude that applying any of the common data quality improvement techniques to a training dataset before its application does not necessarily produce a higher accuracy, compared with using full training datasets. Even though some results of this study reported that some techniques may be able to improve the quality of data, we see them as minority cases that will be less likely occur in practice. Hence, unless having abundant computing resources and timescale, we do not suggest ones to fit their model and data to optimize their estimation processes with any of the commonly-adopted data quality improvement techniques examined in this study.

Since our main findings of this study are contradictory with many of the previous research, we therefore highly suggest future studies to replicate this work or to further investigate these techniques in other areas of application. We strongly believe that a more comprehensive understanding regarding the level of data quality is hugely important for the research communities of empirical software engineering as well as the industry practices.

Chapter 7

Comparing ABE-Best with Other Well-Known SEE Models

In the previous chapter, we coined **ABE-Best** as the ABE model built from best combination of approaches, concluded from all the earlier chapter of this thesis. In this chapter, we compared this **ABE-Best** model with other 7 machine learning models that were commonly adopted as software effort estimators. Even though the study by Keung et al. [9], where the original *Stable ranking method* was proposed, concluded that several variants of ABE0 was ranked among the most accurate model-based effort estimation methods, However, those variants of ABE0 examined in the study by Keung et al. [9] were only ranked among the moderate performers in our studies in the previous chapters. For example, all the variants examined in the study by Keung et al. adopted **UAVG** as solution adaptation technique, where the results of Chapter 4 shows that ABE models adapted with this adaptation technique consistently performed worse than many other techniques. Furthermore, our proposed **LSA-X** in 5 was conclusively the best techniques in general situations. This therefore motivated us to revisit the ranking of ABE models built with knowledge discovered in our studies with other well-known model-based software effort estimators.

7.1. Background of the Effort Estimators based on Machine Learning Model being Compared with ABE in this Study

Machine learning models examined in this study were the same set of models examined in the study of Keung et al. [9]. Specifically, we compared **ABE-Best** with the following 7 machine learning models:

1. Four regression-based methods: **LReg**, **SWReg**, **PCReg**, and **PLSReg**;
2. Two decision tree-based methods: **CART(yes)** and **CART(no)**;
3. One neural networks-based method: **NNet**.

LReg is a simple linear regression algorithm. It is commonly used for modeling linear relationship between a target variable and one or more descriptive variables. **SWReg** or stepwise regression is a process of determining the best fit regression model with its corresponding best fit feature subset from a given dataset. The procedure of **SWReg** is to continually add project features to a collection that represents the best feature subset, or remove them from the collection, until there is no additional improvement after any feature is added to the collection nor is removed from the collection. **PCReg** or principle component regression is a regression method based on principle component analysis (**Pca**) [40]. It is similar to the simple linear regression, but it uses the **Pca** technique to extract components of independent variables and used information based on the extracted variables for estimating the unknown regression coefficients in the regression model. **PLSReg** or partial least squares regression is another regression method based on principle component analysis [40]. **PLSReg** is similar to **PCReg** but **PLSReg** utilized information of the target variable in its calculation. Therefore, its objective function can include the minimizing of the errors between the descriptive variables and the target variable. As reported by Keung et al. in [9] **PLSReg** generally provides better accuracy than **PCReg**.

CART(yes) and **CART(no)** are two commonly seen configuration variants of the classification and regression tree *CART* algorithm. The procedure of the

CART algorithm is to recursively partition the training set based on GINI index, and its goal is to generate a binary decision tree that best explains the target variable. After a full **CART** is generate from the given training set, **CART(yes)** prunes the generate tree using cross-validation to find the subtree that produces the minimum error rate, the this subtree is then used as the resulting model. On the other hand, **CART(yes)** used the full *CART* without any pruning applied.

NNet or neural networks is a learning model that mimics the functions of human nervous system. Artificial neurons called units are placed in layers and each unit in the contiguous are fully connected across the layers to form the network. Input units and output units are at the different ends of the network, and there are one or more layers called *hidden layers* separate the two ends. The task of neural networks is to learn the weight value assigned to each edge in the network that produce the output value as closest to the expected output.

7.2. Evaluation Procedure

The variants of software effort models compared in this study are listed as followed:

1. ABE-Best, an ABE model being tailored with:
    - Similarity measure: **Euclidean**;
    - Feature subset selection: **All**;
    - Approach to identify number of analogues: **Best-k**.
    - Normalization method: **Log**;
    - Solution adaptation techniques: **LSA-X** for datasets for datasets with $r(Pr, PrFactor)$ equal to or greater than 0.30, and **LSA** for datasets with $r(Pr, PrFactor)$ less than 0.30;
2. 7 machine learning models (**LReg**, **SWReg**, **PCReg**, **PLSReg**, **CART(yes)** and **CART(no)**, and **NNet**) configured with:
    - Feature subset selection: **All**, **Sfs**, **Swvs**, **Pca**;

- Normalization method: **None**, **Interval0-1**, **Log**;

Thus, 1 ABE-Best + (7 × 4 × 3 variants of machine learning models) = 85 effort models compared in this study. The comparison procedures are the same as done in Chapter 4, where 2,304 variants of ABE models were evaluated and compared. That is after the 85 variants of software effort models were executed by the leave-one-out approach, their estimation errors were recorded in terms of *MAR*, *MdAR*, *SD*, *LSD*, *RSD*. Then, pairwise comparisons were carried out between each of the 85 variants × 84 others, using the *win-tie-loss* statistics. Finally, the overall performance of each similarity measure was aggregated by the total summations of *wins*, *losses*, and *wins-losses* across all the selected datasets and the 5 robust error measures.

7.3. Result

Table 7.1 shows the ranking results of the 85 effort models examined in this study. The ranking is sorted by the values of *wins-losses* in descending order. The results of this table show that **ABE-Best** significantly outperformed all the 84 other models in terms of *losses* and *wins-losses*. With 393 *losses*, **ABE-Best** performed approximately 3 times better than the second rank. Also in terms of *wins-losses*, leading the second rank by over 400 *wins-losses* was the largest gaps among the gaps between any other two models whose values of *wins-losses* are positive.

