2021

Change Request Prediction and Effort Estimation in an Evolving Software System

Lamees Abdullah Alhazzaa
University of Denver

Change Request Prediction and Effort Estimation in an Evolving Software System

Abstract

Prediction of software defects has been the focus of many researchers in empirical software engineering and software maintenance because of its significance in providing quality estimates from the project management perspective for an evolving legacy system. Software Reliability Growth Models (SRGM) have been used to predict future defects in a software release. Modern software engineering databases contain Change Requests (CR), which include both defects and other maintenance requests. Our goal is to use defect prediction methods to help predict CRs in an evolving legacy system.

Limited research has been done in defect prediction using curve-fitting methods evolving software systems, with one or more change-points. Curve-fitting approaches have been successfully used to select a fitted reliability model among candidate models for defect prediction. This work demonstrates the use of curve-fitting defect prediction methods to predict CRs. It focuses on providing a curve-fit solution that deals with evolutionary software changes but yet considers long-term prediction of data in the full release. We compare three curve-fit solutions in terms of their ability to predict CRs. Our data show that the Time Transformation approach (TT) provides more accurate CR predictions and fewer under-predicted Change Requests than the other curve-fitting methods.

In addition to CR prediction, we investigated the possibility of estimating effort as well. We found Lines of Code (added, deleted, modified, and auto-generated) associated with CRs do not necessarily predict the actual effort spent on CR resolution.

Document Type

Dissertation

Degree Name

Ph.D.

Department

Computer Science

First Advisor

Anneliese Amschler Andrews

Second Advisor

Scott Leutenegger

Third Advisor

Catherine Durso

Keywords

Change request, Effort, Estimation, Evolution, Legacy system, Prediction

Subject Categories

Computer Sciences | Software Engineering

CHANGE REQUEST PREDICTION AND EFFORT ESTIMATION IN AN EVOLVING SOFTWARE SYSTEM

A DISSERTATION

Presented to

the Faculty of the Daniel Felix Ritchie School of Engineering and Computer Science

University of Denver

in Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

by

Lamees Abdullah Alhazzaa

June 2021

Advisor: Prof. Anneliese Amschler Andrews

Author: Lamees Abdullah Alhazzaa
Title: CHANGE REQUEST PREDICTION AND EFFORT ESTIMATION IN AN EVOLVING SOFTWARE SYSTEM
Advisor: Prof. Anneliese Amschler Andrews
Degree Date: June 2021

Abstract

Prediction of software defects has been the focus of many researchers in empirical software engineering and software maintenance because of its significance in providing quality estimates from the project management perspective for an evolving legacy system. Software Reliability Growth Models (SRGM) have been used to predict future defects in a software release. Modern software engineering databases contain Change Requests (CR), which include both defects and other maintenance requests. Our goal is to use defect prediction methods to help predict CRs in an evolving legacy system.

Limited research has been done in defect prediction using curve-fitting methods evolving software systems, with one or more change-points. Curve-fitting approaches have been successfully used to select a fitted reliability model among candidate models for defect prediction. This work demonstrates the use of curve-fitting defect prediction methods to predict CRs. It focuses on providing a curve-fit solution that deals with evolutionary software changes but yet considers long-term prediction of data in the full release. We compare three curve-fit solutions in terms of their ability to predict CRs. Our data show that the Time Transformation approach (TT) provides more accurate CR predictions and fewer under-predicted Change Requests than the other curve-fitting methods.

In addition to CR prediction, we investigated the possibility of estimating effort as well. We found Lines of Code (added, deleted, modified, and auto-generated) associated with CRs do not necessarily predict the actual effort spent on CR resolution.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Professor Anneliese Amschler Andrews for the continuous support of my Ph.D. study and related research, for her patience, motivation, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis. Besides my advisor, I would like to thank the rest of my thesis committee: Professor Scott Leutenegger, Dr. Catherine Durso, and Dr. Krystyna Matusiak, for accepting my invitation to be members of my committee. Dr. Durso was of great help with her insightful comments and encouragement, in addition to assisting me on several occasions when I had difficult questions.

My sincere thanks also go to Professor Andrews, Professor Leutenegger, Dr. GauthierDickey, Dr. Dewri, and Ms. Corley who provided me an opportunity to join their team as GTA's which made it possible to continue my studies having the support. Special thanks to Dr. Sherba and Dr. Edgington who provided me with so much emotional support. I thank my fellow lab mates and colleagues, Afnan Albahli, Aiman Gannous, Ahmad Alhaddad, and Jide Williams, Dr. Lucente for their continuous support. Special thanks to Imam Muhammad bin Saud University for the scholarship and for the support of my colleagues.

My final thanks go to my family: my parents who supported me from the start, my siblings for their continuous and generous support, my small family and especially my kids: Hana, Tyme and Ryan, my relatives, neighbors and friends who cheered for me and provided help when I needed. This has been the most challenging journey of my life and without your generous support, I would not have been here now.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

Software systems provide numerous functionalities and innovative features to businesses and organizations. As the level of competition in businesses increases, the demand for rapid changes in software requirements and functionalities also become more frequent. Thus, the size and complexity of software grow as well, affecting the cost of system deployment and failures.

In situations where maintenance and evolution are based on predefined, competitive contracts, accurate effort (and cost) estimation is crucial, since overestimating will reduce the competitiveness of a bid while underestimating risks losing the organization money. Some of these systems are decades old and represent major assets.

Legacy systems are software systems that are vital to an organization but due to their age may have used outdated techniques. Some of these systems are over 40 years old and are most likely written in an older third generation programming language (FORTRAN, ALGOL, COBOL, C, etc.). Most legacy systems require frequent updates and maintenance to cope with changes in business [21]. The Y2K problem raised significant awareness of legacy systems and caused many governments

and major corporations worldwide to be concerned about the cost and magnitude of modifying their software systems' date format. They had to perform major system updates to change the two digit year representation into four digits in order to avoid system failures before the start of the new millennium.

To maintain the reliability of a legacy software system, management focuses on three main characteristics: quality, schedule and cost. Software quality influences whether and how fast new or modified features can be added, i.e. schedule, and how much it will cost to maintain and evolve legacy software. Software reliability is a key indicator of software quality. Software reliability is defined as "The probability of failure-free software operation for a specified period of time in a specified environment" according to ANSI [90]. Software Reliability Growth Models (SRGM) are used to assess the reliability of software systems by estimating the parameters of the model using existing failure data.

Software evolution refers to the process of repeatedly updating software systems and includes requirement changes or integration of parts during development. Requirement changes could be an enhancement of features, adaption of a system to changing hardware or software, or performance improvements [98]. Change-point is a term used to refer to change in the failure rate of a defect data set.

Our motive is to be able to use analytical methods to predict Change Requests (CRs) in an evolving aerospace legacy system. Analytical methods such as Software Reliability Growth Models (SRGM), have been used in defect prediction which does not consider system enhancement requests.

When applying reliability models to legacy systems, there are issues with the underlying assumptions of SRGM models. Many times these assumptions are violated such as assuming that when defects are fixed no new code was added, or assuming that a given operational profile does not change. This can cause unexpected

changes to the defect rate in a system. Additionally, this can also affect the quality of the defect prediction that could add additional cost for system maintenance and evolution. Stringfellow and Andrews [128] successfully used analytical models in their defect prediction. They propose a selection process to find a candidate model among several analytical models to be used in defect prediction. The use of analytical models has been effective in many systems but the predictions are less accurate in legacy systems that undergo periodic changes due to corrective, perfective or adaptive maintenance and enhancements. When changes occur they affect the defect rate and intensity, therefore the accuracy of the model used for defect prediction decreases.

Currently, reliability models are evaluated based on their sample fitting or short-term predictions, but we will also focus on long-term prediction. Long-term prediction is useful for project managers to take necessary action related to staffing, budgeting, and resource allocation to maintain software quality.

This thesis contributes in novel ways to use defect prediction methods to help predict Change Requests (CR) in an evolving legacy system. CRs include both corrective and perfective requests with a stronger emphasis on enhancements. We also apply a heterogeneous Time Transformation (TT) approach to provide CR prediction and compare it to other curve-fitting CR prediction approaches. We also investigate the ability to estimate effort in order to assist managers in estimating the maintenance costs of evolving software products.

1.2 Research Questions

In order to identify our scope, we developed a number of research questions to be further expanded in our thesis into sub-questions and ultimately answered as we

go along. The following research questions are established to formalize the main objectives of this thesis and define the scope of our work:

- RQ1: Can we predict CRs in an evolving legacy system?
  - RQ1.1: What research exists for failure and defect prediction in evolving software systems?
  - RQ1.2: What are the existing research gaps related to defect prediction during evolution and change?
  - RQ1.3: Can we estimate change-points in an evolving legacy system?
  - RQ1.4: What approaches can we use in CR predictions during evolution and change in legacy systems?
  - RQ1.5: How do these approaches compare?
- RQ2: Can we estimate effort in our legacy systems?
  - RQ2.1: What research exists related to effort estimation during maintenance in evolving software systems?
  - RQ2.2: What are the existing research gaps related to effort estimation during maintenance?
  - RQ2.3: What approaches can be used in effort estimation?
- RQ3: Can we validate the approaches on multiple releases of an actual legacy system?

These research questions are answered throughout the dissertation according to the research agenda below.

1.3 Research Agenda and Contribution

To answer the research questions we conduct the following:

- For RQ1.1 and RQ1.2: We conduct a mapping study of the existing body of literature concerned with failure and defect prediction when software evolves. (Chapter 2)
- For RQ1.3: We provide a practical and efficient method to estimate and detect change-point in a cumulative CR data.(Chapter 4)
- RQ1.4: We analyze data of the aerospace legacy system. (Chapter 3). We then use three curve-fitting approaches for CR prediction in an evolving software system considering change-points. (Chapter 5)
- RQ1.5: We compare an approach that uses Time Transformation (TT) to other existing curve-fitting approaches in terms of effectiveness and accuracy, by comparing their predictive ability (Chapter 5)
- RQ2.1 and 2.2: We describe existing work in effort estimation. (Chapter 2)
- RQ2.3: We investigate the use of two approaches: the COCOMO model and cumulative effort prediction. (Chapter 8)
- RQ3: Validation of generalization of change-point estimation, CR prediction, and effort estimation is applied to releases 1, 2 and 3. (Chapters 4, 7 and 8)

This document is organized as follows: Chapter 2 Covers existing work in failure and defect estimation methods, a systematic mapping study of software reliability growth models that consider evolution and change-points, change-point estimation

Publication	Type	Venue	Publisher	Status	Chapter
A systematic mapping study on software reliability growth models that consider evolution. [7]	Conf.	CSCI'19	World Comp.	Published	Ch. 2
Software Reliability Growth Models that Consider Software Evolution: A Systematic Mapping Study	Chapter	Advances in Computers	Elsevier	Submitted	Ch. 2
Estimating change-points based on defect data. [10]	Conf.	CSCI'18	IEEE	Published	Ch. 4
Change Request Prediction in an Evolving Legacy System: A Comparison [9]	Conf.	CSCE'20 + Springer Nature: Book Series	Springer	Published	Ch. 5
Trade-offs between early software defect prediction versus prediction accuracy. [8]	Conf.	CSCI'19	IEEE	Published	Ch. 6

Table (1.1) Publications

methods, and effort estimation. Chapter 3 explains our case study including system specification and data analysis. Estimating change-points based on CR data is presented in Chapter 4. Then CR prediction approach for evolving systems is presented in Chapter 5, followed by the trade-offs between early defect prediction vs. accuracy of prediction in Chapter 6. Chapter 7 expands the application of the CR prediction approaches to multiple releases and compares the results. Chapter 8 investigates a couple of methods for effort estimation using the COCOMO model and using cumulative effort data and discusses the results of each method. Future work is presented in Chapter 9 and finally conclusions are drawn in Chapter 10. Table 1.1 shows the list of submitted and published papers and their location in this document.

Chapter 2

Background

Software reliability is key in measuring software quality. Software Reliability Growth Models (SRGM), are widely used to predict growth in defect data. Changes due to major fixes or upgrades that cause change in the failure distribution, are called change-points. In an evolving legacy system, change-points could affect the quality of defect prediction using SRGMs. Therefore, it is key to find effective approaches to predict defects during evolution and change.

In this chapter we explain SRGMs, and how they are used in failure and defect prediction. We then explain software evolution and change-point estimation methods. Afterwards, we conduct a systematic mapping study on software reliability growth models that consider evolution. We conduct this mapping study to find and compare solutions in literature for reliability prediction for evolving systems, by collecting and analyzing publications in the field to identify available solutions, gaps and possible future work. Finally, we then highlight how SRGM is used in defect prediction in an evolving legacy system.

2.1 Software Reliability Growth Models (SRGM)

Software reliability research has recently hit the 52 year mark. It was first studied by Hudson in 1967 [61], which provided the basis of reliability engineering in software systems. Software reliability engineering consists of tools, methods and measures to track and predict software reliability [33]. Utilization of Software Reliability Growth Models (SRGM) started in 1972 with the work of Jelinski and Moranda [71]. By the late 1970s and early 1980's many researchers presented theories for predictive reliability practices. Software Reliability Growth Models (SRGM), are widely used in predicting future failures from cumulative failures data in software systems. Some of the commonly used models are the Musa model or the Goel-Okumoto (G-O) exponential model proposed by Goel and Okumoto [45] and Musa et al.[98], the Delayed S-shaped model [141], the Yamada Exponential model [142], and the Gompertz model [84].

To expand more on the history on software reliability engineering, we refer to a recently published survey by Cusick [33]. He demonstrated the progression of software reliability engineering in the past 50 years. He went through the development history and improvement of software reliability engineering including novel techniques and models, significant publications, related journals and venues, main tools, and notable companies who remain active in Software Reliability Engineering (SRE). This survey focused on the historical timeline of software reliability and basic events but lacks information regarding software reliability when evolution and change occur.

Stringfellow and Andrews [128] addressed the use of SRGMs for data from a defect database rather than failures. They proposed an empirical method for selecting SRGMs to help test managers in a software organization make release decisions

Model	Equation	Curve Shape
G-O	$\mu(t) = a(1 - e^{(-bt)}) \quad a \geq 0, b > 0$	Concave
Delayed S-Shape	$\mu(t) = a(1 - (1 + bt)e^{(-bt)}) \quad a \geq 0, b > 0$	S-shaped
Gompertz	$\mu(t) = a(b^{(c^t)}) \quad a \geq 0, 1 > b > 0, c > 0$	S-shaped
Modified Gompertz	$\mu(t) = d + a(b^{(c^t)}) \quad a \geq 0, 1 > b > 0, c > 0, d > 0$	S-shaped
Yamada Exponential	$\mu(t) = a(1 - e^{(-bc^{((1-e^{(-dt)}))})}) \quad a \geq 0, bc > 0, d > 0$	Concave

Table (2.1) SRGMs

based on predicting the number of defects after release. This work was then replicated by Andersson [14]. Andrews and Lucente used SRGMs for predicting help desk incidents [15]. ABB Inc. used software reliability modeling as well for risk management. Their purpose was to compare reliability models and select the most appropriate one to predict field defects in order to establish accurate maintenance planning [85]. These models assume that no major changes in the software occur which could alter the failure process (usually by altering the failure rate). Such events are referred to as change-points which can affect the accuracy of reliability model's predictions.

2.2 Evolution and Change in Software Systems

When project managers and decision makers are dealing with software systems, changes are expected either in the form of upgrades or fixes. Changes in the system cause changes in the predicted number of defects, affecting system reliability. To increase managers' and consumers' confidence in a software system, project managers should be able to predict the effect of change in their system in order to plan ahead in terms of staffing, time management, or even preparation of back up systems. The problem with SRGM is that they do not account for changes in the defect rate. When a change-point occurs, a change in the selected model is required. A change-point is defined as "the point at which fault detection/introduction rate is

changed."[70]. Changes in a legacy system may occur due to corrective or perfective measures.

To define change-points formally, let a sequence of failures, $\zeta_1, \zeta_2, ..., \zeta_n$ be where n is the number of cumulative failures. A change-point τ, exists if $\zeta_1, \zeta_2, ..., \zeta_\tau$ has a failure distribution of F, and $\zeta_\tau, \zeta_\tau + 1, ..., \zeta_n$ has a failure distribution G, where $F \neq G$ and the two sequences of failure data are statistically independent.

2.2.1 Reliability Modeling with the Existence of Change-points

When dealing with evolving systems there are a few ways to handle change-points. Musa eta al. [98] and Lyu [90] gave three main approaches to handle system evolution:

1. Ignore change, by selecting a model that fits the available defect data then assuming that this model is appropriate for predicting defect for the remaining part of the release. This approach is used when the total number and volume of changes is small.

2. Apply changes to the model after a change-point by dividing the defect dataset into stages. When a change occurs the dataset after change is considered a new stage and it is considered a separate dataset where reliability modeling is performed separately. This approach is suitable when changes are large in amount but low in frequency. This method considers each stage as an independent sub-release.

3. Apply failure times adjustment. It is most appropriately used when a software is rapidly changing and if it cannot be divided into separate sub-releases.

2.2.2 Change-point Estimation

While change-point estimation methods are widely discussed in many fields of research such as statistics, psychology and mathematics, research in the field of software reliability modeling is limited ([148] [55] [151][41][64]). A few studies focused on change-point estimation rather than reliability modeling with change-point existence. In the field of change-point estimation using statistical methods, studies discussed the use of control charts for change-point diagnosis ([148][55]). Amiri and Allahyari [12] proposed an overview of control charts in a literature review. They presented different variations of control charts and their underlying techniques. Zhao et al. [148] applied control charts following the idea presented in Huang and Huang [55] with some modifications to the criteria of defining a data point as a change-point candidate. Additionally, Zhao et al. [148], used a progressive adjustment step. When a fix is applied to a change-point, the remaining change-points are re-estimated. Other studies ([151] [41][64]) use the difference in the mean value detected in a regression model or a reliability growth model before and after the change. Zou [151] proposed a change-point estimator likelihood ratio method by comparing defect density among a sequence of time intervals of cumulative defects. The time interval that causes a change in the statistical model used to predict failures is marked as a change-point. This method assumes the existence of a single change-point. Galeano's method [41] estimates multiple change-points. Like Zou [151], he proposed a method comparing failure density of cumulative failure sums of different time intervals. The mean value of the time interval with the highest change is normalized and centered to be then considered as a change-point. He uses a binary segmentation procedure that continues to split data after detecting a change until all change-points are detected [41]. Inoue et al. used arithmetic means factor along

with Laplace factors for detecting change-points [64]. Change-point estimation will be investigated further in Section 4.

2.3 Reliability using SRGM Considering Evolution

When applying the reliability models to legacy systems, there are issues with the underlying assumptions of SRGM models. Many times these assumptions are violated such as assuming that when defects are fixed no new code was added, or assuming that a given operational profile does not change. This can cause unexpected changes to the failure rate in a system. Additionally, this can also affect the quality of the reliability prediction that could add additional cost for system restoration or business compensation. Febrero et al. [40] conducted a systematic mapping study of software reliability modeling in general. A total of 972 works were obtained that focus on software reliability in general.

Our goal is to conduct a more focused work to study emerging solutions that deal with change and evolution in legacy systems. Our objective is to map the body of research in the area and compare solutions. We explore publications in industry and academia, find research affiliations and identify contributions through different venues. We then classify the papers according to solution extent, proposed method and research type, and finally discuss focus and gaps. We selected 63 papers from 1/1/1986 to 5/1/2021 and summarized trends with respect to year, research organizations, publication venues and academia versus industry. Papers are classified according to solution extent, proposed method, and research type. This mapping study is an updated and expanded version of a mapping study published by Alhazzaa and Andrews in 2019 [7]. The previous study was shorter and included

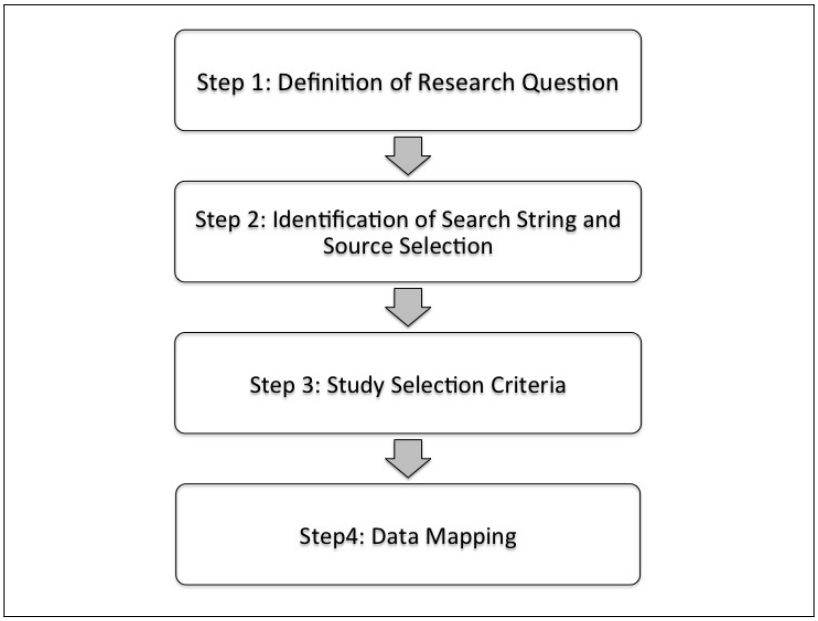


Figure (2.1) Research method work flow

papers until 2018. This version of the study is an expanded version with more details on focus and gaps, and includes papers until 5/1/2021 as well.

We begin by first describing the research method in Section 2.3.1, followed by the study classification scheme in Section 2.3.2. This scheme is applied to the selected papers to present the actual mapping of the field in Section 2.3.3. A discussion of the gaps and the focus is in Section 2.3.4. Section 2.3.5 covers threats to validity. Conclusive remarks are in Section 2.3.6.

2.3.1 Research method

Following the main guidelines by Peterson et al. [106], we propose our systematic mapping going through the following four steps (see Figure 2.1) similar to [17]: Definition of research questions, identification of search string and source selection, study selection criteria and data mapping. Each step is explained in greater detail in the following subsections.

2.3.1.1 Definition of research questions

The overall objective of this study is to investigate and analyze the available body of work that is concerned with failure prediction using software reliability models in legacy systems during evolution and change. Our main goal is to find and classify articles about using software reliability growth models during evolution, understand the methods and models used and the most active research area in the field. We are also concerned with the number of publications and their venues. The research questions below focus on finding available solutions in existing publications, categorizing them according to their source and the year of publication, and finding research gaps to understand what areas of solutions have not been presented thoroughly:

- RQ1: What are the publication trends for reliability models used during change and evolution?
  - RQ1.1: What is the annual number of publications in the field?
  - RQ1.2: What are the main venues of publication in the field?
  - RQ1.3: Which publications are affiliated with academia and which are of an industrial affiliation?
  - RQ1.4: What institutes are the most active in the field according to the number of publications published?
- RQ2: What are the existing research gaps related to solutions of evolution and change in failure prediction using SRGMs?
  - RQ2.1: What solutions have been proposed?
  - RQ2.2: What are the methods used to apply proposed solutions?
  - RQ2.3: What types of contributions have been proposed?

– RQ2.4: What types of research are conducted in the area?

By looking first into research trends we can explore emerging and abandoned trends, the progression of research activity during a specific time span or through a specific research group. The second set of questions covers the most important solutions and approaches presented along with the gaps and underrepresented methods. In addition, we analyze their contributions and their research method.

2.3.1.2 Identification of search string and source selection

In order to find reliable peer reviewed articles we selected articles from the following sources:

- IEEE
- ACM Digital Library
- Elsevier
- Springer
- Wiley

We used search engines in the libraries above in addition to searching for papers using:

- Web of Knowledge
- Google Scholar

Many of the papers we reviewed focused on reliability models but not evolution and change. In addition, evolution and change is referred to using different phrases such as change-point, multi-up gradation, or multiple change-points. Consequently, we used the following search string:

- (Failure prediction OR failure estimation) AND (software reliability models OR SRGM OR software reliability growth models) AND (evolution OR change OR change-point OR multiple change-points OR multi up-gradation)

After performing the search, papers were selected according to the study selection process explained in the following section.

2.3.1.3 Study selection criteria

Criteria Type	Criteria List
Inclusion Criteria	• Journal, conference and workshop papers • Peer reviewed • English language only • Software systems • Published papers available electronically • Statistical methods • Failure prediction using SRGMs • Solve issues of evolution and change
Exclusion Criteria	• Books • Papers that are not peer reviewed • Papers written in languages other than English • Hardware systems • Papers not published/available electronically • Architecture-based or AI-based methods • Change-point estimation techniques • Fields other than software engineering

Table (2.2) Inclusion and exclusion criteria for the mapping study

This step involves selecting relevant studies from the search results. First, inclusion and exclusion criteria are defined (see Table 2.2). For this study, all papers that were considered were peer reviewed, published, available electronically, and written in English. Books are excluded as they provide basic knowledge and not novel work

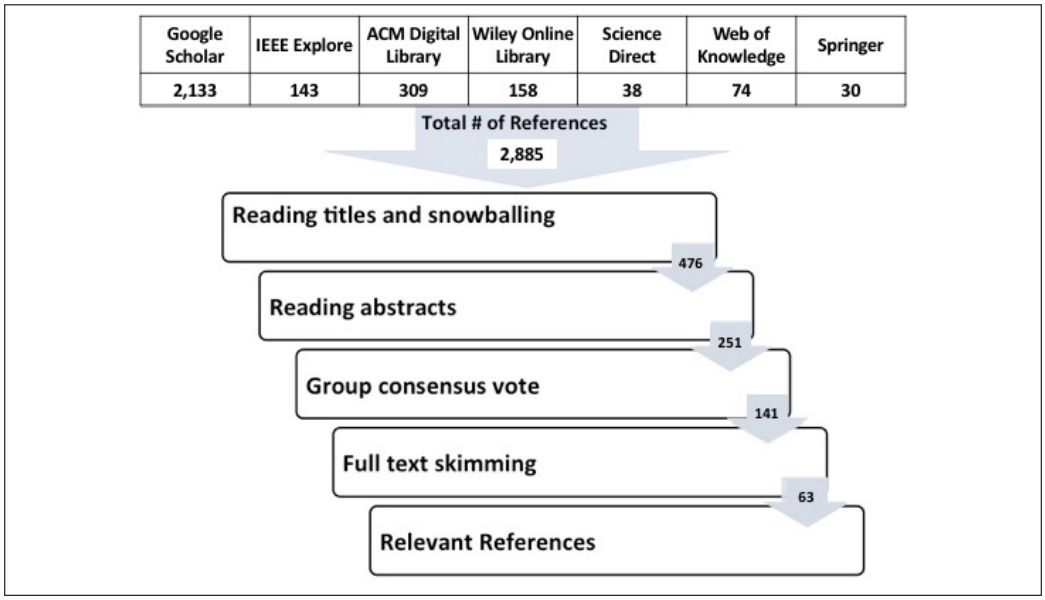


Figure (2.2) Study selection process

and developments in the field. The references would have to address failure prediction in a software system since it is the focus of this work, this means that any hardware related approaches are excluded. Solutions must provide failure prediction during evolution and change. The solutions investigated here are statistical methods and not AI or architecture based solutions, since these methods are irrelevant to our focus area. All research must be in the field of software engineering. Figure 2.2 summarizes the steps in the paper selection. They are explained as follows:

1. We collected references using the search string (mentioned in Section 2.3.1.2). This resulted in 2,885 papers. The results of the research were not bound by dates, so we included all references up to 5/1/2021.

2. We applied inclusion and exclusion criteria as given in Table 2.2. For each included reference, a supplemental search using snowballing was performed in order to find more relevant papers. This resulted in 476 papers.

3. After reading the Abstracts, the papers were then classified into three categories: relevant, irrelevant, and not clear. The relevant papers were papers in the field of software engineering and focused on reliability models with failure prediction and estimation using analytical models or statistical methods. The irrelevant papers on the other hand, were papers in a field other than software engineering, or papers discussing methods that are using artificial intelligence techniques or architecture-based techniques. Papers that exclusively dealt with change-point estimation were also excluded. Some papers did not fall clearly in either the relevant or irrelevant categories at this stage, so we labeled them as "not clear". This left 251 references which were either relevant or not clear.

4. At this point two individuals double checked the papers and performed a consensus vote. The voting was based on the relevance of the paper labeled as "not clear" and whether it follows the inclusion and exclusion criteria stated above. Papers were discussed until a consensus occurred. At the end of this step we were left with 141 relevant references.

5. The remaining papers were skimmed and references of those papers were investigated. This step provided more clarification and understanding of the relevance of the paper. Additional papers were excluded at this point and we were left with 63 papers.

2.3.1.4 Data Mapping

The 63 papers chosen covered the time period of 1/1/1986 – 5/1/2021. Since most SRGM models were introduced between 1983-1987, we find the earliest publication related to our research is 1986. We performed data mapping using four

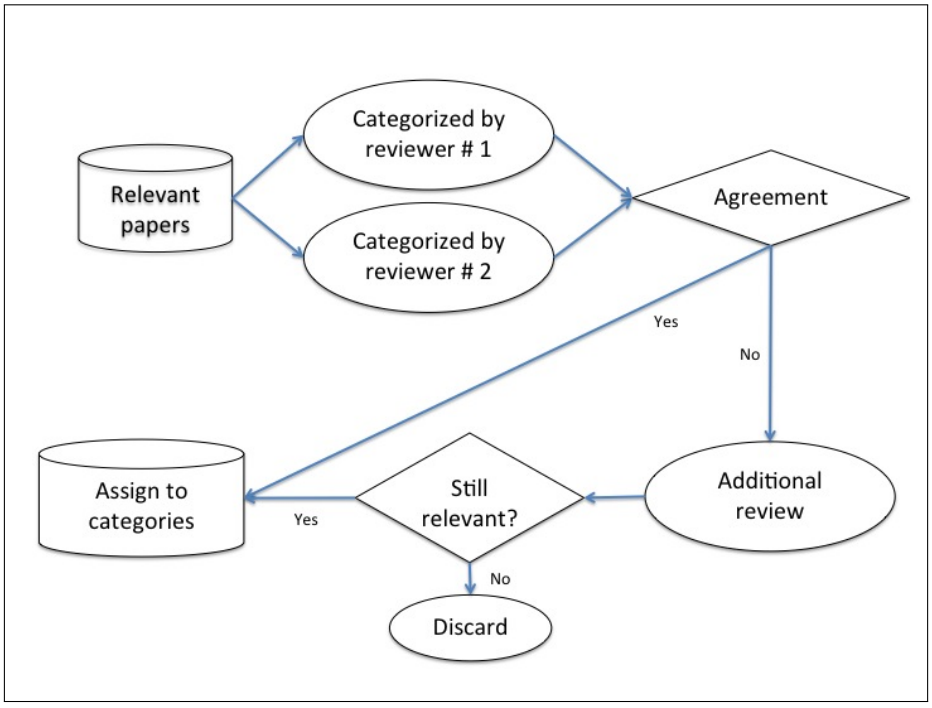


Figure (2.3) Data mapping

categorizations: Solution Extent, Proposed Method, Contribution Type, and Research Type.

Each relevant paper is reviewed by two reviewers and each reviewer suggests the proper categorization of the paper. If they both agree, the paper is assigned to the agreed upon category; if there is no agreement, we skim the text. In some cases, skimming text resulted in the paper being declared not relevant, and therefore discarded (Figure 2.3).

2.3.2 Study Classification Scheme

The study classification is divided into two main categories. The first category is related to the publication trends which highlights the years of publications, the venues, active research organizations and affiliations. These trends answer the first set of research questions in Section 2.3.1.1. The second category is concerned more

with the research content and possible research gaps which are: Solution Extent, Proposed Method, Contribution Type, and Research Type. Detailed descriptions are given below.

2.3.2.1 Solution Extent

Solution Extent discusses what solution is proposed according to the type of evolution and other factors affecting evolution. The focus here is on papers aimed towards failure prediction and estimation using reliability models in legacy systems where changes are expected. The classification of the papers fall into three categories as follows:

- Single change-point: Change-points here refers to the change that happens to the cumulative rate of failures as a result of code addition, deletion, or modification. This change affects the estimation of failures after that point. Many reliability models won't fit the failure rate curve following a change in the same way that it fit before. The papers in this category provide solutions to reliability models that are aimed at systems with only a single change-point throughout the curve of cumulative failures in the dataset. These solutions usually provide a change in the model attribute or change type of model selected whenever a change-point occurs that alters the shape of the curve so that it no longer fits the originally selected model nor its attributes.

- Multiple change-points: This provides a solution to a reliability model that is aimed at systems with multiple change-points in the cumulative failure curve. The major difference here from the single change-point is that the change in model selection and attributes occurs several times as changes happen to the cumulative failure count.

- Multi up-gradation: Multi up-gradation refers to multiple upgrades at designated points throughout the system's lifetime. These changes are scheduled and known previously, but they have the same effect as any change-point. The failure rate is expected to change after each software maintenance task or upgrade and solutions to estimate failure density during evolution are provided accordingly.

2.3.2.2 Proposed Method

The proposed methods refers to the method of selecting the proper reliability model for the proposed solution. They can be of two types:

- Analytical methods: They derive a solution analytically by providing assumptions for the failure and defect removal process, as well as usage profile for the software and developing a model based on these assumptions.

- Curve-fit methods: They make little or no assumptions. They entirely rely on statistical curve-fit methods for one or more types of functions and select the one that fits the best.

2.3.2.3 Contribution Type

Each research paper makes a specific contribution to the field of software reliability estimation for evolving systems, that fall under one or both of thse categories:

- Build a new method or model: This category includes papers that present a new method or a new model for reliability estimation for evolving systems.

- Evaluation of a new or an existing method or model: This category includes papers that evaluate or validate either new or existing methods or models for

reliability estimation for evolving systems. These papers offer an empirical evaluation of a method or model.

2.3.2.4 Research Type

Specifying the research type provides more understanding of the maturity of the proposed work. If a method proposed is validated or evaluated empirically, this produces more confidence in the provided solution. Research type would fall under one of these two major categories:

- Empirical research: This type of research provides direct or indirect observation of an experience. In empirical research an application of a given method or model is demonstrated and results are derived through a formal experience such as a case study, an experiment or a survey.

- Non-empirical research: This is a more theoretical research that provides information without empirical evidence.

For the empirical type of research, the work is evaluated according to empirical study evaluation criteria as proposed by [11]. This required that we look deeper into the following aspects:

- Objectives and hypothesis:
  - Check if software systems are defined (what is assessed and compared).
  - Check if a statistical hypothesis is stated, either one-tailed or two-tailed, the conclusion has to clarify if the hypotheses is rejected, or not, as well as the interpretation of the resulting p-value.

- Targeted application domain(s): The application domain should be mentioned. Some software is used in several domains while other software only considers a single domain.

- Subject system specification: Systems used in a specific domain should be mentioned in terms of size, language used, system specifications, time periods that the defects were collected, number of defects, etc.

- Parameter setting: This evaluates whether and how parameters are set. Many studies use techniques like Maximum Likelihood Estimation (MLE) or Least Square Estimation (LSE). Some specify the use of tools that use one of these methods to do the estimation automatically, such as SPSS statistical software or R. Specifying the details of how parameters are set makes the case study of a higher quality for interested readers.

- Measures of cost and effectiveness:

  - Measuring the effectiveness is measuring the goodness-of-fit of the proposed method and how well it performs. Many measurements can be used to measure effectiveness, some papers use one measure and some use more than one measure. This will be later used as a base of comparison.

  - A cost measure is crucial for any organization to decide if they will adopt a method. Cost is usually measured according to an organization's need, but many define an optimal release policy that takes into consideration several cost attributes such as cost of testing, cost of assets, cost of maintenance, etc.

- Baseline for comparison:

  - Comparing effectiveness of methods, i.e. the number and quality of measures used to evaluate the results. This includes checking whether they are comparing only goodness of fit or if they are evaluating the effectiveness of future predictions. Goodness-of-fit measures the closeness of the proposed model curve against the actual data. Additionally, for effectiveness, predictive validity which is the ability to predict the number of failures from present and past failure behavior, should be used to evaluate the accuracy of the proposed model.
  - Comparing cost of development, testing-effort and maintenance before and after releases in order to assist management in decision making.

- Data analysis:
  - Here we assess whether an empirical study is presenting descriptive statistics, results of hypothesis testing, and whether the actual significance of the results is explained.

- Validity threats:
  - Ideally, papers should address internal validity threats, external validity threats, construct validity threats, and conclusion validity threats.
  - A study is evaluated in terms of presenting these validity threats as follows:
    * No mention: If no threats or validity limitations are mentioned.
    * Some of the validity threats are mentioned in a formal or informal way such as included in one of the sections without specifying them explicitly.
    * Mentioned: If they are explicitly and clearly presented.

- Quality of data: This refers to the data used in the case study. Is data collected from a reliable source? Does it accommodate the purpose of the study? How recent is the data? In our evaluation, we consider data as recent if the time span between the research publication and the data collected does not exceed 15 years.

2.3.3 Study Synthesis and Mapping

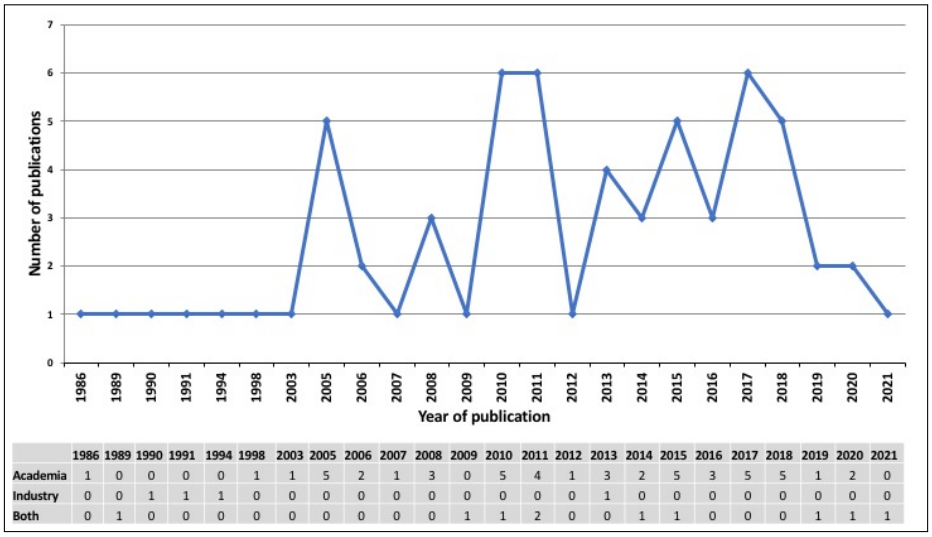
In this section we provide our synthesis of the relative studies we collected. In the following section we present the trends and the potential gaps among the 63 publications of the time period covering 1/1/1986-5/1/2021.

2.3.3.1 Publication Trends between 1986 and 2019

In this section, publications trends are analyzed in terms of distribution over the span of years covered. The main venues of publications are then highlighted, as well as the contributions of academia and industry in the field of study. We then look into the most active research organizations in academia or industry.

Distribution of Publications

Publications between 1986-2021 have increased in the last decade in general. There are some years that had major spikes in the total number of publications in the field, such as in 2010, 2011 and 2015 as shown in Figure 2.4. In 2010, a major interest was given to a new area: Multi up-gradation in software reliability estimation by a group of researchers in schools in India, such as Amity University and University of Delhi. These publications continued through 2011 and the years after. In 2017, many of the publications in the field of multi up-gradation were published, looking into multiple releases and multi-version systems, in addition to



	1986	1989	1990	1991	1994	1998	2003	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
Academia	1	0	0	0	0	1	1	5	2	1	3	0	5	4	1	3	2	5	3	5	5	1	2	0
Industry	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Both	0	1	0	0	0	0	0	0	0	0	0	1	1	2	0	0	1	1	0	0	0	1	1	1

Figure (2.4) Annual distribution of publications

focusing on other systems such as Open Source Software (OSS). Efforts were also made on finding curve-fit solutions between the years of (2017 -2019). The number of publications decreases in 2021. This does not mean that there was less interest, but it could have been that many of those publications are not yet available online and may be difficult to find currently. Publishing is continuous since 2003, which means that researchers are still interested in the field.

Main Publication Venues

Table 2.3 lists the top five venues where papers were published. The rank refers to the quantity of publications for each venue. The venue ranked No.1 is the IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), which has the most publications in the field (seven). These publications have been published throughout different years of the conference. Next comes International Journal of System Assurance Engineering and Management and IEEE Transactions on Reliability. These two journals each published four of the papers of interest. Journal of Systems and Software and IEEE Transactions on Software Engineering, each published three related publications. In total, the primary stud-

Rank	Type	Venue	No. of Articles
1	Conference	IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)	7
2	Journal	International Journal of System Assurance Engineering and Management	4
2	Journal	IEEE Transactions on Reliability	4
3	Journal	Journal of Systems and Software	3
3	Journal	IEEE Transactions on Software Engineering	3

Table (2.3) Top five publications venues

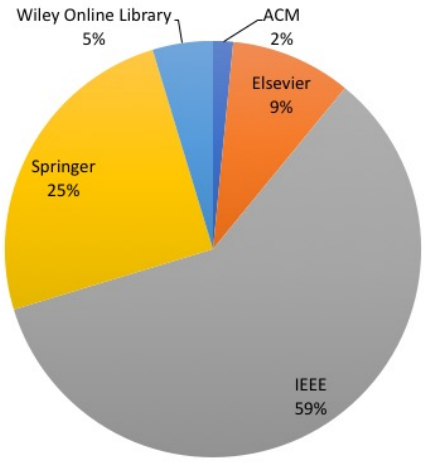


Figure (2.5) Distribution of publications over different publishers

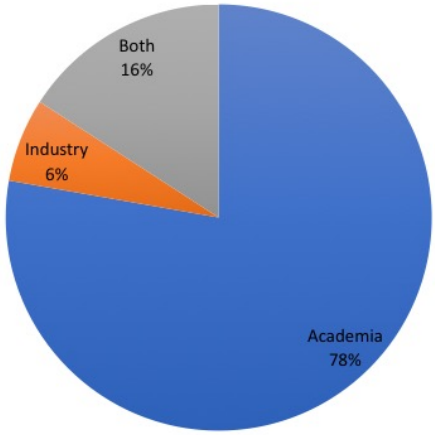


Figure (2.6) Distribution of publications in academia, industry, and both

ies were published in 43 different venues. 33 of the publications were published in journals, 28 were conference papers, and 2 were published in workshops.

Figure 2.5 shows the sponsors of those venues. It shows that 59 percent of the publications were published in IEEE venues, which encompasses the majority of the body of work. Springer published almost 25 percent of the publications, Elsevier nine percent. Finally, Wiley and ACM are five and two percent of the publications respectively.

Academia and Industry Representation

This section concentrates on the affiliation of the authors. If all authors of a paper work in an academic organization, then the paper falls under the Academia classification. If authors were affiliated with an industrial organization then it falls under Industry. In some papers, authors come from both academia and industry so they have both affiliations. Figure 2.6 shows that the vast majority of the work is purely a production of research in academia (78 percent). On the other hand, only 6 percent of the work is purely industrial research and 16 percent reflects a joint effort between academia and industry.

Rank	Institution	Number of Articles
1	University of Delhi, India	14
2	National Tsing Hua University, Taiwan	13
3	Amity University, India	10
4	Tottori University, Japan	6
5	Islamic Azad University, Iran	3
5	Rutgers, The State University of New Jersey, NJ, USA	3
5	The French National Center for Scientific Research, France	3

Table (2.4) Most active research organizations

This shows the high interest of academia in the subject, while industry alone has very limited efforts although it would be of great benefit to industry. The amount of joint work demonstrates that industry could use academic efforts or results to find practical solutions.

Active Research Organization

A total of 51 research organizations are interested in research in this area. Table 2.4 shows that University of Delhi in India contributed with 14 publications. National Tsing Hua University in Taiwan contributed 13 publications. Amity University in India contributed 10 publications. Tottori University in Japan contributed 6 publications. The Islamic Azad University in Iran, Rutgers, The State University of New Jersey, NJ, USA and The French National Center for Scientific Research, France each contributed three publications. Some schools and organizations have joint efforts. Many organizations in India, Taiwan, China, Japan and the USA are interested in this research area.

Figure 2.7 shows the distribution of research groups interested in this area of research around the globe.

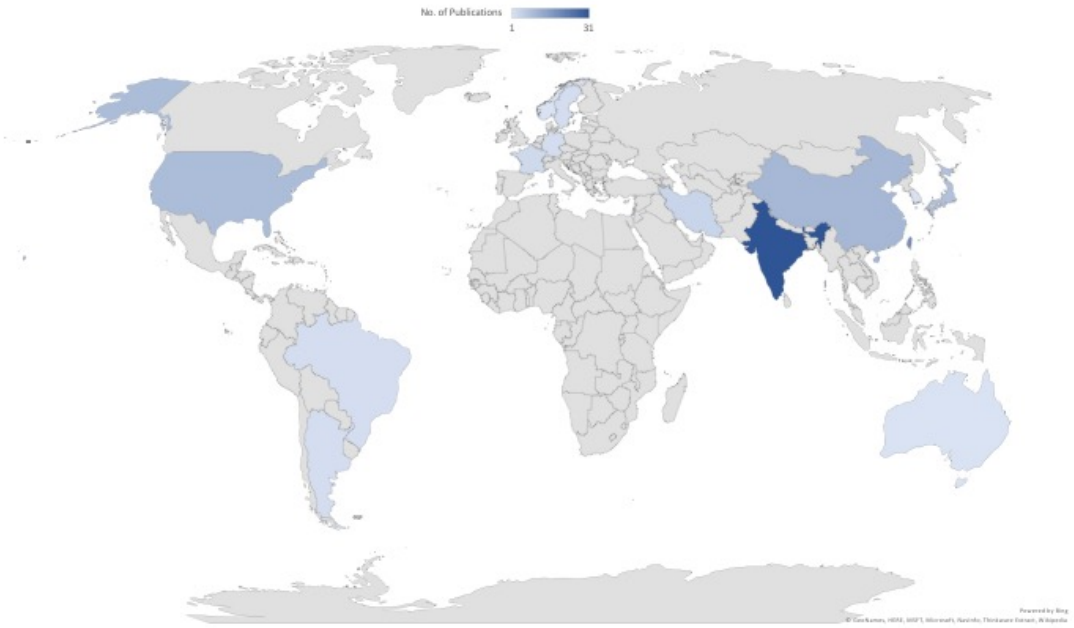


Figure (2.7) Distribution of organizations interested in research of interest

2.3.3.2 Focus and Potential Gaps

A major purpose of this study is to provide a map of publications in the field of failure predictions using reliability models when evolution and change exists, and identifying the gaps in this field in order to determine what research areas could be emphasized more and to suggest potential future work.

The following sections demonstrate the focus of the existing body of work in terms of the solution extent discussed, proposed methods, contribution type, and research type and quality.

Solution Extent

After analyzing the available body of work and looking into the core solution extents discussed in each paper, we concluded our classification to two groups, each of which has three sub-groups as discussed in Section 2.3.2.1 which are: Single change-point, multiple change-points, multi up-gradation. We find that 46% of the publications focus on solutions for multiple change-points, since it provides more

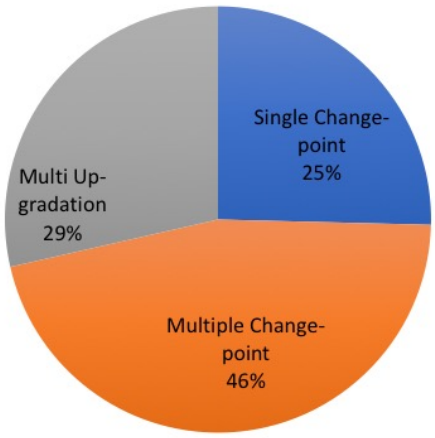


Figure (2.8) Distribution of publications over proposed solutions extents

Solution Extent	**Single CP**	**Multiple CP**	**Multi UG**
Fault Severity		[49]	[131] [44] [122] [123] [121]
Testing Effort	[66] [54] [42] [124]	[83] [89] [86] [88][82]	[78] [79]
Environmental & learning effects	[147]	[31] [32]	[122] [80]
Testing Compression Factor (TCF)		[87] [58]	
Fault Reduction Factor (FRF)			[79] [2]
Code complexity			[125]
Testing coverage	[124]		
Queueing model	[145]	[56]	
Hazard Rate	[69] [65] [64]		
Weighted means		[60] [59] [57]	
Other	[38] [68] [76] [151] [67] [100] [93] [13] [19]	[129] [29] [30] [55] [126] [137] [18] [91] [75] [101] [94] [144] [143]	[103] [77] [81] [43] [149] [50] [53] [4]

Table (2.5) Solution extent

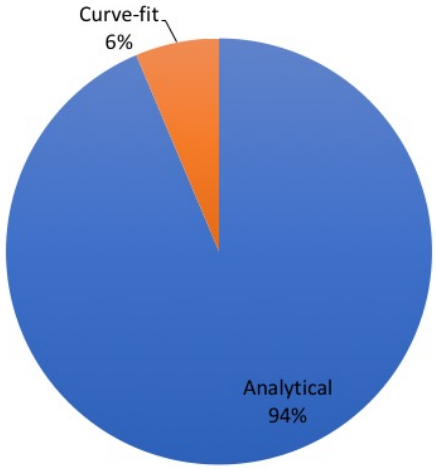


Figure (2.9) Distribution of publications over proposed method

generic and applicable solutions than just single change-point solutions, which contribute 25% to the body of work. Multi up-gradation gained a great deal of interest and represents 29% of the papers in the field.

Some solutions incorporate an additional aspect or dimension into the proposed model, such as testing effort or code complexity. Some add an additional dimension to have two dimensional models such as including testing effort and fault reduction in Kapur et al.[79]. Table 2.5 classifies the research in the field based on solution extent and additional model extensions. More explanation regarding the proposed solutions are explained in Section 2.3.4.1.

Proposed Method

We defined the two types of proposed methods, analytical method and curve-fit method. Analytical methods represent almost 94 percent of the research contributions, while curve-fit methods represent about 6 percent, as shown in Figure 2.9.