In contrast from the results in terms of *losses* and *wins-losses*, the values of *wins* of **ABE-Best** was not the best. The *wins* value of 1,930 was ranked #22 out of the 85 models. However, when we only focused on the *wins* column of this table, we see that almost the entire list of models with positive values of *wins-losses* had the values of *wins* between 1,800 and 2,100. In other word, this result indicates that there was no significant cut off threshold in terms of *wins*. Hence, it appears to be valid to focus and interpret only the ranking lists based on *losses* and *wins-losses* to distinguish superior models from others. Based on the ranking agreement established between the ranking based on *losses* and *wins-losses*, we concluded that **ABE-Best** outperformed all the 85 models examined in this study, in terms of overall performance.

Table 7.1. Results of the comparison between **ABE-Best** and 84 combinations of 7 effort models × 4 feature subset selection methods (FSS) × 3 normalization approaches (Norm), sorted by the sum of their *wins-losses* seen in all performance measures and all data sets.

Rank	Wins	Losses	Wins-Losses	FSS	Norm	Model
1	1937	393	1544		ABE-Best	
2	2210	1108	1102	Pca	Log	SWReg
3	2084	990	1094	Swvs	Interval0-1	SWReg
4	2082	992	1090	Swvs	None	SWReg
5	2180	1099	1081	Swvs	None	PCReg
6	2215	1177	1038	Swvs	None	PLSReg
7	2100	1089	1011	Swvs	Interval0-1	LReg
8	2100	1089	1011	Swvs	None	LReg
9	2045	1061	984	Sfs	Interval0-1	SWReg
10	2130	1256	874	Swvs	Interval0-1	PLSReg
11	1984	1165	819	Sfs	None	SWReg
12	1986	1171	815	All	Interval0-1	SWReg
13	1973	1199	774	All	None	SWReg
14	1988	1259	729	All	Log	SWReg
15	2029	1336	693	Swvs	Log	SWReg
16	1952	1351	601	Sfs	Log	SWReg
17	2015	1445	570	All	None	PLSReg
18	1889	1332	557	Swvs	Log	LReg
19	1939	1388	551	Swvs	Log	PLSReg
20	1999	1449	550	Sfs	None	PLSReg
21	1894	1396	498	Swvs	Interval0-1	PCReg
22	1858	1385	473	Swvs	None	CART(yes)
23	1871	1398	473	Swvs	Interval0-1	CART(no)
24	1858	1385	473	Swvs	None	CART(no)
25	1871	1398	473	Swvs	Interval0-1	CART(yes)
26	1732	1267	465	Pca	Interval0-1	SWReg
27	1954	1498	456	Sfs	None	PCReg
28	1926	1477	449	Swvs	Log	PCReg
29	1721	1277	444	Pca	None	SWReg
30	1945	1504	441	Pca	None	PCReg
31	1924	1545	379	All	None	PCReg
32	1900	1526	374	Pca	None	PLSReg
33	1693	1396	297	Swvs	Log	CART(no)
34	1693	1396	297	Swvs	Log	CART(yes)
35	1713	1501	212	Pca	Interval0-1	PLSReg
36	1769	1645	124	All	Interval0-1	PLSReg
37	1631	1584	47	Pca	Log	PLSReg
38	1503	1488	15	Pca	None	LReg
39	1483	1499	-16	Sfs	Interval0-1	CART(no)
40	1483	1499	-16	Sfs	Interval0-1	CART(yes)
41	1342	1370	-28	Sfs	Log	CART(yes)
42	1342	1370	-28	Sfs	Log	CART(no)
43	1769	1808	-39	Sfs	Interval0-1	PLSReg
44	1493	1537	-44	Pca	Interval0-1	LReg
45	1448	1528	-80	Pca	Interval0-1	PCReg
46	1315	1440	-125	Sfs	None	CART(no)
47	1315	1440	-125	Sfs	None	CART(yes)
48	1285	1435	-150	All	Log	CART(no)
49	1285	1435	-150	All	Log	CART(yes)
50	1281	1439	-158	All	None	CART(no)
51	1281	1439	-158	All	None	CART(yes)
52	1365	1547	-182	Pca	Log	PCReg
53	1268	1466	-198	All	Interval0-1	CART(yes)
54	1268	1466	-198	All	Interval0-1	CART(no)
55	1536	1738	-202	Pca	Log	CART(no)
56	1536	1738	-202	Pca	Log	CART(yes)
57	1551	1772	-221	Pca	Interval0-1	CART(no)
58	1551	1772	-221	Pca	Interval0-1	CART(yes)
59	1526	1748	-222	Pca	None	CART(no)
60	1526	1748	-222	Pca	None	CART(yes)
61	1374	1713	-339	Swvs	Log	NNet
62	1474	1902	-428	Sfs	Log	LReg
63	1401	1856	-455	Pca	Log	LReg
64	1302	1758	-456	All	None	LReg
65	1239	1780	-541	Sfs	None	LReg
66	1259	1802	-543	All	Interval0-1	LReg
67	1253	1805	-552	Sfs	Interval0-1	LReg
68	999	1625	-626	All	Interval0-1	PCReg
69	1423	2059	-636	All	Log	LReg
70	952	1614	-662	Sfs	Interval0-1	PCReg
71	1120	1961	-841	Swvs	Interval0-1	NNet
72	1016	1865	-849	Pca	Interval0-1	NNet
73	1061	1933	-872	Pca	Log	NNet
74	1000	1881	-881	Pca	None	NNet
75	1100	1981	-881	Swvs	None	NNet
76	1218	2118	-900	Sfs	Log	PLSReg
77	855	1791	-936	All	None	NNet
78	1212	2166	-954	All	Log	PLSReg
79	844	1802	-958	All	Interval0-1	NNet
80	765	1728	-963	Sfs	Log	PCReg
81	767	1780	-1013	All	Log	PCReg
82	871	1917	-1046	Sfs	None	NNet
83	883	2076	-1193	Sfs	Interval0-1	NNet
84	784	2370	-1586	All	Log	NNet
85	760	2542	-1782	Sfs	Log	NNet

To conclude whether **ABE-Best** was also performed the best in terms of robustness and had the best generalized performance, we further examined the results by datasets. In this investigation, we only focus on the superior models by assuming them all as the candidates model that would performed the best. From Table 7.1, we defined the cut off threshold at the models falling between rank #9 and rank #10, and determined the models on the top 9 ranks as superior models. This cut off was decided based on the performance in terms of *losses* and *wins-losses*. All the effort models placing above this cut off threshold having their values of *losses* and *wins-losses* consistently higher than all the other models located below this cut off. That is, the values of *losses* of the models on the top 9 ranks were all lower than 1,200 and that of most of the other models were higher than 1,200. Also in term of *wins-losses*, the gap between models at the rank #9 and the rank #10 with the difference of 110 *wins-losses* was larger than that of between other machine learning-based models whose values of *wins-losses* were positive. Therefore, without any further analysis of the ranking agreement performed, we believe that this cut off threshold between rank #9 and rank #10 is the most appropriate to suggest superior models over all others.