In the analytical methods presented, change-points are identified and the failure curve is divided into segments. Then the proposed model is applied with different parameters to fit each segment accordingly. Variations of equations and consider-

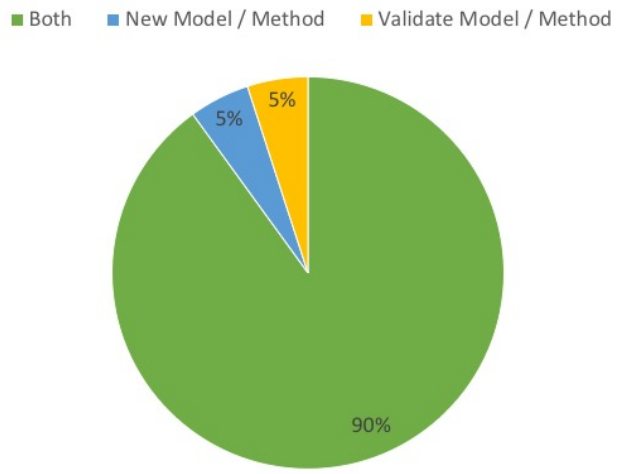


Figure (2.10) Distribution of publications over contribution type

ations were incorporated in the presented papers in order to provide an effective reliability model.

Curve-fitting was mostly presented in works related to reliability models that do not discuss the existence of a change-point. Some earlier works utilized reliability models using curve-fit methods such as the work presented by [128]. In these works different SRGM models were estimated against a given data set and then predictions were made using the best model in terms of accuracy. Li et al. [85] performed curve-fit as part of a quantitative risk management method. Curve-fitting was used for several releases in order to select the best model that fit the dataset. The publications that discusses curve-fit methods [137] [38] [30] [19] focus on finding trends, detecting changes in the cumulative failure curve and fit the best available model after each change occurrence. These efforts are valuable in the study but they are very limited. They merely adjust parameters or choose a new model after each change-point and discard old data, but they do not consider the combined effect of old and new systems on system reliability.

Contribution Type

Papers can make different contributions, such as proposing a new method or model, or validation of a new or existing method or model. Articles could have one or more of those contributions, since an author could propose a new method and evaluate the proposed method in the same article, for instance. Figure 2.10 shows that almost 5 percent of the publications propose a new method or model, another 5 percent validate an existing method or model and 90 percent are papers with a newly proposed method or model along with validation through a case study.

Research Type

Research type is either empirical or non-empirical. As shown in Figure 2.11, almost 95 percent of papers represent empirical research, and only 5 percent are non-empirical. We noticed that the empirical work we found are mostly case studies, and there was a notable lack of experiments. This finding is to be expected, since one cannot expect to produce multiple versions of evolving software simultaneously. This would simply not be financially viable.

Looking into the 60 empirical studies, we evaluated their quality on the basis of [11], with some modifications as discussed in Section 2.3.2.4.

- In terms of the objectives and hypotheses, we found that of 60 publications stated hypotheses or objectives.

- 33 studies give explicit application domain details, where in others we assume that their solutions are generic and would fit most application domains. 13 percent of the publications were for Open Source Systems (OSS), 3 percent were for software based on agile development. 30 percent of the publications discussed multiple releases, multiple version or multiple sprints, while the remainder discussed their methods for a single release.

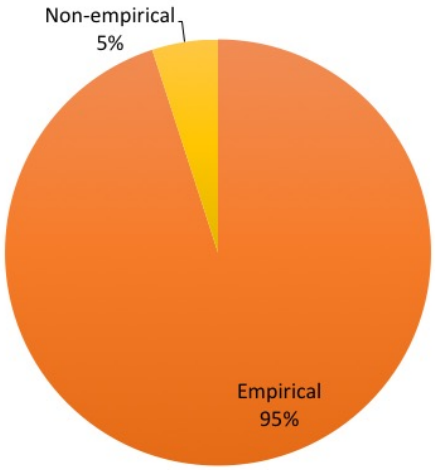


Figure (2.11) Distribution of publications over research type

Evaluation criteria	**Number of publications**
Objectives and hypotheses	60
Target application domain	33
Subject system specification	51
Parameter Setting Specification	52
Measuring cost and effectives: Cost	12
Measuring cost and effectiveness: Effectiveness	57
Baseline for comparison: Goodness-of-fit	56
Baseline for comparison: Predictive Validity	25
Baseline for comparison: Cost	12
Data Analysis	58
Validity threats	4
Quality of Data	10

Table (2.6) A Summary of quality assessment for the available case studies

- Details of subject system specifications such as language used, system specifications, time periods when defects were collected, number of defects, etc., are mentioned in 51 of the studies.

- Parameter settings regarding the method, or the tool used in parameter settings are mentioned in 52 of the studies. These are important for researchers since different tools and methodologies may give different values in estimating parameters; so if a case study were replicated, we could use the same parameters to obtain similar results.

- Among the 60 empirical studies, 57 measured the effectiveness of the proposed method while only 12 proposed a model that also estimates the cost.

- For effectiveness, 56 studies used goodness-of-fit measurement, and 25 of those studies also considered predictive validity. For goodness-of-fit several measurements are used such as Mean Square Error (MSE) and R-square (Coefficient of Multiple Determination), to measure the closeness of failure data to the fitted regression line. In some papers they also measured predictive validity by measuring Relative Error (RE) and Root Mean Square Prediction Error (RMSPE). In addition cost was measured by proposing an Optimal Release Policy for cost, which is evaluated by major release costs such as: cost of development, cost of testing, and cost of maintenance before and after release.

- In terms of data analysis, the majority of the available work presented a decent description of descriptive statistics, results of the case study, and actual significance of the results. Some papers had a minimal description. In many cases this occurred in a short version of the case study in a conference paper which was followed with a more detailed version in a journal paper. All papers

Quality of Data Used	Number of publications
Old data	48
New data	8
Both old data and new data	2
Unknown	2

Table (2.7) A Summary of quality of data in the available case studies

provided different variations of descriptive statistics. 58 out of the 60 papers reported the statistical significance of the results. None of the papers provided results of hypothesis testing in their data analysis or discussion section.

- Out of all the publications, only one of them presented a thorough explanation of the validity threats [82]. A total of four of the studies highlighted validity threats in general.

- Data in the studies available are of a decent quality, i.e. they all come from reliable sources (benchmarks) and no major problems are mentioned regarding the quality of the data collected. The only problem with some publications is that they use old data that may not reflect the accuracy of the model for current software technology. A study published in 2007 using data from a source published in 1980, may not provide realistic outcomes, since techniques evolve over the years causing changes in failure processes. Table 2.7 shows that 80 percent of the publications use old data, 14 percent use new data, 3 percent use two sets of data (old and new) and 3 percent use unknown data, i.e. the source of the data is not reported. The data is considered new if it was published or available within the last decade (2010-2020), otherwise it is considered old.

Relationship between Research Type, Research Contribution, Solution Extent, and Proposed Method

Solution Extent	**Analytical**	**Curve-fit**
Single Change-point	15	1
Multiple Change-points	23	3
Multi up-gradation	18	0

Table (2.8) Number of publications per proposed method and solution extent

Solution Extent	**New method / model**	**Validate method / model**	**Both**
Single Change-point	0	2	15
Multiple Change-points	1	1	26
Multi up-gradation	2	0	16

Table (2.9) Number of publications per contribution type and solution extent

In Table 2.8, we can see that curve-fit papers cover single change-point and multiple change-point solutions. The lack of curve-fit papers leaves a huge gap in available methods of this type.

Table 2.9, shows the papers by solution extent and how many of them provide a new method or validate methods. It appears that the majority of papers provide a new model or method and validates it. Two of the single change-point papers provide a validation of a method/model only. Two of the multi up-gradation papers discuss a new method without validation, possibly due to the novelty of the idea. Two papers dealing with multiple change-points propose a new method only and another only validates a method.

In Table 2.10, we can see that most papers provide some empirical study to validate their methods. There are very few papers that use non-empirical validation, only for single and multiple change-points. This shows most of the work presented is empirically validated. Table 2.11 shows that the non-empirical research are strictly for analytical methods. Table 2.12 shows that most publications provide both pro-

Solution Extent	Empirical	Non-empirical
Single Change-point	30	1
Multiple Change-points	17	3
Multi up-gradation	12	0

Table (2.10) Number of publications per research type and solution extent

Proposed Method	Empirical	Non-empirical
Analytical	56	3
Curve-fit	4	0

Table (2.11) Number of publications per research type and proposed method

posal of a new model or method and as well as validation. The exception is a paper that describes a curve-fit approach that validates a multi-stage method [19].

2.3.4 Discussion

2.3.4.1 Focus

Looking at the progression of research over the years, we find that there is an increased interest in the subject for both academia and industry, but academia has the most contributions. Some of the top organizations conduct cohesive research in specific subjects, which is shown in the progress of research timeline (Fig. 2.12). Notice that the Timeline chart has letter references next to each contribution in square brackets, these are labels that refer to a list of references in Table 2.13.

Contribution Type	Analytical	Curve-fit
New method/model	3	0
Validate method/-model	2	1
Both	54	3

Table (2.12) Number of publications per contribution type and proposed method

Early efforts focused on dividing datasets into smaller sets and performing estimates on a smaller scale ([129][137][29][151]). The French National Center for Scientific Research sponsored research focused on trend analysis to detect change, dividing the failure dataset at the point of the change then applying SRGM to each partition ([75][76][91]). Between 2005 and 2014, National Tsing Hua University in Taiwan provided a body of work by C.Y. Huang. and C.T. Lin. Their focus was on providing a unified theory for SRGM by adding weights of means to the model. They also incorporated testing effort into their models and a Testing Compression Factor (TCF). TCF provides a ratio of change in fault detection between the testing phase and the operational phase ([59][89][54][57][87][88][58][60][83][82]). Huang et al. Investigated the use of queueing model for latent fault correction during software development. They applied the method on software with multiple change-points [56][55]. This work was later carried on by Yao and Zhang [145] by using only finite serving queueing model with various distributions and change-points for defect prediction. Between 2008 and 2014, Tottori University in Japan, Inoue and Yamada, conducted research regarding hazard rate modeling, proposing 2-dimensional models using time and effort and other environmental factors ([65][66][68][67][64][69]). Meanwhile, efforts were made to consider fault severity, learning effects and other environmental factors in the SRGM ([147][49][18][38][53][50][31][32]).

From 2010 until 2019, a remarkable research effort was conducted by researchers at the University of Delhi and Amity University in India in addition to members from Islamic Azad University, Iran and Rutgers, The State University of New Jersey, NJ, USA. They defined the term of multi up-gradation, referring to change due to updates in software systems. They incorporated several factors in their models such as fault severity, testing effort, environmental functions. They also applied their methods to Open Source Systems (OSS) and to multi-release environments. ([77]

[81] [44] [124] [80] [123] [131] [103] [122] [121] [43] [78] [79] [4] [2] [42] [125][149][93]). In addition, they applied fault detection models on agile software environment, where evolution comes for the changing user requirements and Sprints are treated like releases [94] [126]. Most recently, Anand et al. [13] used upgrades and updates from previous releases for defect prediction, in an attempt to enhance predictability of failures. Until 2018, the focus in literature was concerned with predicting failures based on past failure data. The first attempt to look into upgrades besides failures in order to predict system reliability is in the recently published paper by Anand et al. [13].

The most recent works focused on testing previously proposed methods. Nagaraju et al.[101] published a case study to validate the model they proposed in [100]. While Barraza [19] provided a case study applying the multi-stage curve-fit model proposed in [30] on multiple projects. And finally, Yang et al. [144][143] Proposed a reliability model with multiple change-points using Open source masked failure data. Masking failure data is used when the failure cause is unknown. Expectation Maximization (EM) algorithm was used to solve the likelihood function complicated problem in parameter estimation of the models.

2.3.4.2 Gaps

Legacy systems are valuable systems that are costly to maintain but cost more to replace. Having a reliable model to predict the effect of change on a software system's reliability is crucial. With available work focused on analytical methods, it is important to get a better understanding of how robust they are when specific assumptions they make are violated. For example, for a model that assumes perfect debugging, what degree of imperfect debugging will not cause inaccurate results? While some of this information exists for systems without change-points, engineering

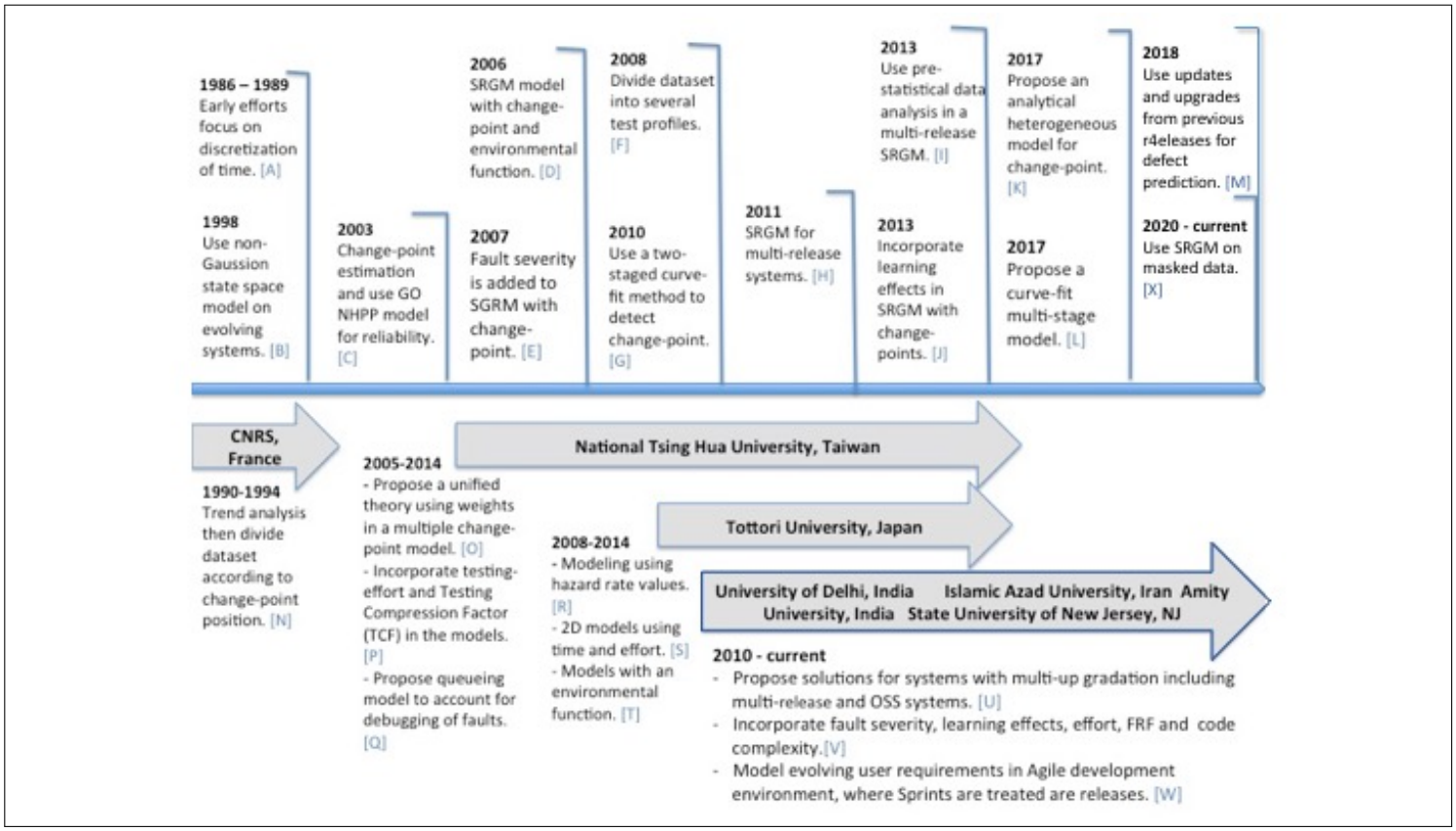


Figure (2.12) A timeline of research progress over the years for the most active research organizations

Reference Label	Corresponding References
A	[129][137]
B	[29]
C	[151]
D	[147]
E	[49]
F	[18]
G	[38]
H	[53]
I	[50]
J	[32][31]
K	[101][100]
L	[19][30]
M	[13]
N	[75][76][91]
O	[59][57][60]
P	[86][89][54][87][88][58][83][82]
Q	[56][55][145]
R	[65][64][69]
S	[66]
T	[68][67]
U	[77][81][43][4][103][42][94][149]
V	[80][44][124][123][122][121][131][78][79][2][125]
W	[126][93]
X	[144][143]

Table (2.13) Reference list of contributions highlighted in the timeline labels figure 2.12

guidelines still have to be developed for evolving and legacy systems. We believe that more effort should be made to use curve-fitting methods for evolving systems as they do not require assumptions. In addition, most of the work focuses on measuring how well the proposed model fits. While predictive ability of a model is briefly discussed in some papers, we would like to see how far can a model predict into the future before losing accuracy. How long can the same model be used and when do I need to use a different model or update its parameters? Moreover, we find that there is a need of high quality empirical work that uses current data and meets all aspects of case study methodology provided by [111] and [11].

While academia has the most contributions, an industrial point of view or more collaborative work between academia and industry would be a rich contribution. These collaborative efforts and recommendations will provide decision makers with better tools to make their systems evolve in a healthy and predictable manner.

We also found that simple single-change point or multiple change-point techniques are of major interest, and discussed in a broad spectrum of organizations. Multi up-gradation was proposed by a more limited number of researchers groups in a specific organization, in fact, the term, multi up-gradation, is exclusively used by them. Overall, looking into the progression of the body of work, it shows that this field is gaining interest and the quality of the proposed work keeps improving.

Finally, we find that current literature focuses on failure or defect prediction and using failure or defect data in evolving software systems. The existing studies do not use databases of Change Requests (CR) which include defects and maintenance requests. Modern software engineering databases contain Change Requests (CR), where prediction ability should be studied in an evolving legacy system.

2.3.5 Threats to Validity

Although we followed a systematic way to map studies in the field, there are some threats to the validity of our study.

The major threat is the possibility of missing important papers due to several restrictions explained in our inclusion and exclusion criteria in Section 2.3.1.3. Excluding papers that are written in languages other than English could have affected our study by not including important information in some of those papers. We could have also missed out on articles not published online since we are restricting our search to articles only available online. In addition, source selection of where papers are published would put this study at risk of leaving out valuable papers that weren't published by the sources selected here. There is also the restriction of excluding papers in fields other than software engineering which may exclude the efforts performed in this area of work that could potentially be applied to software systems. Having papers from other fields would make it more difficult to categorize and focus our efforts in producing a clear mapping study.

The choice of keywords and search strings was made systematically, but if some keywords or terms were missing this would have prevented us from reaching some sources. Forward and backward tracking of references was performed for further assurance that the maximum number of studies get recognized and included.

Finally, there is a possibility of excluding a valuable paper in the review by excluding a paper before skimming. This would be possible if the title or abstract was misleading or incomplete, or by voting to exclude the paper by the group of reviewers before fully skimming the paper. On the other hand, having more than one reviewer reduces the chance of discarding papers that are relevant.

2.3.6 Conclusion and Future Work

As the demand increases on businesses to grow, legacy systems are expected to continue to evolve accordingly. Evolution, updates, and changes get more complex; so it is more likely to find a sudden increase in failure intensity. The challenge then is to find a reliable method to predict system reliability and estimate future failures.

The main objective of this study is to obtain a holistic view of the existing studies in designing, validating and evaluating reliability models for evolving legacy software systems. Throughout this mapping study we identified 60 papers covering a spectrum of approaches of reliability models. These approaches are different in solution extent, techniques, and methods. In proposed methods, papers fell into one of the two main categories, analytical or curve-fit. We found that there is a high focus in the field of analytical methods in software reliability model with evolution, while curve-fit methods are limited although recent work has been published, ([30] [137][38][19]). Solution extent covered single-change point evolution, multiple change-point evolution or multi up-gradation. There is an increase of interest in multi up-gradation. The research in the field provides empirical work. A large proportion of case studies were of a high or a decent quality but some used low quality data, which affected the strength of their outcomes. We then conclude our work with some recommended areas for potential researchers to investigate: (1)Looking into more curve-fit solutions, (2) providing better quality empirical studies, (3) greater involvement of industry, (4) Use SRGMs in predicting change requests in general that include both failures and enhancements in databases to predict future change requests.

2.4 Effort Estimation using CR data

In order for software managers to succeed in managing their software projects they need to effectively control the cost. The accuracy of cost estimation is affected by the accuracy of effort estimation. It is reported that 60% to 80% of projects in industry encounter effort overruns according to a review of surveys conducted by Molokken and Jorgensen [95]. A main cause of the overruns is over-optimistic estimates. Optimistic estimates of effort gives a false sense of security about a project's cost which can turn disastrous for the project budget.

Sehra et al. [114] conducted a systematic mapping study on software effort estimation patterns and research trends for the time period 1996-2016. They concluded that twelve core research areas have been studied, these areas include using size metrics, machine learning techniques, estimation by analogy and others in order to estimate effort. The study lacks any references to studies that used SRGM from defect data to predict effort is not found on the study. In fact, most research uses effort to build a better reliability model to predict failures not the other way around.

To meet our objective, we ask the following research questions:

- RQ1: What are the publication trends for effort estimation in maintenance software systems?
- RQ2: Is CR data used in effort estimation?

To select relevant studies we defined inclusion and exclusion criteria as shown in Table 2.14.

Andrews et al.[16] used SRGM models to predict incidents for help desk operations. Using historical labor data to resolve these incidents, they estimate effort

Criteria Type	Criteria List
Inclusion Criteria	• Journal and conference papers • Peer reviewed • English language only • Software systems • Published papers available electronically • Statistical methods • Effort Estimation for maintenance software systems
Exclusion Criteria	• Books • Papers that are not peer reviewed • Papers written in languages other than English • Hardware systems • Papers not published/available electronically • Architecture-based or AI-based methods • Fields other than software engineering

Table (2.14) Inclusion and exclusion criteria for effort estimation literature

required. In our case, we deal with CRs instead of help desk cases. We predict CRs using reliability models, so we try to use CR data in effort estimation.

2.4.1 Change Request Prediction

According to the mapping study in Section 2.3, literature shows many studies are concerned with finding solutions in terms of goodness-of-fit. The predictive ability for the proposed solutions are measured for short-term predictions, i.e. looking into one or two time units into the future. Rana et al. [109] and Park et al [104] highlighted the issue of limited long-term prediction in research. Andrews et al. [16] used a month-by-month interval to evaluate prediction capabilities of their proposed system. Since long-term prediction is of major concern on this work to provide managers with better tools for release planning, we apply long-term CR predictions in our work.

2.4.2 Effort Estimation

Over the past two decades there has been considerable activity in the area of effort prediction with most approaches being typified as being algorithmic in nature. Well known examples include COCOMO [25] and function points [5]. Albrecht [5][6] has developed a methodology to use function points, which is a weighted sum of the numbers of inputs, outputs, master files, and software inquiries. His work suggested the use of a two-step work-effort validation procedure by estimating SLOC using function points and then using SLOC to estimate the work-effort. In 1997, Niessink and Van Vliet [102] used Function Points in effort prediction as well. Their experiment indicated that the performance of this method gave poor results compared to other effort estimation methods like expert estimation. Recently, Shah and Kama [115] suggested a new Software Change Effort Estimation (SCEE) model that uses a combination of Change Impact Analysis technique (CIA) together with Function Point Analysis (FPA). This work lacks empirical results to estimate the accuracy rate of the new model compared to the existing FPA method.

Evanco [39] used both external and internal measures of maintainability to predict effort. A statistical analysis of fault correction effort was conducted based on those factors: fault locality in the software architecture, software characteristics of the defective components associated with the fault, and cumulative changes made to the software (i.e.,fault correction, enhancement, and adaptation).

Analogy based effort estimation uses similar projects to estimate effort. Similarity is defined as Euclidean distance. The key activities for estimating by analogy are the identification of a problem as a new case, the retrieval of similar cases from a repository, the reuse of knowledge derived from previous cases and the suggestion of a solution for the new case. The method is validated on nine different industrial

datasets by Shepperd and Schofield [116], a total of 275 projects. In cases analogy outperforms algorithmic models based upon stepwise regression. Jorgensen et al. [74] added an adjustment to the analogy based estimation. When a project has an unusual high or low productivity then the estimate is adjusted towards the values of more average projects. AbdelMoez et al. [1] on the other hand dealt with outliers by excluding them to improve their prediction model. Weiss et al. [135] used the same analogy approach to predict bug fixing. Similarity between bugs is compared using the bug description. They combine reported effort for similar bugs as a prediction for the new similar bug. Hassouna and Tahvildari [51] proposed four enhancements to Weiss's method [135]: Data Enrichment, Majority Voting, Adaptive Threshold and Binary Clustering. Data Enrichment infuses additional issue information into the similarity-scoring procedure, aiming to increase the accuracy of similarity scores. Majority Voting exploits the fact that many of the similar historical issues have repeating effort values, which are close to the actual. An adaptive Threshold automatically adjusts the similarity threshold. Binary Clustering is used if the similarity scores are very low, which might result in misleading predictions. Numerical results are presented showing a noticeable improvement over the method proposed in Weiss et al. [135]. Dehghan et al. [37] proposed an approach to create a hybrid model based on selected individual predictors to achieve more accurate and stable results in early prediction of task completion effort and to make sure the model is not bounded to some attributes and consequently is adoptable to a larger number of tasks. The hybrid effort model uses three independent attribute sets: early metadata based attributes, title and description of software tasks. For this study two commercial projects of IBM were analyzed, called RQM and RTC. Better effort estimation results were shown. From the historical bug-fixing data, Zhang et al. [146] proposed an empirical distribution of bug-fixing time using Monte Carlo

Simulation to estimate the total amount of time required. They used a k-Nearest Neighbors (kNN) based method for classifying the bug-fixing time, which can improve the accuracy of existing methods

Calzolari et al. [27] estimated testing effort in CPU time using a predator/prey non-linear dynamic model. Defects in the software are considered prey and programmers solving defects are the predators. At the beginning of a new release a high number of defects will result in a good efficiency in bug detection and removal. However, as the number of defects decreases the effort required to discover any remaining defect increases. The phenomenon starts again with any new software release. The classical predator/prey model was introduced in 1972 by Shimazu et al. [119] to study ecological systems. The proposed model estimates the effort spent in maintenance and testing activities. Results show that prediction error did not exceed 30%.

De Lucia et al. [34] [35][36] used a model that takes size of components in LOC and number of tasks into determining the effort formula. Shihab et al. [118] concluded that using a combination of complexity, size and churn metrics are a better measure of effort than using LOC alone. Using LOC under-estimates the amount of effort required compared to their best effort predictor by approximately 66%. Thung [130] fucosed on code churn size to estimate bug fixing effort.

Jorgensen proposed an extensive review of studies related to expert estimation of software development effort [72]. He then compared the use of expert judgement, formal models, and a combination of these two approaches when estimating software development work effort. Sixteen relevant studies were identified and reviewed. The review found that the average accuracy of expert judgement-based effort estimates was higher than the average accuracy of the models in ten of the sixteen studies. Four of the reviewed studies evaluated effort estimates based on a combination of

expert judgement and models. The mean estimation accuracy of the combination-based methods was similar to the best of that of the other estimation methods [73].

Gokhale and Mullen [47] [46] observed the software defect repair times by observing several factors: Defect characteristics, the assigned personnel, and the resources used for the maintenance. These factors were used to model software defect repair times, and they are characterized by the Laplace Transform of the rate distribution (LTLN). Their results confirm that the LTLN distribution provides a statistically better fit to the observed repair times than either of the two most widely used repair time distributions, the lognormal and the exponential distribution.

Whigham et al. [136] proposed an automatically transformed linear model(ATLM) as a suitable baseline model for comparison against Software Effort Estimation methods. ATLM is simple yet performs well over a range of different project types. ATLM may be used with mixed numeric and categorical data and requires no parameter tuning. Sarro and Petrozziello [112] use linear programming to find a baseline for software effort estimation as well. The results of the study confirm the need to benchmark every other proposal against accurate and robust baselines.

To better estimate the attributes that contribute more in effort estimation Shukla and Misra [120] applied Principal Component Analysis (PCA) to two sets of data for two large sized software systems. Since effort estimation relies on several factors, PCA was used to reduce the data and predict the variance in software maintenance effort. Hayes et al. [52] on the other hand derived a model for estimating adaptive software maintenance effort using correlation analysis.

Several surveys and reviews have been conducted related to the body of research in effort analysis in software systems. In 2014, Rastogi et al. [110] published a survey on software effort estimation techniques. In this paper, a review of general techniques

and models regarding effort estimation has been done. After analyzing 25 papers, they conclude that there is no single technique that can lead to unambiguous results. None of the technique can perform exceptionally well when deployed alone. Hybrid approaches perform better than single approaches. Idri et al. [63] then conducted a systematic literature review of ensemble effort estimation in 2016. An ensemble effort estimation (EEE) technique combines several of the single/classical models found in the Software development effort estimation literature. They performed a systematic review of EEE studies published between 2000 and 2016, and selected 24 studies. They found that EEE methods achieve acceptable results, with mean MMRE ranging from 17.56% to 62.29%. In the same year Usharani et al. [134] also proposed a short survey on software effort estimation. They evaluated fifteen journal papers related to algorithmic methods and prediction methods of software effort estimation. In 2017, Sehra et al. [114] investigated research patterns and trends in software effort estimation during the period 1996 to 2016. Research was classified into twelve core research areas and sixty research trends. The research topics were: application specific estimation, estimation for web applications, project data selection, machine learning techniques, ensemble models, reviews and mapping studies, expert judgement, factors affecting estimation, size metrics and estimation by analogy.

We find that many papers use software metrics and characteristics such as LOC, function points, defect description, etc., to compare project similarity or estimate by analogy. The COCOMO model uses LOC data for effort estimation that has been widely used in industry. Some of the releases of our case study provide this information Effort estimation is further discussed and investigated in Chapter 8.

Chapter 3

Case Study

To perform our case study, we need to understand the subject system and the data we are working with. Since we are dealing with a legacy system, it is important to learn how the system evolved and changed through the years. We are also eager to learn more about the available attributes and data. The main objective of this work is to be able to predict effort and Change Requests (CRs) in a release using historic data from the same release. Therefore, we set our research questions to analyze CR data from a release and use them to predict future CR and effort. The research questions are as follows:

- RQ1: What does the data in the system look like?
- RQ2: Are there any possible relationships among different attributes that could help with CR and effort prediction?

To answer these questions we start by with brief description of the system in Section 3.1. We then describe and analyze data in Section 3.2. Analysis tools are specified in Section 3.3.

3.1 Subject System Specification

We use data of four releases of an aerospace system. It is a legacy system that has been around for three decades. It consists of over 1.2 million of lines of code, with most of the code written in C, C++, Java, and scripted code. There are over 850 components in 23 subsystems. The subsystems are organized by the functionality they provide and are referred as Computer System Configuration Items (CSCI). The subject system was first developed using the Waterfall process, then after a few years of operation, the Spiral development process was adopted. In each release, new requirements are addressed and new functionalities added. Maintenance included corrective maintenance, adaptations (e.g. to new hardware), perfective maintenance (e.g. performance improvements) and enhancements. Most of the maintenance effort was adaptive and corrective while less effort was preventive or perfective. One of the major enhancements of the system occurred when it was upgraded to a new hardware and operating system and code was converted to an object oriented programming language. Therefore, each release consists of changes that were made to add new functionality as well as to correct defects found throughout the development lifecycle including defects found during operation.

CR related metrics generated by Change Requests (CR) are the main focus of this study. Each CR is written to report a problem and is recorded in a CR tracking system (ClearQuest).b The CR tracking system provides work-flow management and CR life-cycle traceability. Software configuration management, version control, and workspace management are provided by ClearCase. While ClearQuest provides CR metrics, ClearCase contains the repository of the actual source code versions driven by CRs. Due to the unavailability of prior historical CR data, this study was based on CR data from four releases.

3.2 Data

3.2.1 Available Data

ClearQuest is a CR tracking system that provides the workflow process of a software Change Request (CR). Each CR has several attributes included in its state within the workflow cycle such as submitted, pending, rejected, approved, closed, etc. CR attributes are recorded in ClearQuest. This provides a detailed description of each CR. A CR is a data record that contains the following attributes which may or may not have been filled in:

1. CR ID: System generated identifier.

2. Request Type: Discrepancy or Enhancement.

   - Enhancement for a changing requirement or customer directed enhancement. Also used for perfective, preventive, and adaptive efforts. The enhancement falls into one of three categories:
     - ANOM: Anomaly during development until integration test is complete.
     - SCR: Software Change Request initiated in the development phase before release.
     - STR: Software Change Request is an operational test request during the customer testing phase.
   - Discrepancy for corrective maintenance. It is referred to as DR.

3. Functional Area: Records the functional area that is affected by this change request.

4. Customer Priority: Indicates when the CR should be delivered. CRs are divided into two categories: CAT I for CRs to be fixed in the current release and CAT II, for CRs to be fixed in the next release. Within each category CRs are sub-categorized based on impact and severity:

    - C1E – CAT I Emergency, if not incorporated severe consequences may result. Needs to be fixed immediately
    - C1U – CAT I Urgent, impact less severe than C1E, but needs to be incorporated immediately.
    - C1R – CAT I Routine. No mission critical impact, correction can be implemented alongside a scheduled maintenance effort.
    - C2E – CAT II Emergency, if needs to be fixed in the next release with no work around.
    - C2U – CAT II Urgent, impact is less severe that C2E, however has reasonable workaround that minimizes impact.
    - C2R – CAT II Routine. No mission critical impact, correction can be implemented in the next release.

5. SLOC Fields: Provides information about the changed code base:

    - SLOC Added: The total lines of added code.
    - SLOC Deleted: The total lines of deleted code.
    - SLOC Generated: The total lines of autogenerated code.
    - SLOC Modified: The total lines of modified code.

6. Actual Effort: Hours spent on the CR.

CR Database Fields	Release 1	Release 2	Release 3	Release 4
ID Number	Y	Y	Y	Y
Change Type / Enhancement	Y	Y	Y	N
Change Type / Discrepancy	Y	Y	Y	Y
Priority	Y	Y	Y	Y
Functional Area	N	N	N	Y
Submit Date	Y	Y	Y	Y
Actual Completion Date	Y	M	M	Y
Actual Effort	M	M	M	M
SLOC Added	N	M	M	Y
SLOC Modified	N	M	M	Y
SLOC Generated	N	M	M	Y
SLOC Deleted	N	M	M	Y

Table (3.1) Attributes available for each release

7. Submission Date: The Date the CR was submitted.

8. Actual Completion Date: The Date the CR was closed.

3.2.2 Data Preparation

3.2.2.1 Data Imputation

After data extraction and definition, data is then prepared for analysis. In our dataset we needed to handle missing data. Dealing with missing data is a sensitive matter in order to not distort a dataset by creating noise or biases. Many factors contribute to the decision of how to handle missing data such as how much data is missing? What type of data is missing? Can we derive missing data values or not? Are there data patterns? In some cases attributes with missing data can be ignored or removed, usually when the amount of missing data is relatively low, i.e. less than 5% as suggested by Pyle [107]. In other cases, missing data is replaced with an estimated value of the data point such as mean or mode. The process of substituting missing data with an estimated value is called data imputation.

In our database we had to deal with missing data in each of the four releases. Table 3.1 summarizes the attributes available for each data record in each release, these attributes are explained in Section 3.2.1. An attribute set without missing data is marked as "Y". If the attribute set has no data, we marked it with "N". If the attribute is available but with missing data we marked it "M". As we can see that Release 1, 2 and 3 "Functional Area" as attribute. Release 4 has no "Change Type" info available. In addition, the first release has no SLOC attributes.

We handled missing data in each release differently. "Actual Effort" and "Actual Completion Date" and "SLOC" fields" are the field we found missing data. We handled each situation as explained below:

- Actual Completion Date has values missing in release 2 and release 3. The missing data in release 2 is about 1% of the data. Since the percentage of missing data is so small and we don't consider completion date has any contribution in determining the actual effort, we ignore those values. All data fields with missing completion date have actual effort and submission data available. In release 3 the percentage of missing completion date is almost 14%. For the fields with missing actual completion date in this release has no data regarding actual effort or even SLOC. When asking the data providers about them they said that "Some completion dates are blank due to CR (s) returned to a "Pending" or "Rejection" states". Therefore, the CRs with missing completion date along with actual effort and SLOC are ignored when performing CR and effort analysis.

- Actual effort in all releases had some unrealistic values such as (0) or (-1). These values were later reported to be human errors caused by manual entry of values by staff. We treated these values as missing data. Although they

were less than 5%, we replaced them with the mean values of actual effort in each release. We found that ignoring the value of hours spent will affect the effort rate so we decided that the mean value would cause less distortion of the data.

- SLOC values were recorded for release 2, 3 and 4. The second release has most of the SLOC values missing. In fact, SLOC data is available for less than 20% of the total CRs. Therefore, imputation of this type of data will not reflect realistic data. Consequentially we decided to not use the SLOC data for release 2. In release 3, SLOC values are missing for up to 28% of the fields. When ignoring the SLOC values for the ignored cases due to missing actual effort and actual completion date (as mentioned in the point 1 of this list), then we are left with almost 13% of SLOC data missing. Since in most cases SLOC is filled for the field that has been updated and the other fields are left such as updating SLOC added with 10 hours and leaving other SLOC as blank means that no work has been done in deletion or modification. In addition, SLOC values have very small number of extreme outliers that could distort the data a lot while they do not represent the mass majority of the data population. So we use the median value of CRs with similar actual effort, or with the closest actual effort and calculate the median of Added SLOC, Modified SLOC, Deleted SLOC and Auto-generated SLOC, to replace missing values.

3.2.2.2 Data Trends

Data trends are patterns of data which changes gradually over time. Outliers represent a few values that lie far from the mainstream data. If data values range

Release	CRs	Weeks	SLOC A	SLOC M	SLOC G	SLOC D
Release 1	486	554	N/A	N/A	N/A	N/A
Release 2	898	433	10417	4079	233794	3818
Release 3	401	472	64627	68423	2096970	14398
Release 4	211	398	69789	113931	1586299	4071

Table (3.2) Number of CRs, Total Number of Weeks and Size of Change per Release

between two boundary values, outliers usually are values that are out of the normal range. In some cases, outlier values create noise, so they are treated like missing values in terms of replacement or leaving them out. In other cases we need outlier values to provide better analysis of a dataset. Box plots are useful graphical representation for describing data behavior. They the median and the lower quartile Q1 and upper quartile Q3 which represents the 25th and 75th percentiles. the difference (Q3 - Q1) is called the interquartile range or IQR. A box plot is constructed by drawing a box between the upper and lower quartiles with a solid line drawn across the box to locate the median. A point that resides beyond these lines is an outlier. Highlighting these terms are essential in describing our data in Section 3.2.3.

3.2.2.3 Data Visualization and Modeling

In order to look for trends, outliers or any relationships among data-points we worked on data visualization and analysis. Data visualization is presenting data and relationships in a graphical format using charts or tables. When data is visualized it is easier to observe trends, to discover noise and outliers and to model data and then find better solutions for the system. We used Microsoft Excel to visualize our data using graphs and charts as demonstrated in Section 3.2.3.

3.2.3 Data Analysis

Exploratory data analysis (EDA) is the first step in processing data. It is an approach to summarize and identify the main characteristics of a dataset, usually with visual methods. John W. Tukey [133], introduced the phrase exploratory data analysis (EDA) as

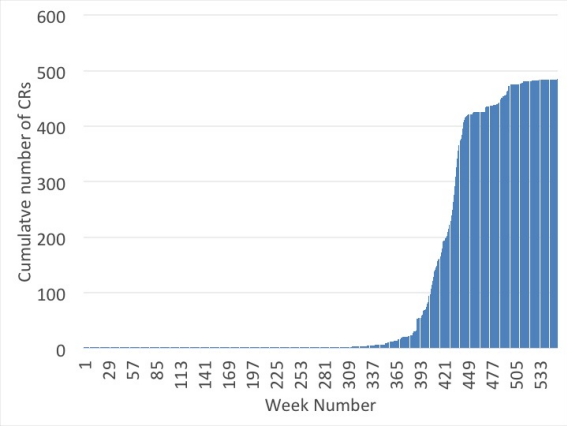
> Procedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and results of (mathematical) statistics which apply to analyzing data.

EDA is beneficial in extracting important variables, detecting outliers, understand underlying structure of a dataset, finding trends, and developing models. In this section, we demonstrate our data analysis for each release by demonstrating some basic statistics and highlighting our observations on the areas that needs to be further investigated.

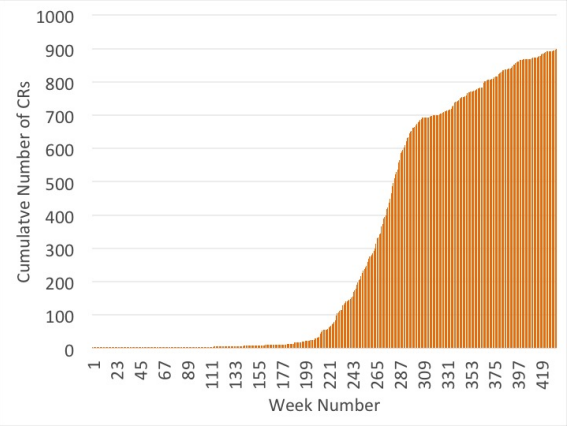
3.2.3.1 Change Requests

For this system we have CR data for four releases. Each release contains hundreds of CRs, and each CR is identified by a CR identifier. A CR contains a record of CR data as explained in Section 3.2.1. Table 3.2 shows the number of CRs collected per release and the total number of weeks for each release. The first release has a total of 486 CRs in 554 weeks. The second release has more CRs, 898 CRs in 433 weeks. Release 3 has a total of 401 CRs in 472 weeks. Release 4 has of 211 CRs in 398 weeks.

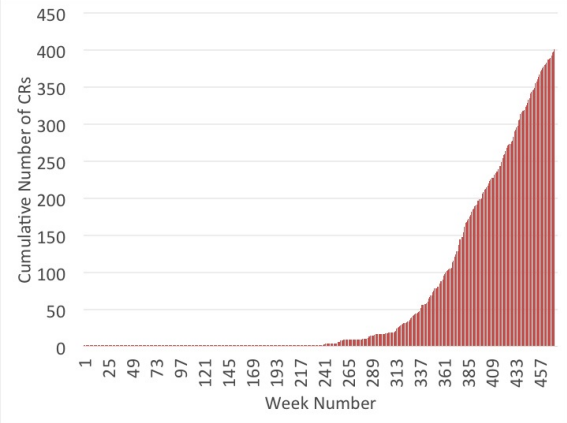
The cumulative number of CRs were collected and grouped on a weekly basis to show the pattern of growth in CRs per release. Figure 3.1 shows the cumulative



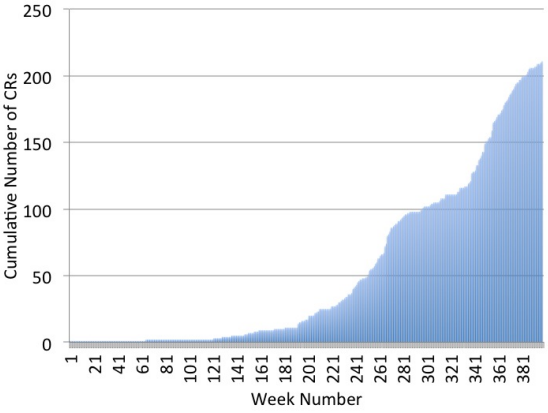
(a) Cumulative No. of CRs for Release 1



(b) Cumulative No. of CRs for Release 2

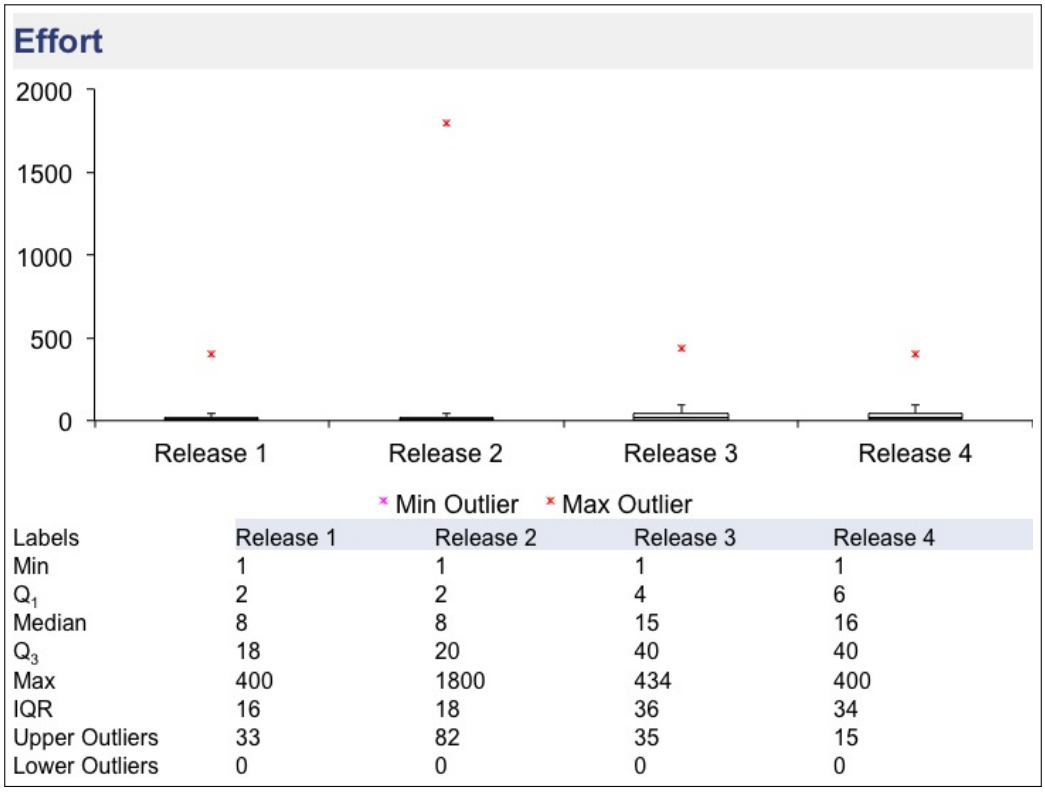


(c) Cumulative No. of CRs for Release 3



(d) Cumulative No. of CRs for Release 4

Figure (3.1) Cumulative Number of CRs for all four releases collected weekly



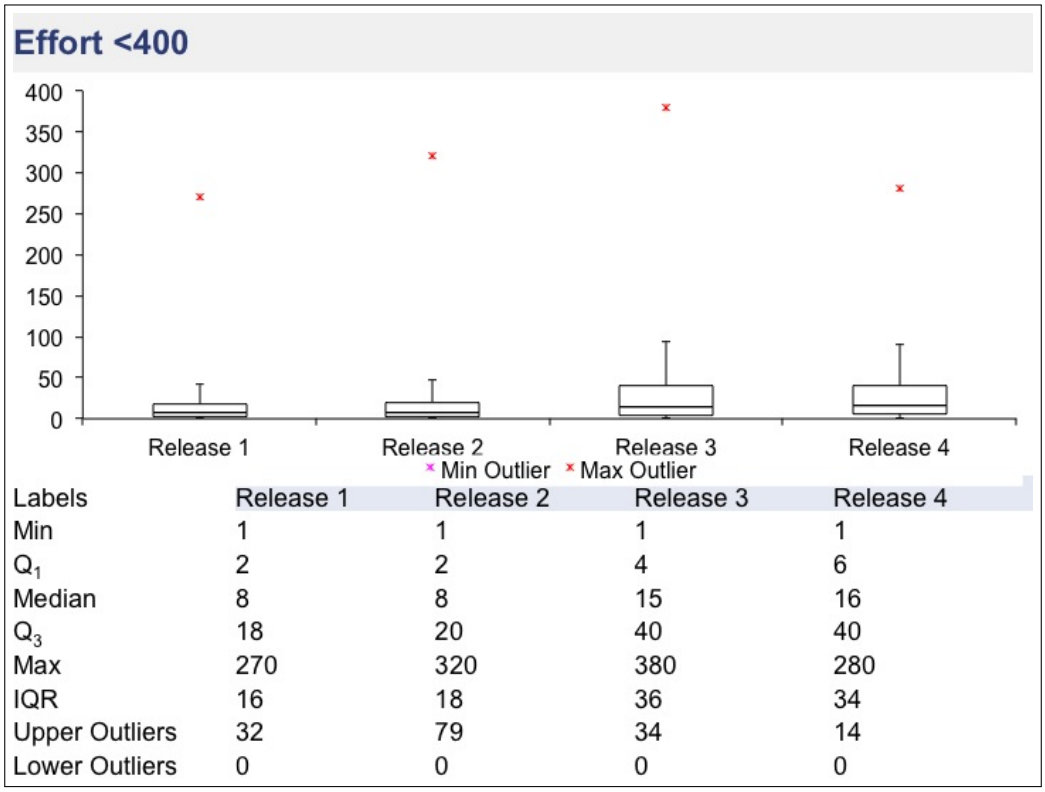
Labels	Release 1	Release 2	Release 3	Release 4
Min	1	1	1	1
Q_1	2	2	4	6
Median	8	8	15	16
Q_3	18	20	40	40
Max	400	1800	434	400
IQR	16	18	36	34
Upper Outliers	33	82	35	15
Lower Outliers	0	0	0	0

Figure (3.2) Actual Effort per release in hours

number of CRs collected for each release. Looking at the patterns we can observe some changes in the growth rate of CRs in all four releases. These changes cause changing CR patterns throughout the release. We identify these changes as change-points, which will be discussed later in Chapter 4.

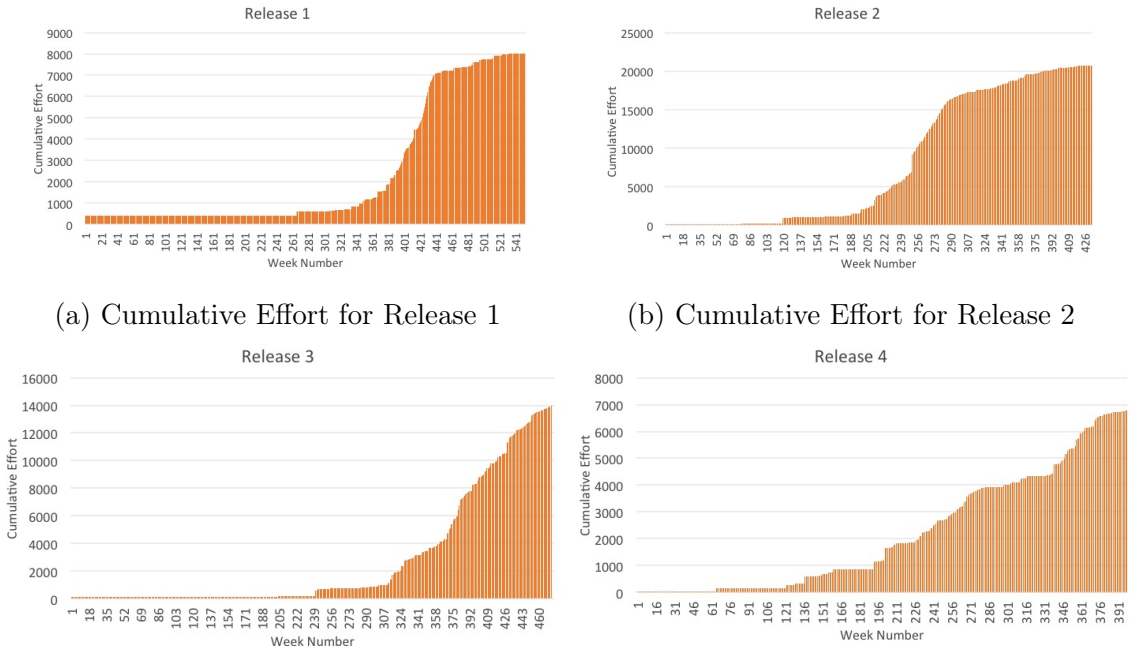
3.2.3.2 Effort

Effort is represented by an integer number of the number of hours for CR resolution. Effort does not reflect calendar-time in our case but man-hours. We refer to effort and number of hours interchangeably in this document. Figure 3.2 shows a boxplot chart of actual number of hours per release. The median effort for Release 1 and Release 2 is 8 hours, the median effort for Release 3 is 15 hours and for Release 4 is 16 hours. The extreme outliers made the chart less clear in terms of effort per



Labels	Release 1	Release 2	Release 3	Release 4
Min	1	1	1	1
Q_1	2	2	4	6
Median	8	8	15	16
Q_3	18	20	40	40
Max	270	320	380	280
IQR	16	18	36	34
Upper Outliers	32	79	34	14
Lower Outliers	0	0	0	0

Figure (3.3) Actual Effort per release in hours, excluding extreme outliers above 400 hours



(a) Cumulative Effort for Release 1

(b) Cumulative Effort for Release 2

(c) Cumulative Effort for Release 3

(d) Cumulative Effort for Release 4

Figure (3.4) Cumulative Number of CRs for all four releases collected weekly

release, therefore we attempt to exclude extreme outliers. Extreme outliers are any data values which lie more than 3 times the interquartile range below the first quartile or above the third quartile, i.e. extreme outliers are data points that are more extreme than Q1 - (3 * IQR) or Q3 + (3 * IQR). Following this process we found that we have 57 outlier values. For better visibility of the boxplots we excluded the extreme top outliers which have a value of 400 hours or greater in Figure 3.3. The removed outliers are of priority C2R (Category 2 Routine) in Release 4, and of type SCR in the first three releases, SCR is a Change Request initiated in the development phase before release for the first three releases.

We collected cumulative effort on a weekly basis to observe the growth of effort in each release. Figure 3.4 shows different trends of cumulative effort growth among releases. We can see a change in growth rate of cumulative effort for each release. These changes are due to changes in the CR rate. In fact looking at the growth

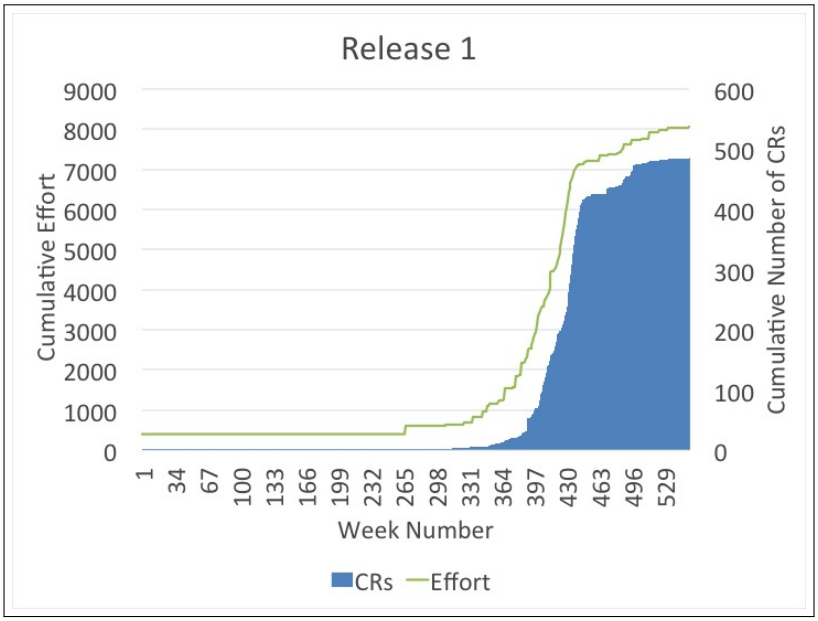


Figure (3.5) Cumulative effort vs. cumulative number of CRs for Release 1

rate of cumulative effort reminds us of the pattern we saw earlier in the CR rate, see Figure 3.1.

To compare the growth of cumulative number of CRs to the cumulative effort we combined them in Figure 3.5 for Release 1, Figure 3.6 for Release 2, Figure 3.7 for release 3, and Figure 3.8 for release 4. We find similarities in the growth rate of cumulative CRs and cumulative effort growth patterns. There are some weeks when the effort curves deviate from the CR curve but in general they follow patterns that are similar in most of the weeks. We can see that the patterns of Release 1 in Figure 3.5 and Release 4 in Figure 3.8 show very similar growth rates of cumulative CRs and cumulative effort, sharing the same change-points in weeks were change-points occur. Release 2 and Release 3 shown in Figures 3.6 and 3.7 relatively, also share similarities between cumulative effort growth and cumulative growth of CRs but with some different change-points.

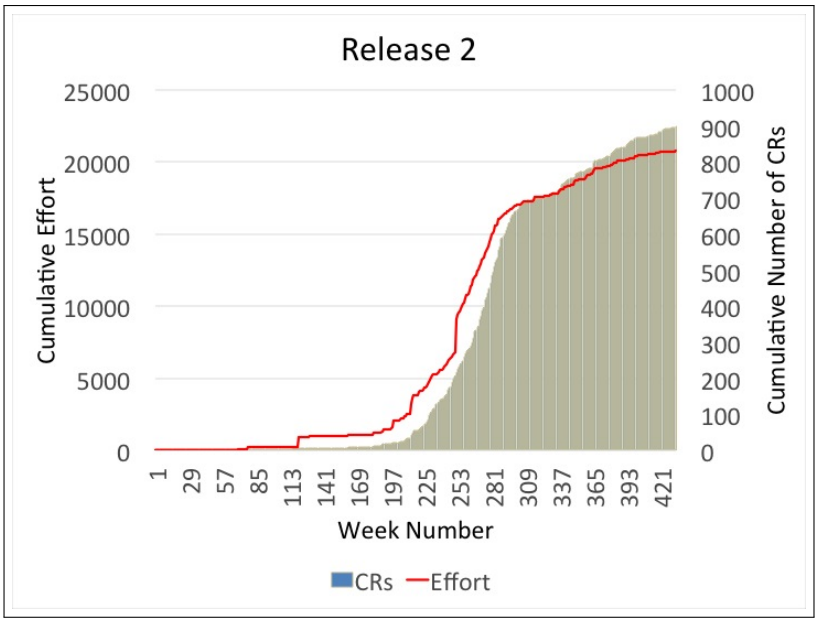


Figure (3.6) Cumulative effort vs. cumulative number of CRs for Release 2

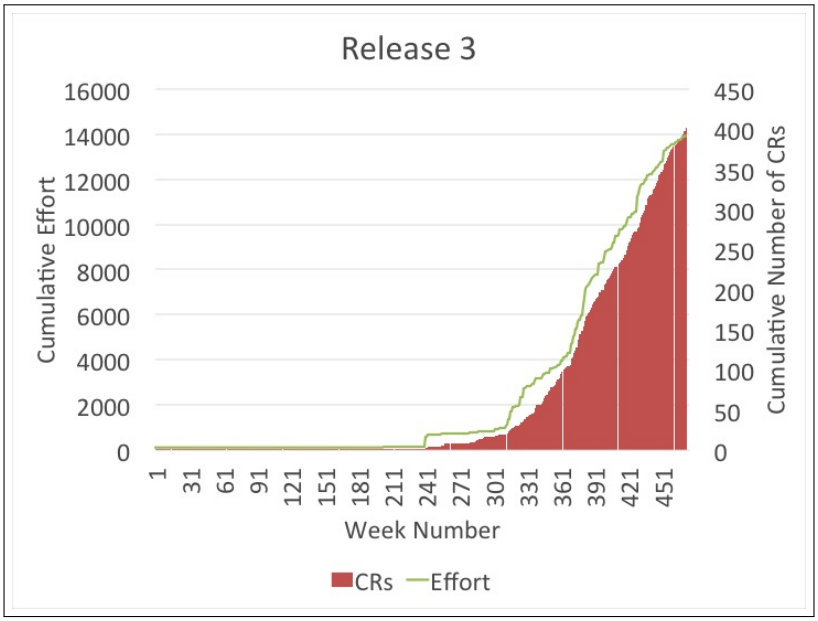


Figure (3.7) Cumulative effort vs. cumulative number of defects for Release 3

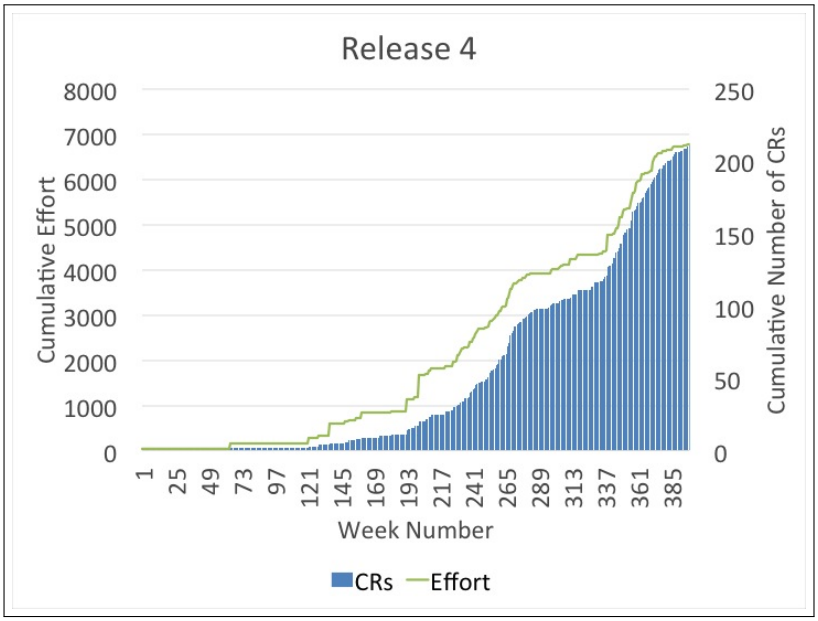


Figure (3.8) Cumulative effort vs. cumulative number of CRs for Release 4

3.2.3.3 Customer Priority

When looking into the data of customer priority, we investigate the data with two questions:

- Q1: Can we find special patterns in customer priority in relation to CRs?
- Q2: Can we find special patterns in customer priority in relation to Effort?

In terms of CRs, there are six levels of priorities for each CR as explained in Section 3.2.1. Table 3.3 shows the total number of CRs found of each specific priority. We find that the highest number of CRs among all releases are of the lowest priority to be resolved in the next release C2R. The lowest number of CRs are of priority C1E, in fact some releases have no CR of this priority, such as Release 3 and Release 4. All four releases have more CRs with C1U priority than C1E CRs. CRs of C1U are about 5% of the CRs in Release 1 and Release 2. They make of only 1.4% of CRs in Release 3 and 2.3% of CRs in release 4. For priority C1R, Release 1 has none,

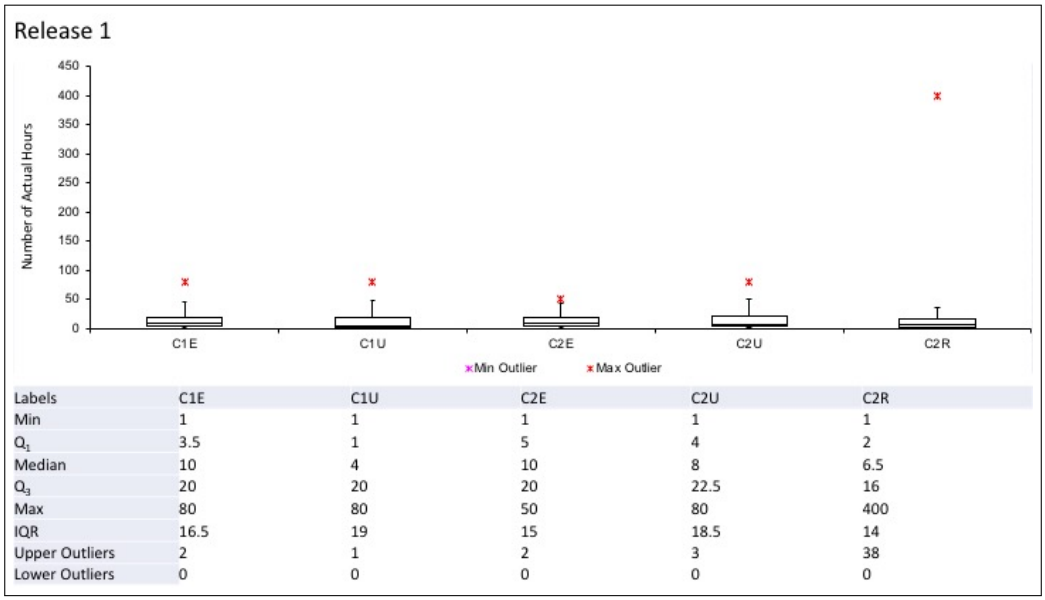
Customer Priority	Release 1	Release 2	Release 3	Release 4
C1E	15	6	0	0
C1U	27	43	6	5
C1R	0	14	13	11
C2E	68	88	11	6
C2U	32	62	20	14
C2R	344	685	351	175
Total No. of CRs	486	898	401	211

Table (3.3) Number of Cases per each Customer Priority

Release 2 has 1.5% of its CRs of this priority, Release 3 has about 3% and Release 4 has about 5%. The CRs that need to be fixed in the same release are less than 9% of total CRs in each release. The highest number of CRs are CRs that need to be fixed in the next release. The C2E CRs are 14%, 10%, 5% and 3% for the releases 1, 2, 3 and 4 relatively. C2U makes about 7% of the CRs in all the releases except the third release where it is about 5% of the CRs. The highest number of CRs are of a C2R priority which ranges between 71% - 88% of the CRs among the releases. This shows that the majority of CRs are of the lowest priority CRs and the lowest level of urgency, which are most likely to be maintenance requests. Thus any future CR is most likely a C2R, but this does not provide enough information to assist in CR prediction.

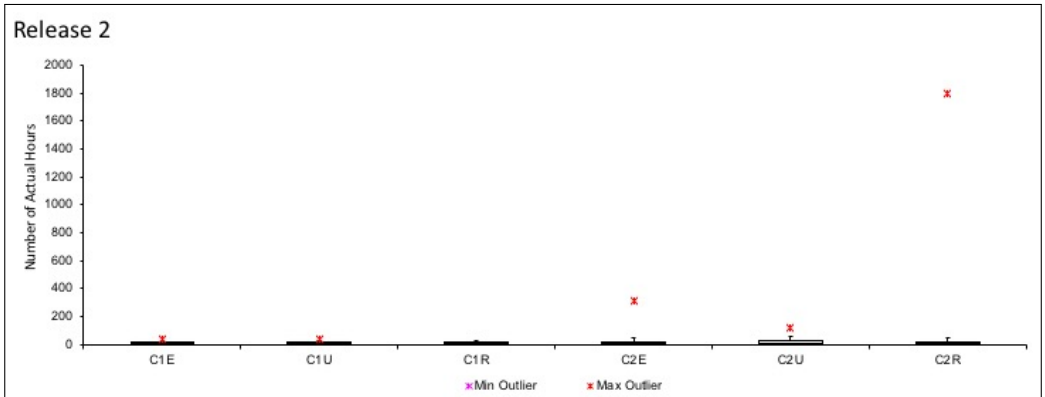
In terms of effort, we want to know if priority has an effect on effort. To visualize the effort spent on each CR according to their priority we demonstrated them in a box plot, see Figures 3.9, 3.10, 3.11 and 3.12.

In the boxplot Figure 3.9 we find that the minimum number of hours is 1 for all priorities and the maximum is 80 for C1E, C1U and C2U. C2E has a maximum of 50 and C2R has a maximum of 400, which is an extreme outlier. The median number of hours shows that C1E and C2E consume the most amount of effort, 10



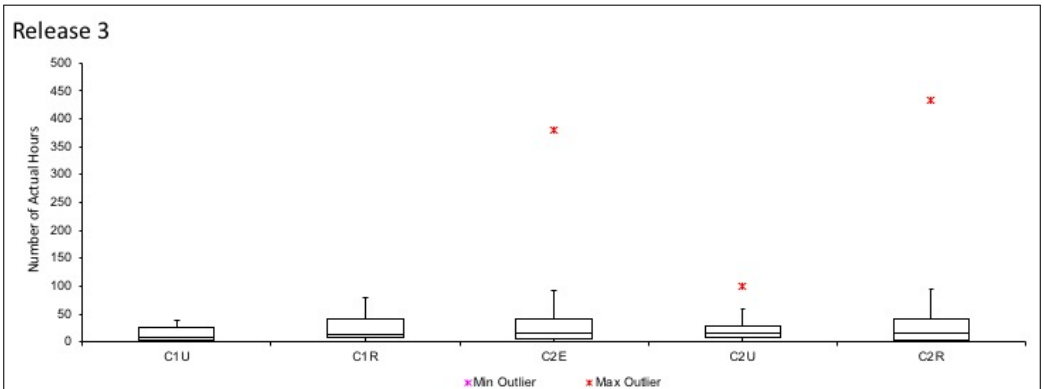
Labels	C1E	C1U	C2E	C2U	C2R
Min	1	1	1	1	1
Q_1	3.5	1	5	4	2
Median	10	4	10	8	6.5
Q_3	20	20	20	22.5	16
Max	80	80	50	80	400
IQR	16.5	19	15	18.5	14
Upper Outliers	2	1	2	3	38
Lower Outliers	0	0	0	0	0

Figure (3.9) Effort per Priority for Release 1



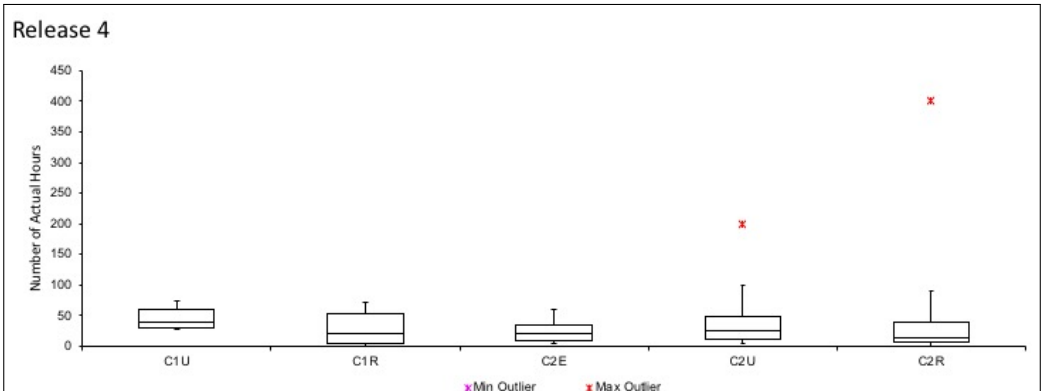
Labels	C1E	C1U	C1R	C2E	C2U	C2R
Min	2	1	1	1	1	0
Q_1	5	3.5	2.75	2	3	2
Median	5	5	8	8	10	8
Q_3	7.25	14	14	20	27	20
Max	40	43	30	310	120	1800
IQR	2.25	10.5	11.25	18	24	18
Upper Outliers	1	7	0	8	2	70
Lower Outliers	0	0	0	0	0	0

Figure (3.10) Effort per Priority for Release 2



Labels	C1U	C1R	C2E	C2U	C2R
Min	1	1	1	1	0
Q_1	3.5	8	5.5	9.25	4
Median	9	12	15	15.5	16
Q_3	25.75	40	40	29.25	40
Max	38	80	380	100	434
IQR	22.25	32	34.5	20	36
Upper Outliers	0	0	2	2	32
Lower Outliers	0	0	0	0	0

Figure (3.11) Effort per Priority for Release 3



Labels	C1U	C1R	C2E	C2U	C2R
Min	29	1	5	4	1
Q_1	30	4.5	8.5	12.75	6
Median	40	20	20	24.5	15
Q_3	60	53.5	36	48	40
Max	75	73	60	200	400
IQR	30	49	27.5	35.25	34
Upper Outliers	0	0	0	1	14
Lower Outliers	0	0	0	0	0

Figure (3.12) Effort per Priority for Release 4

hours, C1U has a median of 4, C2U has a median of 8 and C2R has a median of 6.5 hours.

Figure 3.10 shows the efforts in Release 2. C1E and C1U CRs has a median of 5 hours. C1R, C2E and C2R has a median of 8 hours and finally C2U has a median of 10 hours. The C2 cases has extreme outlier of hundreds and thousands of hours while C1 cases has maximum values that range between 30 and 40 hours.

In Release 3, C2 CRs medians are slightly higher than C1 CRs. With medians around 15 hours for C2 CRs and 9 for C1U and 13 for C1R we find that is no big difference in the number of hours. We also notice that the outliers for C2 CRs consume hundreds of hours which is extremely high compared to C1 CRs which are between 38 and 80 hours.

Finally, we observe the efforts in the fourth release Figure 3.12. The medians of C1U and C2R are 40 hours and the medians of C1R and C2E are 20 hours. The median of C2U is 48. The maximum values of hours in C1U, C1R and C2E are between 60 and 75, while C2U and C2R are 200 and 400 relatively.

When observing the effort required for CRs among different priorities we find that the extreme outliers of hours are higher in C2 CRs than C1 CRs. But when looking into the medians of hours spent on a CR we do not find a clear pattern. As the number of hours grow when priorities get lower in Release 3 we find the opposite happening in Release 4, while the other two releases have no specific pattern. We could not conclude any major observation regarding the relationship between CR priority and effort required per each priority that could assist us in effort prediction.

3.2.3.4 Functional Area

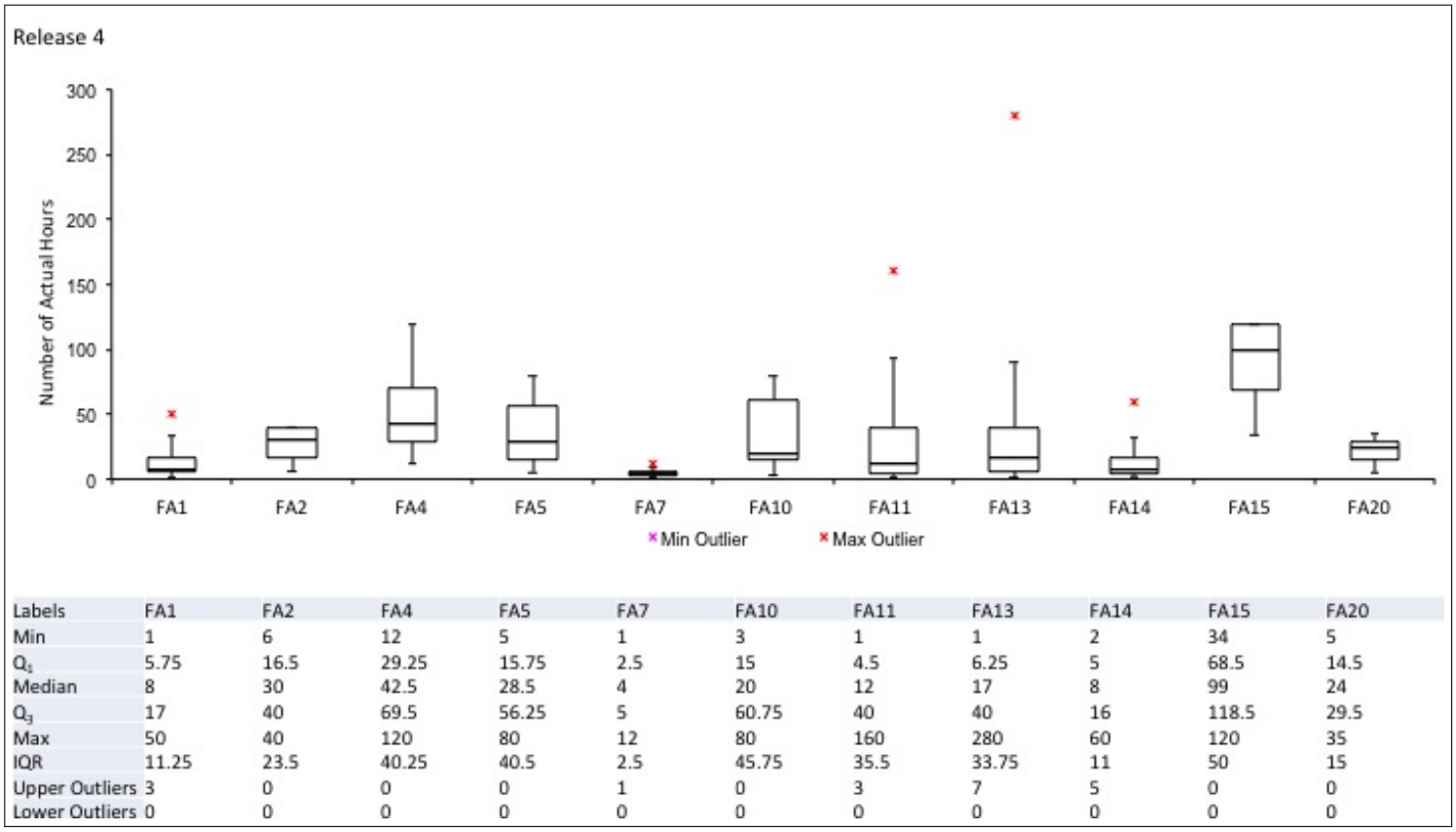
There are twenty different functional areas in the system affected by CRs. Functional areas were specified for CRs only for the fourth release. Table 3.4 shows the

Functional Area	No. of CRs	Total Effort (Hrs)	Average Effort
FA1	20	277	14
FA2	4	106	27
FA3	1	32	32
FA4	10	504	50
FA5	10	365	37
FA6	1	50	50
FA7	7	32	5
FA8	2	231	116
FA9	1	30	30
FA10	12	392	33
FA11	35	1037	30
FA12	1	10	10
FA13	74	2447	33
FA14	21	374	18
FA15	4	352	88
FA16	2	38	19
FA17	1	1	1
FA18	1	40	40
FA19	1	400	400
FA20	3	64	21

Table (3.4) Number of CRs, Total Effort and Average Effort per Functional Area

distribution of the number of CRs per functional area, the total effort per functional area and the average effort used to fix CRs for each functional area. Due to confidentiality requirements, the functional areas were coded as FA+Number code (FA1 to FA20). Like customer priority we look into the functional area data with two questions:

- Q1: Can we find patterns in functional areas with regard to the number of CRs that might make prediction possible?
- Q2: Can we find patterns in functional areas with regard to effort that might make prediction possible?



Labels	FA1	FA2	FA4	FA5	FA7	FA10	FA11	FA13	FA14	FA15	FA20
Min	1	6	12	5	1	3	1	1	2	34	5
Q_1	5.75	16.5	29.25	15.75	2.5	15	4.5	6.25	5	68.5	14.5
Median	8	30	42.5	28.5	4	20	12	17	8	99	24
Q_3	17	40	69.5	56.25	5	60.75	40	40	16	118.5	29.5
Max	50	40	120	80	12	80	160	280	60	120	35
IQR	11.25	23.5	40.25	40.5	2.5	45.75	35.5	33.75	11	50	15
Upper Outliers	3	0	0	0	1	0	3	7	5	0	0
Lower Outliers	0	0	0	0	0	0	0	0	0	0	0

Figure (3.13) Effort per Functional Area for Release 4 Excluding Functional Areas with less than three CRs

To answer the questions we look into the number of CRs per functional area. FA13 area has the highest amount of CRs with 74 CRs. FA11 comes next with 35 CRs. The remaining functional areas have fewer than 22 CRs per release. While some functional areas have a very high number of CRs, this is not true for the average effort in those functional areas. In fact, functional areas with the smallest number of CRs have the highest average efforts. For instance, FA19 has an average effort for its single CR of 400 hours. This CR is a Discrepancy request (DR) of priority of C2R, which is routine maintenance.

Figure 3.13 shows the amount of effort in each functional area as a series of boxplots. We excluded the functional areas that contain only one or two CRs. We find extreme outliers in FA11, FA13, FA14, FA7 and FA1. These CRs are all of type C2R, two of them are Discrepancies. The priorities of the CRs in different functional areas, especially the functional areas with many CRs, vary by type. We find some functional areas such as FA15, FA10 and FA4 have higher median values while other functional areas have very low medians such as the median in FA1, FA7 and FA14. When trying to use a functional area to predict effort we find that most functional areas have low CRs. The highest number of CRs are in FA13 and FA11, which have 74 and 35 CRs respectively. All other functional areas show no more than 20 CRs. Functional areas with a higher number of CRs do not necessarily have the highest effort. As we mentioned earlier, FA19 has the highest average effort, 400. FA8 has the second highest average effort (116), but with only two CRs. There is no obvious relationship between the density of CRs and the average effort per functional area, therefore we will not likely to be able to predict effort by using functional areas.

Release	Change	DR	No. of CRs	Total Effort	Avg. Effort
Release 1	SCR	DR	8	385	48
			41	1609	39
	STR	DR	4	10	3
			70	779	11
	ANOM	DR	2	6	3
			361	5155	14
Release 2	SCR	DR	30	2409	80
			89	3808	43
	STR	DR	5	284	57
			58	1062	18
	ANOM	DR	1	1	1
			715	13239	19
Release 3	SCR	DR	5	641	128
			80	4454	56
	STR	DR	2	172	86
			23	1038	45
	ANOM	DR	2	81	41
			289	7560	26
Release 4		DR	15	1199	80
			197	5583	28

Table (3.5) Number of CRs, Total Effort and Average Effort per Change Type

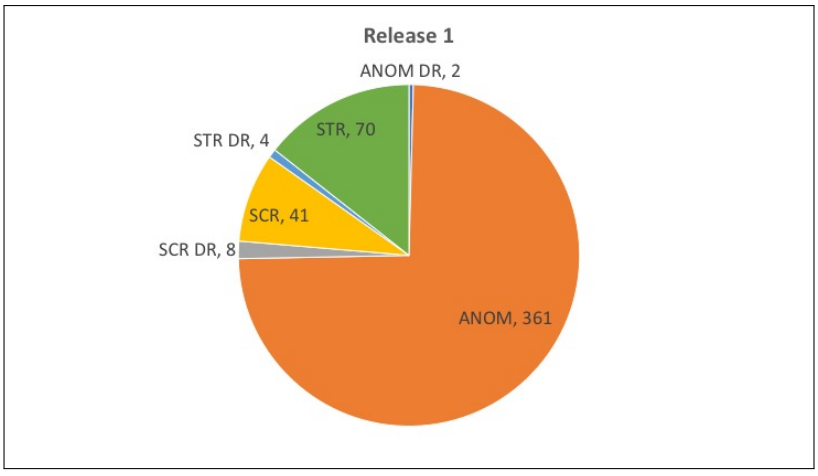


Figure (3.14) Number of CRs per Change Type for Release 1

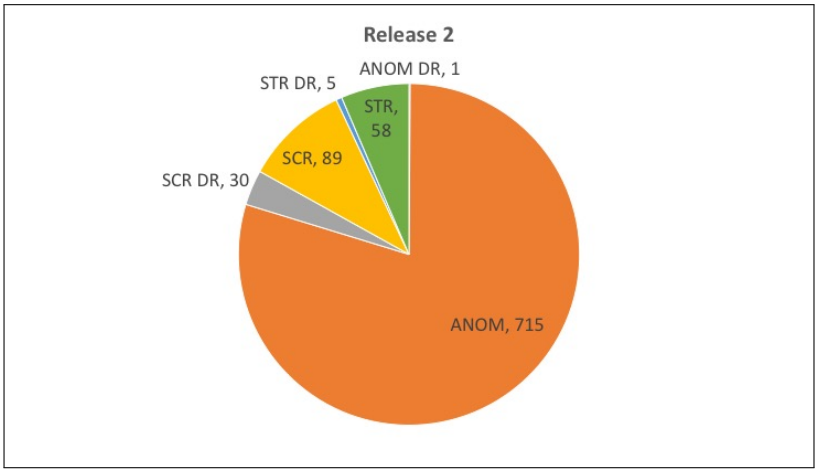


Figure (3.15) Number of CRs per Change Type for Release 2

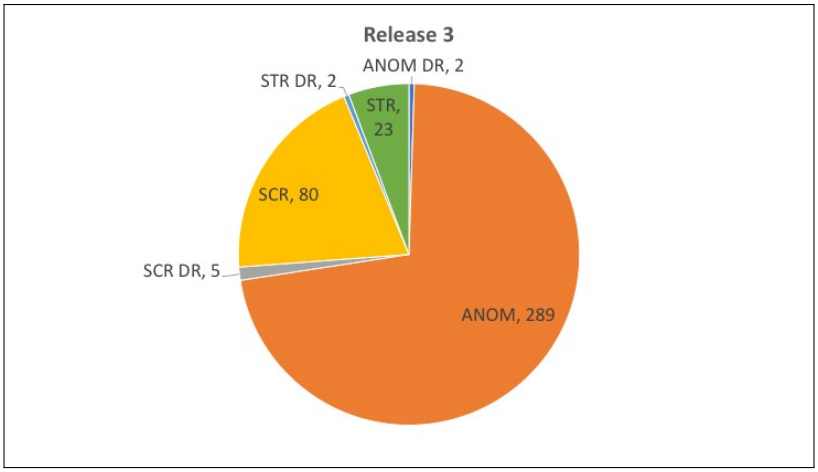


Figure (3.16) Number of CRs per Change Type for Release 3

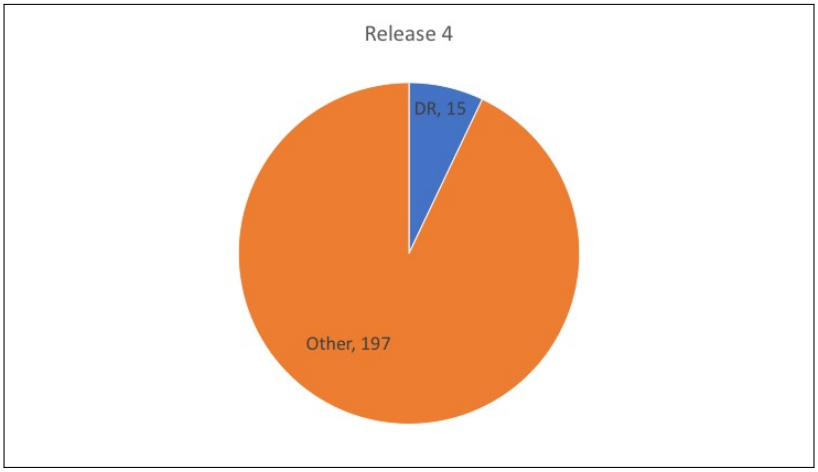


Figure (3.17) Number of CRs per Change Type for Release 4

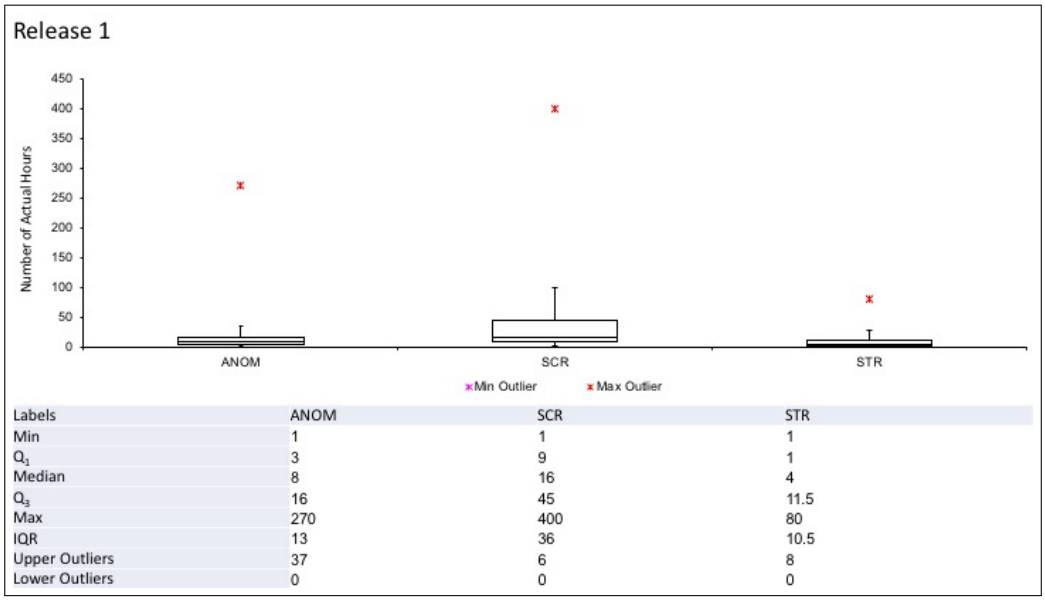
3.2.3.5 Change Type

As explained in Section 3.2.1 a change type of a request type can be an Enhancement under one of the following: ANOM, SCR or STR. Some of these enhancements may or may not be Discrepancies (DR). Like customer priority and functional area we analyze change type in terms of these two questions:

- Q1: Can we find patterns for CRs based on the change type?
- Q2: Can we find patterns for CRs with regard to effort for specific change types?

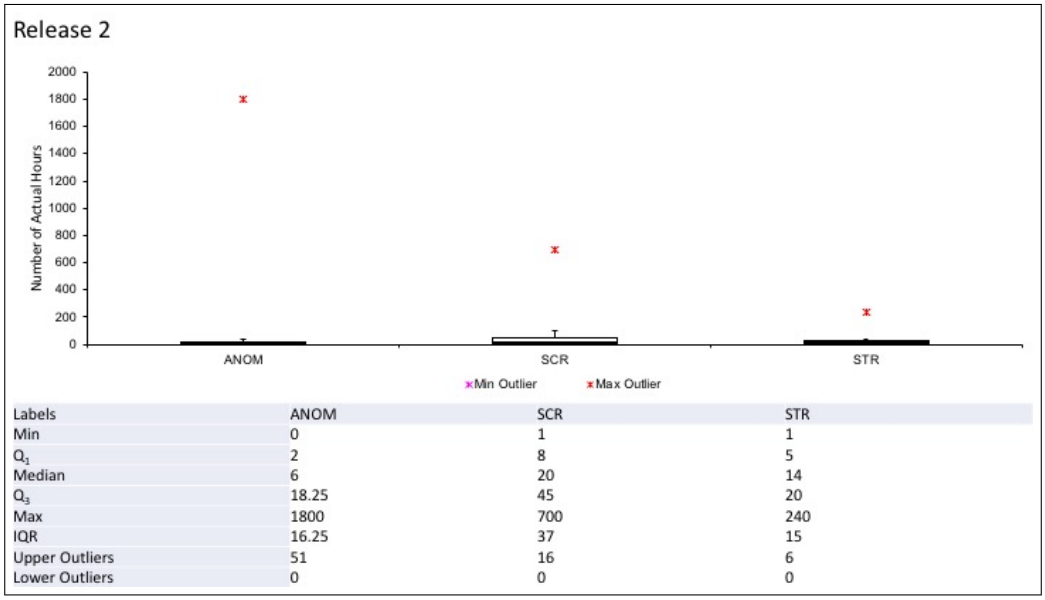
Table 3.5 summarizes the CRs and efforts according to their change type. For each release, we present the number of CRs, total effort and average effort. Figures 3.14, 3.15, 3.16 and 3.17 demonstrates the number of CRs per change type for each release. We find that the highest number of CRs are anomalies in the first three releases, which is about 75% of the CRs. SCR comes next, which is Change Request initiated in the development phase before release, and then STR, which are Change Requests during the customer testing phase. A very small portion of these CRs are DRs. Figure 3.17 shows the number of DRs vs. all other CRs. DRs are about 8% of the total number of CRs in Release 4. We do not have any information about Enhancement types for this release. We find that most CRs are anomalies and non-discrepancies.

Effort varies from release to release. In the first release, (Figure 3.18) we find that SCR accounts for the highest effort, next is ANOM and finally STR. In the second release (Figure 3.19), SCR has the highest median while ANOM has the lowest median. On the other hand ANOM has the highest outlier value. In the third release, STR has the highest median but the lowest value of extreme outlier



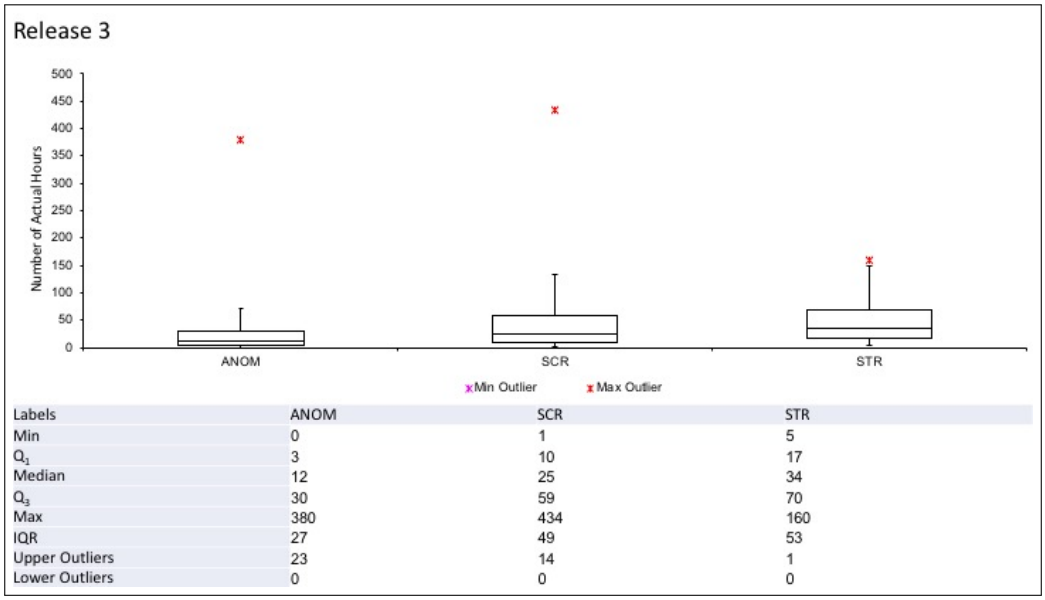
Labels	ANOM	SCR	STR
Min	1	1	1
Q_1	3	9	1
Median	8	16	4
Q_3	16	45	11.5
Max	270	400	80
IQR	13	36	10.5
Upper Outliers	37	6	8
Lower Outliers	0	0	0

Figure (3.18) Boxplot of effort per Change Type for Release 1



Labels	ANOM	SCR	STR
Min	0	1	1
Q_1	2	8	5
Median	6	20	14
Q_3	18.25	45	20
Max	1800	700	240
IQR	16.25	37	15
Upper Outliers	51	16	6
Lower Outliers	0	0	0

Figure (3.19) Boxplot of effort per Change Type for Release 2



Labels	ANOM	SCR	STR
Min	0	1	5
Q_1	3	10	17
Median	12	25	34
Q_3	30	59	70
Max	380	434	160
IQR	27	49	53
Upper Outliers	23	14	1
Lower Outliers	0	0	0

Figure (3.20) Boxplot of effort per Change Type for Release 3

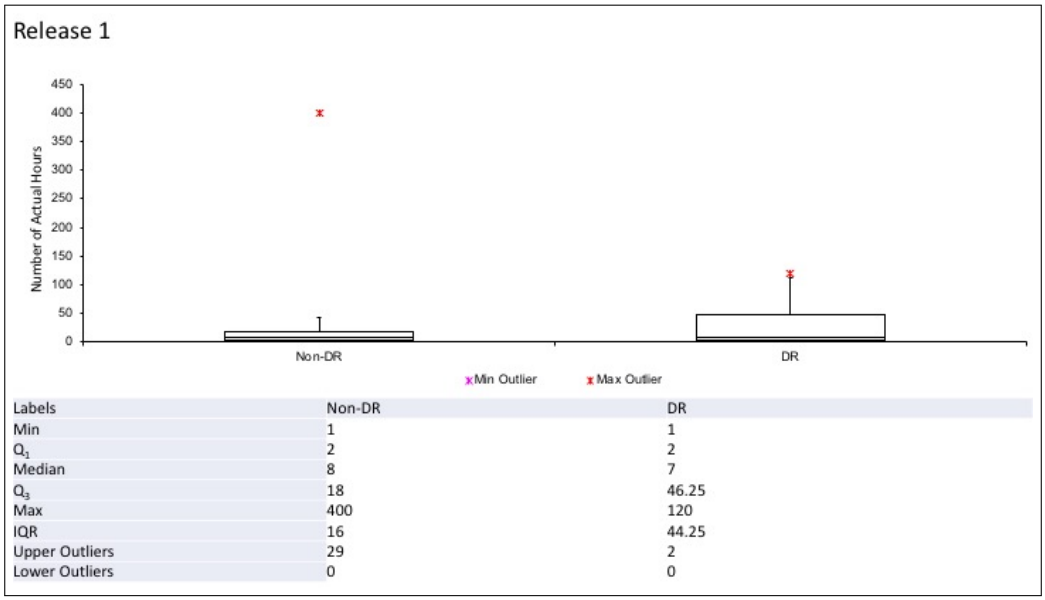
among the other change types, while ANOM has the lowest median while having a relatively high outlier. STR has the highest outlier value, see Figure 3.20.

We then compare the effort spent on DR CRs vs. non-DR CRs for the four releases. Figures 3.21, 3.22, 3.23, and 3.24 show that DR CRs require more effort than other CRs. This observation is consistent for every release. The averages of effort also vary depending on the work required for a CR but there is no clear pattern in Table 3.5 that could lead us to a conclusion in regard to effort estimation.

3.2.3.6 Lines of Code

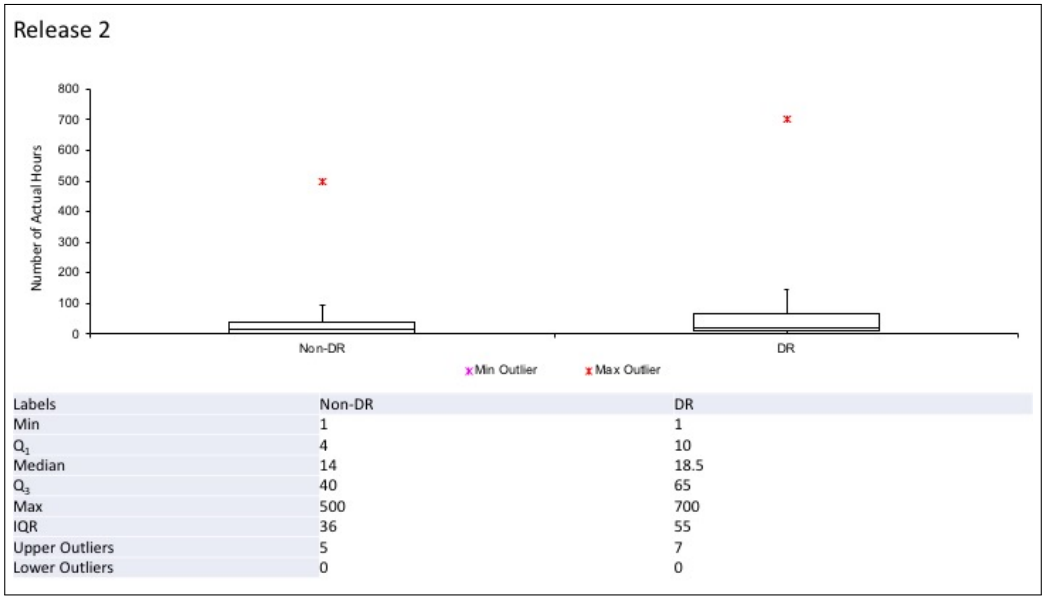
For each CR, lines of Code represent the number of lines of code that are added to the code, modified in the code, were deleted from the code or added using auto-generation tools. As mentioned earlier, the lines of code values were reported for the third and fourth release only. We also investigate if SLOC can be used in CR and effort prediction, so we ask:

- Q1: Can we find patterns in SLOC data?