Fig.7.1 shows box plots of the ranking results and the values of *wins-losses*. Each box aggregates the results from the 12 datasets being used in all the studies of this thesis. The plots of this figure showed only the models on the top 9 ranks of Table 7.1. In the above box plot that indicates the ranking results, **ABE-Best** was among the best models where **SWReg-Pca-Log** and **SWReg-Sfs-Interval0-1** also showed outstanding performance. These 3 models have very close median ranks, and all of them has reached rank #1 for at least one dataset. Slightly better performance of **ABE-Best** is showed by the its rank at the upper quartile and its maximum ranks, where all two **SWReg-Pca-Log** and **SWReg-Sfs-Interval0-1** had clearly worse results. That is the two models had their worse ranks fall at the ranks higher than 60. Therefore, the results of this figure allow us to suggest that **ABE-Best** also performed best in terms of stability.

For the other plot located at the lower part of Fig.7.1, this box plot indicate that **ABE-Best** was clearly the best model of choice as all the values at its upper quartile, its median and its lower quartile were higher than that of all

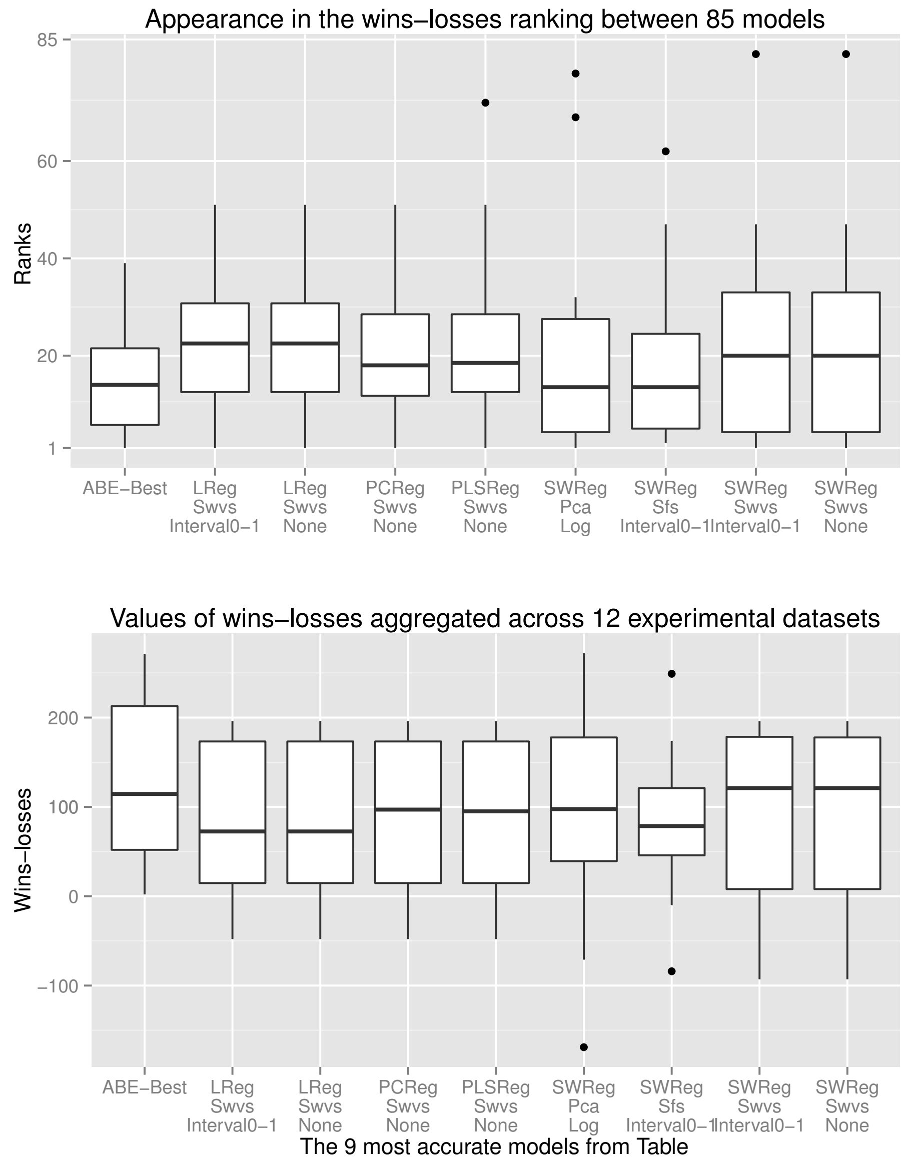

Figure 7.1. Box plots of the ranking and the total values of *wins-losses* of the 9 superior models indicated from the results of Table 7.1. Each plots aggregate the results across all error measures and all the 12 selected datasets.

the other 8 models. More interestingly, in contrast to all the other models, the minimum value of *wins-losses* of **ABE-Best** in all the 12 selected datasets were above 0. This means that **ABE-Best** was never *losses* more than *wins* in all the comparisons with any of the other 84 models. Therefore, this finding leads to a strong conclusion that **ABE-Best** was also the best performer in terms of generalized performance and robustness.

7.4. Discussion

All the results of this chapter strongly recommend our discovered **ABE-Best** to be the estimation model of choice for SEE, whether the required performance are expressed in terms of overall performance, generalized performance, stability, or robustness. Incidentally, before proceeding to conclude **ABE-Best** as the model most suitable for practical use, it may be necessary to address the costs of applying it in the real-world application.

Using a simple time complexity analysis, we do not hesitate to say that **ABE-Best** are among the most lightweight prediction models. This is because, using all features, applying **Log** and adopting **LSA** are calculated at almost no computing cost, the calculation for the **Euclidean** distance and the **Best-k** algorithm are in a complexity of $\mathcal{O}(n^2)$, and finally computation of the Pearson's correlation required by **LSA-X** is in $\mathcal{O}(n^3)$, i.e., its formula shows the same complexity as that of the linear regression, which is know to be an $\mathcal{O}(n^3)$ algorithm. In other word, the computation of all the components of **ABE-Best** are solvable in a polynomial time. Compared among the other commonly-adopted machine learning models, our observation suggests **LReg**, **PCReg**, and **PLSReg** are more lightweight than the other 4 selected models. Based on this simple time complexity analysis as described above, **ABE-Best** can also be classified into this group of the lightweight predictive models. Taken together, the results of this analysis of time complexity strongly suggest a role for **ABE-Best** in the real world application.