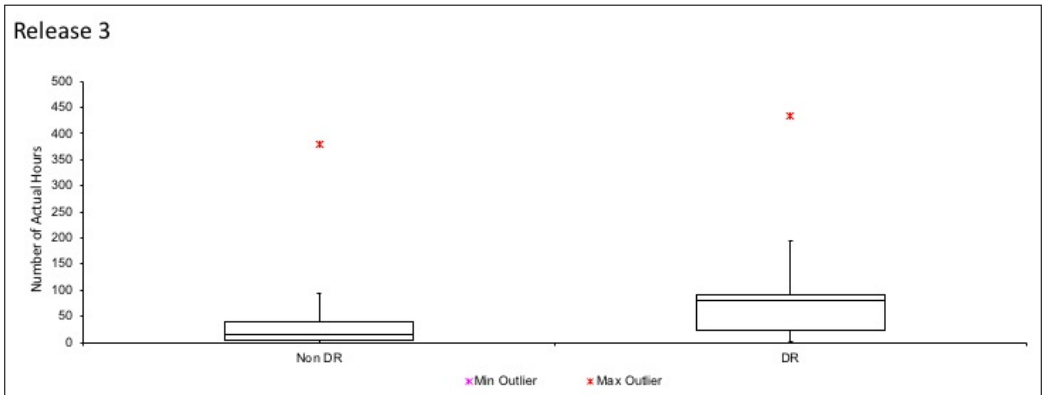
Labels	Non-DR	DR
Min	1	1
Q_1	2	2
Median	8	7
Q_3	18	46.25
Max	400	120
IQR	16	44.25
Upper Outliers	29	2
Lower Outliers	0	0

Figure (3.21) Boxplot of effort for DR CRs (Release 1)



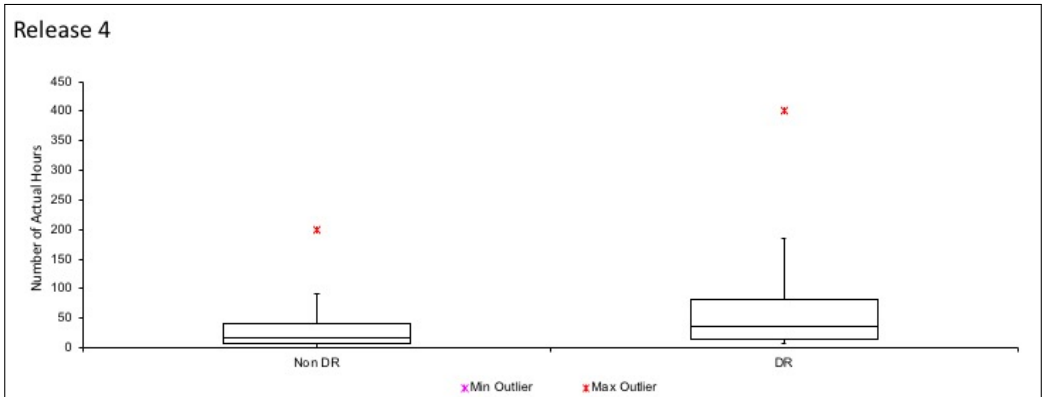
Labels	Non-DR	DR
Min	1	1
Q_1	4	10
Median	14	18.5
Q_3	40	65
Max	500	700
IQR	36	55
Upper Outliers	5	7
Lower Outliers	0	0

Figure (3.22) Boxplot of effort for DR CRs (Release 2)



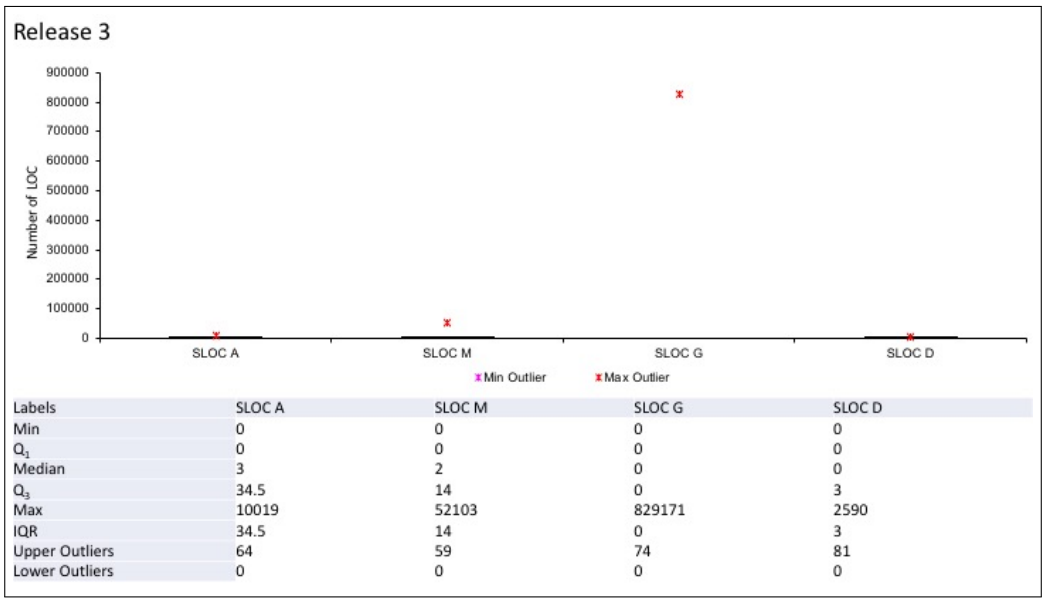
Labels	Non-DR	DR
Min	0	1
Q_1	4	24
Median	16	80
Q_3	40	92
Max	380	434
IQR	36	68
Upper Outliers	33	1
Lower Outliers	0	0

Figure (3.23) Boxplot of effort for DR CRs (Release 3)



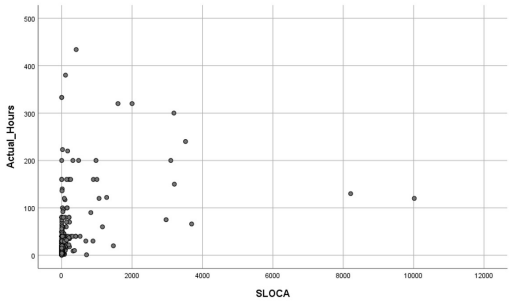
Labels	Non-DR	DR
Min	1	6
Q_1	6	13.5
Median	16	35
Q_3	40	82.5
Max	200	400
IQR	34	69
Upper Outliers	11	2
Lower Outliers	0	0

Figure (3.24) Boxplot of effort for DR CRs (Release 4)

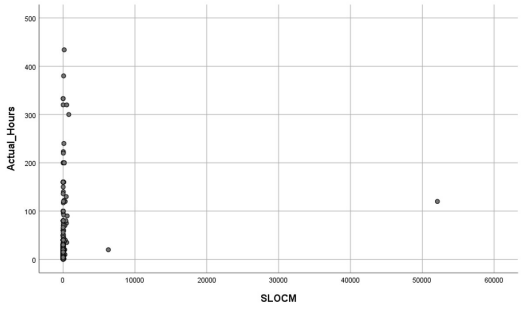


Labels	SLOC A	SLOC M	SLOC G	SLOC D
Min	0	0	0	0
Q_1	0	0	0	0
Median	3	2	0	0
Q_3	34.5	14	0	3
Max	10019	52103	829171	2590
IQR	34.5	14	0	3
Upper Outliers	64	59	74	81
Lower Outliers	0	0	0	0

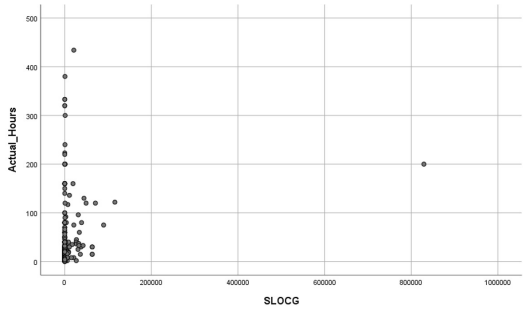
Figure (3.25) Boxplot of the number of SLOC Added, SLOC Modified, SLOC Deleted and SLOC Auto-generated (Release 3)



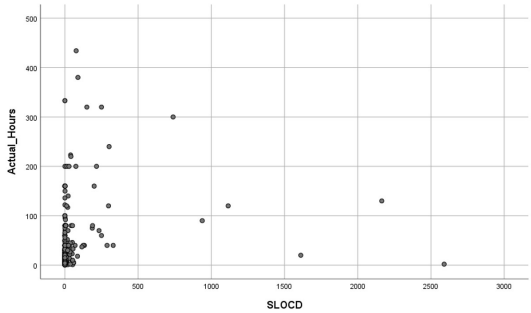
(a) SLOCA vs. Effort (Actual Hours)



(b) SLOCM vs. Effort (Actual Hours)



(c) SLOCG vs. Effort (Actual Hours)



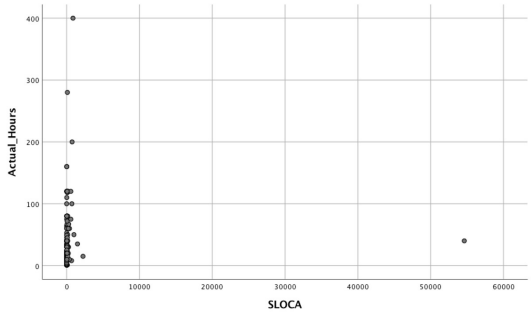
(d) SLOCD vs. Effort (Actual Hours4)

Figure (3.26) Scatter plot of SLOCs vs. Effort (Actual Hours) for Release 3

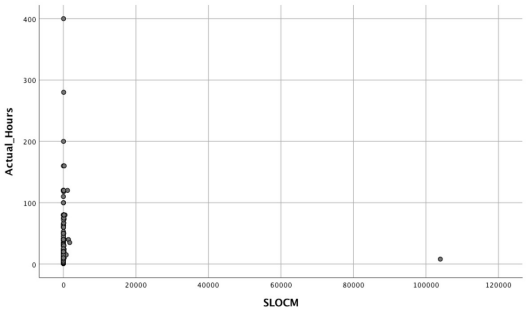
- Q2: How do SLOC data, CR data and effort compare?

We first check the SLOC statistics in Release 3. Figure 3.25 shows boxplot of SLOC for Release 3, (SLOC A) is for added LOC, (SLOC M) is for modified LOC, (SLOC G) is for auto-generated LOC and (SLOC D) is for deleted LOC. The outliers on the charts distort the presentation of the data, so the small values are not shown clearly. Notice the extreme outliers of SLOC Generated. This is because the auto-generated code consists of a large number of SLOC for GUIs. SLOC Modified are driven by merges when a baseline version is updated. We demonstrate the relation between effort and each type of SLOC using scatter plots in Figures 3.26 and 3.27. Figure 3.26 shows that in the third release most CRs report SLOCM and SLOCG that are close to zero regardless of effort. This also applies to SLOCA and SLOCD but with more points that are scattered. The four figures do not show that the plotted points could fit a regression function which makes the possibility of estimating effort using any of the SLOCs unlikely. Figure 3.27 shows the scatter plot of SLOCA, SLOCM, SLOCG and SLOCD vs. effort for the fourth release. The spread of the points are closer to zero in SLOCA, SLOCM and SLOCG, while SLOCD points appear to be more spread. These plots do not show patterns that may lead to a regression model that can be used to predict effort using any of the SLOC data.

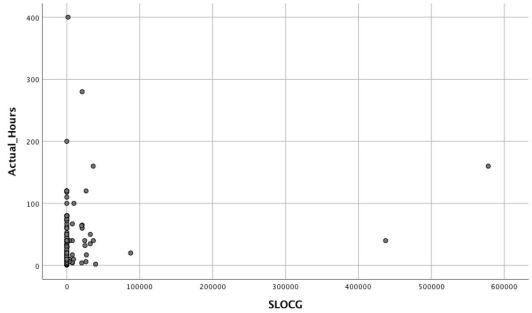
To compare changes in Lines of Code with CR rate we compare cumulative CR rate to cumulative SLOC in a collection of figures, in Figure 3.28. We analyze the cumulative number of Added SLOC, Modified SLOC, Auto-generated SLOC and Deleted SLOC separately and combined. Figure 3.28a shows the growth of CRs along with the growth of added SLOC. We can see that there is a large increase of Added SLOC towards the end of the release. The number of SLOC increases



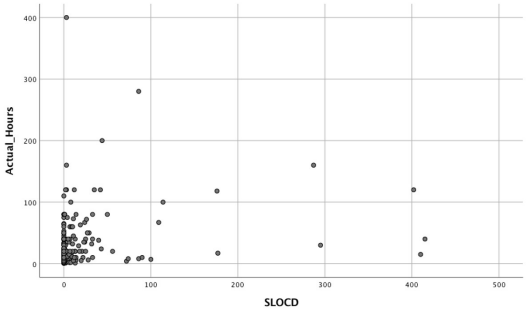
(a) SLOCA vs. Effort (Actual Hours)



(b) SLOCM vs. Effort (Actual Hours)



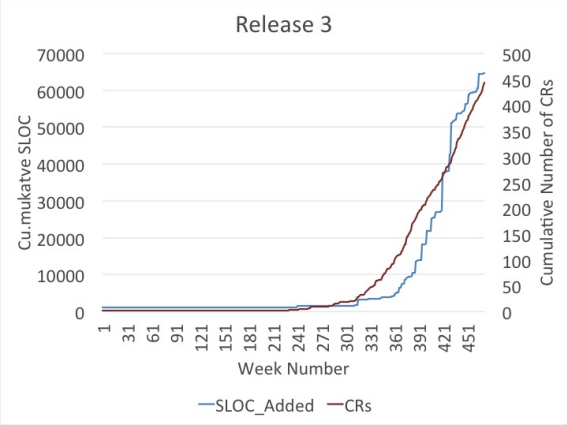
(c) SLOCG vs. Effort (Actual Hours)



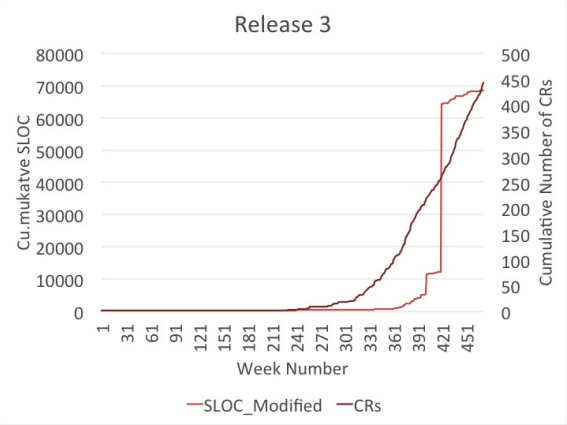
(d) SLOCD vs. Effort (Actual Hours4)

Figure (3.27) Scatter plot of SLOCs vs. Effort (Actual Hours) for Release 4

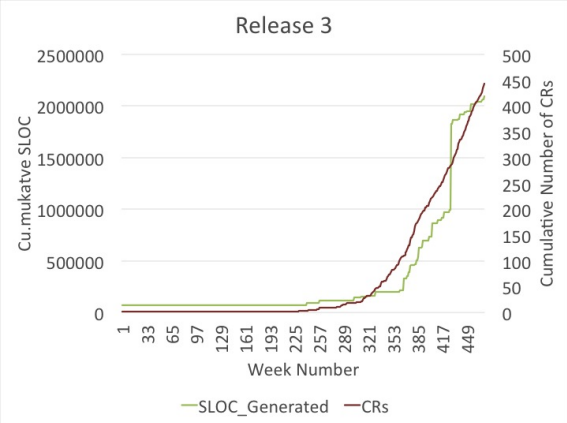
dramatically around week 400. There is another jump between week 420 and 430. Also, cumulative Modified SLOC jumps between week 400 and 420 (Figure 3.28b). Auto-generated SLOC increases at a faster pace around week 370 and continues to gradually increase until after week 420 where a large increase occurs (Figure 3.28c). Cumulative deleted SLOC also starts increasing at a faster pace around week 370 and continues to grow until the end of the release (Figure 3.28d). When looking at the total SLOC we find that in general SLOC modifications increased after week 370. When adding all SLOCs, the highest total amount of change was performed around week 430 (Figure 3.28e) and (Figure 3.28f). Note that we excluded the auto-generated code in Figure 3.31f, since its nature is different than the nature of the other SLOC. Since it is auto-generated, the time required to generate the code is different than the human effort in producing or modifying code and usually the



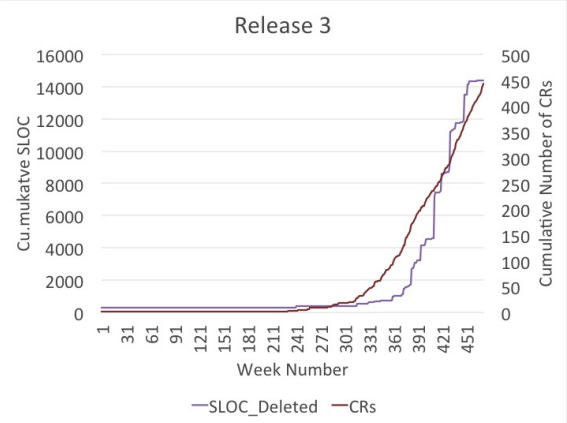
(a) Cumulative No. of CRs vs. Cumulative SLOC Added



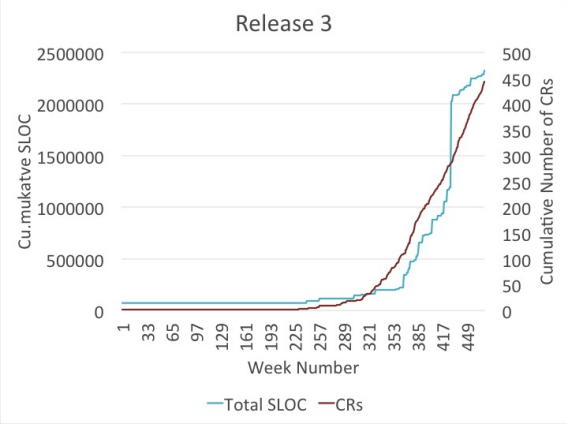
(b) Cumulative No. of CRs vs. Cumulative SLOC Modified



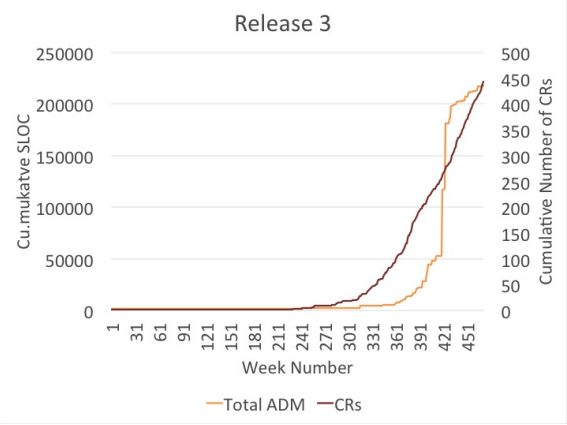
(c) Cumulative No. of CRs vs. Cumulative SLOC Autogenerated



(d) Cumulative No. of CRs vs. Cumulative SLOC Deleted



(e) Cumulative No. of CRs vs. Cumulative Total of SLOCs



(f) Cumulative No. of CRs vs. Cumulative total of SLOC Added, Modified and Deleted

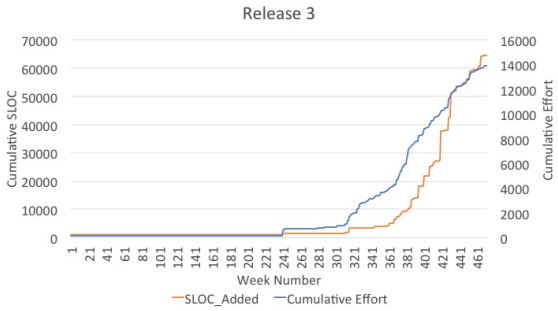
Figure (3.28) Cumulative Number of CRs vs. Cumulative SLOC for Release 3

amount of auto-generated code is much higher than the other values, therefore we wanted to examine changes of code including and excluding auto-generated code. This observation could indicate that there was a maintenance request that took place at around that time that might have caused an increase in CRs and accompanying code change.

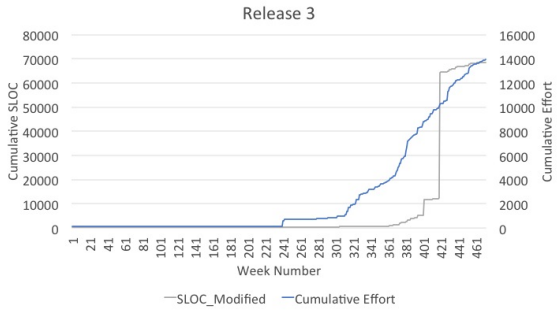
When comparing the cumulative SLOC values to the cumulative effort, we look for possible relationships between the number of SLOC and effort spent on CRs. Figure 3.29 demonstrates the growth of effort compared to the growth of SLOC. We find that there is a slight increase in effort between weeks 420 and 430 where SLOC Added, Modified and Autogenerated increase as well. This increase is also reflected in the total SLOC, both for including and excluding the auto-generated SLOC. Cumulative deleted SLOC also increases. This change in SLOC and the change in effort could assist in predicting effort. When looking into the cumulative effort curve we find other change-points, around week 370 and around week 300. In week 370, we noticed that the total SLOC rates started increasing faster than before, which indicates that more effort has to be expended towards these changes. Also, by week 300, we find that there is a slight increase in SLOC Added, which could be associated with the change in effort.

We then move to the fourth release. The boxplots for SLOC Added, Deleted, Modified and Auto-generated are shown in Figure 3.30. We find that the auto-generated LOC has the biggest outliers but most of the values are zeros. Figure 3.30 shows the maximum outlier as 54,616 for added SLOC, 103960 for modified SLOC, 577,984 SLOC and deleted SLOC is 415.

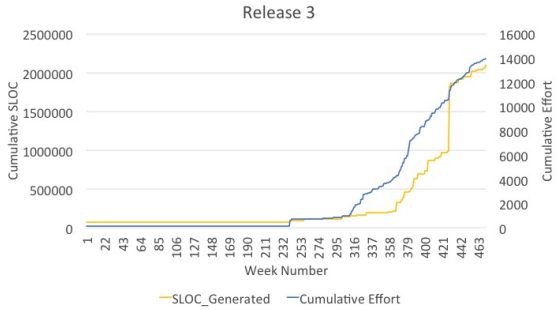
To compare changes in Lines of Code with CRs, we compare cumulative CR rate to cumulative SLOC in a collection of figures (Figure 3.31). We notice that a high number of SLOC is added in week 265 where there is a change in the CR rate



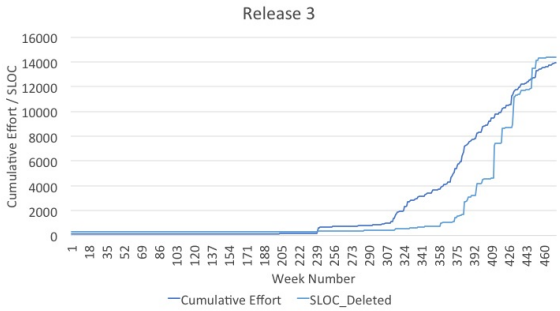
(a) Cumulative Effort vs. Cumulative SLOC Added



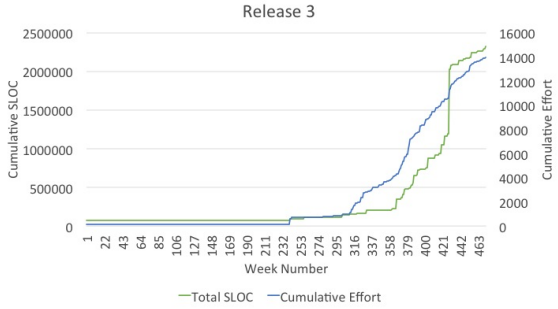
(b) Cumulative Effort vs. Cumulative SLOC Modified



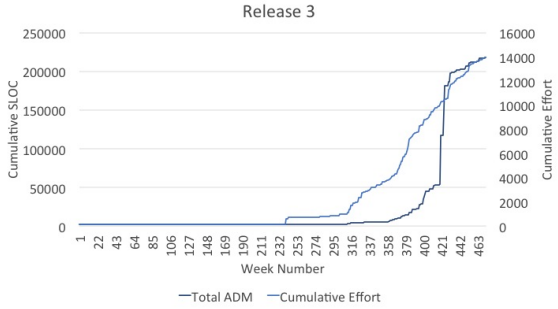
(c) Cumulative Effort vs. Cumulative SLOC Autogenerated



(d) Cumulative Effort vs. Cumulative SLOC Deleted

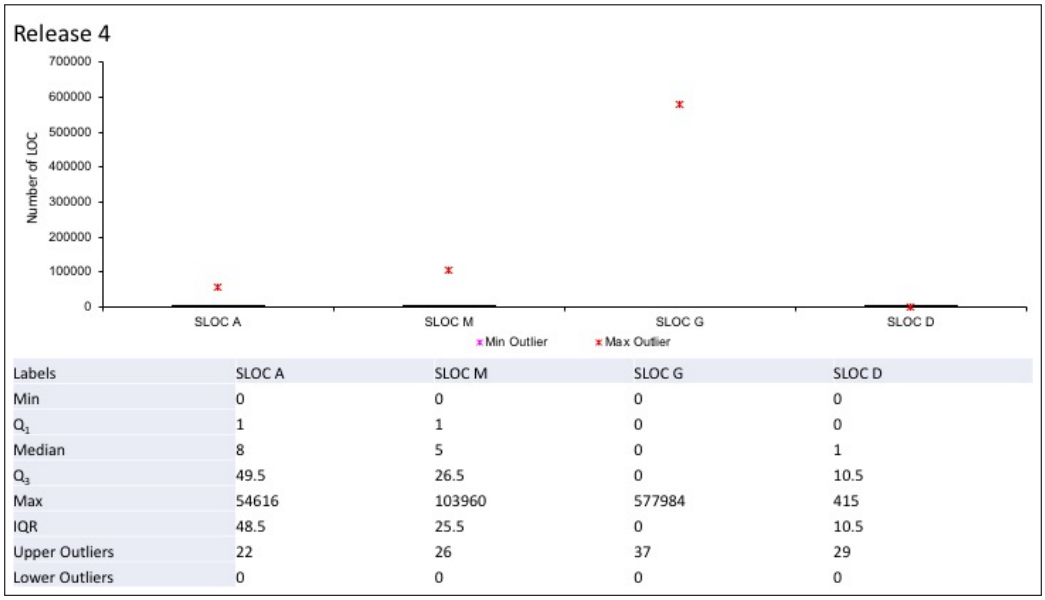


(e) Cumulative Effort vs. Cumulative Total SLOCs



(f) Cumulative Effort vs. Cumulative Total of SLOC Added, Modified and Deleted

Figure (3.29) Cumulative Effort vs. Cumulative SLOC for Release 3

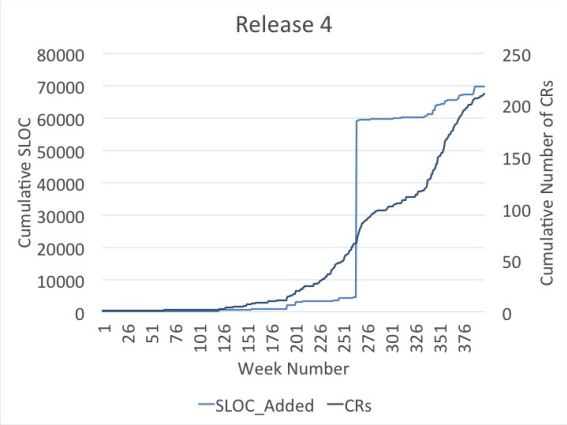


Labels	SLOC A	SLOC M	SLOC G	SLOC D
Min	0	0	0	0
Q_1	1	1	0	0
Median	8	5	0	1
Q_3	49.5	26.5	0	10.5
Max	54616	103960	577984	415
IQR	48.5	25.5	0	10.5
Upper Outliers	22	26	37	29
Lower Outliers	0	0	0	0

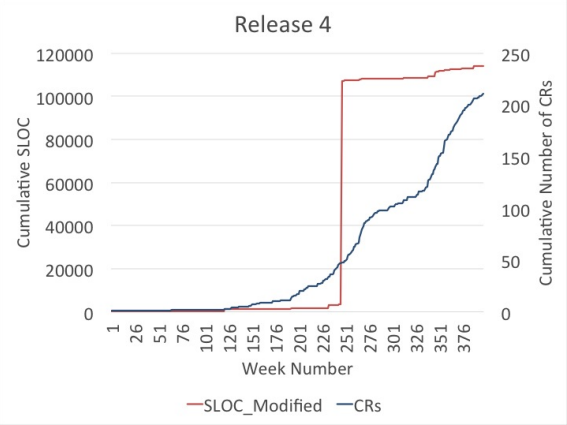
Figure (3.30) Boxplot of the number of SLOC Added, SLOC Modified, SLOC Deleted and SLOC Auto-generated (Release 4)

(Figure 3.31a). SLOC modified has a spike around week 245 just before a change point in the CR curve (Figure 3.31b). Auto-generated code has two major spikes, around weeks 245 and 350 (Figure 3.31c). For SLOC deleted we find that it grows along the cumulative CR curve with changes at times around the changes in CR rate such as the increase around weeks 190, 265 and 340 (Figure 3.31d). When looking at the totals of cumulative SLOC in Figures 3.31e and 3.31f we find that major code change occurred in weeks 245, 265, and 340.

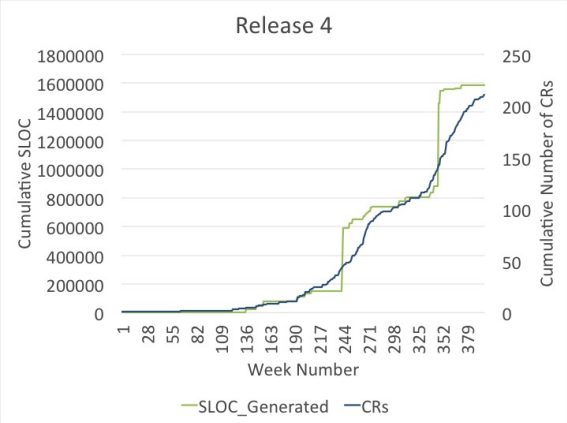
To compare cumulative effort to cumulative SLOC for the fourth release we consider Figure 3.32. When looking into the times where SLOC growth was major around weeks 245, 265 and 340, we find that effort has change-points in those weeks as well. In Figure 3.32a we find a change in week 265, the same week we see a high increase in SLOC added and SLOC Generated. We also see an increase in week 340 where a lot of auto-generated code was added (Figure 3.32c). The change in modified SLOC shows a slight change in effort in Figure 3.32f. For cumulative CRs,



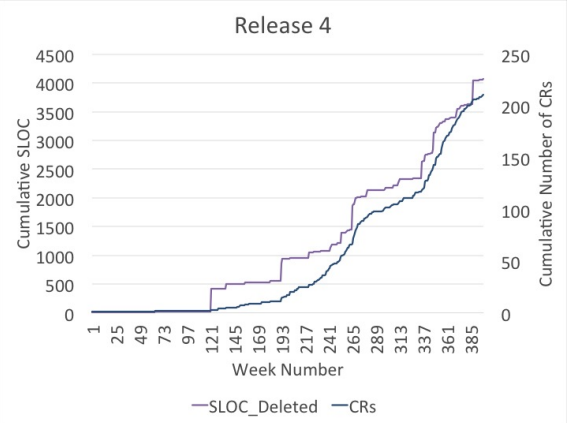
(a) Cumulative No. of CRs vs. Cumulative SLOC Added



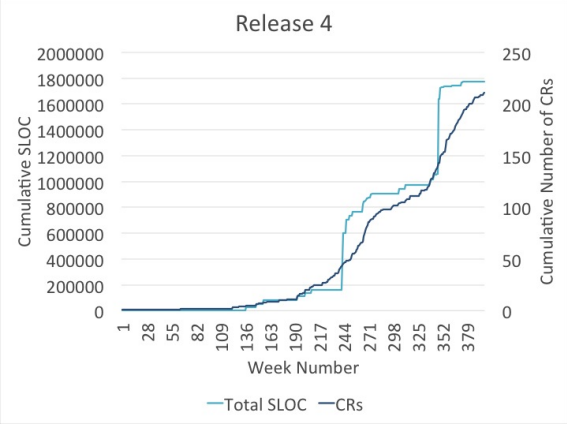
(b) Cumulative No. of CRs vs. Cumulative SLOC Modified



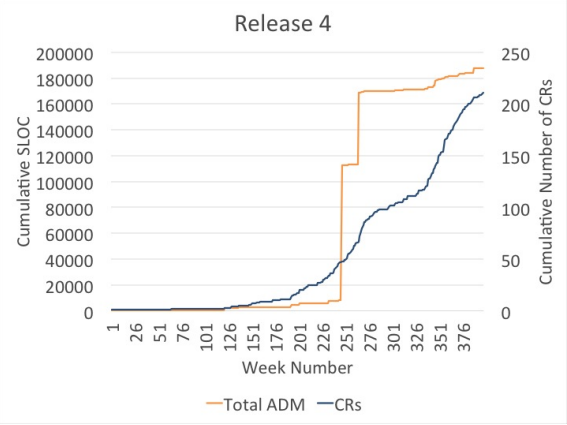
(c) Cumulative No. of CRs vs. Cumulative SLOC Autogenerated



(d) Cumulative No. of CRs vs. Cumulative SLOC Deleted

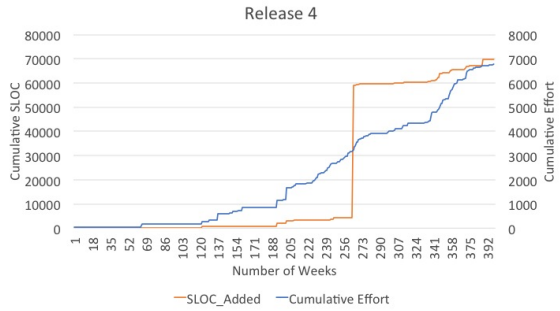


(e) Cumulative No. of CRs vs. Cumulative Total of SLOCs

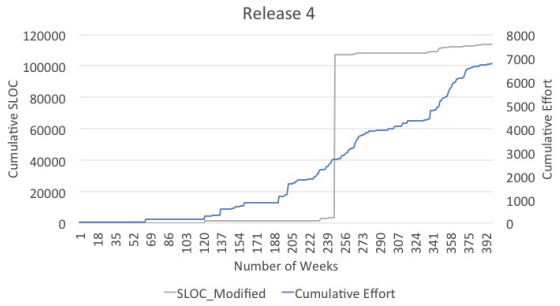


(f) Cumulative No. of CRs vs. Cumulative total of SLOC Added, Modified and Deleted

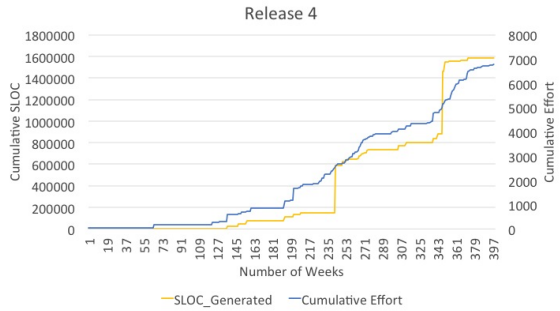
Figure (3.31) Cumulative Number of CRs vs. Cumulative SLOC for Release 4



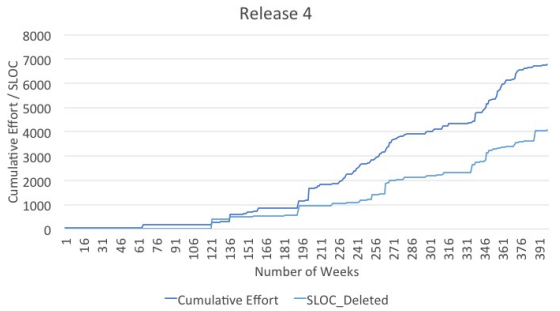
(a) Cumulative Effort vs. Cumulative SLOC Added



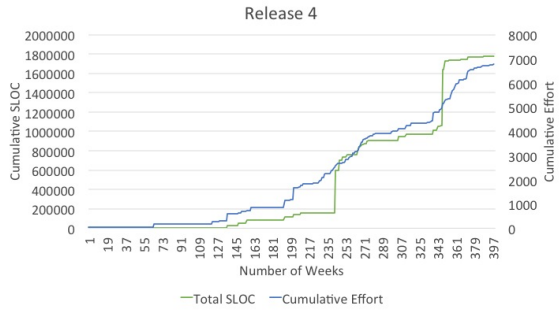
(b) Cumulative Effort vs. Cumulative SLOC Modified



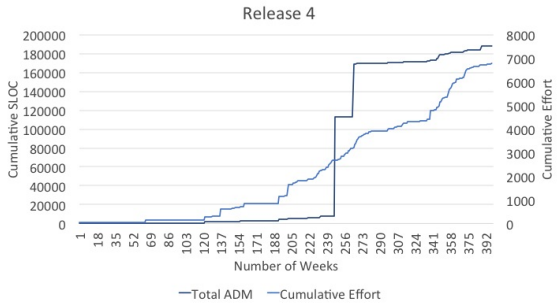
(c) Cumulative Effort vs. Cumulative SLOC Autogenerated



(d) Cumulative Effort vs. Cumulative SLOC Deleted



(e) Cumulative Effort vs. Cumulative Total SLOCs



(f) Cumulative Effort vs. Cumulative Total of SLOC Added, Modified and Deleted

Figure (3.32) Cumulative Effort vs. Cumulative SLOC for Release 4

we see similarities in the growth between cumulative effort and SLOC deleted (Figure 3.32d). We find a lot of similarities between cumulative SLOCs and cumulative effort and cumulative SLOC and cumulative CRs. From our observation we conclude that we can find changes in cumulative CRs and effort by knowing the amount of change in SLOC. SLOC might not be useful in predicting the exact number of CRs or their effort in the future, but it gives a clear indication that change-points can be identified using changes in SLOC. We will investigate this more in Chapter 4.

3.2.3.7 Summary of Data Analysis

At the beginning of this chapter we defined our system and presented relevant data. We also wanted to understand the data in the system so we set up two research questions:

- RQ1: What does the data in the system look like?
- RQ2: Are there any possible relationships among different attributes?

To answer the first question we took the attributes of CRs among four releases of the system and we visualized the ranges and values that these attributes could be in. We then tried to find different relationships among the attributes with having a main goal of finding attributes that assist us in effort estimation. We were looking for relationships between effort and other elements such as customer priority, functional area and change type. For customer priority, we found CRs of certain priorities require more effort than other CRs within a release, but this is not consistent among different releases, for example C1U in Release 3 requires the least amount of effort in comparing with other CRs in release 3, but in Release 4, CRs of type CIU require the most effort compared to other types of CRs in Release 4. We also have information about the amount of CRs of every priority in each release, for example a CR is more

likely to be of priority C2R in any of the releases since most CRs are of this type. In terms of predicting the number of CRs or effort using customer priority, there is no clear indication that predictions can be made using customer priority. These results are similar, when we investigated the use of Change Type. The effort related to different change types have different median values but they are not consistent over different releases. For example STR has the lowest median value in Release 1, while it has the highest median in Release 3. We also found that most CRs are anomalies, not DRs which makes the majority of them to be enhancement requests rather than fixes. By looking to functional areas in Release 4 we find that some of the functional areas with the highest effort have the lowest number of CRs. Many functional areas have very few CRs, (less than 3). Most of the functional areas have less than 20 CRs in total. Generally speaking, there is no indication that some functional areas require more effort than others, there is too much variation. We also explored if we can find any patterns or relationships between the number of CRs and the number of lines of code that were added, deleted, modified and auto-generated. We found that in places where cumulative CRs reflect major change, one or more of the SLOC data has an extremely high value, which indicates that big changes in code are reflected in the number of CRs. When SLOC and effort were compared there was no specific pattern to link the two. But when cumulative CRs where compared to cumulative effort these follow similar growth patterns. Figures 3.5, 3.6, 3.7 and 3.8 show that cumulative effort growth and changes along with cumulative CRs behave similarly. In fact, many of the points where cumulative CR rate changes, cumulative effort changes around the same time. This shows that it is possible to estimate and predict cumulative effort in the same way cumulative CR is estimated and predicted. For further analysis in an attempt to find correlations between our attributes, we perform a Principal Component Analysis next.

Att. ID	Description	Data Type	Range	Role
A1	Customer Priority	Ordinal	1=C2R, 2=C2U, 3=C2E, 4=C1R, 5=C1U, 6=C1E.	Input
A2	DR	Binary	0=False, 1=True	Input
A3	actual effort	Numerical	$0, 1, ...\infty$	Output
A4	SLOC Added	Numerical	$0, 1, ...\infty$	Output
A5	SLOC Modified	Numerical	$0, 1, ...\infty$	Output
A6	SLOC Generated	Numerical	$0, 1, ...\infty$	Output
A7	SLOC Deleted	Numerical	$0, 1, ...\infty$	Output
A8	Elapsed Time = (Completion Date - Submission Date)	Numerical	$0, 1, ...\infty$	Output

Table (3.6) Description of CR Attributes

3.2.4 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. It is a useful multivariate method commonly used to reduce the data to avoid multicollinearity. It can help in reducing the number of variables, and it can also provide support in identifying variables that vary together.

PCA was first invented by Pearson in 1901 [105]. It is mostly used as a tool in exploratory data analysis and for building predictive models. PCA groups correlated variables into a number of factors where each factor accounts for the maximum possible amount of variance for the variables being analyzed. The number of factors extracted may vary depending on the data set and the method chosen for extraction.

In this data set, we have 12 attributes available, but we only use eight attributes in the PCA analysis as shown in Table 3.6. We excluded Change Type/Enhancement, ID Number and Functional Area since they contain text values, not quantitative values. We replaced the Submit Date and Actual Completion Date with

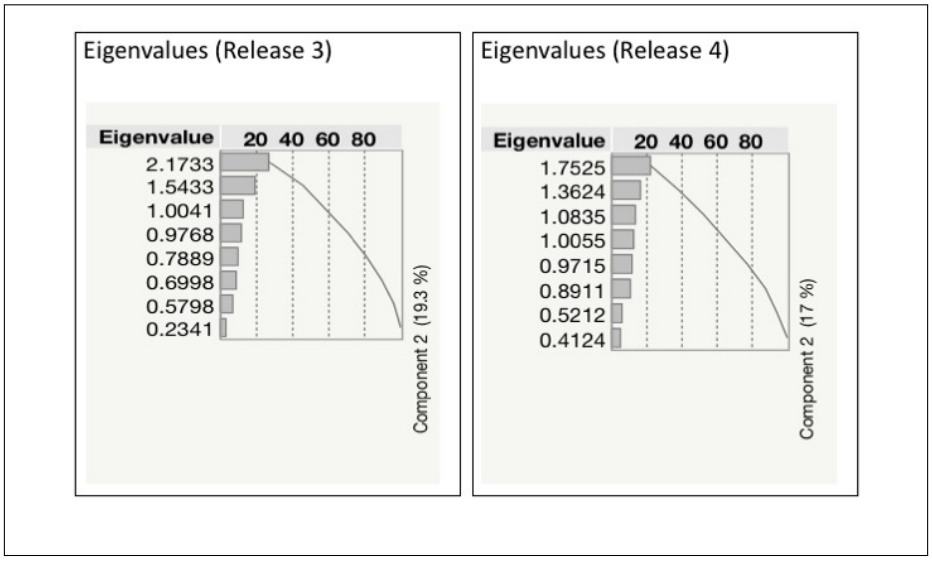


Figure (3.33) PCA eigenvalues for Release 3 and Release 4

Elapsed time. Therefore we end up with eight attributes. We investigate the ability of applying PCA to each release according to the available data.

- The first and second releases only have A1, A2, A3 and A9 available. Therefore PCA is not preformed since it has a small number of attributes and no reduction is useful.

- The third and the fourth releases have all attributes in Table 3.6. PCA was performed for these two releases.

A principal components analysis was conducted using JMP 14 software. Figure 3.33 shows that there are two principal components or factors with eigenvalues greater than 1. The cumulative percentage of variance explained by the two factors for Release 3 is 46.4% and for Release 4 is 38.9 %. According to this result, two components should be retained when performing PCA for the third and fourth release.

Table 3.7 shows the results from the analysis for Release 3. The highest loadings that are above 0.5 are written in boldface. The other loadings with values less than

Attribute	Description	Factor 1	Factor 2
A1	Customer Priority	-0.002716	-0.256239
A2	DR	-0.091123	0.724709
A3	Actual Effort	0.358475	0.554335
A4	SLOC Added	0.919374	0.100707
A5	SLOC Modified	0.756570	-0.044395
A6	SLOC Generate	0.184611	0.262517
A7	SLOC Deleted	0.711486	0.022626
A8	Elapsed Time	-0.067765	0.798463

Table (3.7) PCA for Release 3 with two factors

Attribute	Description	Factor 1	Factor 2
A1	Customer Priority	-0.016738	-0.299683
A2	DR	-0.077985	0.630785
A3	Actual Effort	0.269208	0.722308
A4	SLOC Added	0.761009	-0.202650
A5	SLOC Modified	0.020457	-0.107772
A6	SLOC Generate	0.405228	0.140050
A7	SLOC Deleted	0.881312	0.060560
A8	Elapsed Time	0.127752	0.643290

Table (3.8) PCA for Release 4 with two factors

0.5 are written in gray. Higher loading means higher influence on PCA. For Release 3, we find that A4, A5 and A7 are in one factor and A2, A3 and A8 are under another factor. This says that there is a correlation between SLOC Added, SLOC Deleted and SLOC Modified and another correlation between, DR, actual effort and Elapsed time. There is no indication of correlation between effort in actual effort and Lines of code.

Table 3.8 also shows correlations between A2, A3 and A8, and another correlation between SLOC Added and SLOC Deleted. The results of the analysis do not indicate any correlation between effort and lines of code. The only correlation we can find is that Discrepancy requires higher effort. From the number of actual effort we can predict if a CR is a discrepancy, but we cannot predict number of actual effort if a

CR is a discrepancy. This result does not help in predicting effort. The results of the PCA confirms our conclusion from our previous analysis that the data available is not useful for effort prediction, except for the relationship between cumulative CRs, SLOC types and cumulative effort.

3.3 Analysis Tools

For data analysis and visualization, we used `MS Excel` spreadsheets [92] and some of the available `MS Excel` spreadsheets templates to calculate and present boxplots by Vertex42 [138] that is specialized in creating professionally designed spreadsheet templates for business, personal, home, and educational use. We also used another `MS Excel` template to calculate Control Charts developed by the American Society for Quality [127].

In addition, other statistical and curve fitting tools were used at several points in our research:

- `IBM SPSS` Statistics package used to estimate parameters and curve-fit different models [62].
- `JMP14` is a statistical software tool used to perform PCA from `SAS` [113].
- `R packages` were used for detecting change-points such as the `cpts package` [108].
- `MyCurveFit` was used. It is an open source curve-fitting online tool [99].

.

Chapter 4

Estimating Change-Points for Change Requests

4.1 Problem Statement

In research, several approaches were proposed to deal with change-points and provide better failure prediction [148] [55] [151][41][64]. We attempt to use reliability growth models on data from a Change Request (CR) database. Unlike the work proposed by [148] [55] [151][41][64], we are attempting to predict future CRs based on CRs in a CR database. This database also contains information on code changes, such as lines of code added, deleted, modified or auto-generated. Ultimately, we would like to provide a project manager with an approach to estimate future CRs for various planning cycles (weekly, monthly, etc.). Since software changes during maintenance, be it corrective, adaptive or dealing with enhancements, any CR prediction model needs to deal with change-points.

Software Reliability Growth Models (SRGM) are used to predict growth in defect data. Changes due to major fixes or upgrades that cause changes in the failure distribution, are called change-points. The number and locations of change-points affect reliability modeling. We examine the different change-point estimation methods used in software reliability modeling and compare them to the underlying change

data that caused the change in CR distribution. The results show that data analysis of change in lines of code provides a credible indication of the existence of change-points.

More formally, let a sequence of failures, $\zeta_1, \zeta_2, ..., \zeta_n$ be where n is the number of cumulative failures. A change-point τ, exists if $\zeta_1, \zeta_2, ..., \zeta_\tau$ has a failure distribution of F, and $\zeta_\tau, \zeta_\tau + 1, ..., \zeta_n$ has a failure distribution G, where $F \neq G$ and the two sequences of failure data are statistically independent. In this context we investigate the following research questions:

- RQ1: What methods are used to estimate a change-point in a cumulative failure curve for reliability prediction?
- RQ2: Can we use them to predict change-points for CR data from a CR database?
- RQ3: How do these methods compare to change-points identified by code change in the CR database?

Estimating the locations and number of change-points is a key objective of our work. The results of this work would be used to predict future CRs from a CR database with change-points.

Section 4.2 provides a background on change-points and the change-point estimation methods used. Section 4.2.3 defines our proposed approach by using Lines of Code (LOC) in change-point estimation. Results of applying the existing approaches are in Section 4.3. Discussion is next, Section 4.4. Method validation among the remaining release is in Section 4.5. Validity threats in Section 4.6 followed by the conclusion in Section 4.7.

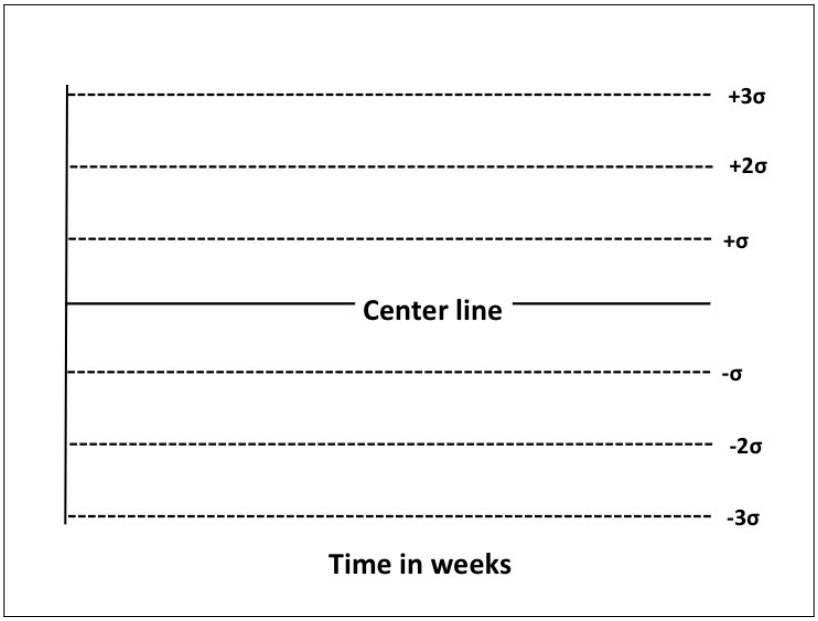


Figure (4.1) Control Charts

4.2 Change-point Estimation Methods

4.2.1 Control Charts

Control charts were first proposed by Dr. Shewhart in 1924 [117]. They are quality control tool that was used to monitor software processes. It has been widely used in many applications including finding change-points in failure data. To explain how control charts work, we first define the basic elements of a control chart. It contains a centerline which represents the average value of data points and upper and lower control lines. On each side of the centerline there are three control lines which are multiples of the standard deviation (σ). They reside respectively as $\pm\sigma$, $\pm 2\sigma$ and $\pm 3\sigma$ (Figure 4.1). Out-of-control data refers to data outside of the assigned control limits, which may indicate a potential change-point. When change is detected, the cause should then be investigated. For example a data point that is above the upper control line $+3\sigma$ is an out of control point. To specify out of control data points and

eventually consider it as a change-point, we follow the criteria presented by Zhao et al.[148]:

- Two out of three successive values are on the same side of the centerline and more than two standard deviations from centerline.
- Four out of five successive values are on the same side of the centerline and more than one standard deviation from centerline.
- At least eight successive values are on the same side of centerline.

4.2.2 Likelihood Ratio Test

A cumulative sum statistic is used to estimate a single change-point. The number and positions of change-points are detected according to the changes in the mean value. Having failure data over a specific time period $[0, n]$, the mean θ is constant. Using the log-likelihood ratio test, changes in the mean are detected and the location of the change is defined as a change-point.