7.4.1 On the Successful of ABE-Best

The results of this study highlight that **ABE-Best** was conclusively better than the other 7 common model-based SEE models by all means of overall performance, generalized performance, stability, and robustness. In our opinion, the successful of **ABE-Best** was mainly because of the *Improved stable ranking method*. With an available of this stable ranking method, we were able to assess the approaches commonly extended to ABE models to improve the estimation performance, and determined the relative best approaches for each of the essential components of ABE models.

In the past, one of the greatest difficulty to determine the superior approaches or combination of approaches that commonly extended to ABE models was mainly due to the existance of conclusion instability issue, where different studies reported greatly diversified results. As a result, a replication of all approaches seems to be the only valid option for such purpose. Furthermore, if one reads through the literature on the approaches that improved the main components or ABE models, somewhat inclusive results were also commonly encounters. This can be observed by examining if approaches with very similar theoretical concepts produced greatly diversified performance results. For example, in a study by Azzeh [45], where the 8 commonly adopted solution adaption techniques were reviewed and replicated, techniques with more similar theoretical concepts than others did not performed with similar performance. For example, in contrast to our results achieved by the *Improved stable ranking method*, **RTM** performed more similar to **GA** than **LSA** in the replicated study by Azzeh [45]. Hence, we strongly believed that the **ABE-Best** model was not discovered before the studies of this thesis.

7.4.2 The 7 Common Model-Based Effort Estimation Methods

Based on the results of Table 7.1 and Fig.7.1, the findings and future directions of the 7 commons effort estimation models are as follows:

Finding 1: SWReg was being both superior effort model and superior feature selection method. If we observe it performance by excluding **ABE-Best**, 6 out of the 8 models on the top 9 ranks of Table 7.1 selected **Swvs** as feature subset selection method, and 4 out of these 8 models were **SWReg**. Furthermore, **SWReg** is the only model of the common 7 models where all its variants had positive *wins-losses* values. Hence, we strongly recommend a more detailed study on stepwise regression for the purposes of SEE to be an interesting continual research topic in this the field.

Finding 2: In contrast to ABE, we could not recommend the single best normalization method for any other models. Table 7.1 shows that all the 3 normalization approaches examined throughout this thesis were almost equally distributed across the ranking of 85 variants of software effort models. And for the superior model, i.e., **SWReg-Swvs**, all of its 3 variants composing from the 3 normalization approaches achieved high ranks in Table 7.1. This means that choosing the right model and the right feature subset selection methods contributes more on the achievable estimation performance when adopting all the other well-known models. Hence, the cost of adopting those models in practice seems to be higher than that of **ABE-Best**.

Finding 3: Fig. 7.1 shows that 8 out of the 9 superior models had reached rank #1 for at least one dataset. This means that even if **ABE-Best** was conclusively the superior methods by all means of overall performance, generalized performance, stability and robustness, there is also situations where it cannot show outstanding performance. This finding suggests two important future directions of SEE research studies to explore (1) a selection guideline for models given different situations, and (2) a methods to build an ensemble of models [2] based on the superior models concluded in this study. The two directions would contribute greatly towards an improvement of the both the process and the maximum achievable accuracy of SEE in practice.

7.5. Conclusion

The study of this chapter compares **ABE-Best** i.e., the ABE model built with the superior approaches concluded by the studies in earlier chapters, with 84 other variants of machine learning-model based effort estimation methods. These

84 variants of effort models were cross-generated from 7 well-known effort estimation models (**NNet**, **LReg**, **SWreg**, **PCReg**, **PLSReg**, **CART(yes)**, and **CART(no)** [2]) $\times$ 4 feature subset selection methods (**All**, **Sfs**, **Swvs**, and **Pca**) $\times$ 3 normalization approaches (**None**, **Interval0-1**, and **Log**). All of feature subset selection methods and normalization approaches were examined with ABE models earlier in this thesis.

Leveraged by our *Improved stable ranking method*, the superior approaches in the 5 most important components of ABE were successfully identified and composed into **ABE-Best**. The evaluation of this study showed that **ABE-Best** consistently outperformed the 84 other variants of software effort models examined in this study by all means of overall performance, generalized performance, stability, and robustness. Hence, we strongly recommend **ABE-Best** for the process of SEE in practice.

Chapter 8

Threats to Validity

8.1. Internal validity

Internal validity questions whether or not different conclusions can be drawn with regard to the different parameter setups for the estimation. There are two possible issues of internal validity in the studies of this thesis. One issue is the single sampling method we used and the other is in regards to the use of pairwise comparisons to determine the ranking results. In this study, we choose the leave-one-out as the sampling method to generate training/test instances, over the other widespread sampling method, N-way cross validation. Following Hastie et al. [84] and Kocaguneli et al. [2], the leave-one-out approach was our approach of choice for all the studies of this thesis because it generates lower bias estimates and higher variance estimates than the cross validation. Most importantly, leave-one-out is a deterministic algorithm so it does not rely on randomization in which the N-way cross validation does. This will greatly contribute towards future replications of all the studies of this thesis.

In regards to the use of pairwise comparisons to determine the ranking results. Mittas and Angelis [25] strongly concerned a simple pairwise comparison based on Nemenyi test, a commonly-used test method in software engineering studies, because it can easily mislead the comparison results due to a lack of statistical power. To avoid this undesirable effect, we adopted *win-tie-loss* statistics [72] in all the experiments where the statistical tests were performed in our studies. For the statistical test method adopted as the essence of the *win-tie-loss* statistics, we selected the Brunner test, one of the most robust alternative of the Wilcoxon rank-sum test based on the confidence interval given Brunner et al.'s *p-hat* metric [56] to produce the statistical significance.

Further than the performance results produced by the *win-tie-loss* statistics, we also focused on finding agreement established across different interpretations of the statistic. Currently and at the time of writing, research studies which analyzed the agreement in pairwise comparisons based on this statistic have continually produced important research conclusions for the communities of SEE such as in a study by Kocaguneli et al. [2]. Therefore, for the present studies of this thesis, we believe that this threat did not impose a threat to the internal validity of this study.