Mathematically, let failure data points ζ_1, ζ_2, ... ζ_n. If a change-point τ exists then the mean of ζ_1, ζ_2, ... ζ_τ is different than the mean of $\zeta_\tau + 1$, $\zeta_\tau + 2$, ... ζ_n The log-likelihood test (LR) for single change-point estimation is

$$LR = max_\tau\{\ell(\zeta_{1:\tau}) + \ell(\zeta_{\tau+1:n}) - \ell(\zeta_{1:n})\} \tag{4.1}$$

Where ℓ represents the reliability model that fits the failure curve. The position of the change-point is estimated as:

$$\tau = argmax\{\ell(\zeta_{1:\tau}) + \ell(\zeta_{\tau+1:n}) - \ell(\zeta_{1:n})\} \tag{4.2}$$

argmax is an abbreviation of arguments of the maxima. This attains the function's largest value which could be an empty set in case of no change-points or one change-point's position or more than one position for several change-points. If we have m number of change-points $\tau = (\tau_0, \tau_1, ..., \tau_{m-1})$ where $\tau_0 = 1$ and $\tau_{m-1} = n$ the likelihood ratio will be:

$$min_{m,\tau}\{\sum_{i=1}^{m}[-\ell(\zeta_{(}\tau_{i-1:\tau_i}))] + \lambda m\} \tag{4.3}$$

Where λ is a constant.

Time is split into successive intervals and the likelihood ratio is computed for each interval. The interval with the highest change in predicted detection of parameters has a change-point. After the first change-point is defined, binary segmentation is used for each time segment, before the change-point and after the change-point to detect multiple change-points until a minimal threshold is reached.

4.2.3 Proposed Approach

Change-point estimation methods that were highlighted so far use their observation of changes in the cumulative CR curve. Instead of relying on the number of CRs we look further into the underlying reason behind the change. By analyzing the number of altered Lines of Code (LOC) in resolving a CR in the system we can predict upcoming change. We start by looking into the outliers in a box plot chart of the lines of code added, deleted, auto-generated and modified as candidate change-points. We then define a threshold for acceptable change in LOC. By analyzing data and measuring the amount of LOC added, deleted, modified or auto-generated in resolving CRs we can then decide if a major change has occurred in the system which indicates a potential change in CR density.

This would then indicate candidate points that are considered potential change-points.

$$LOC_i \geqslant Threshold \tag{4.4}$$

Where i reflects the time when LOC changes where made for the period of time $[0, n]$. $i \in \{1, 2, ..., n\}$. This method requires data analysis rather than statistics.

4.3 Results: Release 4

We apply the change-point estimation methods to actual CR data for the fourth release. There are 211 CRs in 398 weeks.

4.3.1 Applying Control Charts

Using CR data for Release 4, results in the control chart are shown in Fig.4.2. We then follow the criteria for change-point estimation mentioned in Section 4.2.1 for the cumulative CRs. The number of estimated change-points is high, looking at the later weeks of the release alone, we observe 22 estimated change-points and change-ranges. The set of changes-points and change ranges by week are {[204-206], 208, [214, 220], [222,226], 245, 266, 268, [270, 271], [290, 295], 303, [311, 312], [318, 326], 328, 332, [334, 335], 339, 343, 345, [349, 350], [356, 357], [370, 377], 393}.

4.3.2 Applying the Likelihood Ratio Test

Using the R `changepoint` package we estimated multiple change-points. The package performed the calculations and then highlighted the estimates for weeks 201, 239, 265 and 341. (Fig. 4.6) The vertical lines in the cumulative CR curve represent the change-point estimates. The results show four estimated change-points. We can

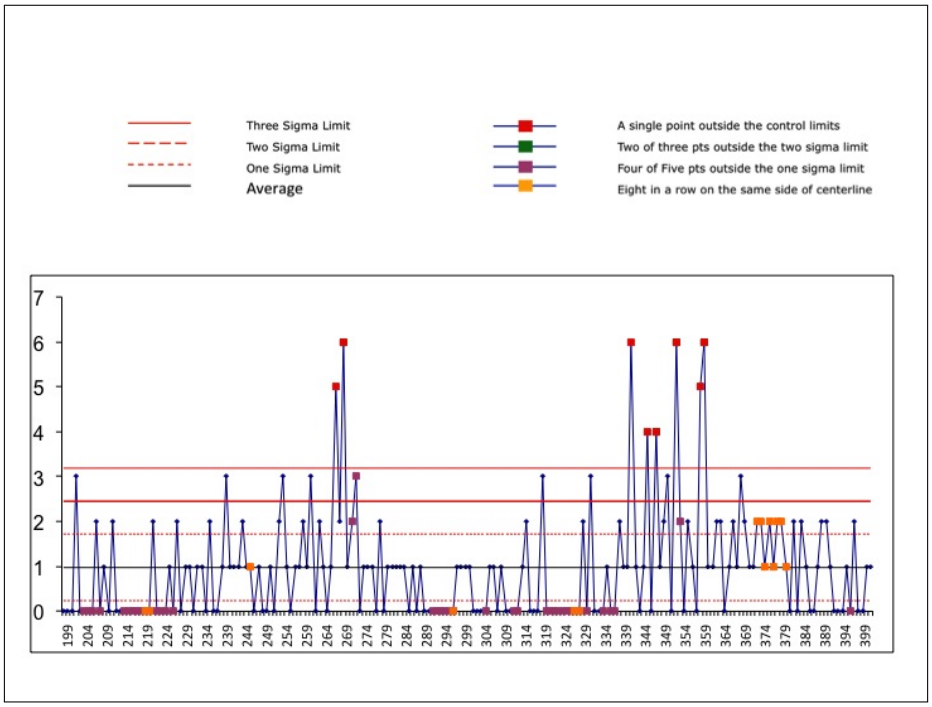


Figure (4.2) Change-point estimation using Control Charts after 200 weeks

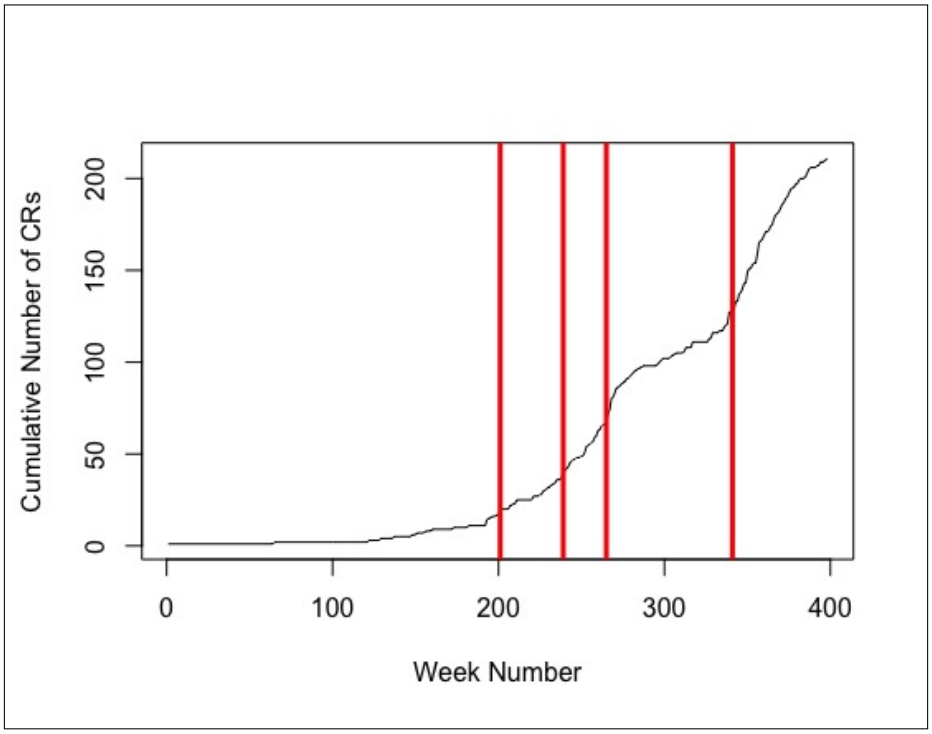


Figure (4.3) Estimated change-points using segmentation comparison (Release 4)

consider these points as times where major changes occur since the CR distribution and increase rate changes after the existence of these change-points.

4.3.3 Estimating Change-points using LOC

Investigating the CR data of the fourth release can also be used to identify candidate change-points. We observed extreme changes in Lines of Code according to the outliers of a box plot representing the LOC. We define our thresholds for added and modified LOC as 10,000 LOC. Auto-generated LOC has a higher threshold since the auto-generated code usually generates thousands and tens of thousands of LOC. The threshold of auto-generated code is 100,000 LOC. The deleted LOC have no extreme outliers. The maximum number of deleted LOC is 415. Using those threshold values we identify 4 change-points as follows:

- In week 265 SLOC Added = 54616. In the same week SLOC Deleted also was high when it reached its maximum value = 415.
- In week 247 SLOC Modified = 103960.
- In week 243 SLOC Generated = 437244.
- In week 348 SLOC Generated = 577984.

4.4 Discussion

Our results show that using control charts detected a high number of change-points in our release, more than 20 change-points in 200 weeks. This could be useful for some applications but not for reliability modeling. Having too many change-points that are adjacent cannot provide us with enough data to fit a model in a

time-series. As a rule of thumb one should have at least 50 data points before starting to model.[26] According to our objectives of finding change-points, control charts over-estimate the number of change-points found. Zhao et al.[148] had suggested performing "progressive adjustments", i.e. after fixing each estimated change-point, control charts are used to estimate remaining change-points. The adjustment could reduce the number of remaining change-points as we progress but it requires a large amount of processing, up to $O(n^2)$.

On the other hand using the likelihood ratio estimation method provides fewer estimations where change-points occur. These results could be used in reliability modeling. The disadvantage of this method is that it requires rigorous computation to find the changes in means. Additional effort could be required to curve-fit models prior to change-point estimation which increases the overhead of computation. With binary segmentation the time could be reduced to $O(nlogn)$

Our proposed method using change in LOC gives an estimate of four change-points as well. This method requires minimal computations prior change-point identification and highlights change-points that are reflective of the actual change in the system. This method requires some data analysis and definition of thresholds which determines when LOC is considered high. It is a beneficial approach in change-point estimation for decision makers, since they could expect beforehand when change-points are going to occur. This could take $O(n)$ of processing time to search for values that exceed the threshold.

We find that the likelihood estimation method and estimating change-points using LOC method both provided results that are very close in number and in locations. They both provided us with four change-points and three of the four locations where close in time, i.e. within seven weeks. Since the change-point values are "estimates", we cannot accurately verify if a certain change-point is the

correct location, but by simply looking into the results in terms of comparing change-points estimated against the growth of cumulative CRs, we find that the likelihood method estimated change where there are actual changes in the cumulative CR curve (Fig.4.3). This is also true when we use LOC changes. In fact we can use both methods to cross-validate each another, since one detects changes using CR data and the other estimates changes from potential causes of change. This case study was published by Alhazzaa and Andrews [10].

4.5 Validation

To validate the change-point estimation method we extend our use of it to the remaining releases of the software system. Unfortunately, we only have one more release where LOC data is available to apply the LOC approach to, which is Release 3. This method is not applicable to the first two releases, since they do not report LOC changes. We identify change-points for Release 1 and Release 2 using the Likelihood Ratio Test only.

4.5.1 Release 3

For the third release, we can estimate change-points by investigating the high spikes of SLOC Added, Modified, Deleted and Auto-generated, (Figure 3.28). We apply the same measures applied to the fourth release in defining change-points and we find the following:

- In week 420 SLOC Added = 10019. In the same week SLOC Modified = 52103.
- In week 429 SLOC Generated = 829171

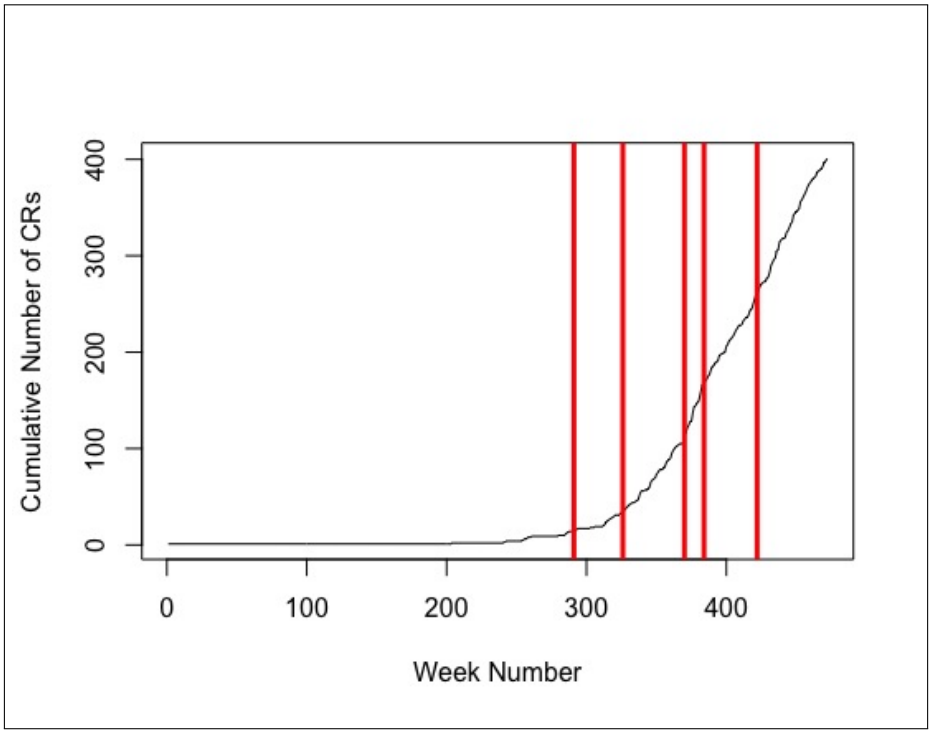


Figure (4.4) Estimated change-points using segmentation comparison (Release 3)

- In week 367 SLOC Generated = 115737.
- In week 387 SLOC Generated = 89751.

According to SLOC Added, SLOC Modified and SLOC Generated we understand that major changes in the code have occurred in weeks 420 and 429. Since the weeks are less than 10 weeks apart, we consider 420 to be the change-point, since it was the time where the first major change occurred. The second greatest outlier value for SLOC Generated is in week 367, which could indicate that it is a point where major change occurred. We apply the likelihood ratio method to the third release to see if we get similar results. Figure 4.4 shows the cumulative number of CRs for the third release on a weekly basis, with red vertical lines that show the estimated change-points. According to this method we find that 5 change-points were estimated in weeks 291, 326, 370, 384 and 422. We find that week 422 is close

to the 420 selected as a change-point using SLOC. Week 370 is also close to week 367 where major SLOC Generated code has occurred. When investigating the three remaining points we find that week 384 is a candidate change-point and around that week SLOC Generated has a value of 89751 which is the third highest amount for generated code in this release. This means that in those weeks major change in the CR rate occurred. We did not include that point earlier since it was under the threshold value we've chosen earlier but it still is a good candidate for a change-point, therefore we choose the earlier point (week 384). The two change-points in weeks 291 and 326 were not estimated according to the LOC method. When we analyzed the LOC data for those weeks we found that the values of LOC were all zeros, which means that they were imputed because the data was missing for these weeks or the values registered by developers were all zeros. As discussed in Section 3.2.2.1 this release has more missing SLOC data than Release 4. It also has lots of inaccurate data especially in the earlier weeks, since many CRs reported non-zero effort value while the SLOC added, deleted modified and auto-generated for that CR were all zeros. This was a pattern found in the earlier weeks of Release 3, and that pattern makes us question the quality of these data. Therefore, since these change-points were not detected or estimated due to lack of SLOC values we use the results of the segmentation method to assign two additional change-points for weeks 291 and 326 in addition to the change-points we found previously. So our final set of change-points is [291, 326, 370, 420].

4.5.2 Release 2

Release 2 is different than the two other releases discussed earlier. It lacks SLOC data since it was mostly missing so it was discarded. Therefore we can estimate

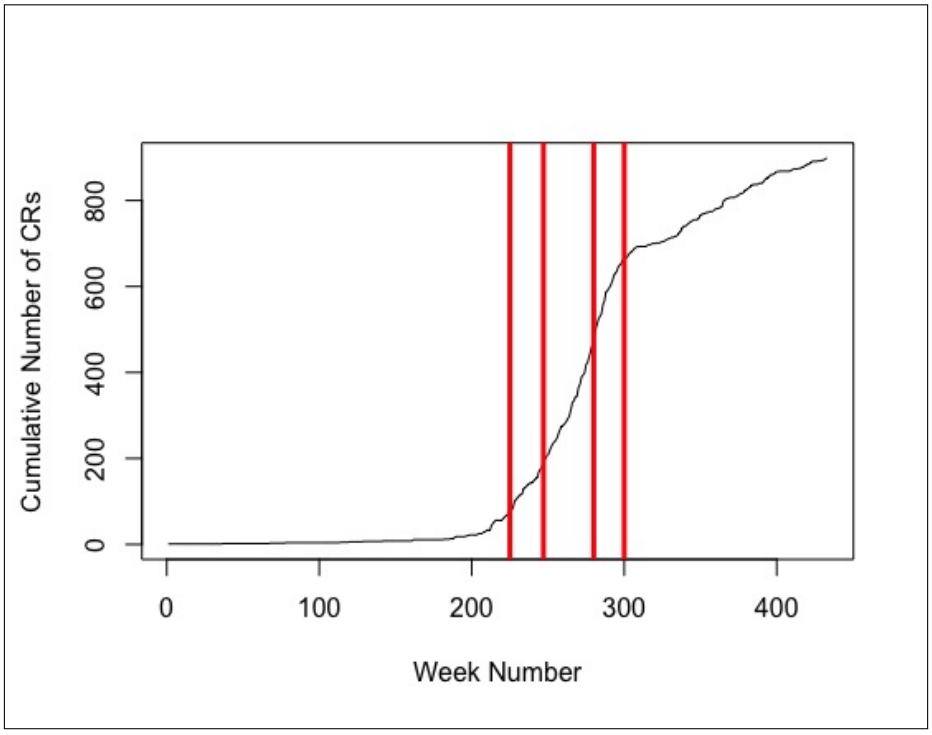


Figure (4.5) Estimated change-points using segmentation comparison (Release 2)

change-points using the likelihood ratio test and the results are shown in Figure 4.5. The estimated change-points are: 225, 247, 280, and 300.

4.5.3 Release 1

The first release is the oldest release in the system. Back then SLOC data was not reported. We used the likelihood ratio test to find change-points. Figure 4.6 shows four change-points: 129, 149, 176, and 193. These change-points are not the actual values, in fact they are shifted. In this release the first CR was recorded in 1999 and the second was recorded in 2005. These 266 weeks made a huge gap in the growth of the CRs, which made the highlighted change-points not clear in the graph. Therefore we kept 5 weeks at the beginning of the release unchanged, and we removed 261 weeks. When estimating change-points according to the modified

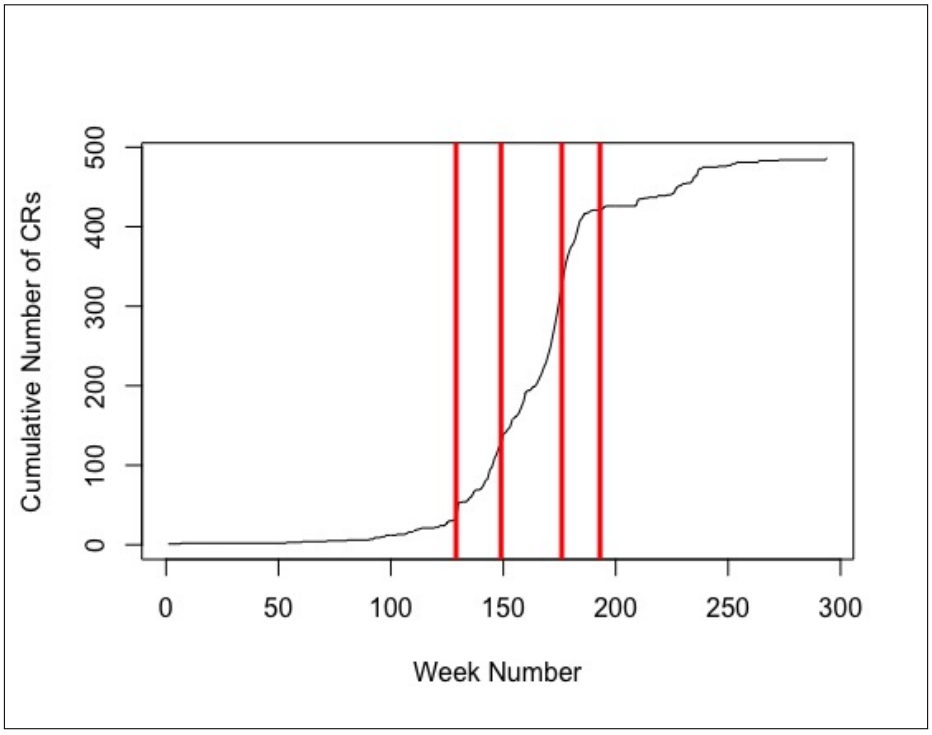


Figure (4.6) Estimated change-points using segmentation comparison (Release 1)

chart it gives the following weeks: 129, 149, 176, and 193. When shifting the weeks back to their original positions, these change-points are going to be as follows: 390, 410, 437 and 454 [1].

4.6 Validity Threats

We address threats to validity in our work according to the criteria described by Wohlin et al.[139]. External validity is concerned with generalization of the findings. While we applied our case study to an actual evolving large software system, other evolving software may not show the same cumulative CR behavior. This depends partly on the types of changes such as code change patterns, stability, frequency,

[1]R gave a warning message that the accuracy of estimating these change-points might be compromised if we used the original data.

size, and inherent quality. Therefore we do not claim that our results are generalizable. Construct validity according to Wohlin et al. [139] refers to the relation between theory and observation. Construct validity is concerned with the extent of operational measures that reflect what the researcher had in mind such as the nature and the quality of the CR database including reporting of change duplicates, accuracy of reporting, etc. In situations where safety-certification requires re-auto-generation of all auto-generated code, amount of change effecting CR behavior may be artificially inflated. However, since we re-estimate CR rate for reliability prediction purposes, we correct any threats related to that. Obviously, the CR data for the older releases had more issues. But there are also some questions of missing data and incorrectly reported data (see Chapter 3).

4.7 Conclusion

In our work, we examine methods proposed in previous research for change-point estimation for the purpose of CR prediction using reliability models. These methods rely on cumulative CR data to predict change. We suggested using changes in LOC to estimate change-points. We found that we could use this method to provide a reliable prediction of change-points in a software release, while it requires less computation than the other two methods. The likelihood ratio method provides results that are close to the results of our proposed approach in terms of the number of change-points and their locations, but it requires more computation than our approach. The control charts method on the other hand over-estimates the number of change-points. Having a large number of change-points is not useful when using reliability models, since models will keep changing according to the change-points and this will affect model fitting and the ability of the fitted model to predict CRs accu-

rately. In addition, it requires more computation to perform estimations with fewer change-points. Using information about changes in Lines of Code provides a good and practically intuitive indication of change-point locations. Other characteristics of CRs such as CR priority or type of incident, may impact failure rate or density and they could be analyzed as well. The results of our estimated change-points is used in the following chapters for defect prediction when change-points exist in CR data.

Chapter 5

Change Request Prediction: A Comparison

5.1 Problem Statement

Software Reliability Growth Models (SRGM) are used for failure prediction. Curve-fitting approaches have been successfully used to select a fitted reliability model among candidate models for defect prediction. Limited research has been done in CR prediction using curve-fitting methods on evolving software systems, with one or more change-points. Change-points are changes due to major fixes or upgrades that cause a change in the failure distribution. Previous approaches mostly focus on sample-fitting and short-term predictions. We focus on providing a curve-fit solution that deals with change-points but yet considers long-term prediction of data for a software release. We use a heterogeneous method that selects models before and after change-points and then performs Time Transformation (TT) to account for change. We then compare our solution to existing curve-fitting solutions in terms of their predictive ability. Our data show that the TT approach provides better CR predictions than other existing curve-fitting approaches.

Many studies and empirical studies were performed using SRGM analytical methods to estimate failures in software systems. Using analytical methods is based

on a number of assumptions about operational profile, defect fixes, perfect or imperfect debugging, etc. Many of these assumptions are violated when legacy systems undergo frequent evolution. Our goal is to benefit from the use of SRGM in CR prediction rather than failure prediction, without using any prior assumptions. The number of failures does not necessarily represent the number of CRs. One CR may include one or more failures. Some "CR" databases also include change information due to enhancements, and these really are more like "change" databases.

Stringfellow and Andrews [128] were successful in using a selection method for SRGMs on defect data from a defect database during system testing to make release decisions. But when applying SRGM to predict CRs in legacy systems they fail to provide long-term CR prediction, especially as the CR rate changes due to evolution. Their approach is one of the few studies which uses curve-fitting to select reliability models. Curve-fit methods perform a regression and evaluate the applicability of a set of candidate models with few or no assumptions about operational profile, CR fixes, etc.

We apply a multi-stage curve-fitting approach using Time Transformation (TT), first introduced by Musa [98]. By splitting a data set into several stages based on change-points and then curve-fitting each stage, we use TT to account for changes as if they had occurred at the beginning of the release. TT adjusts parameters of the model chosen after change behavior as if a change was accounted for since the beginning of the release.

We compare the curve-fitting methods using a Goodness-of-fit evaluation of the model and its predictive ability. Although a number of solutions have been proposed for reliability modeling with change-points, the predictive ability assessment is mostly limited to short-term predictions (i.e. a next step prediction). This is not practical in regard to practices in industry. Project managers hope for these models

Domain	Analytical	Curve-fit
No Change-points	Many solutions starting with model inventors [96] [142][141] [48]	Stringfellow and Andrews [128]
Change-points	[129] [29] [151] [147] [49] [18] [53] [50] [32] [31] [101] [100] [13] [75] [76] [91] [59] [57] [60] [86] [89] [54] [87] [88] [58] [83] [56] [55] [145] [65] [64] [69] [66] [68] [67] [77] [81] [43] [4] [103] [42] [94] [150] [80] [44] [124] [123] [122] [121] [131] [78] [79] [2] [125] [126] [93]	[137] [38] [30][19]
Change-points using TT	Musa [98]	Our work

Table (5.1) Research areas and gaps

to assist them in predicting CRs further into the future to assist them in resource planning. Therefore, we provide longer-term predictions in comparing the models we use in the case study.

- RQ1: Can we predict CRs in an evolving legacy system using curve-fitting approaches?
- RQ2: What curve-fitting approaches can we use in CR predictions during evolution and change in legacy systems?
- RQ3: How do these approaches compare?

We use SRGM methods that are used for failure prediction to predict future CRs. We use real CR data to compare the performance of different curve-fit methods in CR prediction. We incorporate the idea of TT into the curve-fitting approach to provide more accurate long-term CR predictions. Table 5.1 highlights the contribution of our work compared to other contributions in the field of software reliability and CR prediction. This work has been published in [9].

The remainder of the Chapter is organized as follows: Section 5.2 describes existing failure and CR modeling and prediction methods in the presence of change-points. Section 5.3 defines our proposed approach. We then compare predictive abilities of the approaches in Section 5.4 , followed by the threats to validity in Section 5.5.

5.2 CR Modeling and Prediction

We can divide modeling approaches into analytical approaches and curve-fit approaches. An analytical approach derives a solution analytically by providing assumptions regarding failures, failure repair and software use and then developing a model based on these assumptions. A curve-fit approach selects a model based on the best curve-fit with few or no assumptions. This approach relies entirely on empirical curve-fitting using one or more types of functions. Both approaches are seen in the literature for software reliability. One of the major contributions for curve-fitting methods is by Stringfellow and Andrews [128]. They performed a curve-fitting approach on defect data. This method was not concerned with evolving systems though and no change-point considerations were considered. Chi et al.[30] proposed a multi-stage model that segregates release times based on change-points. Whenever a change in failure rate is detected, a new modeling phase is applied as if the data after the change-point was isolated from the old data. The most recent model selected is the one used for prediction. This does not work in the case of frequent change-points, as not enough data is available to determine model parameters.

In the mapping study (Section 2.3) literature shows many studies are concerned with finding solutions in terms of goodness-of-fit. The predictive ability for the

proposed solutions are measured for short-term predictions, i.e. looking into one or two time units into the future. Rana et al. [109] and Park et al [104] highlighted the issue of limited long-term prediction in research. Andrews et al. [16] used a month-by-month interval to evaluate prediction capabilities for future incident prediction for a help desk rather than defect prediction for software. Since long-term prediction is a major concern in this work, we will try to adopt this method in our future forecasts.

5.3 Proposed Approach

We are looking into three curve-fitting approaches to predict defects when change-points exist. Our purpose is to find the approach that provides the most accurate predictions with the least amount of under-predicted values. Under-prediction is risky for management, when more CRs occur than they predicted, as they may have failed to plan for adequate resources. We describe three curve-fit approaches for SRGM estimation. We will then use these three approaches in our case study for CR prediction and compare their predictive ability.

5.3.1 Approach 1: Curve-fitting approach

This approach uses a cumulative number of CRs over a time period to find a fitted model among several SRGM candidates. When a model is selected it is then used to predict CRs for the remainder of the release. This process was first proposed by Stringfellow and Andrews [128]. Using the SRGM in Table 2.1, we select a model that best fits the CR data. When a model are estimated, it is evaluated as follows:

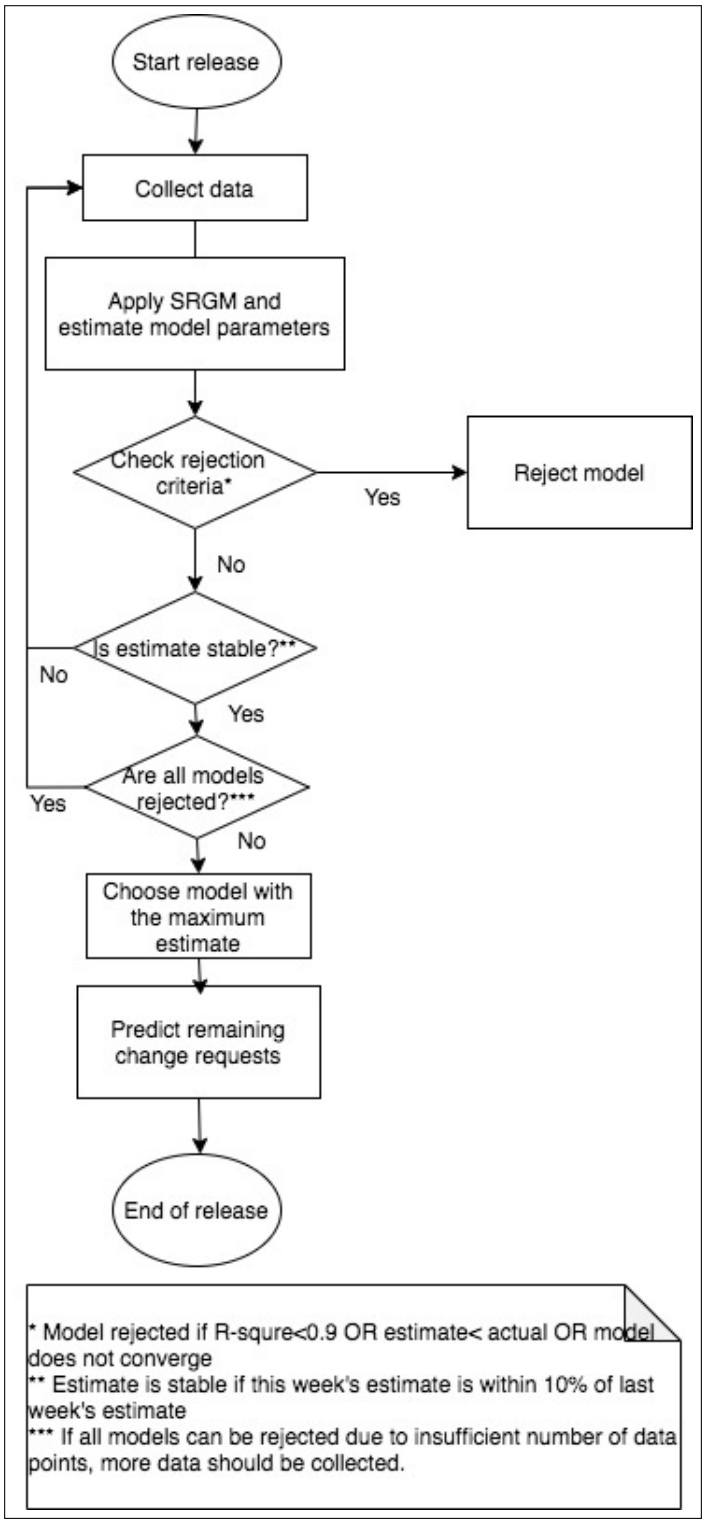


Figure (5.1) Model selection and CR estimation using Approach 1 [128]

- Goodness-Of-Fit (GOF) using R^2. The threshold according to Stringfellow and Andrews [128] for R^2 is 0.90 or above. Only models that meet the threshold will be considered in the prediction stage.

- Prediction Stability by checking the stability of the prediction for a certain week compared to a previous week. The estimated value for a week should be within 10% of the estimated value for the previous week. This is percentage is subjectively chosen as a rule of thumb by Wood [140].

- Prediction ability by checking the relative error[1] in prediction. Error is calculated as follows:

$$Error = (Estimated - Actual) \tag{5.1}$$

  while

$$RelativeError = (Error/Actual) \tag{5.2}$$

To apply the curve-fitting method to our system we use the process shown in Figure 5.1. After collecting cumulative CRs in each week t, the curve-fit program estimates model parameters by attempting to fit the model to the data. If a fit cannot be performed, due to the model's not being appropriate for the data or due to insufficient data, the model is rejected. A sufficient number of data points is determined subjectively. Most curve-fitting tools require at least five data-points for the tool to start estimating model parameters and fitting them to the existing data.

If a model's predictions for expected number of total CRs are lower than the actual number of CRs already found and have been consistently so in prior weeks,

[1]relative error value is calculated using the absolute value of an error over the actual value. In our case we need to keep track of negative values (under-predictions). Therefore we calculated the relative error with the real error value instead.

the model chosen is inappropriate for the data and should not be used in future weeks. If used, it would underestimate the number of remaining CRs and give a false sense of security. If there is at least one stable model, then the model with the highest R^2 value is chosen for CR prediction.

This approach does not take into consideration the existence of change-points, which can affect the quality of the predictions. Changes in a software system can change the rate of CRs occurring which affects estimation of future CRs.

5.3.2 Approach 2: Multi-stage approach

The multi-stage approach was applied by Chi et al. [30] to defect data. Although the effectiveness of the predictions has not been discussed thoroughly in their work, we find the solution to be interesting to apply to our CR data in order to avoid poor predictions when change-points occur.

For the multi-stage approach we use the same curve-fitting approach in Section 5.3.1 after each change-point. i.e if a model is selected to perform predictions and a change-point occurs, we are required to fit a new model. After each change-point, we use the curve fitting approach in Figure 5.1 to estimate a new model as if the data after change was in a separate release. This method assumes that a dataset is divided into stages. Each stage has its own fitted model for CR prediction.

Let S be a dataset of the cumulative number of CRs in a release over time. This dataset has a number of change-points n. Change-points divide the dataset into $n + 1$ stages, where each stage is referred to as s_i, and $1 \leq i \leq n + 1$. A change-point exists at time T_i, where the total number of CRs for the i^{th} stage is D_i. For each stage s_i, a reliability model is selected $\mu_i(t)$. When a change-point is found at time T_i , a new model is estimated for the next stage. Once model $\mu_{i+1}(t)$

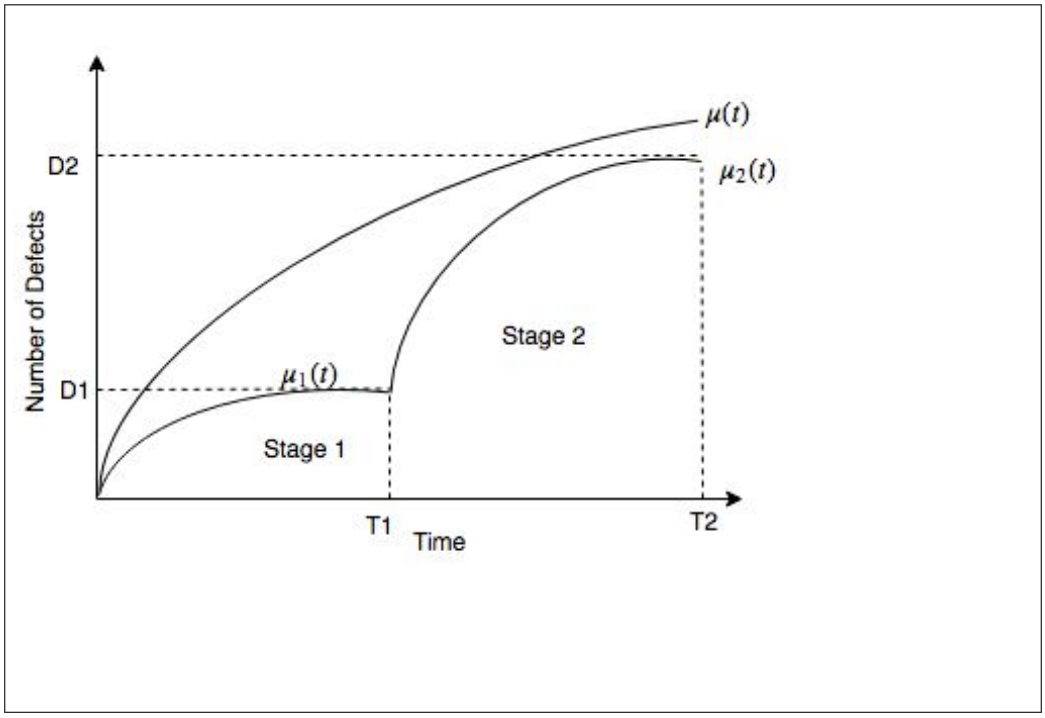


Figure (5.2) Multi-stage model transformation.

is selected then it will be then used starting at s_{i+1} for CR prediction. The process repeats for each stage until the end of the release (see Figure 5.3).

This method overcomes the issues of selecting a single model in the curve-fitting method in Section 5.3.1 [128]. A disadvantage of this method that it does not consider each stage as a part of a whole release. This might affect the accuracy of the CR predictions. When stages are short, there may not be enough data to select a model and determine parameters according to the selection criteria in Figure 5.1.

5.3.3 Approach 3: Multi-stage approach with Time Transformation

To overcome the issue with the multi-stage approach presented in Section 5.3.2, we apply a *Time Transformation (TT)* technique. The idea of time transformation

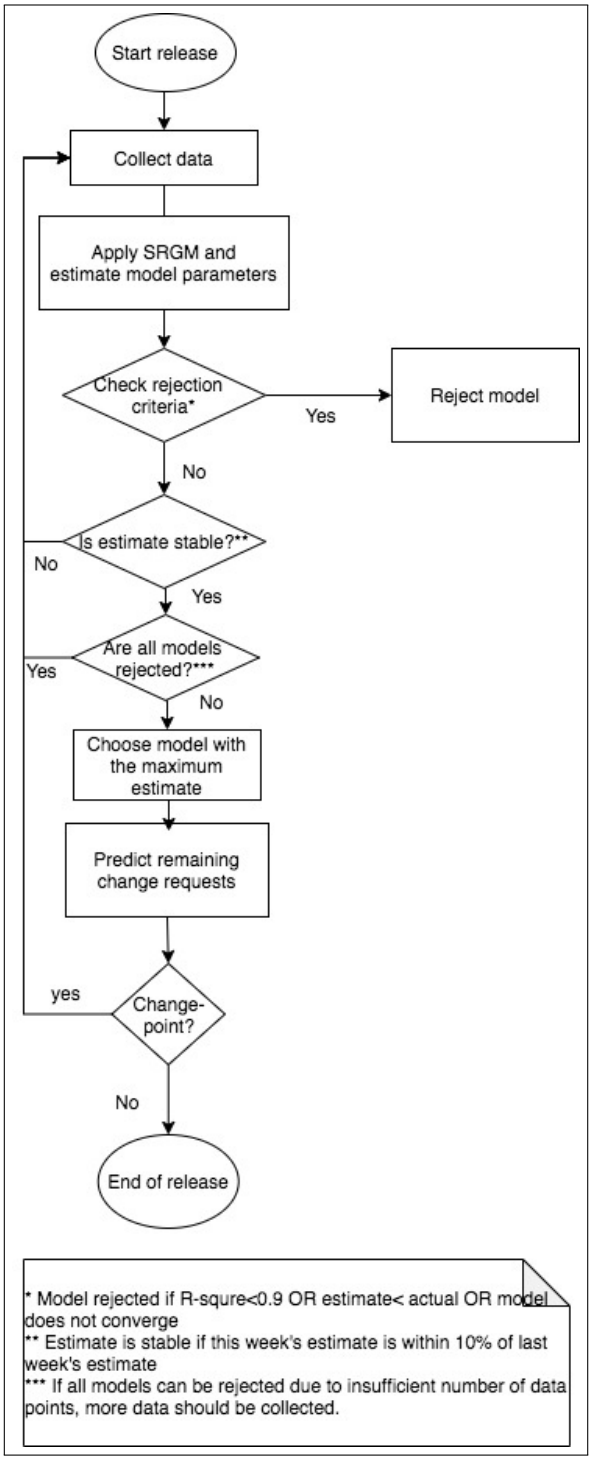


Figure (5.3) Model selection and CR estimation using Approach 2

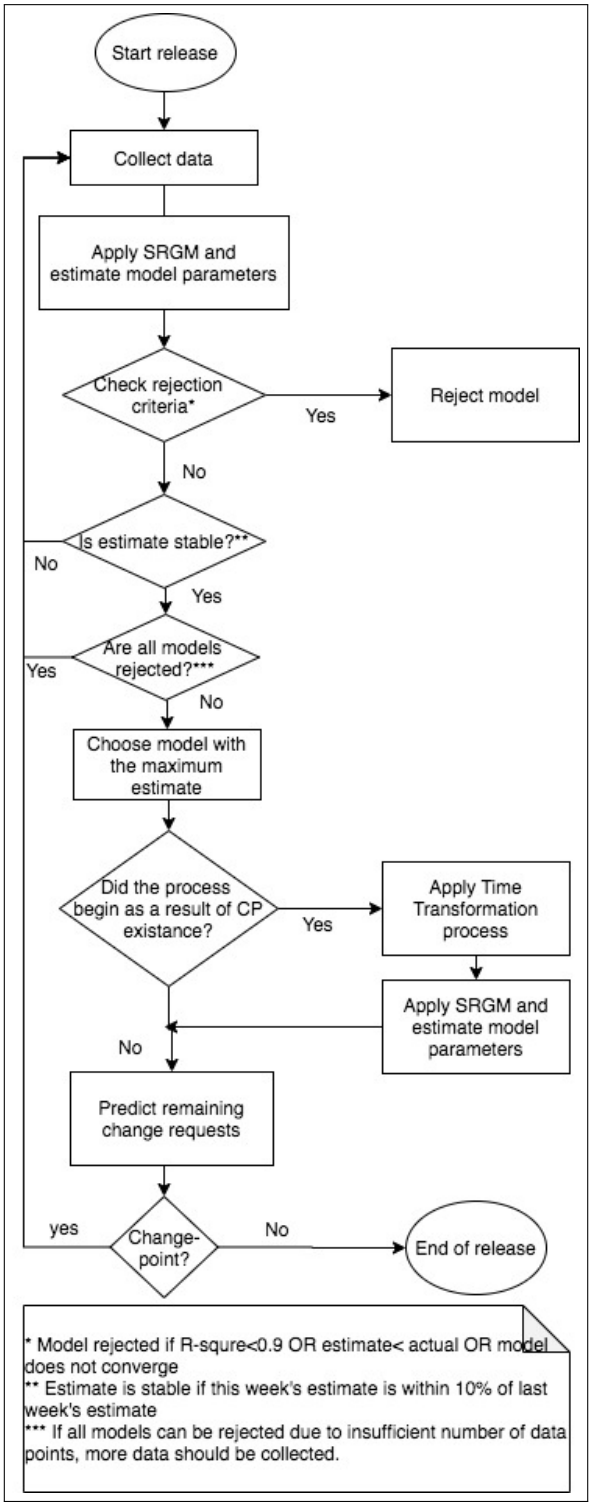


Figure (5.4) Model selection and CR estimation using Approach 3

was introduced by Musa et al. [98] to transform into failure times rather than times between failure occurrences before the change-point into new times consistent with the model applied after the change point. Failure time adjustment transforms the data to account for code changes. The problem in evolution is that when a significant amount of code is changed, the rate of cumulative CR growth changes. In the multi-stage method proposed by Chi et al.[30], we would discard any CR data before the change and we would start all over again after a change-point as if it was a separate release. Approach 3 accounts for code change. Before a change-point, the growth rate of cumulative number of CR is different than the growth rate afterwards. TT calculates a model using the new transformed time, which is calculated using the cumulative CR rate using the parameters of the model before the change-point and the parameters of the model after the change. Typically adding a significant amount of code should increase the CR rate.

When the idea of time transformation was proposed by Musa et al. [98] it was proposed on an analytical model using the same model type before and after change. We plan to build a heterogeneous curve-fit approach that can use a combination of different models to provide the current TT model. In addition, Musa et al. [98] used the model on failure data, we use TT on CR data which is different than failures.

To introduce our approach we explain the process as shown in Figure 5.4. The approach starts similar to Approach 1, with the addition of time transformation after a change point is detected. When a model is selected after a change-point, time transformation is performed and new parameters are determined for the new model before using it for CR prediction.

Our goal is to transform $\mu_i(t)$ to $\mu(t)$, where $\mu(t)$ is the curve after TT, see Figure 5.2. Let $\mu_i(t)$ be the model selected initially using the curve-fitting approach. At T_i changes in code are applied and the CR detection rate changes. T_i is the change-

point for stage s_i, where i is the number of change-points, $1 \leq i \leq n$, and n is the total number of stages.

- s_1 represents the stage before the first change-point T_1
- s_2 represents the stage after the first change-point T_1 and before the second change point T_2.
- $s_{(n+1)}$ represents the last stage after the last change-point.

To perform time transformation on $\mu_i(t)$ to produce the TT model $\mu(t)$ we calculate transformation time $t*$ for each time unit j in the timeline, $1 \leq j \leq m$, m the total number of weeks in the software release. For each stage let D_i represent the total number of cumulative CRs in stage i that occurred at time T_i. Stage 1 has D_1 cumulative CRs which were found by week T_1, while stage 2 has $D_2 - D_1$ cumulative CRs which were found in weeks $T_1 + 1$ to T_2.

To perform the time transformation on the data up to T_2, let $\mu_1(t)$ be the model selected until T_1 and let $\mu_2(t)$ be the model selected after the first change-point according to approach 2. We need to transform the time according to $\mu_1(t)$ and derive a transformed version of $\mu_2(t)$, to obtain the model $\mu(t)$. Let:

$$\mu_1(t) = \lambda(t) \tag{5.3}$$

$$\mu_2(t) = \alpha(t) \tag{5.4}$$

We calculate translated time for CRs before the change $\hat{t}_j$ for $\mu_2(t)$. We assign $\mu_2(t)$ to $\mu_1(t)$ to get the value of the translated time

$$\hat{t}_j = \alpha^{-1}(\lambda(t_j)) \tag{5.5}$$

We then calculate the expected amount of time τ it would have taken to detect D_1 CRs if the new code was part of the original code. By assigning

$$D_1 = \mu_2(\tau) \tag{5.6}$$

$$D_1 = \alpha(\tau) \tag{5.7}$$

$$\tau = \alpha^{-1}(D_1) \tag{5.8}$$

To calculate translated time for CRs observed after the insertion of the new code, we start by asking the question how much time is required for the new model to observe D1 CRs? Then all CR times between T1 and T2 are transformed using the equation below:

$$t* = \hat{t} - (T_1 - \tau) \tag{5.9}$$

The value of τ is less than T_1. For times $t > T_1$, the transformed data consist of the observed CR counts at the translated times. Finally, the new curve $\mu(t)$ is calculated using the new, transformed data.

Let us illustrate this with an example. Assume that the change point T1 occurs in week 264, where the cumulative CRs D1 is 66. Assume that, before the change-point the Modified Gompertz model was selected as

$$M_1(t) = d_1 + a_1(b_1^{(c_1^t)}) \tag{5.10}$$

Assume that after the change-point the Gompetz model is selected

$$M_2(t) = a_2(b_2^{(c_2^t)}) \tag{5.11}$$

Assume that we use each model to find predictions for a week after the change-point we find that in week 267, $M_1(t)$ estimates 66 CRs while $M_2(t)$ estimates 84 CRs.

When performing TT, we calculate $\hat{t}_j$ using the following equation:

$$a_2 * (b_2^{(c_2^{\hat{t}_j})}) = d_1 + a_1 * (b_1^{(c_1^{t_j})}) \tag{5.12}$$

$$a_2 * (b_2^{(c_2^{\hat{t}_j})}) = 66 \tag{5.13}$$

Therefore, $\hat{t}_j = 224$. Using the number of CRs D1, we calculate τ. The value of $\tau = 223$, when calculated as follows :

$$a_2 * (b_2^{(c_2^{\tau})}) = 66 \tag{5.14}$$

We finally calculate the new time as

$$t*_j = 224 - (264 - 223) \tag{5.15}$$

The new $t*_j$ used for $M(t)$ is 183 with 55 cumulative CRs. Fitting the Gompertz model creates a prediction of 75. Using this example, $M_1(t)$ estimated the number of CRs to be 66 which is lower than the actual number of CRs 74. $M_2(t)$ estimated the number of CRs to be 84 which is higher than the actual number of CRs. After TT, the estimated number of CRs is 75, which is higher than the actual but with a smaller error.

Week No.	No. of CRs	G-O		DSS		Gompertz		Yamada		M Gompertz	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
140	5	3	0.66	4	0.5	4	0.81	3	0.65	5	0.86
141	5	3	0.67	4	0.53	4	0.82	3	0.66	5	0.87
142	5	3	0.67	4	0.56	4	0.83	3	0.67	5	0.88
143	5	4	0.68	4	0.59	4	0.83	3	0.67	5	0.89
144	5	4	0.69	5	0.61	4	0.84	4	0.68	5	0.89
145	5	4	0.69	5	0.63	4	0.85	4	0.69	5	0.9

Table (5.2) Estimation using SRGM and the GOF value

Week No.	No. of CRs	G-O		DSS		Gompertz		Yamada		M Gompertz	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
247	48	47	0.44	47	0.74	48	0.94	N/A	N/A	47	0.57

Table (5.3) Full re-stimation using SRGM and the GOF value for stage 2

5.4 Results: Release 4

After collecting and organizing data for Release 4, we used each curve-fitting approach and recorded our results. Tables 5.2, 5.3, 5.4 and 5.5 demonstrates the estimated value of each SRGM for each stage. The table shows a number that represents the week of Release 4, Number of cumulative CRs for each week, and

Week No.	No. of CRs	G-O		DSS		Gompertz		Yamada		M Gompertz	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
268	80	72	0.14	73	0.27	74	0.58	72	0.14	76	0.73
269	81	74	0.15	75	0.29	77	0.61	74	0.15	78	0.77
270	83	76	0.16	76	0.31	79	0.67	76	0.16	81	0.81
271	86	77	0.17	78	0.33	81	0.69	77	0.71	83	0.83
272	86	78	0.19	80	0.35	83	0.72	78	0.18	85	0.86
273	87	80	0.2	81	0.38	85	0.76	80	0.2	87	0.88
274	88	81	0.21	82	0.4	86	0.74	81	0.21	89	0.89
275	89	82	0.23	83	0.43	88	0.8	82	0.22	91	0.9

Table (5.4) Full re-stimation using SRGM and the GOF value for stage 3

Week No.	No. of CRs	G-O		DSS		Gompertz		Yamada		M Gompertz	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
354	154	150	0.34	152	0.6	156	0.88	N/A	N/A	155	0.89
355	154	151	0.37	153	0.65	157	0.8	N/A	N/A	156	0.86
356	159	153	0.37	154	0.65	160	0.87	N/A	N/A	158	0.9
357	165	154	0.33	156	0.59	162	0.9	N/A	N/A	162	0.91
358	166	156	0.33	158	0.58	165	0.93	N/A	N/A	165	0.93
359	167	157	0.33	160	0.59	167	0.94	N/A	N/A	167	0.94

Table (5.5) Full re-estimation using SRGM and the GOF value for stage 4

the estimated number of CRs, the column headed "Est." and the R^2 value of the five models used in this case study Goel-Okumoto model (G-O), Delayed S-Shaped model (DSS), Yamada Model, Gompertz Model and Modified Gompertz Model (M Gompertz).

5.4.1 Applying the Curve-fit Approach

We apply curve-fitting according to Approach 1 to the CR database of the case study using the *MyCurveFit* tool. Table 5.2 shows the weeks were models started fitting data. In week 140, the number of actual CRs is 5. The G-O model estimated only 3 CRs and the R^2 value is only 0.66, which is beneath our threshold, so this model is rejected at this stage. The Delayed S-shaped model estimates only 4 CRs and the R^2 value is only 0.5. The Gomperz model estimates 4 CRs and the R^2 value is 0.81. The Yamada model estimates 3 CRs and the R^2 value is 0.65. And finally the Modified Gompertz estimates 5 CRs which is equal to the actual number of CRs but the R^2 value is only 0.86 which is less than 0.9. The process proceeds to collect data for another week, 141. It rejects all the models as well according to their low R^2 values, which means that more data is collected until week 145. By week 145 the Modified Gompertz model is selected because its R^2 value meets the minimum

threshold requirement of 0.9, the number of estimated CRs is equal to the number of actual CRs and prediction stability is within range since the estimated value for week 145 is within 10% of the value of the previous week. By selecting the Modified Gompertz model we then use it to predict CR in future weeks. Notice that some of the R^2 values gradually change due to adding additional data points. The CR predictions throughout all the stages is shown in Table 5.6 and it will be further explained in Section 5.4.4.

5.4.2 Applying the Multi-stage Method Curve-fit Approach for Change-points

Using this method, we use the same curve-fitting method we used in Section 5.3.2 to predict CRs for the first stage. We refer to the period before the existence of any change-points as "Stage 1". When a change-point exists, we start estimating a new curve after the change and the new curve is used then for CR prediction in the future. This method considers the time period after change as "Stage 2". This applies for multiple change-points, and each time a change-point occurs a new stage is declared. Using the multi-stage method Modified Gompertz is selected for Stage 1 according to Table 5.2. In week 243, a change in the CR rate occurs. We apply the curve-fitting method for the new stage starting from week 243. The minimum number of data points we need to collect to start fitting using our curve-fitting tool is 5 data points. Therefore, we start our first curve-fitting in week 247. In week 247, the Gompertz model has an R^2 value of 0.94 and an estimated value of 48 which matches the actual value, see Table 5.3. We use this model for CR predictions from this point forward until a change occurs.

After the second change-point in week 265 a Modified Gompertz model is selected until week 275, see Table 5.4. After the third change-point in week 357, the R^2 value of both the Gompertz model and Modified Gompertz model is within the acceptable threshold. But these models are rejected due to having the estimated value less than the actual value of cumulative number of CRs. By week 359 all three conditions for selecting a model apply for both Gompertz and modified Gompertz. For this stage the Gompertz model is selected since the d value of the modified Gompetz is equal to zero which makes it a Gompertz model. See Table 5.5.

5.4.3 Applying the Multi-stage Method Curve-fit Approach with Time Transformation for Change-points

This approach starts like the previous curve-fitting approach until a change-point occurs. Then a new curve-fitting is performed to select a new model for the CR data after change. When the new model is selected Time Transformation is performed to adjust the parameters of the final model. After the first change-point, a Gompertz model was selected in a way similar to the multi-stage approach in Section 5.4.2. Time-transformation is then applied to the parameters of the Gompertz model to adjust the parameter of the Gompertz model. The new Gompetz model has an R^2 value of 0.94, so it is used to perform predictions of CRs. Likewise after the change-point in week 264, Time transformation is applied to the Modified Gompertz because $R^2 = 0.97$. Finally after the third change-point the Gompertz model is used after Time Transformation with $R^2 = 0.93$. The resulting model is then used for CR prediction.

Week	Actual CRs	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
145	5	CRs	6	7	7	8	9	10
		RE	0.00	0.00	-0.13	-0.11	0.00	0.11
247	48	CRs	94	104	115	128	141	155
		RE	0.92	0.89	0.92	0.94	0.91	0.80
275	89	CRs	188	207	228	250	275	301
		RE	1.04	1.16	1.33	1.55	1.81	1.95
359	167	CRs	943	1023	1108	1199	1296	1401
		RE	5.83	6.06	6.39	6.73	7.13	7.64

Table (5.6) CR Predictions and Relative Errors for six months into the future using Approach 1

Week	Actual CRs	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
145	5	CRs	6	7	7	8	9	10
		RE	0.00	0.00	-0.13	-0.11	0.00	0.11
247	48	CRs	51	54	57	61	64	68
		RE	0.04	-0.02	-0.05	-0.08	-0.16	-0.26
275	89	CRs	100	110	120	131	143	155
		RE	0.08	0.13	0.18	0.25	0.31	0.34
359	167	CRs	177	187	198	209	221	234
		RE	0.03	0.04	0.06	0.09	0.12	0.17

Table (5.7) CR Predictions and Relative Errors for six months into the future using Approach 2

Week	Actual CRs	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
145	5	CRs	6	7	7	8	9	10
		RE	0.00	0.00	-0.13	-0.11	0.00	0.11
247	48	CRs	51	55	58	62	66	71
		RE	0.04	0.00	-0.03	-0.06	-0.12	-0.21
275	89	CRs	99	108	118	129	140	153
		RE	0.07	0.11	0.17	0.24	0.30	0.33
359	167	CRs	174	183	192	201	211	221
		RE	0.01	0.02	0.03	0.04	0.07	0.10

Table (5.8) CR Predictions and Relative Errors for six months into the future using Approach 3

5.4.4 Comparing Predictive Ability

We compare the number of cumulative CRs for every month for a period of six months after a model was selected. Approach 1 does not consider change-points [128], approach 2 starts curve-fitting at each change-point [30], and approach 3 uses TT at each change-point. In every stage, after model selection, the model is then used for a longer term (six months) CR prediction. We show the results in Tables 5.6,5.7 and 5.8. They are structured as follows: The first column of the table shows the last week before prediction. We used week 145 where the model was estimated for the first stage and the weeks after are the weeks where estimation stopped for each of the stages. The second column represents the number of CRs of that specific week. The next column gives the number of predicted CRs after each month, for up to six months, i.e. (+1 mo.) means predictions after the first month, (+2 mo.) is after two months. The last columns record the relative error value. When the relative error equals zero that means that the predicted number of CRs matches the actual number of CRs. When it is negative, it means that the predicted number of CRs is less than the actual number of CRs, which indicates that the model under-predicted the number of CRs and is rejected.

In Table 5.6 the first row shows week 145, which is the week where the Modified Gompertz model was selected as a model to provide CR predictions. The first two months have a relative error of zero, which means predictions are accurate. Afterwards, the relative error ranges from (-0.13) to (0.11). We then test the predictions of the model after each change-point for six months ahead. We find that the range of the relative error varies and can reach a value of 7.64, which is very high compared to the other approaches.

In Table 5.7, we show the relative error before the first change-point in week 145, which is the same for Approach 1, since no changes in model selection have been made yet. Week 247 is the week where a model was selected for the second stage and prediction started. The model has a relative error value of 0.04. The relative error in the following months range from (-0.05 to -0.26). As the we get further in time, the relative error increases, showing less accurate predictions. After week 275, relative error is 0.8 after one month and 0.34 for a six month prediction. Finally after the last change-point, the relative error starts with 0.03 after one month to 0.17 after six months. In Table 5.7 we see that the predictions in general have low relative error values compared to approach 1, especially when performing predictions after change-points. This represents an improvement over Approach 1 results. Table 5.8 shows the predictions and relative errors after performing TT. We also found that the relative error is relatively low compared to Approach 1 as well. In comparison with Approach 2, TT improves the relative error values. In week 247, the relative error value after two months is zero rather than the negative error value if Approach 2 had been applied. We then notice a decrease of relative error values for every month, which makes this method an improvement in terms of finding better predictions. Looking into the predictions after week 275 and week 359 all show an improvement of relative error values. We highlighted the relative error values of the monthly prediction in Table 5.7 in comparison with the relative error in Table 5.8. The approach that provides worse relative error value among the two approaches, Approach 2 and Approach 3, is highlighted in red and the approach with the better relative error is highlighted in green. We excluded Approach 1 from this comparison because the relative error values are higher than the other two approaches, so the results are not comparable.

We revisit our research questions stated in the introduction:

- RQ1: Can we predict CRs in an evolving legacy system using curve-fitting approaches?

  From our case study we find that we can use curve-fitting defect prediction approaches in CR prediction. The results are promising in predicting CR similar to predicting defects.

- RQ2: What curve-fitting approaches can we use in CR predictions during evolution and change in legacy systems?

  Musa et al. [98] provided general guidelines on how to deal with evolution. We considered those guidelines together with adapting them. We enhanced Stringfellow and Andrews' [128] curve-fitting method that was successful in defect prediction by enhancing it to consider change-points. Chi et al.[30] provided a case study that used a multi-staged method that would re-estimate a new model after each change-point. Musa et al. [98][97] proposed the idea of considering change-points and time transformation to consider the whole release. We found that the enhanced curve-fitting method provides prediction with low error.

- RQ3: How do these approaches compare?

  When change-points are ignored, CR prediction error increases dramatically as shown in Table 5.6. Dividing the release into stages and applying the curve-fitting approach provides more accurate results and lower error especially after change-points. The issue with this method is that it discards old data and starts modeling over with new data points. This gives fewer data points to use in curve-fitting, which affects the quality and reliability of the results. Adding a TT step to the existing curve-fitting approach accounts for all the data in

the release, and uses curves before and after change to find a third curve that accounts for change as if it had existed from the beginning of the release. By comparing the TT method to the multi-stage curve-fitting method in Tables 5.7 and 5.8 we find that the relative error is smaller when TT is applied in all the months except for the third month after week 247 where the multi-stage method provide a smaller relative error. In general we find relative error values are more likely to stabilize or decreases over time when multi-stage or TT approaches are applied compared to the first approach. We also find that the TT approach is superior to the multi-stage approach in providing lower relative error values.

In trying to find what is the best solution, there is no straightforward answer. Each approach is suitable for a specific type of data. If a release has minimal changes that do not affect the CR rate, then Approach 1 would be a suitable approach. When evolution exists the choice is between Approach 2 and Approach 3. Approach 2 provides a simple solution that re-estimates models as required. This is beneficial if at each stage there are enough data-points to perform the curve-fitting. It is not recommended to use when change-points are frequent. The problem with this approach is that under-estimating of CRs is likely to occur due to over-fitting. In addition, sometimes a model is selected early in a particular stage based on very few data points. This could lead the curve-fitting tool to settle on a model that poorly predicts future CRs. Approach 3, using TT overcomes the issues in Approach 2. After a change-point, when a model is selected, TT includes data from the beginning of the release to estimate the new model parameters and this overcomes the risk of curve-fitting with too little data. TT also reduces the risk of over-fitting models and causing under-estimation. In CR prediction, a model that frequently underestimate

CRs is not desirable and introduces the risk of management not being prepared for the number of CRs in the future. So we find that TT is a good fit in an evolving release to provide both short-term and longer term CR prediction. In industry there are many systems that evolve during a release for a variety of reasons. Our aerospace system is one of them.

5.5 Validity Threats

We follow the guidelines by Runeson et al.[111] in defining our validity threats. An external validity threat is concerned with the generalization of our results. Although we used approaches on an evolving system we do not claim that it will produce similar outcomes for all other software systems. We claim that our solution works best for evolving systems. The amount of change and the frequency of changes play a key role in defining change-points and how much these methods are of improvement. Our data was collected by a third party which means the researchers have less control over the quality of the data, which is a threat to internal validity. Construct validity refers to the relation between theory and observation. We observed that some models fit data better than others. This may be affected by the number of data-points used for model estimation and selection. If the size of data used is too small we may be at risk of selecting a model that does not predict very well.

5.6 Conclusion

In this case study, we investigate the use of three different curve-fitting approaches that have been used in defect prediction for predicting Change Requests

(CR) instead. We tested their ability to predict future CRs using a CR database of an evolving aerospace legacy system. We then compared the predictive ability of each of the curve-fitting approaches for Release 4 in an effort to find the approach with the most accurate prediction of CRs. The predictions were performed monthly for up to six months after a model was selected. We applied the curve-fitting approach [128] that does not account for change-points. The predictions showed a low relative error at first, but as soon as the release evolved, the predicted number of CRs were much higher than the actual number of CRs. The second approach applied was a multi-stage approach that segments the dataset whenever a change-point is found. The multi-stage model based on the work presented by Chi et al. [30] had proven to give lower relative error in the predicted values but often future CRs are underestimated. Underestimation of the number of CRs puts an organization at risk for not being prepared for the volume of work. Finally, the use of Time Transformation (TT) first proposed by Musa et al. [98] [97] along with the curve-fit approach has shown predictions with lower relative error than both of the other approaches and with fewer under-predicted CRs. The idea of TT has not been widely used in literature. Before our work, it was demonstrated only with analytical, homogeneous models. The assumptions upon which these models are based on are not met when CRs are considered. For industrial databases that contain CRs for both defects and enhancements, curve-fit methods are more realistic since they only select an appropriate model according to the given dataset without the assumptions made by analytic models. This chapter used only one release (Release 4). We will provide further validation by applying these approaches and comparing the results for the remaining releases in Chapter 7.

Chapter 6

Early Prediction Versus Accuracy

6.1 Problem Statement

After release, managers also need to predict future CRs to gauge software quality and adjust staffing levels for maintenance and evolution. They should be able to predict CRs early on. In the literature, researchers focus on model estimation techniques and prediction capability, but not much attention was concentrated on when to predict? During system testing, a rule of thumb states that 60% of the test plan should have been executed before starting to apply reliability models, which refers to 60% of the percent of calendar test time [140]. When the system is released, it is unclear what would constitute the right point to start predicting further CRs, especially when the software also evolves. In the field of statistics, Bentler and Chou [23] provided an oversimplified guideline based on previous work from Bentler [22] to serve as a rule of thumb for the number of observations per parameters estimated in a model. They mentioned that a sample size could go as low as 5 observations but sees the use of at least 10 observations is more appropriate. Aguinis and Harden [3] suggest that a minimum of 10 observations is recommended for obtaining trustworthy estimates of parameters. In statistics this rule of thumb is a benchmark on

adequacy of dataset size in terms of number of observations required in regression models. Specifically, this rule suggests having a minimum of 10 observations per predictor variable[1] [132]. An observation is a value of something of interest counted for a study such as a person's height, a bank account value at a certain point in time, or number of defects in a certain month. For example: the balance of a bank account in each month is an observation, the following are four observations:

- January = 1000 Dollars
- February = 1300 Dollars
- March = 800 Dollars
- April = 1100 Dollars

Since we are dealing with cumulative CRs over week time units, we consider the cumulative CRs per time unit (week) to be the observation that we are discussing in this chapter.

Therefore, we set up our research questions as follows:

- RQ1: When do we start predicting CRs during operation?
- RQ2: How does the number of observations affect the prediction accuracy of CR prediction model?

We use Release 4 of our case study to evaluate the quality of early predictions. We discuss different scenarios of performing early predictions vs. predicting CRs later in a release. We then provide some recommendations for software project managers to assist them CR predictions. The remainder of the chapter discusses

[1]Predictor variable is the independent variable used in regression analyses, which is time in this case measured by weeks

our approach in section 6.2. Then we discuss our results in section 6.3 and validity threats in Section 6.4. Finally, we draw conclusions in Section 6.5.

6.2 Approach

6.2.1 Data Collection

Our data represents number cumulative number of CRs per week, which are our observations. Since the least number of observations to perform regression is 5 observations we require that five weeks of cumulative CRs are collected before estimating any model. We also require that collected data has at least 5 unique CRs, which is the minimum number of CRs to start applying any model. In some cases we have several weeks with no additional CRs that will cause the cumulative CRs function to be linear. Knowing that the models we are dealing with are non-linear we require five unique CRs to be collected before starting to model. This case occurs at the beginning of each release where one CR is found in one week then a few weeks later another CR is added. Therefore, we require five unique CRs at the beginning of each release before estimating a model.

6.2.2 Model Estimation

Once enough data is collected, we use them in model estimation. We follow the multi-stage curve-fitting approach explained in Section 5.3.2. Curve-fitting is performed for cumulative CRs until an SRGM is selected to be used for future CR prediction. When a change-point occurs another curve-fitting is performed to find a new SRGM that fits the new dataset after the change-point. Change-points occur when changes in cumulative CR growth rate take place due to a system fix or

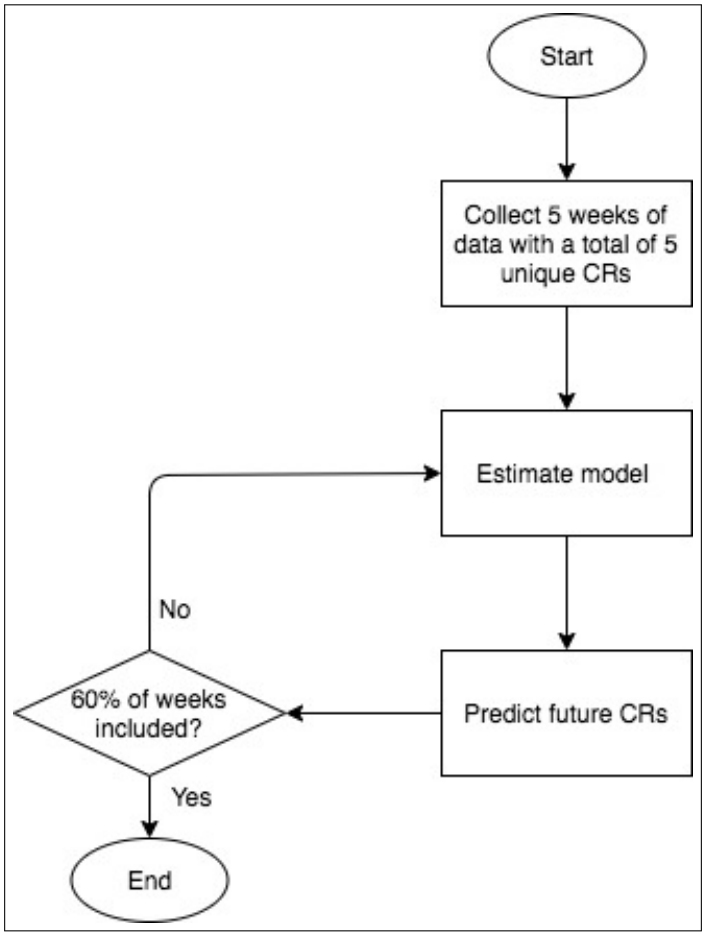


Figure (6.1) Repeated model selection process from 5 weeks of unique CRs up to 60% of the weeks of a stage

maintenance. Change-points were previously defined and discussed in more detail in Chapter 4.

6.2.3 Predicting CRs

After model estimation, a model is selected to be used in CR prediction based on the method in Section 6.2.2. We apply the same process after collecting data for one more week and we measure the accuracy of prediction then. This process is repeated for 60% of the weeks of every stage as shown in Figure 6.1 and the Algorithm 1.

Algorithm 1 Repeated model selection process from 5 weeks of unique CRs up to 60% of the weeks of a stage

Result: Find relative Error of selected models starting from collecting the first 5 weeks of unique CRs up to 60% of a stage in a release in order to compare their prediction ability and conclude the optimum amount of data collected to estimate more accurate models

Data: Collect at least 10 weeks of unique 5 CRs or more from the current stage

1: **while** less than 60% of the weeks of the stage collected **do**
2: Estimate a model using one of the model estimation approaches
3: Record the week when a model was selected
4: Use model for CR prediction for six months
5: Calculate relative error of every month
6: **end while**=0

6.3 Results

For this case study we apply curve-fitting to data in the CR database of Release 4 for SRGM selection. Then we use the selected model for long-term prediction of CRs. This process is applied at the beginning of the release until a model is selected for prediction and after the existence of a change-point. Since we have three change-points in weeks (243, 265, 348), we applied model selection four times, (see Chapter 4). Afterwards, we compare the predictive ability of each selected model. Model Selection was performed using a multi-stage approach that has been described in Section 5.3.2.

When the first 5 CRs were collected, the multi-stage model is applied according to the process in Figure 6.1. The results of model estimation and prediction of each stage is explained below:

Week	Actual CRs	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
145	5	CRs	6	7	7	8	9	10
		RE	0.00	0.00	-0.13	-0.11	0.00	0.11

Table (6.1) CR Predictions and Relative Errors of the Modified Gompertz model for Stage 1

Week	Actual CRs	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
247	48	CRs	51	54	57	61	64	68
		RE	0.04	-0.02	-0.05	-0.08	-0.16	-0.26
256	56	CRs	59	64	68	73	78	83
		RE	-0.06	-0.03	-0.15	-0.15	-0.12	-0.11

Table (6.2) CR Predictions and Relative Errors of two Gompertz models for Stage 2

6.3.1 Stage 1

For stage 1, after collecting 5 unique CRs we apply the curve-fitting process. The first model selected was the Modified Gompertz in week 145. By this time 60% of the weeks in the stage time has been used for model estimation. Table 6.1 shows the long-term prediction by month of this selected model. For the first two months the relative error is zero. Afterwards, the relative error is negative which means that the model is under predicting. For this stage we cannot test the predictive ability of the 5 rule-of-thumb or 10 rule-of-thumb or compare it with the results of the selected model, since it was not applicable due to the limited number of unique CRs early in the stage.

6.3.2 Stage 2

In Stage 2, we applied the process in Figure 6.1 and we selected the Gompertz model in week 247. This stage started in week 243, data was collected for five weeks (243, 244, 245, 246, 247). This model used five weeks of cumulative CRs. The next

Week	Actual CRs	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
275	89	CRs	100	110	120	131	143	155
		RE	0.08	0.13	0.18	0.25	0.31	0.34

Table (6.3) CR Predictions and Relative Errors of the Modified Gompertz model for Stage 3

model selected was in week 256, (after 13 CRs were collected). This happened when 66% of the time in Stage 2 has elapsed. The first model represents the 5 rule-of-thumb and the second model is a little over 10 weeks, which can be considered as 10 rule-of-thumb and it is at a point where 66% of the stage weeks are included.

Table 6.2 shows that the RE in predictions for the two Gompertz model, the first was selected in week 247 and the other is in week 256. We find that the relative error for early prediction is better for the first model but then the RE is negative. In general we see that the second model provides better predictions in later months which indicates that it provides better long term predictions. In general we find that this stage is short. A change-point occurs in week 265, which makes prediction values to be poor.

6.3.3 Stage 3

Table 6.3 shows the predictions of the Modified Gompertz model that was selected after the second change-point in week 275. When this model was selected on;y 8% of the weeks were used for model estimation, (13 weeks). No other model was later selected for this stage. In general we find that the RE values are positive ranging from 0.08 to 0.34.

Week	Month					
	+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
359	0.03	0.04	0.06	0.09	0.13	0.17
360	0.03	0.05	0.07	0.09	0.14	0.19
361	0.04	0.05	0.07	0.1	0.14	0.19
362	0.03	0.05	0.08	0.11	0.15	0.19
363	0.03	0.05	0.07	0.1	0.15	0.18
364	0.03	0.05	0.07	0.1	0.15	0.17
365	0.03	0.04	0.06	0.1	0.14	0.17
366	0.03	0.04	0.07	0.09	0.13	0.17
367	0.03	0.04	0.07	0.1	0.13	0.17
368	0.03	0.04	0.07	0.11	0.13	0.17
369	0.03	0.04	0.07	0.11	0.13	0.18
370	0.03	0.05	0.07	0.1	0.14	0.17
371	0.03	0.05	0.08	0.1	0.14	0.18
372	0.03	0.06	0.09	0.1	0.14	0.18
373	0.03	0.05	0.08	0.11	0.15	0.18
374	0.04	0.06	0.08	0.11	0.14	0.18
375	0.03	0.06	0.08	0.12	0.15	
376	0.04	0.07	0.08	0.12	0.16	
377	0.04	0.07	0.09	0.13	0.16	
378	0.04	0.07	0.1	0.12	0.16	

Table (6.4) CR Predictions and Relative Errors of Gompertz models for Stage 4

6.3.4 Stage 4

According to the process in Figure 6.1, the first model was selected in week 359, the Gompertz model. Since the process keeps collecting data by week and estimating models, we found a fitted model in every week for at least 60% of the stage. We collected the data from week 359, which is the 11^{th} week in the stage until week 377 which as at 60% of the total weeks in that stage. To better demonstrate the results we show the RE value of month-by-month predictions for six months for each of these models except for the last three models. We cannot apply predictions of a sixth month for the models selected in weeks 376, 377 and 378 since the end of the stage occurs before six months. Table 6.4 shows the RE values of each model selected in a specific week. We find that over the weeks the RE values of predictions after each month are very close. In fact comparing the RE values of week 11 to the RE values of week 30, we find that the RE values in week 11 are better. This shows that with only 11 data points a prediction can perform as well as or sometimes better than a model that is selected later on.

6.3.5 Discussion

In this section we discuss the results and revisit the research questions stated in the introduction of this paper.

6.3.5.1 RQ1: When can we predict CRs during operation?

This question has no definitive answer. A small set of data can introduce the risk of giving false predictions or under-estimating future CRs. Predicting future CRs also depends on how early a model is selected based on the data. We found that in

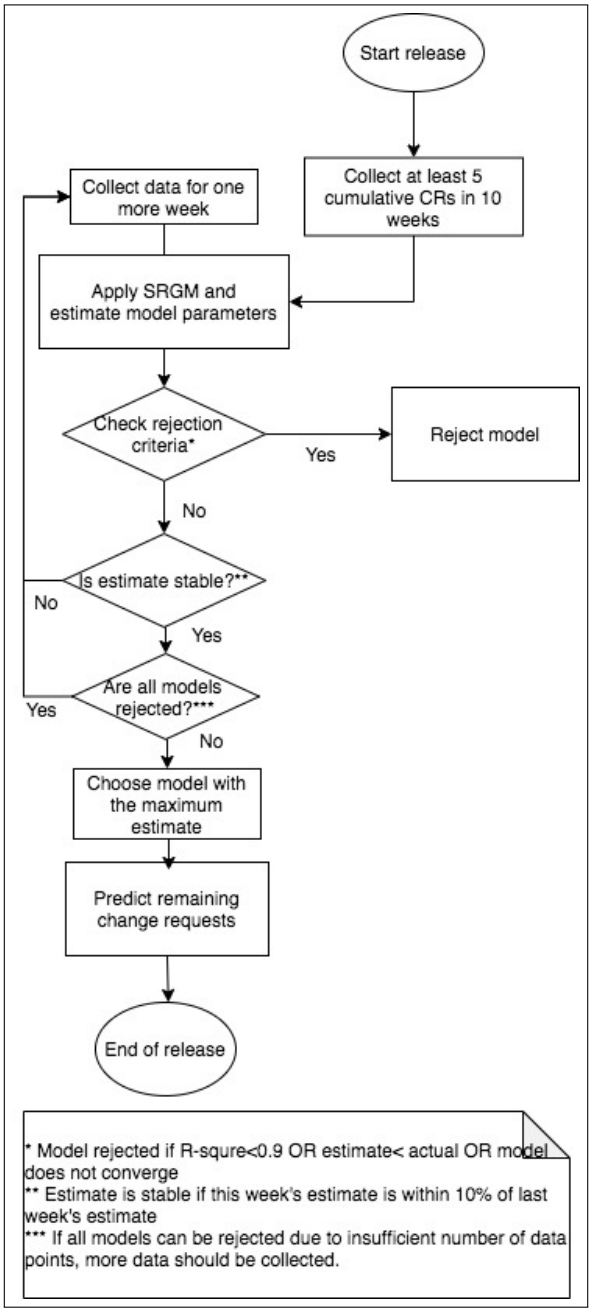


Figure (6.2) Modified model selection And CR Estimation using Approach 1

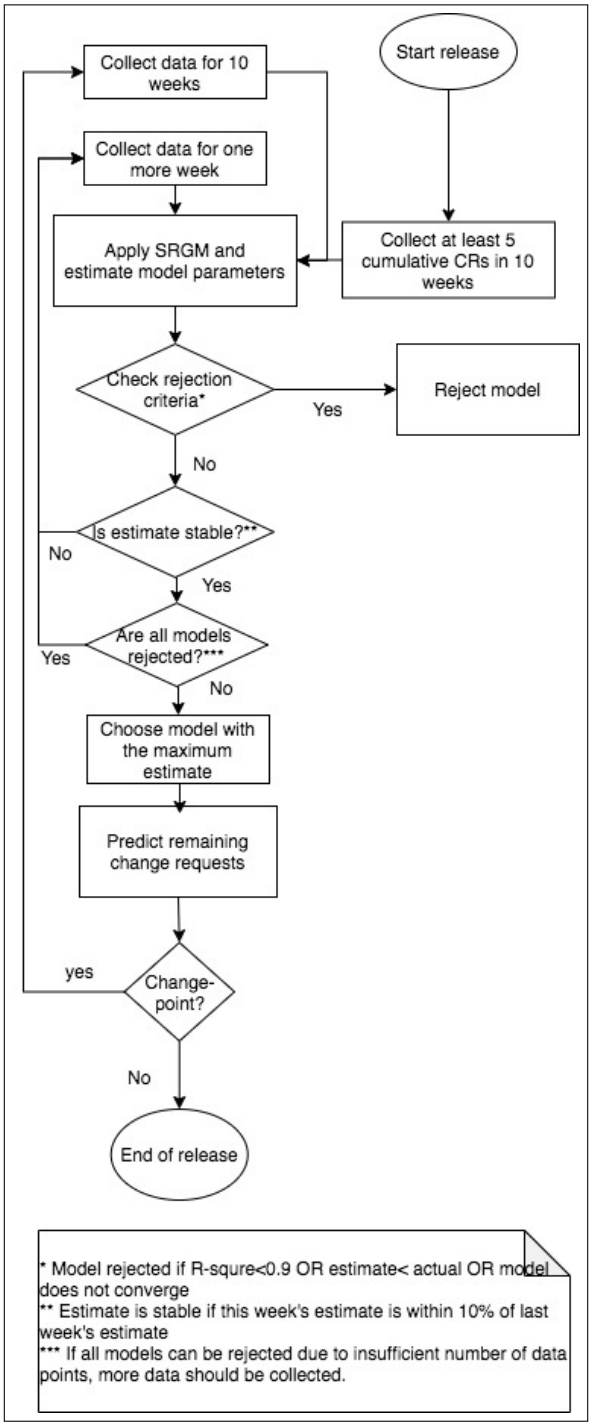


Figure (6.3) Modified model selection And CR Estimation using Approach 2

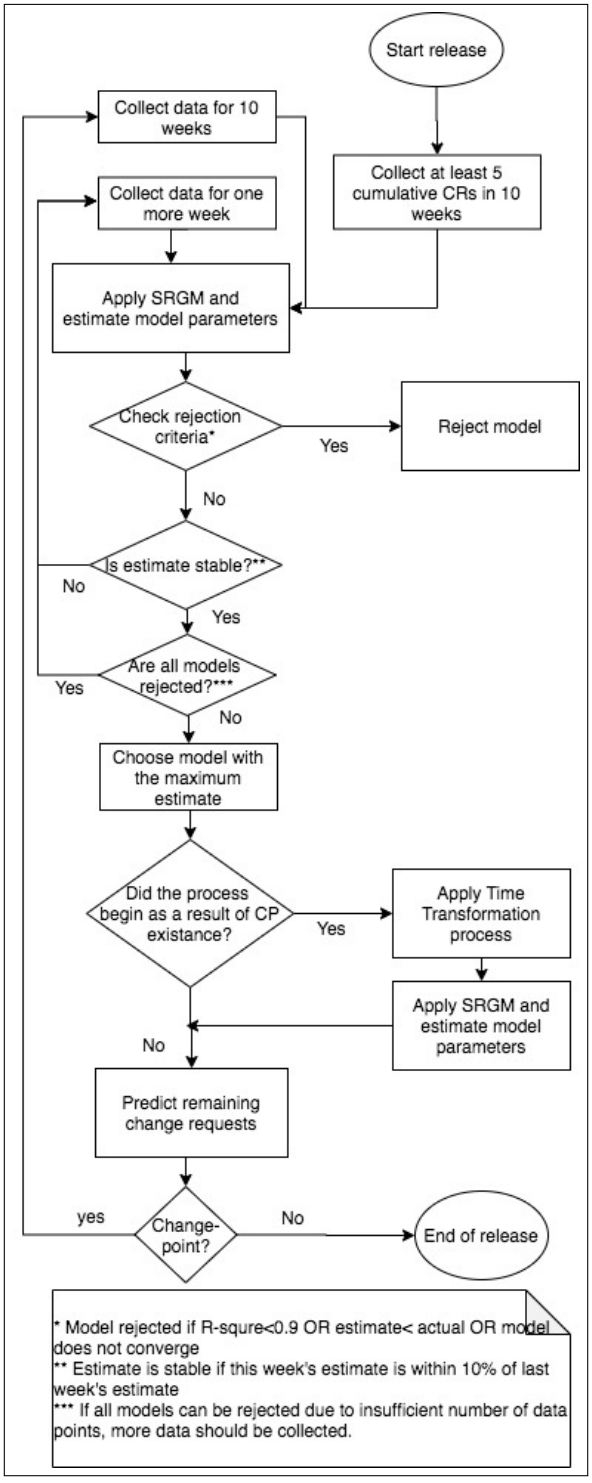


Figure (6.4) Modified model selection And CR Estimation using Approach 3

some stages, models were selected as early as 5 weeks and in others we needed 145 weeks to fit a model. When a model is selected, it can be used for predictions.

6.3.5.2 RQ2: How does the number of data-points affect the prediction accuracy of a CR prediction model?

We are looking for a trade-off to find a point where we can find accurate predictions as early as possible. For this case study we applied our approach described in Figure 6.1.

In Stage 1, we find that the model selected at 60% has small REs in the first two months. The RE values of the next two months are low as well but negative. Then they become positive and low for the next two months. In Stage 2, a model was selected based on only 5 observations (CRs), but looking at the predictions in the future we find that it fails to provide reliable predictions after the first month. The model that was selected later on did not perform well also, but looking into the long term predictions we find that the model selected at 60% provides better RE values. For his specific stage, we find that it is a short stage (under six months) so the six months predictions are mostly negative due to having a change-point before after about four months. In stage 3, the model is selected at 13 weeks which is about 24% into the stage. It has positive RE values which range from 0.08 to 0.34. Finally, we find that stage 4 has RE values that are small and positive when a model is selected after 11 weeks and better than the RE values after 60% of the data was collected.

We find that it is possible to find a model that fits a data-set with only 5 data-points, however, the accuracy of the model in predicting future CRs is questionable. On the other hand, we find that some models that used 11 or 13 weeks provided more reliable predictions than a model that used 145 weeks. Comparing the prediction accuracy of these four models, one was selected after 60% of the stage was collected,

one was selected after five weeks of collection and two were selected after a little over 10 weeks of data, we find that the quality of prediction of the models selected after at least 10 weeks are better than the models selected by 5 weeks and comparable to the models selected at 60%. Managers need to weigh the risk of not being able to predict CRs for long periods of time before deciding that "enough data had been collected" against the risk of having predictions that are less accurate. Therefore, we modify the three approaches of models estimation in Figures (5.1, 5.3 and 5.4) to include a minimum of 10 weeks of of observations before modeling with at least 5 collected CRs. Figures (6.2, 6.3, and 6.4) show the three approaches of model estimation after applying the suggested the modification.

6.4 Threats to Validity

We address these threats according to the criteria described by Wohlin et al.[139]. External validity is concerned with generalization of the findings. While we applied our approach to the evolving large software system described in Chapter 3, other evolving software may not show the same CR behavior. Model selection and parameter estimation depend solely on the specifics of the CR data and existence of change-points. Therefore we do not claim that our results can be generalized. However, we applied our method repeatedly to data that has different CRs patterns to confirm that the results were consistent. Conclusion validity focuses on how sure we can be that the treatment we used in an experiment really is related to the actual outcome we observed. We don't claim that there is a linear relationship between the amount of data used for model selection and the accuracy of prediction. We provide guidelines that a researcher might find useful with the consideration of varying results.

6.5 Conclusion

Early planning is key for managers of the operational software. We find that early model selection runs the risk of poor long-term predictions but in general they give managers an idea of how their systems are performing, which assist them in planning. From our case study we found that selecting a model based on ten weeks rule-of-thumb provides more accurate CR predictions than five weeks rule-of-thumb. This observation was applied to the model selection approaches described in Chapter 5 to produce the models in Figures (6.2, 6.3, and 6.4), which will be used in them following Chapter 6.

Chapter 7

Multi-release Change Request Prediction

7.1 Introduction

When software products are released they are mostly not developed with the full set of functionalities. Some requirements and further enhancements are developed in the later releases. Therefore, most large software experience multiple release. The aerospace system in our case study is no exception to that. In Chapter 3 we described the system which has four releases. Each of these releases has cumulative CR data over time. Each release has change-points which were estimated in Chapter 4. On Chapter 5 we applied three curve-fitting approaches explained in Section 5.3.1, Section 5.3.2 and Section 5.3.3 to Release 4 and compared the three approaches according to their predictive ability. We found that the TT approach has a lower relative error than the other curve-fitting approaches. We wish to expand our observations and apply the three approaches to the remaining releases. So our research questions for this chapter are:

- RQ1: Can we generalize the use to the three approaches for curve-fitting to other releases?

- RQ2: Can we generalize our findings that TT approach performs better than the other curve-fitting approaches to other releases?

To answer these questions we use the same methodology of applying model estimation and CR prediction to the three remaining releases: Release 1, Release 2 and Release 3. We then compare the prediction ability for the three approaches within each release and finally we compare and discuss the results among the releases.

7.2 Release 3

This release is a long release that spans 472 weeks and shows 401 CRs. When applying the change-point estimation we found that four change-points were estimated in weeks 291, 326, 370 and 420. We can then divide this release into 5 stages:

- Stage 1: From week 1 to week 289
- Stage 2: From week 291 to week 325
- Stage 3: From week 326 to week 369
- Stage 4: From week 370 to week 419
- Stage 5: From week 420 to week 472

On each stage we start with 10 weeks of collected data before estimating a model for the stage. This is due to our recommended finding of having at least 10 weeks of data to get more reliable results in Chapter 6, so the 10 observations rule-of thumb is applied on the number of weeks collected for SRGM estimation. Whenever we find a fitted model then we use that model for future CR prediction from that point forward. We compare the predictive ability of each model by calculating the relative

error of month-to-month prediction for six months into the future. Notice that for a model to fit CRs, there should be at least 5 cumulative CRs collected, which is the minimal five observations rule-of-thumb to select a model.

7.2.1 Model Estimation

7.2.1.1 Approach 1

For this approach we do not consider change-point data. The main concern here is to fit an SRGM model on the cumulative CRs for this release starting from the first week. Once a model is selected we use it for future CR predictions. Table 7.1 shows the progression of fitting the five SRGMs to the data once 5 CRs were collected. By week 271, The Modified Gompertz model has an R^2 of 0.90 and an estimated value that is greater than or equal the actual value. Therefore, this model is used for CR prediction using this approach.

7.2.1.2 Approach 2

The Modified Gompertz model selected in Approach 1 is used in CR prediction using this approach as well but that selection changes after the occurrence of change-points. We call this time period, Stage 1. Therefore, when a change-point is estimated for the week of 291, a new SRGM selection process takes place to select a model that would be used for CR predictions until the following change-point, and this will be our second stage.

From the week of 291 we collect ten more weeks of cumulative CRs then we apply the SRGM selection process weekly until we find a fit model. From week 300 we start examining the Goodness-of-fit and of each model. If no model fits then we collect CRs for another week and then we try to fit the models again until we find the

Week No.	No. of CRs	G-O		DSS		M Gompertz		Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
255	6	2	0.27	3	0.2	4	0.88	4	0.88	3	0.51
256	6	2	0.29	3	0.25	4	0.88	4	0.88	3	0.52
257	6	2	0.3	3	0.29	5	0.88	5	0.88	3	0.52
258	8	3	0.31	3	0.32	5	0.86	5	0.86	3	0.51
259	8	3	0.31	3	0.35	5	0.85	5	0.85	4	0.51
260	8	3	0.31	4	0.38	5	0.85	5	0.85	4	0.52
261	9	3	0.31	4	0.39	6	0.84	6	0.84	4	0.53
262	9	3	0.31	4	0.41	6	0.83	6	0.83	4	0.54
263	9	3	0.32	4	0.42	6	0.85	6	0.85	5	0.56
264	9	3	0.32	4	0.43	7	0.85	7	0.85	5	0.57
265	9	3	0.32	4	0.44	7	0.84	7	0.84	5	0.59
266	9	3	0.33	4	0.46	7	0.87	7	0.87	6	0.61
267	9	3	0.33	4	0.47	8	0.87	8	0.87	6	0.63
268	9	3	0.34	5	0.48	8	0.87	7	0.87	6	0.65
269	9	3	0.34	5	0.49	8	0.87	8	0.85	6	0.66
270	9	3	0.35	5	0.5	8	0.88	8	0.86	6	0.67
271	9	4	0.35	5	0.51	9	0.9	8	0.86	7	0.69

Table (7.1) SRGM Estimation for Stage 1 the GOF value for Release 3: Approach 1

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
300	17	17	0.22	17	0.35	17	0.59	18	0.59	17	0.54
301	17	17	0.23	17	0.36	18	0.55	18	0.55	18	0.56
302	17	17	0.25	17	0.37	18	0.51	18	0.52	18	0.53
303	18	17	0.28	17	0.43	18	0.59	18	0.59	18	0.6
304	18	17	0.3	17	0.47	18	0.64	18	0.64	18	0.65
305	18	17	0.33	17	0.51	18	0.67	18	0.67	18	0.68
306	19	17	0.34	18	0.54	19	0.73	19	0.73	16	0.73
307	19	18	0.35	18	0.57	19	0.77	19	0.77	16	0.77
308	19	18	0.36	18	0.59	19	0.8	19	0.8	16	0.8
309	19	18	0.38	18	0.61	19	0.81	19	0.81	17	0.82
310	19	18	0.39	18	0.64	19	0.82	19	0.83	19	0.84
311	19	18	0.41	18	0.66	20	0.83	19	0.83	19	0.85
312	20	18	0.41	19	0.67	20	0.85	20	0.86	20	0.87
313	21	18	0.4	19	0.66	20	0.87	20	0.87	20	0.87
314	24	19	0.33	19	0.56	21	0.81	22	0.82	21	0.81
315	25	19	0.28	20	0.5	22	0.79	23	0.81	22	0.79
316	26	19	0.26	20	0.47	23	0.8	23	0.8	23	0.8
317	27	20	0.24	20	0.44	24	0.81	24	0.81	24	0.81
318	28	20	0.23	21	0.43	25	0.82	25	0.83	25	0.83
319	29	20	0.22	21	0.41	26	0.84	27	0.84	26	0.85
320	30	21	0.22	22	0.4	27	0.86	27	0.86	27	0.86
321	31	21	0.21	22	0.39	29	0.87	28	0.87	29	0.88
322	31	22	0.21	23	0.39	30	0.88	29	0.88	30	0.89
323	31	22	0.21	23	0.4	30	0.9	30	0.89	30	0.9
324	32	22	0.21	24	0.4	31	0.91	31	0.9	31	0.91
325	34	23	0.21	24	0.4	32	0.92	31	0.92	32	0.92

Table (7.2) SRGM Estimation for Stage 2 the GOF value for Release 3: Approach 2

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
335	45	41	0.19	42	0.35	45	0.89	45	0.87	44	0.73
336	46	42	0.2	42	0.37	46	0.9	46	0.9	44	0.76

Table (7.3) SRGM Estimation for Stage 3 the GOF value for Release 3: Approach 2

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
379	148	132	0.16	134	0.31	146	0.85	143	0.87	147	0.95
380	148	134	0.17	136	0.32	149	0.87	146	0.89	149	0.95

Table (7.4) SRGM Estimation for Stage 4 the GOF value for Release 3: Approach 2

most fit model. In week 323 in Table 7.2, we see that both Gompertz and Yamada model meet the minimum threshold value of $R^2 = 0.9$, but it estimates 30 CRs less than the actual value of 31 CRs. Next, the process proceeds by collecting more data to find the same issue in week 324, where Yamada and Gompertz have estimated CRs that are less than the values of the actual CRs. This also continues for week 325 (the last week of this stage). This means that no SRGM model was selected for this stage since none of the models were good enough. Therefore, we continue using the Modified Gompertz model selected in the Stage 1 for CR predictions.

For stage 3, we find that in week 336 both the Gompertz model and the Modified Gompertz have an R^2 value of 0.9 and an estimated number of CRs that is equal to the actual, 46. This is also within 10% from the previous value. Any of these models can be selected and for this stage, the Gompertz model was selected, (Table 7.3).

In Stage 4, the R^2 value of the Yamada model by week 379 is 0.95 which makes it a candidate to be selected for this stage but the estimated CRs is lower than the

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
429	277	270	0.43	273	0.73	278	0.947	278	0.94	278	0.94

Table (7.5) SRGM Estimation for Stage 5 the GOF value for Release 3: Approach 2

actual number of CRs. Since no other model meets the the criteria, we continue to collect more CR data for another week. The Yamada model still maintains an acceptable R^2 and the estimated CR value is greater than the actual CR values, in addition to having the estimated number of CRs within 10% from the previous one. Therefore, the Yamada model is selected for this stage, (see Table 7.4).

Table 7.5 shows the SRGM estimates for week 429, which is the tenth week after the fourth change-point. Since we needed 10 weeks to collect data before running the model estimation process, we found that by this week we already have three candidate SRGMs. The Gompertz, the modified Gompertz and the Yamada all have an R^2 value of 0.94 and an estimated value that is greater than the actual value. For this stage the Gompertz model was selected.

So for this release we have selected four different SRGMs for Stages 1, 3, 4 and 5. Stage 2 continues to use the SRGM used in Stage 1 since no new SRGM was selected.

7.2.1.3 Approach 3

Approach 3 uses the selected model for each stage after a change-point occurrence and applies Time Transformation (TT) calculations to calculate a new time which presumably would have been the time of the CRs if changes were accounted for since the beginning of the release.

(Actual Week	**Actual CRs**	$M_1(t)$ **Estimation**	$M_2(t)$ **Estimation**	**Week after TT**	$M(t)$ **Estimation**
326	34	33	37	322	36
327	36	33	38	323	36
328	38	34	38	324	37
329	39	35	39	325	38
330	41	36	40	326	39
331	42	37	41	327	40
332	43	37	42	328	41
333	44	38	43	329	43
334	44	39	44	330	44
335	45	40	45	331	45
336	46	41	46	332	46

Table (7.6) CR estimation with TT for Stage 3 of Release 3

For the first and second stage no TT is required, since they are using the SRGM of the first stage and no new SRGM is selected after change. For the third stage, $M_1(t)$ is the Modified Gompertz model selected for the first stage and $M_2(t)$ is the Gompertz model selected for the third stage. Table 7.6 shows the values of the estimated CRs using $M_1(t)$ and $M_2(t)$ and then the value of CRs for $M(t)$ using the new time after transformation (TT), notice that the time here is the week number. In week 326, the actual number of CRs is 34, $M_1(t)$ predicts that the number of CRs is 33 and $M_2(t)$ estimates 37 CRs. Using these two models we calculate a time after Time Transformation (TT), as explained in Section 5.3.3. Using the newly transformed values of the weeks we estimate CR values using $M(t)$ which is a newer version of $M_2(t)$ with new times and parameters. $M(t)$ is then used for CR prediction until the next stage, Stage 4. When Stage 4 begins we use $M(t)$ from Stage 3 as $M_1(t)$ and the Yamada model selected for Stage 4 as $M_2(t)$. We use $M_1(t)$ and $M_2(t)$ in calculating TT, which is then used to estimate CRs in $M(t)$, (see Table 7.7). Finally, $M(t)$ from the previous stage is used as $M_1(t)$ and the Gompertz model is in $M_2(t)$ to calculate TT to be used by $M(t)$ for the final stage, (Table 7.8).