8.2. External validity

External validity questions whether the results can be generalized. In this study, we used a wide range of datasets from two distinct sources which are the tera-PROMISE software repository [67] and the ISBSG dataset [62]. In total of 582 software projects in 12 datasets, the selected datasets are varied in both size, characteristic, and distribution, as they came from different organizations and were developed in different periods of time. In a systematic review by Kitchenham et al. [4], the median value of the samples commonly used in effort estimation literature is only 186. Hence, we believe that both the number and diversity of datasets used in this study offers a higher degree of validity than many other studies in the area of SEE, and is sufficient to justify generalizing our results.

8.3. Construct validity

Construct validity questions whether or not we are measuring what we intend to measure. SEE studies are often criticized for inadequate use of performance measures such as error measures. For example, use of *MMRE* by itself is commonly seen in the literature [24], despite widespread criticism of *MMRE* as an inappropriate and biased measure. To minimize the construct validity possible introduced by this common issue, we used only the robust error measures suggested from a review study by Foss et al. [24] in our studies. Furthermore, as suggested by Kitchenham et al. in [58], the solely use of the performance measures without an appropriate statistical test applied can mislead the comparison results. Hence, we the Brunner test [56] as the essence of the robust evaluation procedure adopted in all the studies of this thesis.

Chapter 9

Final Remarks

9.1. Future Work

In this section, we emphasize the interesting and important future research directions from our viewpoints, regarding the results and discussions from all the previous chapters.

9.1.1 Future Directions of ABE

The name ABE-Best was coined as being the ABE model tailoring by the common approaches that perform best in general situations. In our opinion, there is abundant room for further progress in improving ABE in many other possible directions, such as to combine ABE with other machine learning-based prediction models or to adopt other models as a part of ABE. Recently and at the time of writing, building ensemble of multiple methods is strongly recommended by Kocaguneli et al. [2] as a comprehensive and effective way to further increase the maximum achievable estimation performance of the existing effort models. The successful approaches suggested by Kocaguneli et al. [2] is to build an ensemble of models by aggregating the estimated effort values from multiple successful models, evaluated prior to the estimation. It is non-trivial that the ABE-Best model determined in the studies of this thesis can further increase the achievable performance of the ensemble of models, where successful ensemble models were suggested be built from successful solo models.

To aim for greater possibility of the performance improvement, in our opinion, we rather see that a more effective way to combine multiple models is to exploit the main advantage of each individual. From the results of the study of Chapter 7, significantly different performance was clearly seen between ABE and any other selected models, whereas the performance between those other models themselves was not much different. This may have dropped a hint that ABE and any other models maybe based on very different theoretical concepts. We see that the process of ABE can be considered mainly as having two steps: similar project cases retrieval and project cases adaptation. These two steps can also be viewed as dissimilar project case filtering and following by the effort modeling. Compared with other models, theoretically, the cases adaptation procedures commonly used in the effort modeling step of ABE may not be much sophisticate as other machine learning models; yet, ABE was proven to be a very successful effort estimator. This can imply that filtering the dissimilar project cases have played a very important role and contributed to the success of ABE. Based on this speculation, it is very interesting for us to further exploit this mechanism with other effort models. That is either applying the first step of ABE (the retrieval of similar project cases) prior to adopt a successful effort model, such as **SWReg**, or to adopt a successful effort model as a case adaptation method for ABE, have a potential to significantly increase the maximum achievable performance of SEE.

9.1.2 On the Data Quality Improvement of Software Engineering Predictive Modeling Data

Other important and interesting future direction stemmed from the studies of this thesis is to further study the influence of data quality on the estimation accuracy. Specifically, the results of the study in Chapter 6 indicated clearly that we are still a long way from being able to define a standard methodology to improve the data quality of empirical software engineering datasets. That is, while applications of data quality improvement techniques have known to be one of the performance factors of software engineering predictive modeling activities, our findings were greatly deviated from this common belief. Two studies of Bosu and MacDonell [77, 78] extensively reviewed other studies in empirical software engineering and suggested notes in agreement with our findings where data quality problems can be addressed more generally. Implications derived from their studies are that we not only lack the method to handle these contaminated data, but we also lack the standard measure to quantify the level of the overall quality of a dataset. This means that whether ones could suggest a procedure or method to eliminate the data points being contaminated in the entire dataset based on one main aspect such as being inconsistent, we would still unable to know whether the overall level of quality of the entire dataset are also increased. In other word, a definitive answer to a question "How can we know when the quality of data is sufficiently high?" is yet available.

Fitting multiple estimation models with a dataset and comparing the estimation performance before and after applying a data quality improvement technique [79,85], such as the procedure adopted in our study of Chapter 6, has been one of the most common procedures to measure the performance of data quality improvement techniques. However, we would hesitate to say that this evaluation procedure is sufficiently reliable. This is because it still lacks an adequate theoretical basis as to justify whether the improved prediction performance is not coincidental caused by any other factors. Moreover, based on a taxonomy of data quality challenges in empirical software engineering study proposed by Bosu and MacDonell [77], there are remaining many data quality issues yet explored in SEE research studies. Hence, we suggest future works to further underline the importance of the quality of data and to seek for adequate approach to measure the overall quality of datasets. In our opinion, a better understanding of data on its quality and a more availability of data quality improvement techniques will greatly contribute to the SEE processes as well as any other domains related to model-based prediction and estimation.

9.2. Conclusion

Many long-standing questions subject to a debate in SEE were mainly due to instability performance conclusions, where various research studies reported divergent performance results of different approaches proposed to improve the accuracy. The research studies presented in this thesis are aimed at overcoming the instability of the performance conclusions. Leveraged by our improvement to the “Stable ranking method” [9], by improving the theoretical validity of the method and having it supported by empirical evidences, all the studies carried out under this thesis were able to answer many important unanswered questions in regard to ABE, such as “What are the essential components of ABE that are strongly associated with its estimation performance?”, “In each dimension of the essential components of ABE, which are the approaches being superior to others?”, “Among all the possible combinations of approaches, can we determine the best one?”, “What is the theoretical concepts that made ABE become more accurate when the most similar case is deviated from the new cases?”, “Does an application of data quality improvement techniques prior to performing an estimation consistently improve the estimation accuracy of ABE model?”. Being able to answer these questions would greatly contribute to both the industry and research communities. For example, the importance of an effort estimation task make it critical for industry practices to perform the estimation with the most accurate and reliable effort model possible. Also, for the research communities, an ability to indicate superior approaches from the existing studies would make a better benchmark standard for any future proposals of software effort estimation methods.