(Actual Week	**Actual CRs**	$M_1(t)$ **Estimation**	$M_2(t)$ **Estimation**	**Week after TT**	$M(t)$ **Estimation**
370	114	104	117	369	117
371	116	106	119	370	120
372	121	109	122	371	123
373	124	111	126	372	126
374	128	114	129	373	129
375	128	116	132	374	132
376	137	119	135	375	135
377	144	122	139	376	139
378	144	125	142	377	142
379	148	127	145	378	145
380	148	130	149	379	149

Table (7.7) CR estimation with TT for Stage 4 of Release 3

(Actual Week	**Actual CRs**	$M_1(t)$ **Estimation**	$M_2(t)$ **Estimation**	**Week after TT**	$M(t)$ **Estimation**
420	254	371	256	409	256
421	259	379	259	410	259
422	260	387	261	411	261
423	264	396	263	412	263
424	268	404	266	413	266
425	271	413	268	414	268
426	272	422	271	415	270
427	273	431	273	416	273
428	273	441	276	417	275
429	277	450	278	418	278

Table (7.8) CR estimation with TT for Stage 5 of Release 3

These estimated models are then used to provide CR predictions in the future. In the following section we demonstrate and compare the results of predictions using the three approaches we applied.

7.2.2 CR Prediction

To compare the predictive ability of the three approaches we compare their ability to predict CRs on a monthly basis after an SRGM selection. Table 7.9 shows the predictive ability of the selected models. The first column shows the week in which a model was selected for each stage, the next column shows the actual number of CRs in that week. The "Prediction" column specifies if the row shows the predicted number of CRs or the Relative Error (RE) of the prediction in that month. The

Week	Actual CRs	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
271	9	CRs	10	11	12	13	14	16
		RE	0.11	0.22	0.20	0.00	0.00	-0.06
336	46	CRs	44	48	53	57	62	67
		RE	-0.21	-0.17	-0.23	-0.27	-0.23	-0.26
380	148	CRs	107	115	124	133	143	153
		RE	-0.36	-0.35	-0.34	-0.32	-0.31	-0.29
429	277	CRs	246	261	278	295	313	332
		RE	-0.16	-0.15	-0.13	-0.10	-0.09	-0.07

Table (7.9) CR Predictions and Relative Errors for six months into the Future using Approach 1 for Release 3

last column has six sub-columns. Each column shows the number of predicted CRs and the RE value for the specific month.

The Modified Gompertz model was selected in week 271. After one month of the model selection we find that the model provides a prediction with a Relative Error (RE) of 0.11. After two months, the RE value doubles to 0.22. After three months the RE value decreases to 0.2. After four months and five months the RE is zero, which means that the actual matches the predicted value. After six months of the model selection the model begins to under-predict CR values, and RE is -0.06.

The following rows in the table show CR prediction of this model after each change-point in the weeks where different models were selected using the multi-stage model. We included these predictions in those weeks in this table for comparison purposes. We would like to analyze and compare the performance of Approach 1 compared to the other two approaches. In general, we find that the RE value becomes more and more negative, an indicator that the prediction is poor as time keeps progressing in Stage 3 and Stage 4. By Stage 5, RE values range from -0.07 to -0.16.

When we look at the predicted values for Approach 2 in Table 7.10 we find that the prediction ability of the models is much better than the previous approach. In

Week	Actual CRs	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
271	9	CRs	10	11	12	13	14	16
		RE	0.11	0.22	0.20	0.00	0.00	-0.06
336	46	CRs	50	54	69	78	81	90
		RE	-0.11	-0.07	-0.14	-0.18	-0.15	-0.17
380	148	CRs	164	181	199	218	240	263
		RE	-0.02	0.02	0.05	0.11	0.17	0.22
429	277	CRs	288	298	309	320	331	343
		RE	-0.02	-0.03	-0.03	-0.03	-0.04	-0.04

Table (7.10) CR Predictions and Relative Errors for six months into the Future using Approach 2 for Release 3

Week	Actual CRs	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
271	9	CRs	10	11	12	13	14	16
		RE	0.11	0.22	0.20	0.00	0.00	-0.06
336	46	CRs	51	56	62	68	75	82
		RE	-0.09	-0.03	-0.10	-0.13	-0.07	-0.09
380	148	CRs	160	175	193	211	232	254
		RE	-0.04	0.00	0.04	0.07	0.16	0.19
429	277	CRs	288	298	309	320	332	344
		RE	-0.02	-0.03	-0.03	-0.03	-0.03	-0.03

Table (7.11) CR Predictions and Relative Errors for six months into the Future using Approach 3 for Release 3

week 336, the Gompertz model was selected to predict CRs in the future. Although this model the REs are negative, which means that the model under-predicts CRs, the RE is lower than for Approach 1 and closer to the actual values. In week 380, the Yamada model was selected. Looking at the predictions and RE values of this model compared to the same weeks using Approach 1, we find that this model performs a lot better. In fact this model stops under-prediction from the second month on. This is also true for the fourth model selected for the last stage, where the RE value are even smaller and closer to the actual values, although they tend to under-predict CR numbers. These values provide an improvement in prediction ability compared to the predictions of Approach 1, which seems to diverge drastically as time progresses.

Table 7.11 shows the month-to-month predictions for Approach 3 application. The first row is the same as Approach 1 for the first stage. After week 336 the RE values range from-0.03 to -013. Comparing the the RE values for this stage of Approach 2 vs. Approach 3 we find that RE values of Approach 3 has lower RE values. A month after week 380 we find that the RE value for Approach 3, (-0.04), is worse than RE value in Approach 2, (-0.02). Unlike the first month RE values of this release, Approach 3 for the remaining months are better than Approach 2. After week 277 we find the month-to-month predictions for both approaches are equal until +4 months, the RE values of Approach 3 are better than Approach 2 for the predictions after five months and after six months.

Notice that we highlight Table 7.10 and Table 7.11 with green when the RE value of an approach is better than the RE value of the other approach and highlighted them with red if the RE value is worse and were left without highlight when they are equal. This color scheme only highlights RE tables for Approach 2 and 3 since their results are comparable. The RE values in Table 7.9 for Approach 1 are highlighted since they are always greater than the RE values of the other approaches.

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
114	5	5	0.88	6	0.54	5	0.89	5	0.89	5	0.89
115	5	5	0.88	6	0.55	5	0.91	5	0.91	5	0.89

Table (7.12) SRGM Estimation for Stage 1 the GOF value for Release 2: Approach 1

7.3 Release 2

This release is 433 weeks long with 898 CRs. When applying the change-point estimation we found that four change-points were estimated in weeks 225, 247, 280, and 300. We can then divide this release into 5 stages:

- Stage 1: From week 1 to week 224
- Stage 2: From week 225 to week 246
- Stage 3: From week 247 to week 279
- Stage 4: From week 280 to week 299
- Stage 5: From week 300 to week 433

As we did before start SRGM selection after 10 weeks of CRs. Whenever we find a fitted model we use that model for future CR prediction from that point forward. We compare the predictive ability of each model by calculating the RE of month-to-month prediction for six months into the future.

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
235	132	108	0.15	110	0.28	127	0.887	127	0.87	117	0.6
236	136	111	0.16	113	0.29	132	0.9	132	0.89	122	0.63
237	140	113	0.16	113	0.3	138	0.92	137	0.91	126	0.65
238	142	116	0.17	117	0.31	143	0.93	143	0.93	130	0.68

Table (7.13) SRGM Estimation for Stage 2 the GOF value for Release 2: Approach 2

7.3.1 Model Estimation

7.3.1.1 Approach 1

For this approach we do not consider change-point data. Table 7.12 shows the progression of fitting the five SRGMs to the data once 5 CRs were collected. By week 115, The Gompertz model and the Modified Gompertz model has an R^2 of 0.91 and an estimated value of CRs that is greater than or equal the actual number of CRs. In this case, both models are the same since the parameter d in the modified Gompertz model is zero, reducing it to the Gompertz model.

7.3.1.2 Approach 2

After the first change-point occurrence we need to start fitting a different SRGM that fits the new stage. From week 255 on more cumulative CR data is collected and then model fitting is performed. Table 7.13 shows that the Gompertz model has an R^2 that meets the minimum threshold since week 236 but it does not meet the full criteria to be selected as a model until week 238, when the estimated value is greater than or equal the actual value. This is true for the Modified Gompertz as well, since in week 237 the Modified Gompertz has an acceptable R^2 value but the estimated number of CRs is less than the actual value. By week 238, both the

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
257	259	229	0.24	233	0.46	258	0.98	257	0.98	254	0.97
258	268	233	0.24	238	0.45	266	0.99	264	0.99	262	0.97
259	275	237	0.24	243	0.45	273	0.99	272	0.99	269	0.97
260	277	241	0.25	247	0.46	281	0.99	279	0.99	276	0.98

Table (7.14) SRGM Estimation for Stage 3 the GOF value for Release 2: Approach 2

Gompertz and the Modified Gompertz are suitable for selection. The Gompertz model was selected for this stage.

Starting in week 257, the R^2 values are 0.98 for the Gompertz and the Modified Gompertz and 0.97 for the Yamada model for Stage 3, as shown in Table 7.13. But the estimated numbers of CRs were all less than the actual number of CRs for that week. Therefore these models are not selected and more data is collected. By week 260, both the Gompertz model and the modified Gompertz model meets the acceptance criteria. The Gompertz model was then selected for predicting future CRs for this stage.

In Stage 4, the Gompertz model and the Yamada model meet the acceptance criteria in week 290. They both have an R^2 value of 0.98 and the estimated number of CRs is greater than or equal to the actual number of CRs, (see Table 7.15). Selecting either of them is suitable. The Yamada model was selected for this stage.

For the final stage, we find that all models except the G-O model meet the acceptance criteria. Table 7.16 shows that by week 310 the R^2 of four of the SRGMs exceeds the threshold of 0.9, and the estimated number of CRs is greater than or equal to the actual number of CRs. The Gompertz model was selected for CR prediction.

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
290	596	553	0.29	563	0.54	599	0.98	588	0.98	601	0.98

Table (7.15) SRGM Estimation for Stage 4 the GOF value for Release 2: Approach 2

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
310	693	691	0.85	697	0.96	697	0.96	697	0.96	696	0.95

Table (7.16) SRGM Estimation for Stage 5 the GOF value for Release 2: Approach 2

For this release curve-fitting was successful for every stage, so we were able to fit five different models for the five stages.

7.3.1.3 Approach 3

Using Approach 2, SRGMs are selected for each stage. Approach 3 uses the selected model for each stage after a change-point occurrence and applies TT calculations to find a model using a new version of the time.

We start applying TT from the second stage of the release. Using the models chosen for the first stage we calculate a newly transformed time value in weeks then we use this new time in the new SRGM selected for the current stage. For the second stage, $M_1(t)$ is the Gompertz model selected for the first stage and $M_2(t)$ is the Gompertz model selected for the second stage. Table 7.17 shows the values of the estimated CRs using $M_1(t)$ and $M_2(t)$ and then the value of CRs for $M(t)$ using the new time after transformation (TT) for stage 2. Table 7.18 shows the TT values and the estimates of the new model for Stage 3. Table 7.19 shows the TT

(Actual Week	**Actual CRs**	$M_1(t)$ **Estimation**	$M_2(t)$ **Estimation**	**Week after TT**	$M(t)$ **Estimation**
225	75	21	87	221	91
226	79	21	90	222	95
227	85	21	94	223	99
228	98	21	98	224	103
229	105	22	101	225	107
230	108	22	105	226	111
231	113	22	110	227	116
232	116	22	114	228	120
233	116	23	118	229	125
234	130	23	123	230	130
235	132	23	128	231	135
236	136	23	133	232	141
237	140	24	138	233	146
238	142	24	143	234	152

Table (7.17) CR estimation with TT for Stage 2 of Release 2

(Actual Week	**Actual CRs**	$M_1(t)$ **Estimation**	$M_2(t)$ **Estimation**	**Week after TT**	$M(t)$ **Estimation**
247	191	213	192	247	197
248	198	220	198	248	203
249	204	229	204	249	209
250	208	237	211	250	216
251	217	246	217	251	222
252	226	255	223	252	228
253	233	264	230	253	235
254	238	274	237	254	242
255	242	284	244	255	249
256	248	294	251	256	256
257	259	304	259	257	264
258	268	315	266	258	271
259	275	326	274	259	279
260	277	338	282	260	287

Table (7.18) CR estimation with TT for Stage 3 of Release 2

values and the estimates of the new model for Stage 4. Table 7.18 shows the TT values and the estimates of the new model for Stage 5.

7.3.2 CR Prediction

To compare the predictive ability of the three approaches we compare their ability to predict CRs on a monthly basis after model selection. Table 7.21 shows the predictive ability of the Gompertz model month-to-month after the model was selected in week 122. The predicted number of CRs matches the actual number after a month of selecting the model, since the relative error value (RE) is zero. After two

(Actual Week	**Actual CRs**	$M_1(t)$ **Estimation**	$M_2(t)$ **Estimation**	**Week after TT**	$M(t)$ **Estimation**
280	486	494	490	280	500
281	496	507	500	281	510
282	510	521	510	282	519
283	523	534	521	283	528
284	530	548	532	284	538
285	536	563	543	285	548
286	557	577	554	286	557
287	566	592	566	287	567
288	587	608	577	288	578
289	590	623	589	289	588
290	596	639	601	290	598
291	602	656	613	291	609

Table (7.19) CR estimation with TT for Stage 4 of Release 2

(Actual Week	**Actual CRs**	$M_1(t)$ **Estimation**	$M_2(t)$ **Estimation**	**Week after TT**	$M(t)$ **Estimation**
300	662	711	664	299	669
301	665	723	667	300	672
302	669	735	670	301	675
303	675	748	674	302	679
304	679	760	677	303	682
305	681	773	680	304	685
306	686	786	684	305	688
307	689	799	687	306	692
308	692	812	690	307	695
309	692	826	694	308	698
310	693	839	697	309	701

Table (7.20) CR estimation with TT for Stage 5 of Release 2

months the model under-predicts the number of CRs where RE is -0.14. The model continues to give accurate predictions after three months and after four months of selection, since RE is zero. After change-points, we find that the RE value for the months following the weeks of 238, 260, 290 and 310 keeps decreasing. In fact the predicted number of CRs is smaller than the actual number of CRs. This shows that the quality of the prediction gets worse as time progresses in the release.

Table 7.22, shows the RE values of predictions using Approach 2. The RE values highlighted in green are lower than the RE values of the same month when Approach 3 is applied. The RE values highlighted in red, are the months where Approach 2 predictions where worse than Approach 3, this also applies to Table 7.11. We find that the prediction ability of the models is much better than the prediction of Approach 1. In week 238, the Gompertz model was selected to predict CRs in the future. We find that model to have RE values ranging from 0.08 to 0.15. These RE values provide more accurate predictions than the model after TT in Approach 3, Table 7.11. In Stage 3, the Gompertz model provides predictions with lower RE values after 1 month and after 2 months of prediction than the model after TT. After TT, the model has smaller RE values for the remaining months. In fact the model before TT shows under-predicted CRs unlike the predictions of the model after TT was performed. After week 290, we find that the Yamada model provides better predictions using TT in general. Finally, the last stage predictions for both approaches are mostly equal or have minor differences. This comparison between the RE values of Approach 2 in Table 7.22 and Approach 3 in Table 7.23 shows that there is not one approach that is always superior to the other in terms of prediction ability.

We then look into the results of CR prediction for this release using approach 3, Table 7.34. The first row is not different than the model selected for Approach 1

Week	Actual CRs	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
122	6	CRs	6	6	7	7	7	8
		RE	0.00	-0.14	0.00	0.00	-0.13	0.00
238	142	CRs	27	28	29	31	32	34
		RE	-0.82	-0.84	-0.86	-0.87	-0.88	-0.88
260	277	CRs	35	36	38	40	42	44
		RE	-0.88	-0.90	-0.90	-0.91	-0.91	-0.92
290	596	CRs	49	51	54	56	59	62
		RE	-0.92	-0.92	-0.92	-0.92	-0.91	-0.91
310	693	CRs	62	64	67	70	74	77
		RE	-0.91	-0.91	-0.90	-0.90	-0.90	-0.89

Table (7.21) CR Predictions and Relative Errors for six months into the Future using Approach 1 for Release 2

Week	Actual CRs	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
122	6	CRs	6	6	7	7	7	8
		RE	0.00	-0.14	0.00	0.00	-0.13	0.00
238	142	CRs	165	191	220	252	289	329
		RE	0.08	0.07	0.06	0.06	0.08	0.15
260	277	CRs	309	344	381	421	465	512
		RE	0.03	0.00	-0.02	-0.01	-0.04	-0.03
290	596	CRs	651	705	762	822	887	955
		RE	0.03	0.08	0.14	0.20	0.28	0.38
310	693	CRs	711	725	739	753	768	783
		RE	0.03	0.04	0.05	0.07	0.08	0.09

Table (7.22) CR Predictions and Relative Errors for six months into the Future using Approach 2 for Release 2

Week	Actual CRs	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
122	6	CRs	6	6	7	7	7	8
		RE	0.00	-0.14	0.00	0.00	-0.13	0.00
238	142	CRs	177	205	237	274	315	362
		RE	0.16	0.15	0.14	0.15	0.18	0.27
260	277	CRs	321	358	399	445	494	548
		RE	0.07	0.04	0.03	0.05	0.02	0.03
290	596	CRs	653	699	748	799	853	910
		RE	0.03	0.06	0.11	0.16	0.23	0.31
310	693	CRs	715	728	742	756	770	785
		RE	0.03	0.04	0.06	0.07	0.08	0.10

Table (7.23) CR Predictions and Relative Errors for six months into the Future using Approach 3 for Release 2

and 2 for the first stage, therefore there are no differences to be discussed. After the following two stages we find that the RE values for Approach 3 are much better than RE for the same months of Approach 1 but worse than the RE values of Approach 2. To better compare Approach 2 and Approach 3 since their results are comparable we highlighted the weeks when one of these approaches performed better in green and the approach that performed poorly with red. We find that Approach 3 performed worse than Approach 2 in two stages after change-points and the two approaches performed equally well for the last stage. Therefore, we can say that the results of Approach 2 were better than Approach 3 for this release.

7.4 Release 1

This release covers 554 weeks with 486 CRs. When applying the change-point estimation we found that four change-points were estimated in weeks 390, 410, 437 and 454. We can then divide this release into 5 stages:

- Stage 1: From week 1 to week 389
- Stage 2: From week 390 to week 409
- Stage 3: From week 410 to week 436
- Stage 4: From week 437 to week 453
- Stage 5: From week 454 to week 554

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
332	5	2	0.24	3	0.09	3	0.52	4	0.92	3	0.51
333	5	2	0.25	3	0.14	3	0.52	4	0.92	3	0.52
334	5	2	0.27	3	0.18	3	0.53	4	0.92	3	0.52
335	5	2	0.28	3	0.21	3	0.53	5	0.93	3	0.53

Table (7.24) SRGM Estimation for Stage 1 the GOF value for Release 1: Approach 1

7.4.1 Model Estimation

7.4.1.1 Approach 1

Table 7.24 shows the progression of fitting the five SRGMs to the data once 5 CRs were collected. By week 335, the Modified Gompertz model has an R^2 of 0.93 and an estimated number of CRs that is greater than or equal the actual number of CRs. Therefore, this model is used for CR prediction using this approach.

7.4.1.2 Approach 2

After the first change-point occurrence in week 390 we collect data for ten more weeks of cumulative CRs then we apply the SRGM selection process weekly until we find a fit model. From week 399 we start examining the Goodness-of-fit and of each model, as shown in Table 7.25. In Stage 2 we kept repeating the process of collecting data and trying to fit a model until we reached the end of the stage, where a new change-point is introduced. No SRGM was selected for this stage.

For stage 3, we find that the Gompertz model has the highest R^2 value since week 419 but the estimated number of cumulative CRs is smaller than the actual number of cumulative CRs. This continues until week 424 where the estimated value is equal to the actual value (Table 7.26).

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
399	69	60	0.12	61	0.23	66	0.81	66	0.81	64	0.63
400	70	61	0.13	62	0.25	68	0.85	68	0.84	66	0.67
401	74	62	0.13	63	0.25	70	0.86	70	0.8	68	0.69
402	80	64	0.12	65	0.24	73	0.8	73	0.85	70	0.67
403	83	66	0.12	67	0.24	76	0.86	76	0.86	72	0.66
404	94	68	0.11	69	0.21	79	0.81	79	0.8	75	0.6
405	98	70	0.1	71	0.2	83	0.79	83	0.79	78	0.6
406	108	72	0.09	74	0.19	87	0.76	87	0.75	82	0.57
407	114	75	0.09	77	0.18	91	0.74	91	0.74	86	0.55
408	122	78	0.09	80	0.17	96	0.73	96	0.73	89	0.54
409	128	80	0.08	83	0.17	101	0.72	101	0.72	94	0.53

Table (7.25) SRGM Estimation for Stage 2 the GOF value for Release 1: Approach 2

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
419	180	159	0.15	161	0.3	174	0.94	171	0.88	166	0.66
420	192	162	0.14	164	0.27	180	0.91	177	0.83	171	0.61
421	194	165	0.14	168	0.27	185	0.91	181	0.83	175	0.61
422	194	168	0.14	171	0.28	190	0.92	190	0.92	178	0.62
423	198	171	0.15	173	0.29	195	0.94	195	0.93	182	0.64
424	199	173	0.15	176	0.3	199	0.94	194	0.88	185	0.66

Table (7.26) SRGM Estimation for Stage 3 the GOF value for Release 1: Approach 2

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
446	416	387	0.2	391	0.37	413	0.94	414	0.95	400	0.69
447	417	391	0.21	395	0.39	420	0.95	420	0.95	405	0.71

Table (7.27) SRGM Estimation for Stage 4 the GOF value for Release 1: Approach 2

Table (7.28) SRGM Estimation for Stage 5 the GOF value for Release 1: Approach 2

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
463	426	427	0.39	427	0.39	427	0.38	427	0.39	427	0.38
464	426	427	0.36	427	0.36	427	0.35	427	0.36	427	0.35
465	426	427	0.33	427	0.33	427	0.32	427	0.34	427	0.32
466	426	427	0.31	427	0.31	427	0.3	427	0.32	427	0.3
467	426	427	0.29	427	0.29	427	0.28	426	0.3	427	0.28
468	426	427	0.27	427	0.27	427	0.26	427	0.28	427	0.26
469	426	426	0.26	426	0.26	426	0.25	426	0.27	426	0.25
470	434	428	0.32	428	0.32	428	0.32	428	0.32	428	0.32
471	435	430	0.42	430	0.41	430	0.42	430	0.42	430	0.42
472	435	431	0.5	431	0.5	431	0.51	431	0.51	432	0.5
473	436	432	0.57	432	0.57	432	0.58	432	0.58	432	0.58
474	436	433	0.63	433	0.63	433	0.63	433	0.63	434	0.63
475	437	434	0.68	434	0.67	435	0.68	435	0.68	435	0.68
476	437	435	0.71	435	0.71	435	0.72	435	0.72	436	0.72
477	437	436	0.74	436	0.74	436	0.75	437	0.77	436	0.75
478	437	437	0.77	437	0.76	437	0.77	437	0.79	437	0.77
479	439	438	0.79	438	0.79	438	0.8	438	0.81	438	0.79
480	439	438	0.81	438	0.81	438	0.82	439	0.83	438	0.82

Continued on next page

Table 7.28 – continued from previous page

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
481	439	439	0.83	439	0.83	439	0.83	440	0.85	439	0.83
482	439	440	0.84	439	0.84	440	0.84	440	0.85	440	0.84
483	440	440	0.85	440	0.85	440	0.76	441	0.86	440	0.86
484	440	441	0.86	441	0.86	441	0.86	441	0.87	441	0.86
485	441	441	0.87	441	0.87	441	0.87	442	0.88	442	0.87
486	443	442	0.88	442	0.87	442	0.89	443	0.88	442	0.89
487	449	443	0.88	443	0.88	443	0.88	444	0.89	444	0.88
488	451	445	0.88	445	0.87	445	0.88	440	0.89	445	0.88
489	452	446	0.88	446	0.88	446	0.88	434	0.9	446	0.88
490	454	448	0.88	447	0.88	448	0.88	449	0.9	448	0.88
491	454	449	0.88	449	0.88	449	0.89	450	0.91	449	0.89
492	455	450	0.89	450	0.88	451	0.9	452	0.91	450	0.9
493	455	451	0.9	451	0.89	452	0.9	453	0.92	452	0.9
494	457	453	0.9	453	0.9	454	0.91	454	0.93	453	0.91
495	463	454	0.9	454	0.9	455	0.9	456	0.93	455	0.9
496	464	456	0.9	456	0.9	458	0.9	458	0.93	456	0.9
497	473	458	0.88	458	0.88	460	0.89	461	0.92	459	0.89
498	473	459	0.88	460	0.87	461	0.88	463	0.91	461	0.88
499	475	461	0.87	462	0.87	463	0.88	465	0.91	463	0.88

Continued on next page

Table 7.28 – continued from previous page

Week No.	No. of CRs	G-O		DSS		Gompertz		M Gompertz		Yamada	
		Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2	Est.	R^2
500	475	462	0.87	465	0.87	465	0.89	467	0.92	465	0.89
501	475	463	0.87	466	0.88	468	0.89	469	0.92	467	0.89
502	475	464	0.87	468	0.89	469	0.9	471	0.93	469	0.9
503	475	465	0.87	470	0.89	471	0.9	473	0.93	470	0.9
504	475	466	0.88	471	0.9	472	0.91	474	0.94	472	0.91
505	475	468	0.88	472	0.9	473	0.91	476	0.94	473	0.91

In Stage 4, both Gompertz and Modified Gompertz had an R^2 value of 0.95 and an estimated value of 420 (see Table 7.27). Since both models were equal, we can selecting either of them. The Gompertz model was selected for this stage.

Table 7.28 shows that by week 505, four of the SRGMs show an R^2 that exceeds the threshold of 0.9, but only the Modified Gompertz model has an estimated number of CRs that is greater than or equal to the actual number of CRs. Therefore, for this release four SRGMs were selected for the first, third, fourth and fifth stage. The selection process failed to find a fit SRGM for Stage 2.

7.4.1.3 Approach 3

Using Approach 2, SRGMs were selected for each stage while Approach 3 uses the selected model for each stage and applies Time Transformation calculations. For

(Actual Week	**Actual CRs**	$M_1(t)$ **Estimation**	$M_2(t)$ **Estimation**	**Week after TT**	$M(t)$ **Estimation**
410	139	26	143	409	145
411	141	26	147	410	148
412	145	27	150	411	151
413	148	27	154	412	155
414	157	28	158	413	158
415	160	29	162	414	161
416	162	29	165	415	165
417	166	30	169	416	168
418	173	30	173	417	172
419	180	31	177	418	176
420	192	32	182	419	180
421	194	33	186	420	183
422	194	33	190	421	187
423	198	34	195	422	191
424	199	35	199	423	195

Table (7.29) CR estimation with TT for Stage 3 of Release 1

(Actual Week	**Actual CRs**	$M_1(t)$ **Estimation**	$M_2(t)$ **Estimation**	**Week after TT**	$M(t)$ **Estimation**
437	342	255	354	435	352
438	356	260	360	436	358
439	366	266	367	437	365
440	374	271	373	438	372
441	378	277	379	439	378
442	385	282	386	440	385
443	396	288	392	441	393
444	407	294	399	442	400
445	411	300	406	443	407
446	416	306	413	444	415
447	417	312	420	445	422

Table (7.30) CR estimation with TT for Stage 4 of Release 1

the first and second stage no TT is required, since they are using the SRGM of the first stage and no new SRGM is selected after the change. Using the models chosen for the first two stages, we calculate a newly transformed time value in weeks then we use this new time in the new SRGM selected for the current stage. For the third stage, Table 7.29 shows the values of the estimated CRs using $M_1(t)$ and $M_2(t)$ and then the value of CRs for $M(t)$ using the new time after transformation (TT). Table 7.30 shows estimated values for Stage 4 and Table 7.31 shows estimated values of Stage 5.

(Actual Week	**Actual CRs**	$M_1(t)$ **Estimation**	$M_2(t)$ **Estimation**	**Week after TT**	$M(t)$ **Estimation**
496	464	976	463	501	463
497	473	992	464	502	464
498	473	1008	466	503	465
499	475	1025	467	504	467
500	475	1041	468	505	468
501	475	1058	470	506	470
502	475	1075	471	507	471
503	475	1092	473	508	473
504	475	1110	474	509	474
505	475	1127	476	510	476

Table (7.31) CR estimation with TT for Stage 5 of Release 1

7.4.2 CR Prediction

To compare the predictive ability of the three approaches we compare their ability to predict CRs on a monthly basis after model selection. Table 7.32 shows the predictive ability of the Modified Gompertz model month-to-month after the model was selected in week 335. The predicted number of CRs matches the actual number after a month of selecting the model, since the relative error value (RE) is zero. After two months the model under-predicts the number of CRs where RE is -0.17. The model continues to give accurate predictions after 3 months and after four months of selection since RE is zero. The prediction ability decreases after 5 months when RE is -0.22 and is worse with an RE of -0.33 by the sixth month. The next rows in Table 7.32 show CR predictions after each change-point. We included these predictions on those weeks on this table for comparison purposes. We would like to analyze and compare the performance of Approach 1 compared to the other two approaches. In general, we find that the RE value keeps getting worse and the quality of the prediction is poor as time keeps progressing.

When we look at the predicted number of CRs for Approach 2 in Table 7.33 we find that the prediction ability of the models is much better than the previous approach. In week 424, the Gompertz model was selected to predict CRs in the

future. Although this model has a negative RE, which means that it keeps under-predicting CRs, the error rate is lower than RE in Approach 1 and closer to the actual values. In week 447, the Gompertz model was selected. Looking at the predictions and RE values of this model compared to the same weeks using Approach 1, we find that this model performs a lot better. The predicted values are closer to the actual values and the RE values are lower. This is also true for the fourth model selected for the last stage, where the RE values are even smaller and closer to the actual values. These values provide great improvement in prediction ability compared to the predictions of Approach 1, which seems to diverge drastically as time progresses.

We then look into the results of CR prediction for this release using approach 3, Table 7.34. The first row is not different than the model selected for Approach 1 and 2 for the first stage, therefore there are no differences to be discussed. Afterwards we find that the RE values for Approach 3 are much better than RE for the same months of Approach 1 but worse than the RE values of Approach 2. To better compare Approach 2 and Approach 3 since their results are comparable we highlighted the weeks when one of these approaches performed better in green and the approach that performed poorly with red. We find that Approach 3 performed worse than Approach 2 in two stages after change-points and the two approaches performed equally on the last stage. Therefore, we can say that the results of Approach 2 were better than Approach 3 for this release.

7.5 Discussion

When we applied the SRGM estimation using a curve-fitting method and CR prediction on a single release back in Chapter 5, we had the chance to observe and compare prediction results for a single release. Although the initial results concluded

Week	Actual CRs	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
335	5	CRs	5	5	6	7	7	8
		RE	0.00	-0.17	0.00	0.00	-0.22	-0.33
424	199	CRs	38	41	45	49	53	58
		RE	-0.83	-0.84	-0.86	-0.87	-0.87	-0.86
447	417	CRs	62	67	73	79	86	93
		RE	-0.85	-0.84	-0.83	-0.81	-0.80	-0.79
507	476	CRs	191	205	220	236	253	271
		RE	-0.60	-0.57	-0.54	-0.51	-0.48	-0.44

Table (7.32) CR Predictions and Relative Errors for six months into the Future using Approach 1 for Release 1

Week	Actual CRs	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
335	5	CRs	5	5	6	7	7	8
		RE	0.00	-0.17	0.00	0.00	-0.22	-0.33
424	199	CRs	218	239	262	286	312	340
		RE	-0.02	-0.09	-0.20	-0.24	-0.23	-0.19
447	417	CRs	449	480	513	547	584	623
		RE	0.07	0.13	0.20	0.28	0.37	0.43
507	476	CRs	482	489	496	503	511	519
		RE	0.01	0.02	0.03	0.05	0.06	0.07

Table (7.33) CR Predictions and Relative Errors for six months into the Future using Approach 2 for Release 1

Week	Actual CRs	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
335	5	CRs	5	5	6	7	7	8
		RE	0.00	-0.17	0.00	0.00	-0.22	-0.33
424	199	CRs	212	231	250	271	294	318
		RE	-0.05	-0.12	-0.23	-0.28	-0.28	-0.24
447	417	CRs	454	487	523	561	602	644
		RE	0.08	0.15	0.23	0.32	0.41	0.48
507	476	CRs	487	492	498	504	513	521
		RE	0.01	0.02	0.03	0.05	0.06	0.07

Table (7.34) CR Predictions and Relative Errors for six months into the Future using Approach 3 for Release 1

that Approach 3 performs better than Approaches 1 and 2 in CR prediction, we could not generalize our findings without applying those approaches to several releases. Therefore we applied the three approaches to the three remaining releases of the system. We applied all three curve-fitting approaches described in Sections 5.3.1, 5.3.2 and 5.3.3 to select models, and then used the selected models to predict future CRs. To answer the the research questions asked at the beginning of the chapter:

- RQ1: Can we generalize the use to the three approaches for curve-fitting on other releases in a multi-release system?

  Regarding Approach 1, which is the curve-fitting approach that does not consider change: what we have observed in the four releases, it is applicable and provides CR predictions with low RE in many cases for up to six months to the future see Tables 7.32, 7.21 and 7.9. This approach performs well if no change-points were introduced. After the change-points, and as the time of the release progresses, this model starts to provide poor predictions. In many cases, the model diverges from the actual cumulative CR curve.

  The second approach, the multi-stage model, selects different models after the introduction of each change-point. This approach has an advantage over Approach 1, as it provides better accuracy of cumulative CR prediction especially after change-points. On the other hand, we noticed that this approach failed several times to find a suitable model for a certain stage after change, which reduces our chances of making any predictions for that stage. This happened in the second stage of Release 1 and the second stage of Release 3. When we are not able to make predictions for these stages, it deprives us from understanding our system's behavior for a period of time. When using the first approach we would have used the initial model selected regardless of the stage

even if these predictions carry a higher error rate. If approach 3 had been used, it would still use the model before change until a new model is selected to be later processed. Notice that these problems occurred in stages where change-points happened close together. In the first release the two change-points were 20 weeks apart, while in Release 3 the two change-points where 35 weeks apart.

- RQ2: Can we generalize our findings that the TT approach performs better than the other curve-fitting approaches?

  To make a generalization regarding an approach's prediction ability, we need to find a pattern of repeated behavior to make a conclusive statement. When Approach 1 is applied to an evolving system we find that it has relatively low RE values before evolution occurs. The ranges of RE per release for the first month-to-month predictions are:

  - Release 1: -0.33 to zero
  - Release 2: -0.14 to zero
  - Release 3: -0.06 to 0.22
  - Release 4: -0.13 to 0.11

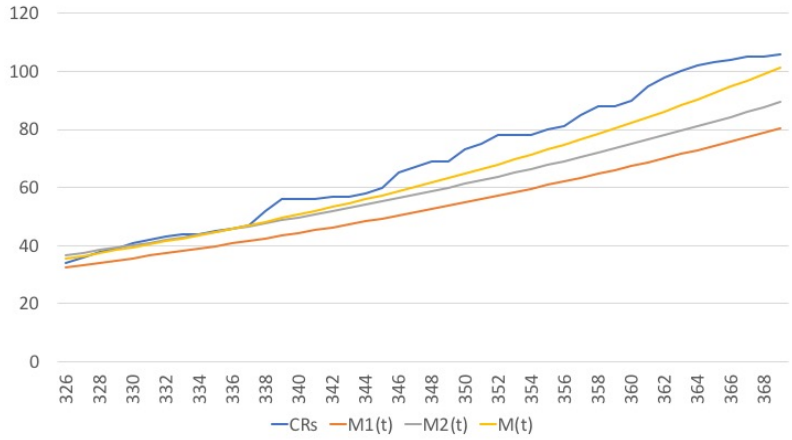
  These RE values increase or decrease dramatically after change-points. In all four releases we find that Approach 1 has higher RE values than both Approach 2 and Approach 3 after change-points. Regarding Approach 2, we find that the predictive ability of this approach is higher than Approach 1, since the RE values is usually lower. It also provides predictions with smaller RE values in many stages than the RE values of predictions provided by Approach 3, such as for the second stage of Release 2 or predictions for the second and third stages

of Release 1. On the other hand, Approach 3 provided prediction with better RE values in some other cases. Approach 3 was better than Approach 2 for most of the predictions for Releases 3 and 4. It also provided better month-to-month predictions for Release 2. But for Release 1 we found that in some stages Approach 3 predictions had equal RE values to Approach 2 or even worse than Approach 2. Therefore, it is difficult to generalize findings regarding the superiority of one approach over the other with regards to Approach 2 and Approach 3.

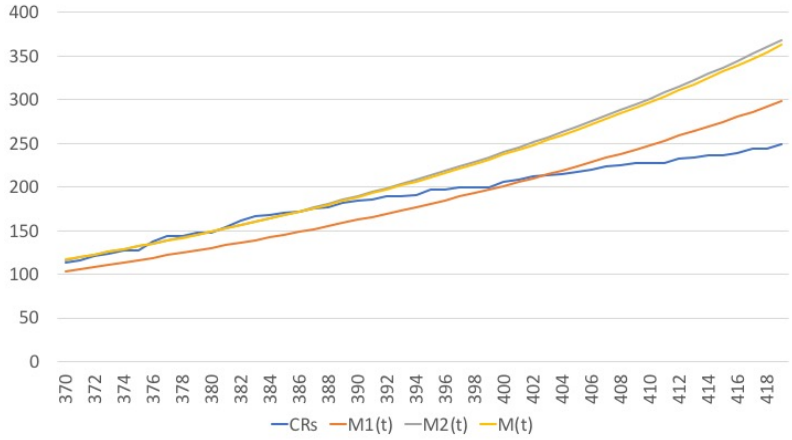
Using Time Transformation (TT) should provide an improvement to the prediction by adjusting the time of the model. When TT was applied to the third release, it provided great improvement to CR prediction by reducing the RE. Figure 7.1 shows how $M(t)$ is closer to the actual CRs for Stage 3 from $M_1(t)$ which is the model selected for the first stage using Approach 1 and $M_2(t)$ which is the curve selected for Stage 3 using Approach 2. In Stages 4 and 5, the $M(t)$ provides very slight improvement over $M_2(t)$ with being closer to the actual curve.

For the second release (Figure 7.2), we find that in general $M(t)$ and $M_2(t)$ provide very predicted values that are very close. The charts do not show one model providing great improvement over the other.

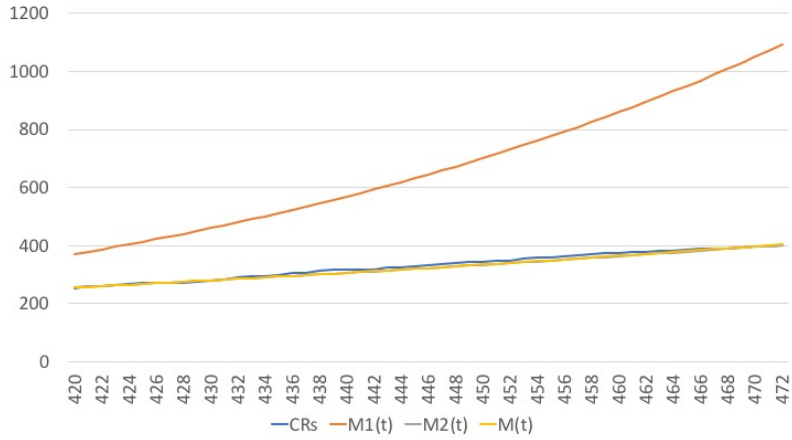
Figure 7.3 shows the TT models for Release 1. We can see that in Figure 7.3a $M(t)$ is close to $M_2(t)$ but is slightly shifted under the curve for the actual cumulative CRs. This causes the model to underestimate future CRs. This is due to the effect of using both $M_1(t)$ and $M_2(t)$ which cause $M(t)$ to be between the two models. $M_1(t)$ is the model selected by Approach 1, which eventually fell below the CR curve after the change-point. In the following stage, the models were closer to the curve



(a) $M_1(t)$, $M_2(t)$ and $M(t)$ in Stage 3

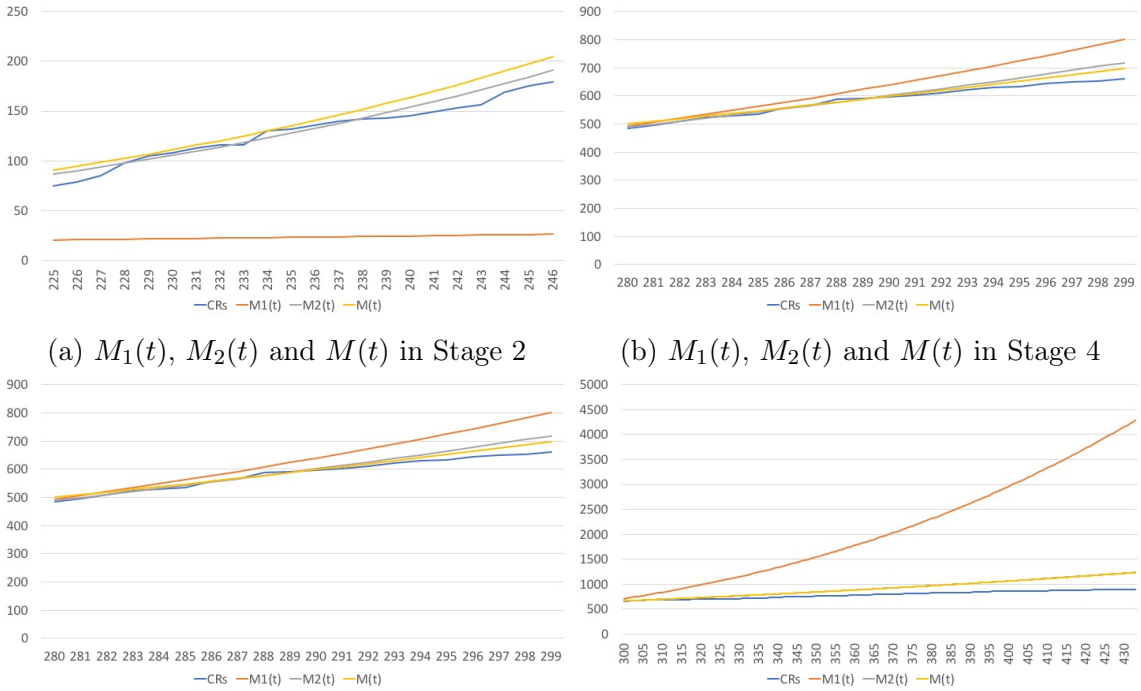


(b) $M_1(t)$, $M_2(t)$ and $M(t)$ in Stage 4



(c) $M_1(t)$, $M_2(t)$ and $M(t)$ in Stage 5

Figure (7.1) TT models per stage for Release 3



(a) $M_1(t)$, $M_2(t)$ and $M(t)$ in Stage 2

(b) $M_1(t)$, $M_2(t)$ and $M(t)$ in Stage 4

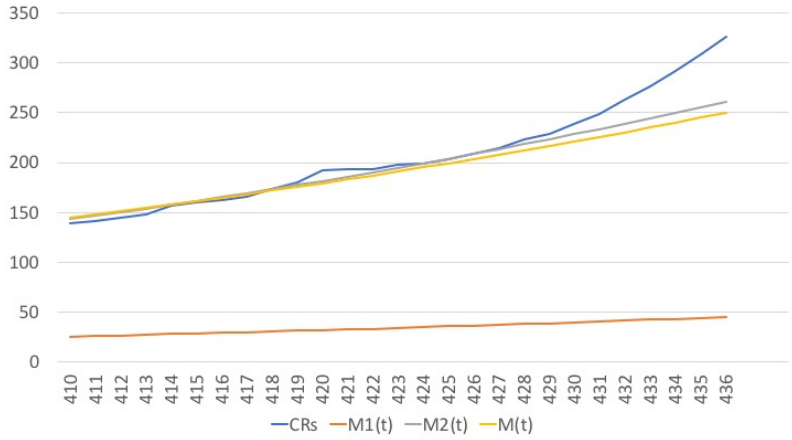
(c) $M_1(t)$, $M_2(t)$ and $M(t)$ in Stage 4

(d) $M_1(t)$, $M_2(t)$ and $M(t)$ in Stage 5

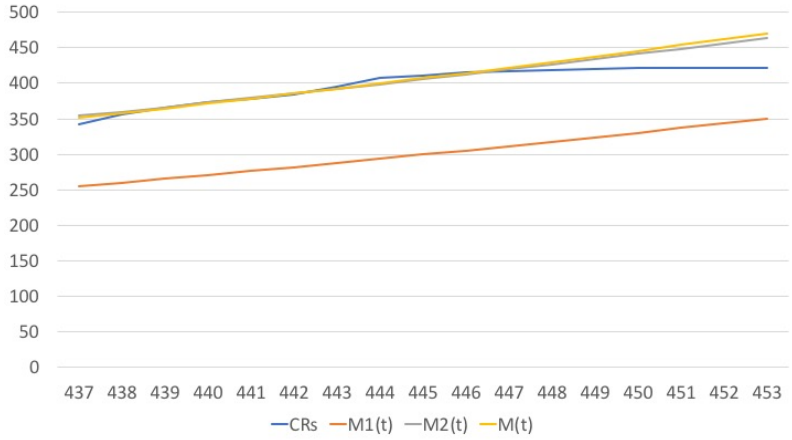
Figure (7.2) TT models per stage for Release 2

for the actual cumulative CRs, which brings $M(t)$ closer to it and provides more accurate predictions.

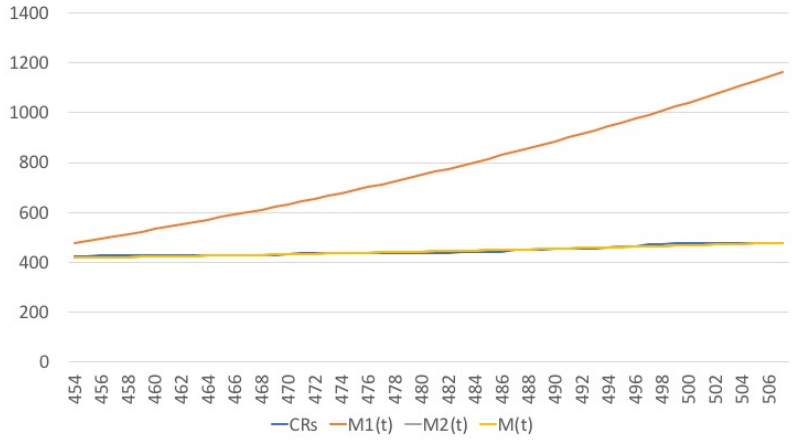
This demonstrates that although Approach 3 sometimes provides improvements for cumulative CR predictions, the quality of those predictions are affected by the quality of the selected models prior to performing TT. Therefore, it is clear that there is no straightforward answer that applies to any dataset. Each approach is suitable for a specific type of data. If a release has minimal changes that do not affect the CR rate, then Approach 1 would be a suitable approach. When evolution exist, the choice is between Approach 2 and Approach 3. Approach 2 provides a simple solution that re-estimates models as required. This is beneficial if at each stage there are enough data-points to perform the curve-fitting. It is not recommended when change-points are frequent. The problem with this approach is that under-



(a) $M_1(t)$, $M_2(t)$ and $M(t)$ in Stage 3



(b) $M_1(t)$, $M_2(t)$ and $M(t)$ in Stage 4



(c) $M_1(t)$, $M_2(t)$ and $M(t)$ in Stage 5

Figure (7.3) TT models per stage for Release 1

estimating of CRs is likely to occur due to over-fitting. In addition, frequent changes might lead us to not having enough data to fit a model and those cases happened in Releases 1 and 3. Approach 3, using TT overcomes this issue in Approach 2. After a change-point, when a model is selected, TT includes data from the beginning of the release to estimate the new model parameters and this overcomes the risk of curve-fitting with too little data. TT also reduces the risk of over-fitting models and causing under-estimation. But there is still a risk of under-estimation or providing poor predictions depending on the behavior of the models selected prior to TT.

7.5.1 Validity Threats

We still cannot claim that this approach is generalizable, which is an external validity threat. Although we used our method on several releases of an evolving system, the outcomes of these releases don't follow the same pattern. There is a general outcome that we claim it improves predictions but this is solely dependent on the behavior of the release in terms of cumulative CRs and on the selected models. We try to overcome this threat by repeating the process of modeling among multiple releases to provide more confidence in the results. We also do not claim that it will produce similar outcomes for other software systems. This is an aerospace system with a CR database that has many more enhancement requests than defect resolution requests. Other systems in different disciplines might not have the same type of data. Therefore, we cannot guarantee that similar findings are going to be found. This also can be a Construct threat to validity, which refers to the relation between theory and observation. Depending on the type of the collected data and the density of data other systems models might fit differently.

Chapter 8

Effort Estimation

8.1 Problem Statement

Effort estimation is key in any software development organization. Estimating effort is essential to software managers to determine the cost of software development and maintenance. This helps them in making informed decisions regarding staffing, future maintenance decisions, and overall business related decisions. As critical as it is choosing the right method for estimating software effort, it is challenging to find the right method that suits a specific project. Software projects are very diverse in the amount and type of effort they consume. Our case study has data regarding the number of CRs, CR priority, type of change, functional area (for release 4 only), Lines of Code, effort, submission date, and completion date, (see Section 3.2.1). To examine the most suitable effort estimation method for our system we first address the following research questions:

- RQ1: Can we use one of the existing methods for software effort estimation?
- RQ2: How accurately do the existing effort estimation methods predict effort for an evolving system?

- RQ3: Are there any alternative approaches that provide better estimation of effort?

According to the background conducted in Section 2.4.2, we find that most common algorithmic software effort estimation methods used in literature basically use a COCOMO based method [25], Function Point Analysis (FPA) based method [5] or estimation by analogy based method [116]. We lack information regarding function points in our data. We don't know if a CR is analogous to any other CRs because CR description is not available. Therefore, the COCOMO model is the only model that is potentially applicable to our case study.

We identify and explore the use of the COCOMO model in Section 8.2. Next, we discuss cumulative effort prediction in Section 8.3. The research questions are answered in Section 8.4, followed by threats to validity in Section 8.5. Finally we draw conclusions in Section 8.6

8.2 Effort Estimation Using COCOMO

The COCOMO model uses SLOC data to estimate effort. The Basic COCOMO equation takes the form of :

$$E = aS^b \tag{8.1}$$

Where E is effort, S refers to the lines of code, a is a coefficient and b is an exponent. COCOMO model is widely used in different forms. The three forms of COCOMO models are: basic, intermediate and detailed. The basic model uses Lines of code to estimate effort. The intermediate model takes more cost drivers into account. These drivers are attributes of the product, the hardware, the project and the personnel. These values are used to calculate the Effort Adjustment Factor (EAF), which is then

multiplied with the formula for the basic model to estimate effort. This equation is calculated as follows:

$$E = (aS^b) * EAF \tag{8.2}$$

The detailed model incorporates the characteristics of the intermediate model with an assessment of the cost drivers for each step of the software engineering process. These include planning and requirements, system design, detailed design, module code and test, etc.[24][20].