From all the studies throughout this thesis, the following findings are our concluding remarks:

1. We strongly recommend our proposed *Improved stable ranking method* for future research studies where multiple prediction/estimation methods are reviewed and compared as well as for studies aiming to propose a new method.

2. We recommend **ABE-Best**, which is a model adopted with the following approaches to be a new standard benchmark for future research studies in SEE, as well as to be the model of choice for industry practice:

    - Apply **Log** transformation to all the continuous features of a dataset.
    - Use all project feature i.e., no feature subset selection is needed.

- Adopt the **Euclidean** distance function as the similarity measure in the cases retrieval process.
- For solution adaptation process, use our proposed **LSA-X** technique for datasets with $r(Pr, PrFactor)$ i.e., the highest correlation strength between productivity and its influential factor variable, equal to or greater than 0.30; otherwise, use the simple **LSA** technique.
- Determine number of analogues by fitting a dataset using the Baker's **Best-k** approach [38].

3. Compared with 7 other common effort estimation models (**NNet**, **LReg**, **SWreg**, **PCReg**, **PLSReg**, **CART(yes)**, and **CART(no)**), **ABE-Best** significantly outperforms by means of overall performance, generalized performance, stability, and robustness.

4. Productivity plays an important role in giving success estimation. A further study with more focus on Productivity is therefore suggested for ABE as well as other model-based effort estimation methods.

5. Many well-known data quality improvement techniques appear not to be suitable for ABE. Therefore, in our opinions, more extensive studies on this topic needs to be undertaken before the association between data quality and the estimation performance is more clearly understood. Notwithstanding, we strongly believe that a more comprehensive understanding regarding the level of data quality is hugely important for the research communities of empirical software engineering as well as the industry practices.

References

[1] E. Kocaguneli, T. Menzies, J. Keung, D. Cok, and R. Madachy, "Active learning and effort estimation: Finding the essential content of software effort estimation data," *IEEE Trans Softw Eng*, vol. 39, no. 8, pp. 1040–1053, 2013.

[2] E. Kocaguneli, T. Menzies, and J. Keung, "On the value of ensemble effort estimation," *IEEE Trans Softw Eng*, vol. 38, no. 6, pp. 1403–1416, 2012.

[3] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Trans Softw Eng*, vol. 33, no. 1, pp. 33–53, 2007.

[4] B. Kitchenham, E. Mendes, G. H. Travassos, *et al.*, "Cross versus within-company cost estimation studies: A systematic review," *IEEE Trans Softw Eng*, vol. 33, no. 5, pp. 316–329, 2007.

[5] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Information and Softw Technology*, vol. 54, no. 1, pp. 41–59, 2012.

[6] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Trans Softw Eng*, vol. 23, no. 11, pp. 736–743, 1997.

[7] J. Keung, "Software development cost estimation using analogy: A review," in *Proceedings of the 2009 Australian Software Engineering Conference*, pp. 327–336, 2009.

[8] J. W. Keung, B. Kitchenham, D. R. Jeffery, *et al.*, "Analogy-x: providing statistical inference to analogy-based software cost estimation," *IEEE Trans Softw Eng*, vol. 34, no. 4, pp. 471–484, 2008.

[9] J. Keung, E. Kocaguneli, and T. Menzies, "Finding conclusion stability for selecting the best effort predictor in software effort estimation," *Automated Software Eng*, vol. 20, no. 4, pp. 543–567, 2013.

[10] E. Mendes, I. Watson, C. Triggs, N. Mosley, and S. Counsell, "A comparative study of cost estimation models for web hypermedia applications," *Empirical Softw Eng*, vol. 8, no. 2, pp. 163–196, 2003.

[11] J. J. Cuadrado-Gallego, P. Rodríguez-Soria, and B. Martín-Herrera, "Analogies and differences between machine learning and expert based software project effort estimation," in *Proceedings of the 11th ACIS International Conference on Software Engineering Artificial Intelligence Networking and Parallel/Distributed Computing*, pp. 269–276, IEEE, 2010.

[12] X. Ren, Y. Dai, and L. Zhou, "Application in effort estimation of collaborative filtering," in *Proceeding of the 6th International Symposium on Computational Intelligence and Design*, pp. 330–333, IEEE, 2013.

[13] F. Walkerden and R. Jeffery, "An empirical study of analogy-based software effort estimation," *Empirical Softw Eng*, vol. 4, no. 2, pp. 135–158, 1999.

[14] E. Kocaguneli, T. Menzies, A. Bener, and J. W. Keung, "Exploiting the essential assumptions of analogy-based effort estimation," *IEEE Trans Softw Eng*, vol. 38, no. 2, pp. 425–438, 2012.

[15] M. Riaz, E. Mendes, E. Tempero, and M. Sulayman, "Using cbr and cart to predict maintainability of relational database-driven software applications," in *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, pp. 132–143, ACM, 2013.

[16] E. Paikari, B. Sun, G. Ruhe, and E. Livani, "Customization support for cbr-based defect prediction," in *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*, p. 16, ACM, 2011.

[17] K. El Emam, S. Benlarbi, N. Goel, and S. N. Rai, "Comparing case-based reasoning classifiers for predicting high risk software components," *J Syst Softw*, vol. 55, no. 3, pp. 301–320, 2001.

[18] T. M. Khoshgoftaar, N. Seliya, and N. Sundaresh, "An empirical study of predicting software faults with case-based reasoning," *Softw Qual J*, vol. 14, no. 2, pp. 85–111, 2006.

[19] J. Li and G. Ruhe, "Decision support analysis for software effort estimation by analogy," in *Proceeding of the 3rd International Workshop on Predictor Models in Software Engineering*, pp. 6–12, IEEE, 2007.

[20] L. Angelis and I. Stamelos, "A simulation tool for efficient analogy based cost estimation," *Empirical Softw Eng*, vol. 5, no. 1, pp. 35–68, 2000.

[21] N.-H. Chiu and S.-J. Huang, "The adjusted analogy-based software effort estimation based on similarity distances," *J Syst Softw*, vol. 80, no. 4, pp. 628 – 640, 2007.

[22] Q. Liu, X. Chu, J. Xiao, and H. Zhu, "Optimizing non-orthogonal space distance using pso in software cost estimation," in *Proceedings of the 38th Annual Computer Software and Applications Conference*, pp. 21–26, IEEE, 2014.

[23] T. Menzies, O. Jalali, J. Hihn, D. Baker, and K. Lum, "Stable rankings for different effort models," *Automated Softw Eng*, vol. 17, no. 4, pp. 409–437, 2010.

[24] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A simulation study of the model evaluation criterion mmre," *IEEE Trans Softw Eng*, vol. 29, no. 11, pp. 985–995, 2003.

[25] N. Mittas and L. Angelis, "Ranking and clustering software cost estimation models through a multiple comparisons algorithm," *IEEE Trans Softw Eng*, vol. 39, no. 4, pp. 537–551, 2013.

[26] Y.-S. Seo and D.-H. Bae, "On the value of outlier elimination on software effort estimation research," *Empirical Softw Eng*, vol. 18, no. 4, pp. 659–698, 2013.

[27] B. Boehm, C. Abts, and S. Chulani, "Software development cost estimation approaches—a survey," *Annals of Softw Eng*, vol. 10, no. 1-4, pp. 177–205, 2000.

[28] M. Shepperd, "Software project economics: a roadmap," in *Proceedings of the 2007 Future of Software Engineering*, pp. 304–315, IEEE, 2007.

[29] M. Shepperd, C. Schofield, and B. Kitchenham, "Effort estimation using analogy," in *Proceedings of the 18th International Conference on Software engineering*, pp. 170–178, IEEE Computer Society, 1996.

[30] B. W. Boehm, *Software Engineering Economics.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 1st ed., 1981.

[31] E. Alpaydin, *Introduction to machine learning.* MIT press, 2014.

[32] G. Nagpal, M. Uddin, and A. Kaur, "A comparative study of estimation by analogy using data mining techniques," *JIPS*, vol. 8, no. 4, pp. 621–652, 2012.

[33] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *AI Commun*, vol. 7, no. 1, pp. 39–59, 1994.

[34] B. Kitchenham and E. Mendes, "Software productivity measurement using multiple size measures," *IEEE Trans Softw Eng*, vol. 30, no. 12, pp. 1023–1035, 2004.

[35] D. R. Wilson and T. R. Martinez, "Improved heterogeneous distance functions," *J Artif Int Res*, vol. 6, no. 1, pp. 1–34, 1997.

[36] C. Kirsopp, E. Mendes, R. Premraj, and M. Shepperd, "An empirical analysis of linear adaptation techniques for case-based prediction," in *Proceedings of the 5th international conference on Case-based reasoning: Research and Development*, pp. 231–245, 2003.

[37] E. Mendes, N. Mosley, and S. Counsell, "A replicated assessment of the use of adaptation rules to improve web cost estimation," in *Proceedings of the*

International Symposium on Empirical Software Engineering, pp. 100–109, 2003.

[38] D. R. Baker, “A hybrid approach to expert and model based effort estimation,” Master’s thesis, Lane Department of Computer Science and Electrical Engineering, West Virginia University, 2007.

[39] L. C. Briand and I. Wieczorek, “Resource estimation in software engineering,” *Encyclopedia of software engineering*, 2002.

[40] J. Li, G. Ruhe, A. Al-Emran, and M. M. Richter, “A flexible method for software effort estimation by analogy,” *Empirical Softw Eng*, vol. 12, no. 1, pp. 65–106, 2007.

[41] P. Phannachitta, J. Keung, A. Monden, and K. Matsumoto, “Improving analogy-based software cost estimation through probabilistic-based similarity measures,” in *Proceedings of the 20th Asia-Pacific Software Engineering Conference*, pp. 541–546, IEEE, 2013.

[42] C.-J. Hsu and C.-Y. Huang, “Comparison and assessment of improved grey relation analysis for software development effort estimation,” in *Proceedings of the 2006 International Conference on Management of Innovation and Technology*, vol. 2, pp. 663–667, IEEE, 2006.

[43] D. Ju-Long, “Control problems of grey systems,” *Syst Control Lett*, vol. 1, no. 5, pp. 288–294, 1982.

[44] Y. F. Li, M. Xie, and T. N. Goh, “A study of the non-linear adjustment for analogy based software cost estimation,” *Empirical Softw Eng*, vol. 14, no. 6, pp. 603–643, 2009.

[45] M. Azzeh, “A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation,” *Empirical Softw Eng*, vol. 17, no. 1-2, pp. 90–127, 2012.

[46] M. Jørgensen, U. Indahl, and D. Sjøberg, “Software effort estimation by analogy and r̈egression toward the mean;” *J Syst Softw*, vol. 68, no. 3, pp. 253–262, 2003.

[47] Y. Li, M. Xie, and T. Goh, "Bayesian inference approach for probabilistic analogy based software maintenance effort estimation," in *Proceedings of the 14th IEEE Pacific Rim International Symposium on Dependable Computing*, pp. 176–183, IEEE, 2008.

[48] B. Sigweni and M. Shepperd, "Feature weighting techniques for cbr in software effort estimation studies: a review and empirical evaluation," in *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, pp. 32–41, ACM, 2014.

[49] E. Mendes, N. Mosley, and S. Counsell, "Early web size measures and effort prediction for web costimation," in *Proceedings of the 9th International Software Metrics Symposium*, pp. 18–39, IEEE, 2003.

[50] I. Myrtveit and E. Stensrud, "A controlled experiment to assess the benefits of estimating with analogy and regression models," *IEEE Trans Softw Eng*, vol. 25, no. 4, pp. 510–525, 1999.

[51] L. C. Briand, K. El Emam, D. Surmann, I. Wieczorek, and K. D. Maxwell, "An assessment and comparison of common software cost estimation modeling techniques," in *Proceedings of the 21st International Conference on Softw Eng*, pp. 313–322, ACM, 1999.

[52] E. Rashid, S. Patnaik, and V. Bhattacharya, "Enhancing the accuracy of case-based estimation model through early prediction of error patterns," in *Proceedings of the 2013 International Symposium on Computational and Business Intelligence*, pp. 208–212, IEEE, 2013.

[53] E. Kocaguneli and T. Menzies, "Software effort models should be assessed via leave-one-out validation," *J Syst Softw*, vol. 86, no. 7, pp. 1879–1890, 2013.

[54] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 1137–1143, 1995.

[55] D. W. Zimmerman, "Statistical significance levels of nonparametric tests biased by heterogeneous variances of treatment groups," *J Gen Psychol*, vol. 127, no. 4, pp. 354–364, 2000.

[56] E. Brunner, U. Munzel, and M. L. Puri, "The multivariate nonparametric behrens–fisher problem," *J Stat Plan and Inf*, vol. 108, no. 1, pp. 37–53, 2002.

[57] R. Wilcox, *Modern statistics for the social and behavioral sciences: A practical introduction.* CRC press, 2011.

[58] B. Kitchenham, "Robust statistical methods: why, what and how: keynote," in *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, p. 1, ACM, 2015.

[59] A. J. Albrecht and J. E. Gaffney, "Software function, source lines of code, and development effort prediction: A software science validation," *IEEE Trans Softw Eng*, vol. 9, no. 6, pp. 639–648, 1983.

[60] A. Bakır, B. Turhan, and A. B. Bener, "A new perspective on data homogeneity in software cost estimation: a study in the embedded systems domain," *Softw Qual Journal*, vol. 18, no. 1, pp. 57–80, 2010.

[61] B. Kitchenham and K. Känsälä, "Inter-item correlations among function points," in *Proceedings of the 15th International Conference on Software Engineering*, pp. 477–480, IEEE, 1993.

[62] R. Baggen, J. P. Correia, K. Schill, and J. Visser, "Standardized code quality benchmarking for improving software maintainability," *Softw Quality Journal*, vol. 20, no. 2, pp. 287–307, 2012.

[63] C. F. Kemerer, "An empirical validation of software cost estimation models," *Commun ACM*, vol. 30, no. 5, pp. 416–429, 1987.

[64] K. Maxwell, *Applied Statistics for Software Managers.* Englewood Cliffs, NJ. Prentice-Hall, 2002.

[65] Y. Miyazaki, M. Terakado, K. Ozaki, and H. Nozaki, "Robust regression for developing software estimation models," *J Syst Softw*, vol. 27, no. 1, pp. 3–16, 1994.

[66] T. Menzies, D. Port, Z. Chen, J. Hihn, and S. Stukes, "Validation methods for calibrating software effort models," in *Proceedings of the 27th international conference on Software engineering*, pp. 587–595, 2005.

[67] T. Menzies, M. Rees-Jones, R. Krishna, and C. Pape, "tera-promise: one of the largest repositories of se research data." `http://openscience.us/repo/index.html`, Jun 2015.

[68] E. Mendes and C. Lokan, "Replicating studies on cross-vs single-company effort models using the isbsg database," *Empirical Softw Eng*, vol. 13, no. 1, pp. 3–37, 2008.

[69] E. Kocaguneli, T. Menzies, and J. W. Keung, "Kernel methods for software effort estimation - effects of different kernel functions and bandwidths on estimation accuracy," *Empir Softw Eng*, vol. 18, no. 1, pp. 1–24, 2013.

[70] M. V. Kosti, N. Mittas, and L. Angelis, "Alternative methods using similarities in software effort estimation," in *Proceedings of the 8th International Conference on Predictive Models in Software Engineering*, pp. 59–68, 2012.

[71] B. Kitchenham and E. Mendes, "Why comparative effort prediction studies may be invalid," in *Proceedings of the 5th International Conference on Predictor Models in Softw Eng*, p. 4, ACM, 2009.

[72] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J Mach Learn Res*, vol. 7, pp. 1–30, 2006.

[73] M. Shepperd and G. Kadoda, "Comparing software prediction techniques using simulation," *IEEE Trans Softw Eng*, vol. 27, no. 11, pp. 1014–1022, 2001.

[74] G. Aquino and S. L. Meira, "An approach to measure value-based productivity in software projects," in *Proceedings of the 5th International Conference on Quality Softw*, pp. 383–389, Aug 2009.

[75] J. Kijne, T. Tuong, J. Bennett, B. Bouman, T. Oweis, *et al.*, "Ensuring food security via improvement in crop water productivity," *Challenge Program on water and Food: Background Paper*, vol. 1, 2003.

[76] M. Tsunoda, A. Monden, H. Yadohisa, N. Kikuchi, and K. Matsumoto, "Software development productivity of japanese enterprise applications," *Information Technology and Management*, vol. 10, no. 4, pp. 193–205, 2009.

[77] M. F. Bosu and S. G. MacDonell, "A taxonomy of data quality challenges in empirical software engineering," in *Proceeding of the 22nd Australasian Softw Eng Conference*, pp. 97–106, 2013.

[78] M. F. Bosu and S. G. MacDonell, "Data quality in empirical software engineering: A targeted review," in *Proceeding of the 17th International Conference on Evaluation and Assessment in Softw Eng*, pp. 171–176, 2013.

[79] P. Phannachitta, A. Monden, J. Keung, and K. Matsumoto, "Case consistency: a necessary data quality property for software engineering data sets," in *Proceeding of the 19th International Conference on Evaluation and Assessment in Softw Eng*, p. 19, 2015.

[80] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection*, vol. 589. John Wiley & Sons, 2005.

[81] J. Agulló, C. Croux, and S. Van Aelst, "The multivariate least-trimmed squares estimator," *Journal on Multivariate Analysis*, vol. 99, no. 3, pp. 311–338, 2008.

[82] D. Beeferman and A. Berger, "Agglomerative clustering of a search engine query log," in *Proceeding of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 407–416, 2000.

[83] L. Lan, N. Djuric, Y. Guo, and S. Vucetic, "Ms-knn: protein function prediction by integrating multiple data sources," *BMC bioinformatics*, vol. 14, no. Suppl 3, p. S8, 2013.

[84] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning: Data mining," *Inference and Prediction*, 2009.

[85] K. Toda, A. Monden, and K. Matsumoto, “Fit data selection based on project feature for software effort estimation models,” in *Proceeding of the International Conference on Advances in Computer Science and Engineering*, pp. 82–88, 2010.

[86] J. Keung, “Empirical evaluation of analogy-x for software cost estimation,” in *Proceedings of the 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 294–296, ACM, 2008.

[87] N. Cliff, “Dominance statistics: Ordinal analyses to answer ordinal questions.,” *Psychol Bull*, vol. 114, no. 3, p. 494, 1993.