While the advanced versions of the COCOMO model that include project attributes could enhance the accuracy of its results, none of this information is available in our case study. Each entry in the CR database reports Added SLOC (SLOCA), Modified SLOC (SLOCM), autogenerated SLOC (SLOCG) and Deleted SLOC (SLOCD). Thus $S = SLOCA + SLOCM + SLOCG + SLOCD$. This lead to the following equation:

$$E = a * ((SLOCA + SLOCM + SLOCG + SLOCD)^b) \tag{8.3}$$

If we assume that the effort for completing a CR is the sum of the effort for adding code, modifying code, deleting code and auto-generating code, plus some effort to analyze the task, then we have the following equation:

$$E = a_0 + a_1 SLOCA^{b_1} + a_2 SLOCM^{b_2} + a_3 SLOCG^{b_3} + a_4 SLOCD^{b_4} \tag{8.4}$$

We use these two regression equations to find the most appropriate model for effort estimation. To measure the effectiveness of the regression model we use the Goodness-of-Fit measure, R^2. We require R^2 to be greater than 0.9 for an acceptable model. Beside the equations above we also apply the model with SLOCA, SLOCM

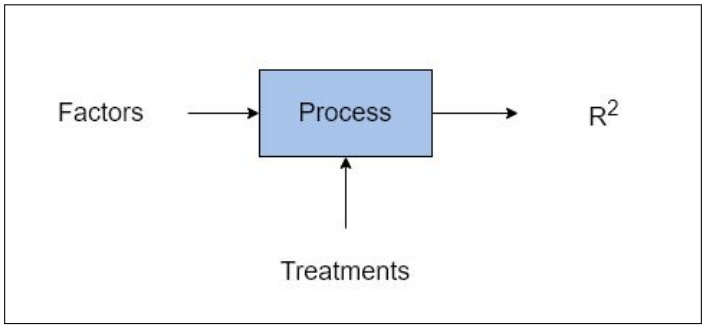


Figure (8.1) Case Study Design

and SLOCD only in some cases to test the fit without the influence of SLOCG. The reason behind that is that auto-generated SLOC values are much higher than SLOCA, SLOCM and SLOCD. SLOCG is auto-generated code and may not track well with human effort. This lead to the following alternative equations:

$$E = a * ((SLOCA + SLOCM + SLOCD)^b) \tag{8.5}$$

and

$$E = a_0 + a_1 SLOCA^{b_1} + a_2 SLOCM^{b_2} + a_4 SLOCD^{b_4} \tag{8.6}$$

We analyze Release 4 and Release 3. Both releases contain SLOC Added, SLOC Modified, SLOC Deleted and SLOC Generated in addition to Effort data. The first and second releases do not report SLOC data.

8.2.1 Case Study Design

This case study is designed based on the standard design for experiments by Wolin et al. [139]. We use different sets of data as independent variables to be processed using different equations to estimate effort and calculate the R^2 value. The results of the process are then analysed and discussed. These independent variables are called factors and the equations are treatments. We want to compare

the four equations in Section 8.2 against each others with having a combination of different factors.

8.2.2 Results: Part 1

We divided case study into two parts. The first part has two factors and four treatments are classified as the following:

- Factor A: Data
  - A1: Include all Data
  - A2: Include only valid data
- Factor B: Outliers
  - B1: Outliers included
  - B2: Outliers Excluded
- Treatments (T): Equations
  - T1: Equation 8.3
  - T2: Equation 8.5
  - T3: Equation 8.4
  - T4: Equation 8.6

The different combinations of factors and treatments are applied as shown in Table 8.1

Case No.	Factor A	Factor B	Treatment
1	A1	B1	T1
2	A2	B1	T1
3	A1	B2	T1
4	A2	B2	T1
5	A1	B1	T2
6	A2	B1	T2
7	A1	B2	T2
8	A2	B2	T2
9	A1	B1	T3
10	A2	B1	T3
11	A1	B2	T3
12	A2	B2	T3
13	A1	B1	T4
14	A2	B1	T4
15	A1	B2	T4
16	A2	B2	T4

Table (8.1) Factor and Treatment combinations for Release 4

8.2.2.1 Release 4

Release 4 is 398 weeks long and has 211 CRs. We used effort data and SLOCA, SLOCM, SLOCG and SLOCD data to fit the four models represented in the equations 8.3-8.6. When applying the case study we exclude the factor A2, because we don't have to exclude invalid data for this release. The data are all valid and included, therefore, we apply cases: 1, 3, 5, 7, 9, 11, 13, and 15 from Table 8.1.

Table 8.2 shows the R^2 values after applying these cases: 1, 3, 5, 7, 9, 11, 13, and 15 from Table 8.1 to Release 4 CR data.

We find that applying model 8.3 has an R^2 of 0.115 when using the full CR dataset and an R^2 of 0.11 when outliers were excluded. Using model 8.5 results in an R^2 of 0.098 and 0.174 when outliers are excluded. Applying model 8.4 gives an R^2 of is 0.172 for the full CR dataset and R^2 of 0.495 when outliers are excluded. Using the model based on the equation 8.6 results in an R^2 of 0.141 for the full

Case No.	Variable	R^2
1	a=18.356, b=0.123	0.115
3	a=17.849, b= 0.131	0.11
5	a=19.009, b=0.157	0.098
7	a=11.027, b=0.289	0.174
9	a0 = 28.55, a1= 23.982, a2= -22.335, a3=4.261, a4= 16.589, b1=0.125, b2=-0.331, b3=0.221, b4=-0.174	0.172
11	a0 = 28.302, a1= 0.812, a2= 0.12, a3=4.164, a4= 0.65, b1=0.481, b2=0.863, b3=-8.278e-10, b4=0.349	0.495
13	a0 =20.223, a1=16.953, a2= 3.705, a4= 9.342, b1=0.143, b2=-3.885e-7, b4=0.2	0.141
15	a0 = 11.66 , a1= 9.019 , a2= -0.011, a4= -6.877, b1=0.37, b2=1.184, b4=-7.317	0.21

Table (8.2) The results of using different Factors and Treatments (Release 4)

CR dataset and R^2 of 0.21 when outliers are excluded. In general, we find that all four models have R^2 values that are much lower than the threshold for acceptable models.

8.2.2.2 Release 3

We apply the case study to Release 3. In Release 3, we noticed that this release has a number of CRs where effort is greater than one hour, but the values of SLOCA, SLOCM, SLOCG and SLOCD are all zero or missing (had to be imputed according to Section 3.2.2.1). Almost 13% of CRs had missing SLOC data and 9% of CRs reported SLOC values of all zeros. This means 22% of CRs are invalid for proper effort estimation. We excluded these CRs and applied the models to the remaining 311 CRs. Therefore We apply the case study to all 16 cases.

Case No.	**Variable**	R^2
1	a=33.369, b=-1.009e-8	0.215
3	a= 18.25, b=0.147	0.059
5	a= 27.59, b=-4.478e-9	0.2
7	a= 9.197, b=0.342	0.185
9	a0 = 1.017, a1=9.477, a2=4.348, a3=6.957, a4 =7.296, b1=0.324, b2=-1.935, b3=0.07, b4=0.158	0.349
11	a0 =19.248, a1=3.172, a2=-2.35, a3=3.846, a4=6.307, b1=0.474, b2=-0.597, b3=-1.006e-6, b4=0.227	0.601
13	a0 =9.327 , a1=9.251 , a2= -6.401, a4=7.394, b1=0.331, b2=0.157, b4=0.152	0.242
15	a0 =239.882, a1=3.491, a2=-232.463, a4=5.857, b1=0.476, b2=-0.009, b4=0.192	0.355

Table (8.3) The results of using different Factors and Treatments (Release 3) with all data (Factor A1)

Case No.	**Variable**	R^2
2	a=17.366, b=0.158	0.173
4	a= 18.335, b=0.147	0.139
6	a= 13.434, b=0.268	0.282
8	a= 10.058, b=0.323	0.296
10	a0 = 5.768, a1=6.137, a2=2.336, a3=-1.32e-14, a4 =11.501, b1=0.373, b2=-7.039, b3=2.459, b4=-0.22	0.407
12	a0 =5.062, a1=6.478, a2=1.003, a3=14.111, a4=7.039, b1=0.357, b2=-2.637, b3=-5.123e-8, b4=0.179	0.228
14	a0 =42.776 , a1=7.604 , a2= -39.764, a4=-0.001, b1=0.367, b2=-0.119, b4=1.518	0.347
16	a0 =279.595, a1=6.512, a2=-275.26, a4=6.001, b1=0.38, b2=-0.007, b4=0.164	0.338

Table (8.4) The results of using different Factors and Treatments (Release 3) with only valid effort data (Factor A2)

Table 8.3 shows the results of applying cases 1, 3, 5, 7, 9, 11, 13, and 15 Where Factor A includes all data, both valid and invalid (Factor A1). The results show that R^2 values are below the acceptable threshold, ranging from 0.059 to 0.60.

Table 8.4 shows the results of applying the remaining cases where only data with valid effort data is included (Factor A2). R^2 values range from 0.139 to 0.407. Therefore, this attempt did not succeed in providing an acceptable model either.

8.2.3 Results: Part 2

Since the results of Part 1 in Section 8.2.2 did not provide us with fitted models we investigate the ability to apply these models on subsets of the data. We apply the case study with different combinations of factors and treatments according to the data available for each release.

8.2.3.1 Release 4

We investigate if effort estimation by type of CR is more successful. Therefore, looked into Priority, Functional Areas and Duration of a CR if they could be used as factors along with the two basic models that include all SLOC data. i.e. we have one factor and two treatments applied each time.

We collected CRs that are in the top 5 maintenance prone areas (FA1 - FA10 - FA11 - FA13 - FA14), these are areas with more than 10 CRs. There are 163 CRs in the top five maintenance prone Functional Areas. The remaining 48 CRs are in the remaining Functional Areas. This top maintenance prone areas are processed alone as one factor. We also divide the CR data into groups based on their priority. The reasoning behind that is that CRs that share the same priority for instance may require a certain effort that is different than for CRs from other priorities.

We found that about 83% of the CRs are of priority C2R (routine maintenance). Therefore, we divided the CRs into two factors, one with priority of C2R and the other containing CRs of the remaining priorities. The second group has only 36 CRs. Lastly, we investigated whether there was a common effort pattern based on how long a CR was open. Some of the CRs were resolved within a year and some took longer than that. We divided the dataset into two factors, one has CRs that were resolved within a year (177 CRs), and the other group that took more than a year, (34 CRs).

To summarize, the factors and treatments for Release 4 for this part are:

- Factor A: Data
  - A1: CRs for the top 5 Functional Areas
  - A2: CRs for all Priorities except "C2R"
  - A3: CRs with Priority "C2R" only
  - A4: CRs with duration of more than one year
  - A5: CRs with duration of within a year
- Treatments (T): Equations
  - T1: Equation 8.3
  - T2: Equation 8.4

The results of applying the different combinations of factors and treatments are shown in Table 8.5

Table 8.5 shows that fitting the model based on equation 8.3 to the top 5 maintenance prone areas has an R^2 of 0.287 only, while fitting the model based on equation 8.4 has an R^2 of 0.147. It also reports a very low R^2 for both sets of priorities (less

Factors and Treatments	**Variable**	R^2
A1 T1	a = 13.964, b = 0.132	0.287
A1 T2	a0=23.642, a1=0.002, a2=1.61e-5, a3=7.96e-5, a4=0.153, b1=136, b2=0.452, b3=0.398, b4=-3.519e-8	0.147
A2 T1	a=22.122, b=0.107	0.123
A2 T2	a0=-122.582, a1=137.085, a2=123.582, a3=281.089, a4=0.98, b1=-5856.142, b2=33138.147, b3=0.008, b4=2.734	0.484
A3 T1	a=17.603, b=0.126	0.116
A3 T2	a0=2354.765, a1=335.006, a2=8.539, a3=-2329.19, a4=-0.355, b1=196649.463, b2=677, b3=-2.625e-9, b4=1.362	0.85
A4 T1	a1=34.704, b1=0.114	0.122
A4 T2	a0=32.106 , a1=0, a2=-4.896r-10, a3=-18.531, a4=20.857, b1=2.698, b2=4.383, b3=822276.133, b4=-46.647	0.51
A5 T1	a=15.828, b=0.113	0.123
A5 T2	a0=2 , a1=16.5, a2=-2.248, a3=10.703, a4=3.248, b1=-2066781.355, b2=190707.082, b3=28851.005, b4=-49.157	0.32

Table (8.5) The results of Part 2 of factors and treatment combinations (Release 4)

than 0.2). The value of R^2 is much higher when the model is based on equation 8.4. R^2 for routine maintenance (C2R) is 0.85. This is almost good enough to accept the model. However, for other groups of CRs this model shows only very small R^2 values. For equation 8.3 the results are (R^2 is 0.122 and 0.123, respectively). Using the model based on equation 8.4 improves the fit somewhat (R^2 is 0.51 and 0.32, respectively). Neither has a good enough fit.

8.2.3.2 Release 3

We also investigate if effort estimation by type of CR provides better results. Therefore, looked into Change Type, Priority, and Duration of a CR if they could be used as factors along with the two basic models that include all SLOC data. i.e. we have one factor and two treatments applied each time.

CR data are divided according the Change Type. Notice that Change Type data is available for CRs of Release 3 but not Release 4, while Functional Area is available for Release 4 only. The reasoning behind that grouping is that CRs that share the same Change Type may require similar effort. We found that about 73% of the CRs are Anomalies which are CRs collected during development until integration test, while SCR and STR are CRs collected afterwards. Therefore, we divided the CRs into two subgroups, one with Anomalies (291 CRs), and the other with both SCR and STR (110 CRs). When dividing the dataset according to priority we find that 87% of CRs are of C2R priority, (351 CRS), and the remaining CRs are of other priorities, (50 CRs). Dividing the CRs according to duration shows 355 CRs that were open for a year or less, and 46 CRs for longer than one year.

To summarize, the factors and treatments for Release 3 for this part are:

- Factor A: Data

  - A1: CRs for Anomalies
  - A2: CRs for SCRs and STRs
  - A3: CRs for all Priorities except "C2R"
  - A4: CRs with Priority "C2R" only
  - A5: CRs with duration of more than one year
  - A6: CRs with duration of within a year

- Treatments (T): Equations
  - T1: Equation 8.3
  - T2: Equation 8.4

The results of applying the different combinations of factors and treatments are shown in Table 8.6. CRs of C2R priority has an R^2 of 0.393 when model based on equation 8.4 is applied, and 0.213 for the remaining priorities, while it has an R^2 of 0.191 when equation 8.3 is applied for C2R and 0.97 for the remaining priorities. Based on equation 8.3 to Anomalies and non-anomalies. The R^2 values are 0.125 and 0.263, respectively and 0.321 and 0.262 when equation 8.4 is applied. The results for the remaining factors regarding the duration are not better, R^2 values range from 0.156 to 0.479. This highlights that regardless of the releases and the types of CRs, these models cannot be used for effort estimation. All R^2 values are far below the threshold of 0.9.

This investigation was performed in an effort to find out if the different versions of COCOMO type models can be used in effort estimation. Obviously from the R^2 values, the models were not a success in terms of fitting data for any release. Model 8.4 has better R^2 values than model 8.3, but in general the R^2 value is less than the

threshold. Therefore, we cannot use these models in effort estimation and we need to find an alternative approach.

8.3 Cumulative Effort Prediction

In Chapter 5, we were able to predict the number of cumulative CRs in a release based on SRGMs fit to cumulative CR data. This method only requires the availability of number of CRs over time. Since we noticed that cumulative effort follows similar growth pattern to CR growth with the existence of change-point (Figures 3.5,3.6,3.6, 3.8), we would like to investigate the ability to use cumulative effort in predicting future effort. This method would include all four releases since it does not depend on the availability of any other attributes such as SLOC, which makes it more applicable and generalizable.

To predict cumulative effort with the existence of change-points we would apply a multi-stage model that uses regression to find a fit model, and then uses the selected model to predict effort until a change-point occurs.

Figure 8.2 shows the process of cumulative effort prediction which is loosely based on the CR prediction process in Chapter 5. Like cumulative CRs we should have a minimum of 5 CRs to start model estimation and after each change-point we need at least 10 weeks to start prediction. After collecting effort data we check if any regression model fits the data. For a model to be considered fit, we need to measure the Goodness-of-fit of R^2 of 0.9 or greater. Also, the estimated effort should be greater than or equal the actual effort. If a model was found to be fit then it is used for future effort prediction, otherwise more data should be collected. The model is used for prediction until a change-point is found, where the whole model estimation process restarts. This whole process continues until the end of the

Factors and Treatments	**Variable**	R^2
A1 T1	a=15.302, b=0.146	0.125
A1 T2	a0=16.065, a1=1.931, a2=0.147, a3=-0.001, a4=0.107, b1=0.518, b2=0.147, b3=1.009, b4=0.522	0.321
A2 T1	a=27.883, b=0.139	0.263
A2 T2	a0=45.644, a1=-17.953, a2=-14.074, a3=-26.01, a4=1.294, b1=-36.438, b2=-448.952, b3=4385.306, b4=1.172	0.262
A3 T1	a=17.734, b=0.144	0.097
A3 T2	a0=822.097, a1=4.689e-165, a2=-820.091, a3=0.121, a4=1.247, b1=4.969, b2=-0.014, b3=0.511, b4=0.565	0.213
A4 T1	a=17.531, b=0.157	0.191
A4 T2	a0=-303.446, a1=10.413, a2=307.523, a3=3.019e-24, a4=10.623, b1=0.32, b2=-0.004, b3=4.304, b4=0.131	0.393
A5 T1	a1=0.065, b1=0	0.208
A5 T2	a0=-3.369 , a1=16.315, a2=15, a3=64.158, a4=-6.337, b1=0.291, b2=-1545.214, b3=1874.871, b4=2111054.639	0.244
A6 T1	a=17.727, b=0.148	0.156
A6 T2	a0=26.084 , a1=0.101, a2=0.013, a3=0.117, a4=-0.007, b1=0.448, b2=-3.949, b3=-7.61e-8, b4=0.542	0.479

Table (8.6) The results of Part 2 of factors and treatment combinations (Release 3)

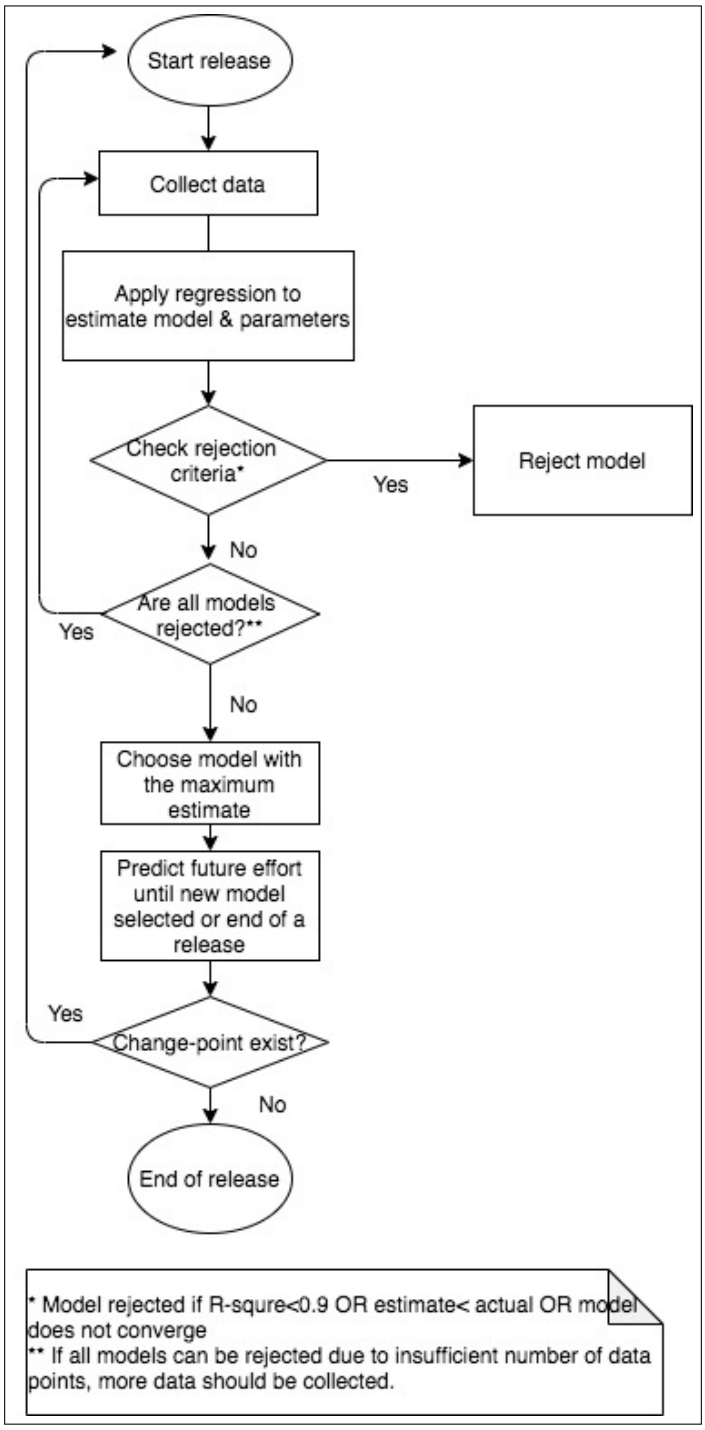


Figure (8.2) Cumulative effort prediction process

release. To apply this process we do three major steps, first we estimate change-points for each release, then we apply model estimation and prediction and finally we compare prediction ability of these models among the four releases.

8.3.1 Change-point Estimation

Since not all releases have SLOC data to use in identifying change-points we apply the likelihood ratio test explained in Section 4.2.3 using `cpts package` in R [108] to identify change-points in the four releases.

Four change-points were estimated for Release 4, (see Figure 8.3). These change-points are in weeks 136, 201, 255, and 338. The first change-point was estimated after only 4 CRs. Fitting a model for fewer than 4 CRs is not practical if non-linear regression is used. It is too early to predict the behavior of the cumulative effort in order to make an acceptable prediction. Therefore, we do not consider 136 a change-point, so our list of change-points for Release 4 is: 201, 255 and 338.

Release 3 has four estimated change-points as well (see Figure 8.4), these are in weeks 327, 337, 369 and 406. Weeks 327 and 337 are just 10 weeks apart which is the minimum number of weeks needed to start predicting. Due to these change-points being very close together we will use only the first as a change-point in our study. Therefore, the list of change-points for Release 3 are 327, 369 and 406.

Release 2 has four estimated change-points as shown in Figure 8.5. These change-points are in weeks 212, 236, 262 and 282.

Figure 8.6 shows the estimated change-points for Release 1. Four change-points were estimated in weeks 349, 383, 396 and 423.

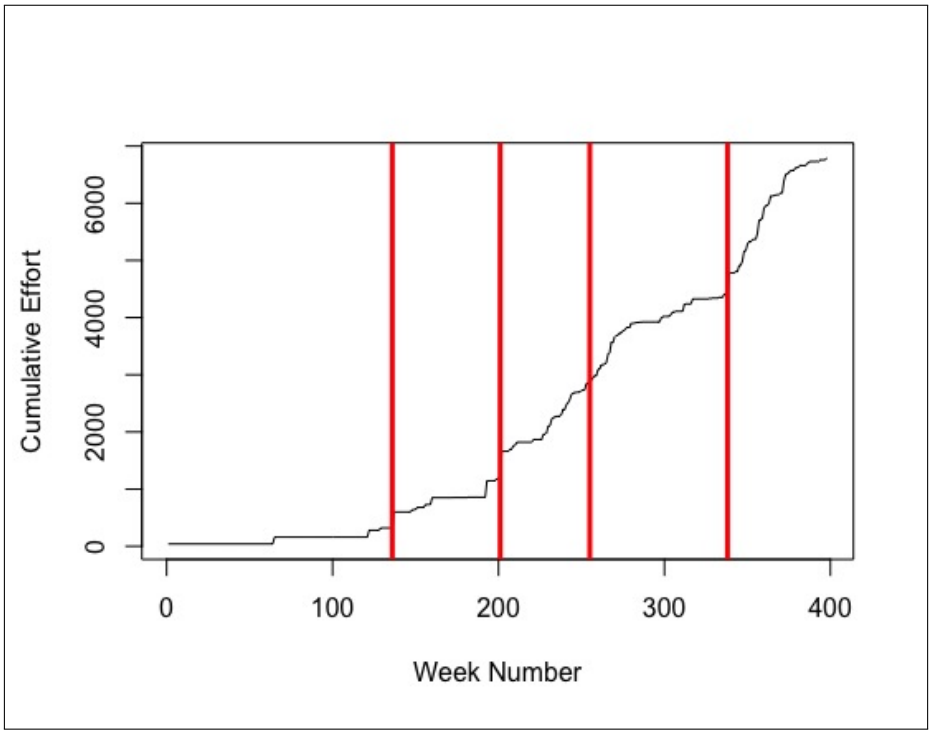


Figure (8.3) Estimated change-points in cumulative effort for Release 4

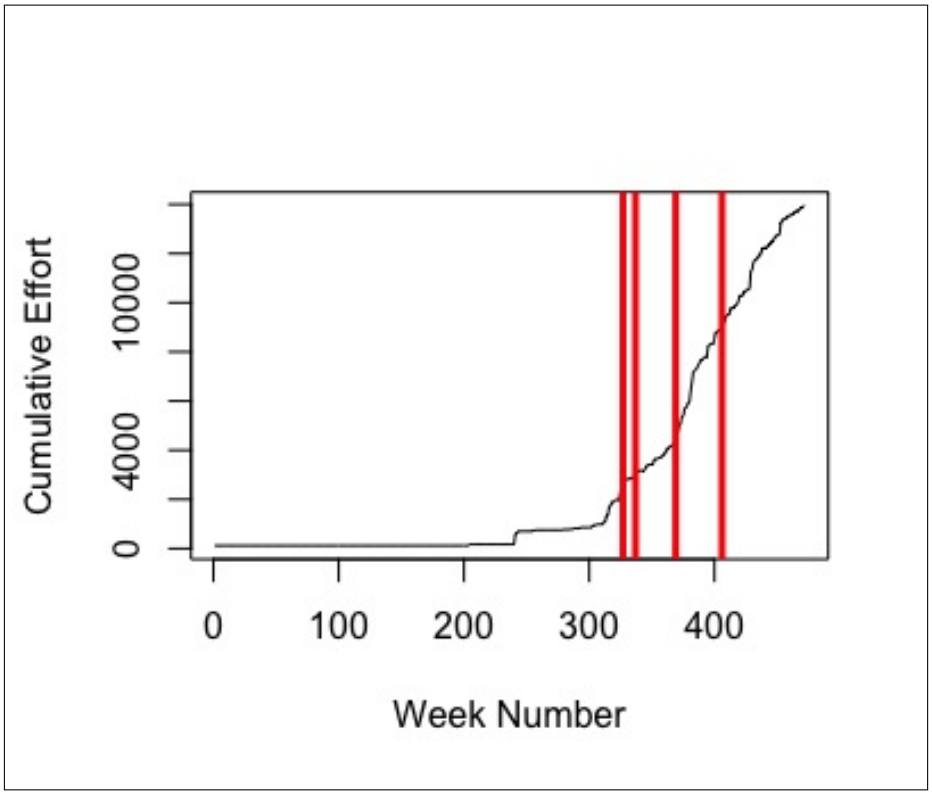


Figure (8.4) Estimated change-points in cumulative effort for Release 3

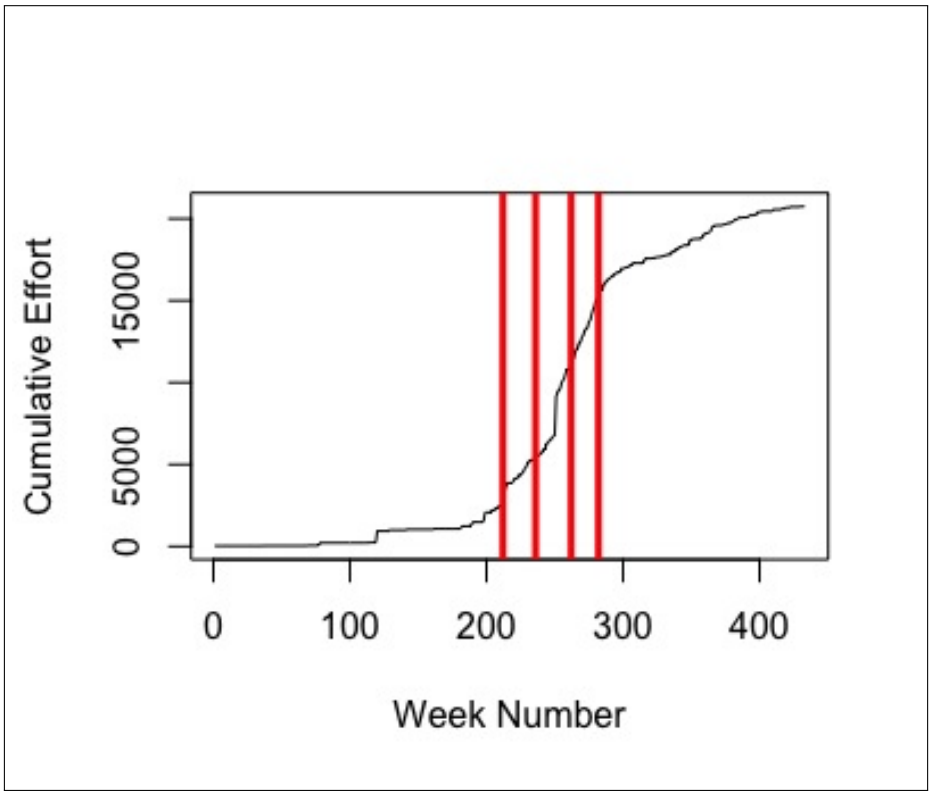


Figure (8.5) Estimated change-points in cumulative effort for Release 2

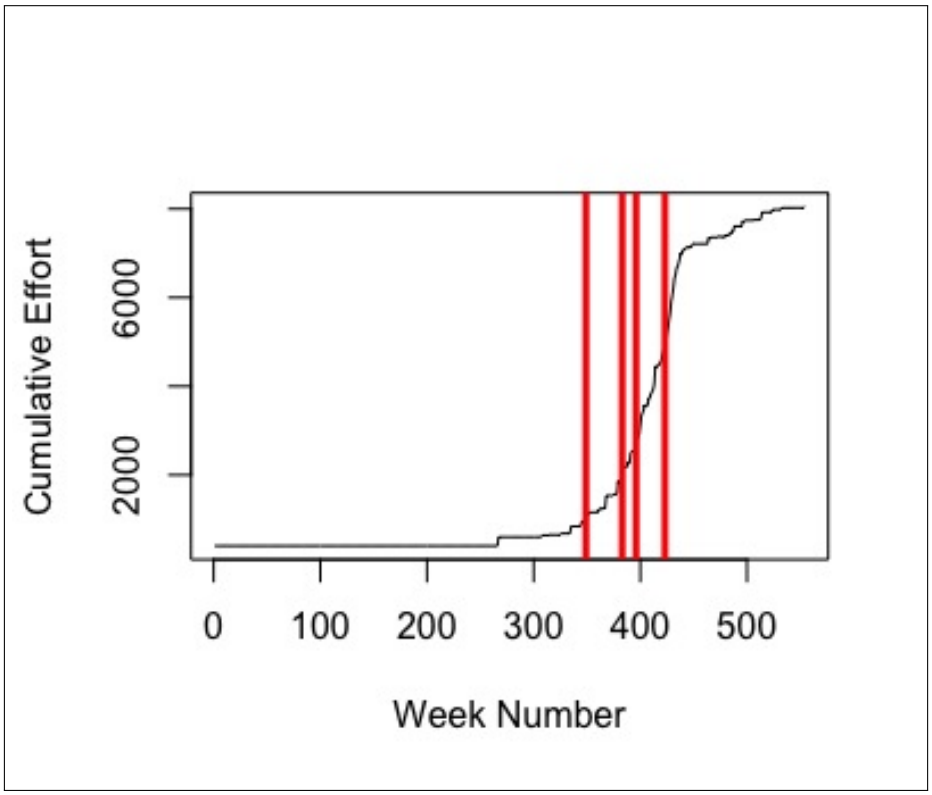


Figure (8.6) Estimated change-points in cumulative effort for Release 1

8.3.2 Model Estimation

After estimating change-points in each release, we then apply the process of model estimation and effort prediction, (Figure 8.2). For model estimation we use the Curve Estimation command in `IBM SPSS` [62]. This is one of the regression commands that takes time in weeks as an independent variable and uses previous cumulative effort data to predict a fit model among 11 different models which are: linear, logarithmic, inverse, quadratic, cubic, power, compound, S, logistic, growth and exponential. Each of these models are described in IBM SPSS Statistics V24.0 documentation [28].

8.3.2.1 Release 4

This release was divided into four stages, based on the estimated change-points (Section 8.3.1), these stages are as follows:

- Stage 1: From week 1 to week 200.
- Stage 2: From week 201 to week 254.
- Stage 3: From week 255 to week 337.
- Stage 4: From week 338 to week 398.

From the beginning of this release cumulative effort data was collected for weeks with CRs. Model estimation was performed and tested if any model has an R^2 of 0.9 or greater. We also require the estimated values to be greater than or equal to the actual value for a fit model. If none of the models were found to be fit, more data was collected then model estimation was performed again until either a model was found fit or we reach the next change-point or the end of the release.

Table 8.7 shows the results of model estimation. "Week" shows the week when the model was selected and "Actual" shows the actual cumulative effort of that week. For each model we show the R^2 value and the estimated cumulative effort for that week. The R^2 value and estimated value highlighted in cyan identify the selected model.

For Stage 1, by week 153, the Cubic model had an R^2 of 0.9 and an estimated value of 693, which is greater than the actual values. The model was selected for cumulative effort prediction starting from week 154 until the next change-point. In week 247, the linear, the cubic and the quadratic models all had a valid R^2, but the linear model had an estimated value that is less than the actual value, so it is not a good fit. Both the quadratic model and the cubic models are good candidates, so we chose the cubic for Stage 2. For stage 3, the models had an R^2 of 0.96 by week 264 and all of the estimated values are greater than the actual cumulative effort, the Inverse model was chosen. Similarly, for Stage 4, by week 352 all models were viable and we chose the quadratic model.

8.3.2.2 Release 3

This release was divided into four stages as follows:

- Stage 1: From week 1 to week 326.
- Stage 2: From week 327 to week 368.
- Stage 3: From week 369 to week 405.
- Stage 4: From week 406 to week 472.

Table 8.8 shows the results of model estimation for each stage. By week 310, the Cubic model was selected, since this model has an R^2 of 0.9, which is the highest

Stage	Week	Actual	Estimation	Model										
				Linear	Log.	Inverse	Quad.	Cubic	Compound	Power	S	Growth	Exp.	Logistic
1	153	680	R^2	0.68	0.37	0.05	0.87	0.92	0.87	0.57	0.09	0.87	0.87	0.87
			Est.	427	286	187	594	693	512	234	121	512	512	512
2	247	2694	R^2	0.90	0.89	0.88	0.94	0.94	0.89	0.89	0.88	0.89	0.89	0.89
			Est.	2567	2546	2525	2711	2718	2607	2582	2557	2607	2607	2607
3	264	3179	R^2	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
			Est.	3214	3213	3212	3214	3214	3217	3216	3216	3217	3217	3217
4	352	5331	R^2	0.90	0.90	0.90	0.90	0.91	0.90	0.90	0.90	0.90	0.90	0.90
			Est.	5335	5332	5330	5337	5371	5343	5341	5338	5343	5343	5343

Table (8.7) Effort model estimation for Release 4

Stage	Week	Actual	Estimation	Model										
				Linear	Log.	Inverse	Quad.	Cubic	Compound	Power	S	Growth	Exp.	Logistic
1	310	984	R^2	0.57	0.25	0.02	0.87	0.90	0.61	0.28	0.02	0.61	0.61	0.61
			Est.	639	420	286	977	1112	540	290	196	540	540	540
2	348	3417	R^2	0.93	0.93	0.93	0.93	0.93	0.90	0.90	0.91	0.90	0.90	0.90
			Est.	3422	3418	3414	3422	3390	3444	3440	3436	3444	3444	3444
3	378	5766	R^2	0.97	0.97	0.97	0.97	0.97	0.95	0.95	0.96	0.95	0.95	0.95
			Est.	5904	5901	5899	5904	5904	5945	5943	5940	5945	5945	5945
4	415	9794	R^2	0.92	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.92
			Est.	9872	9871	9870	9872	9872	9878	9877	9876	9878	9878	9878

Table (8.8) Effort model estimation for Release 3

value among the other models and the estimated cumulative effort is 1112 which is greater than the actual cumulative effort. In Stage 2, the linear, the logarithmic, the inverse, the quadratic and the cubic models all have the highest R^2 values. The inverse model has an estimated cumulative effort that is lower than the actual, so it is excluded. From the four remaining candidates, the logarithmic model was selected. In week 378, The inverse model was selected for Stage 3 and in week 415, Stage 4, the inverse model was selected.

8.3.2.3 Release 2

Release 2 has the most stages:

- Stage 1: From week 1 to week 211.
- Stage 2: From week 212 to week 235.
- Stage 3: From week 236 to week 261.
- Stage 4: From week 262 to week 281.
- Stage 5: From week 282 to week 433.

Table 8.9 highlights results for model estimation. By week 160, a few candidate models are acceptable, and we chose the exponential model, which had an R^2 of 0.9 and an estimate of 1326. For the second stage, some models have an acceptable R^2 but the estimated cumulative effort is less than the actual, except for the cubic model. The cubic model was selected for the third stage as well. since it had the highest R^2 value of 0.97 and an estimated cumulative effort of 6368. By week 271, all models fit the data for stage 4, we selected the Inverse model. In week 291, the model for the final stage was selected to be the S model among all the other models which are all fit.

Stage	Week	Actual	Estimation	Model										
				Linear	Log.	Inverse	Quad.	Cubic	Compound	Power	S	Growth	Exp.	Logistic
1	160	1035	R^2	0.73	0.38	0.05	0.89	0.89	0.90	0.58	0.09	0.90	0.90	0.90
			Est.	918	588	358	1275	1271	1326	407	153	1326	1326	1326
2	232	5248	R^2	0.90	0.90	0.91	0.91	0.95	0.84	0.84	0.85	0.84	0.84	0.84
			Est.	5191	5177	5164	5131	5414	5306	5290	5274	5306	5306	5306
3	245	6356	R^2	0.92	0.92	0.91	0.92	0.97	0.93	0.92	0.92	0.93	0.93	0.93
			Est.	6255	6252	6249	6258	6367	6260	6257	6254	6260	6260	6260
4	271	12910	R^2	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.98	0.98
			Est.	12964	12960	12956	12964	12964	12988	12983	12979	12988	12988	12988
5	291	16424	R^2	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94
			Est.	16529	16527	16524	16529	16529	16537	16535	16533	16537	16537	16537

Table (8.9) Effort model estimation for Release 2

Stage	Week	Actual	Estimation	Model										
				Linear	Log.	Inverse	Quad.	Cubic	Compound	Power	S	Growth	Exp.	Logistic
1	348	839	R^2	0.53	0.23	0.02	0.84	0.90	0.54	0.24	0.02	0.54	0.54	0.54
			Est.	593	510	460	727	798	579	495	449	579	579	579
2	359	1157	R^2	0.91	0.90	0.90	0.91	0.91	0.90	0.90	0.90	0.90	0.90	0.90
			Est.	1164	1163	1163	1165	1166	1172	1172	1171	1172	1172	1172
3	395	2295	R^2	0.89	0.89	0.90	0.89	0.89	0.88	0.88	0.88	0.88	0.88	0.88
			Est.	2333	2331	2330	2333	2333	2360	2359	2357	2360	2360	2360
3	411	3559	R^2	0.96	0.96	0.96	0.96	0.98	0.97	0.97	0.97	0.97	0.97	0.97
			Est.	3578	3573	3569	3582	3669	3618	3613	3608	3618	3618	3618
4	432	5338	R^2	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
			Est.	5260	5258	5256	5261	5341	5268	5266	5264	5268	5268	5268

Table (8.10) Effort model estimation for Release 1

8.3.2.4 Release 1

This release was divided into four stages:

- Stage 1: From week 1 to week 348.
- Stage 2: From week 349 to week 382.
- Stage 3: From week 383 to week 395.
- Stage 4: From week 396 to week 422.
- Stage 5: From week 423 to week 554.

For this first stage data was collected until the end of the stage and no model was selected. For Stage 1 Table 8.10 shows model estimation of the last week (348). We find that the only model that has an R^2 of 0.9 or above is the cubic model, but the estimated cumulative effort (798) is less than the actual (839). Therefore, model selection for this stage was not successful. By contrast, the remaining stages each had several models that meet the requirement of accepted models. In week 359, the linear model was selected. For Stage 3, the Inverse model was selected on week 395. The cubic model was selected for Stage 4 in week 411, and for Stage 5 in week 432.

8.3.3 Effort Prediction

After selecting a model for each stage, these models are used for predicting effort for the future. To measure the predictive ability for each model we observed future predictions from the week the model was selected for up to 6 months into the future. We compared the predicted values for each month to the actual cumulative

Week	Actual Effort	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
153	680	Effort	768	848	935	1027	1126	1232
		RE	0.05	0.00	0.10	0.21	0.32	0.45
247	2694	Effort	3103	3469	3908	4427	5033	5735
		RE	0.14	0.21	0.31	0.39	0.49	0.55
264	3179	Effort	3359	3501	3639	3773	3903	4029
		RE	-0.06	-0.05	-0.04	-0.03	0.00	0.03
352	5331	Effort	5589	5818	6049	6284	6520	6760
		RE	0.02	-0.02	-0.01	0.02	0.02	0.03

Table (8.11) Effort Predictions and Relative Errors for six months into the Future for Release 4

effort. Then we calculated the relative error. Relative error (RE) is calculated as $RelativeError = ((estimated - actual)/actual)$. [1]

Table 8.11 shows four main columns, the first column shows the week when a model was selected in the fourth release. The second column shows the actual effort by that week, the third column shows the predicted effort on a month-to-month basis for six months, and the fourth column shows the RE value of each prediction month-by-month. After week 153, the RE value is very small in the first three months, i.e. less than or equal 0.1. After the third month RE starts increasing to 0.21 then 0.32 and 0.45. The model is applicable for the 6 months and it does not show that it under-predicts effort. After week 247, the cubic model has a positive error that is larger than 0.1 which increases gradually into the future. The error rates are acceptable since they are relatively small (less than 1) and positive. After week 254, the inverse model has a very small error value but it under-predicts the actual cumulative effort sometimes. Likewise, the last stage shows small RE but some of them are negative. In general, REs are small for up to six months of prediction, which makes effort prediction accuracy high.

[1]Relative error value is calculated using the absolute value of an error over the actual value. In our case we need to keep track of negative values, therefore we calculated the relative error with the real error value instead.

Week	Actual Effort	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
310	984	Effort	1170	1230	1292	1357	1423	1492
		RE	-0.14	-0.35	-0.34	-0.42	-0.49	-0.48
348	3417	Effort	3567	3714	3859	4003	4145	4286
		RE	-0.02	0.01	0.01	-0.03	-0.03	-0.15
378	5766	Effort	6526	7140	7741	8330	8907	9472
		RE	-0.03	-0.03	0.00	0.07	0.07	0.07
415	9794	Effort	10200	10525	10843	11155	11462	11763
		RE	0.01	0.02	0.03	-0.04	-0.03	-0.04

Table (8.12) Effort Predictions and Relative Errors for six months into the Future for Release 3

Table 8.12 shows the predictions of cumulative effort for Release 3. By observing the RE values of each stage, we find that in some months these models tend to have negative RE values, but in general the RE values are very small. After week 310, the cubic model was selected, but it did not perform well compared to the other models in this release, (the inverse model and the logarithmic model). The month-to-month prediction shows that all values were under-predicted and the RE is higher than the other models.

For Release 2, Table 8.13 shows the month-to-month predictions of every stage. The selection of the exponential model for the first stage gave predictions with RE values ranging from 0.35 to 0.78. In week 323 the cubic model was selected. The RE ranges from 0.29 to 2.68 for predictions. After week 245, the cubic model gave predictions with relatively high RE values that were 0.29 and 0.7 in the first two month and increased to values greater than one starting in the third month. The RE values of this model are the highest in this release. For the third stage, a cubic model was also selected, the RE values started small in the first couple of months but increased to 2.4 by the sixth month. The inverse model selected for the fourth stage has very low error rates. RE values range between -0.03 to 0.03, the lowest RE values among the other models in this release. The S model has relatively low

Week	Actual Effort	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
160	1035	Effort	1444	1571	1708	1857	2017	2191
		RE	0.35	0.46	0.59	0.73	0.88	0.78
232	5248	Effort	6961	9581	13603	19345	27126	37262
		RE	0.29	0.70	1.17	1.92	1.89	2.68
245	6356	Effort	7809	10481	14845	21348	30437	42556
		RE	0.16	0.10	0.45	0.96	1.57	2.40
271	12910	Effort	13717	14455	15173	15871	16550	17210
		RE	0.00	-0.01	-0.03	-0.01	0.01	0.03
291	16424	Effort	17066	17595	18125	18656	19189	19723
		RE	0.02	0.04	0.06	0.08	0.11	0.14

Table (8.13) Effort Predictions and Relative Errors for six months into the Future for Release 2

RE when predictions were made after week 291, ranging from 0.02 after one month and gradually increasing to reach an RE of 0.14.

Finally, Table 8.14 shows the RE values of models selected for Release 1. The linear model selected after week 359 has RE values ranging from 0.12 to 0.25. The inverse model used to predict effort after week 395 (Stage 2) has a lower range of RE, (-0.10 to 0.01), but it mostly under-predicts effort. The cubic model selected after week 411 which has a low RE value after a month then this RE value increases dramatically. In fact, by month 5 and month 6 we find that the RE value is high and the predicted values are negative due to the shape of the cubic curve. These predicted values are invalid for cumulative effort. After week 432 has a zero RE value after one month and it gradually increases by month until it reaches 0.56 after six months. In general, the selected models show acceptable predictive ability with small RE values, but some performs poorly as time progresses.

In some cases, no model is selected due to failing to find a good fit model or because of not having enough data to start fitting a new model in a certain stage. In this case we continue using a model selected from a previous stage. We also want to maintain having a low RE value to have accurate prediction. Therefore, we

Week	Actual Effort	Prediction	Month					
			+1 mo.	+2 mo.	+3 mo.	+4 mo.	+5 mo.	+6 mo.
359	1157	Effort	1294	1424	1554	1684	1814	1944
		RE	0.12	0.18	0.25	0.10	0.18	0.24
395	2295	Effort	2559	2783	3003	3218	3430	36372
		RE	0.01	-0.01	-0.10	-0.10	-0.09	-0.10
411	3559	Effort	3796	3645	3011	1749	-288	-3247
		RE	0.01	-0.10	-0.33	-0.63	-1.06	-1.55
432	5338	Effort	6055	6922	7908	8978	10099	11237
		RE	0.00	0.04	0.13	0.27	0.42	0.56

Table (8.14) Effort Predictions and Relative Errors for six months into the Future for Release 1

propose Algorithm 2 to maintain having the best fitted model as much as possible. In case no model was fitted for a stage then we use a previously selected model. To maintain the accuracy of the model, a threshold should be defined to what is considered as a change-point and another threshold for acceptable range of RE.

Algorithm 2 Maintaining the accuracy effort estimation throughout a release

1: Model selected on Week (W_i) for effort prediction
2: **repeat**
3: Check for changes on Week (W_{i+1})
4: **if** change $>$ threshold **then**
5: Apply change-point algorithm after collecting CRs for at least 10 more weeks after change-point and 5 unique CRs (in the meantime use old model for prediction);
6: **else**
7: Apply selected model for next month and check relative error (RE);
8: **if** RE $>$ threshold AND RE $<$ 0 **then**
9: re-estimate model;
10: **end if**
11: **end if**
12: **until** End of Release =0

This algorithm can be applied to CR prediction as well to maintain better predictions of future CRs.

8.4 Discussion

For this aerospace system, we can only apply the basic COCOMO model due to lack of cost driver data. The data we used was SLOCA, SLOCD, SLOCM and SLOCG. To incorporate all these pieces of data regarding SLOC we interpreted the COCOMO model two different ways, Eq. 8.3 and Eq. 8.4. These two models were tested for Release 3 and Release 4.

To discuss the results we revisit the research questions stated at the beginning of this chapter.

- RQ1: Can we use one of the existing methods for software effort estimation?

  The existing models can be used and have been widely used in research and industry for estimating effort. In our case study we applied two versions of the basic COCOMO model to our system data in order to predict effort. We tried to perform a regression of the model on the available SLOC data from Releases 3 and 4, The Goodness-of-fit shows that the models poorly fits the existing data. For both releases we tried the model on the full set of CRs and their attributes. Then we excluded SLOG from the equation. This did not improve Goodness-of-fit. Then we excluded the outliers in an effort to reduce distortion. These different scenarios did not change the fact that the models poorly fit the data. We applied the models on different portions of the data by splitting the dataset based on priority, change type, elapsed time and functional area. Fitting the model was not successful for any of these data groupings. Therefore, applying COCOMO was unsuccessful.

- RQ2: How accurately do the existing effort estimation methods predict effort for an evolving system?

When using COCOMO, we are not detecting or estimating change-points, so evolution points are not clear and the model does not address evolution in effort estimation. We dealt with the issue of evolution in previous chapters (4 and 5), when predicting the number of cumulative CRs in a release. We highlighted the weeks that a change-point was likely to occur and used that to apply CR prediction in a multi-stage release. Inspired by success in predicting CRs we apply a multi-stage approach to predict cumulative effort in a release where change-points are identified and models are selected in each stage to predict effort.

- RQ3: Are there any alternative approaches that provides better estimation of effort?

  We find that the multi-stage approach that has been used to predict cumulative effort has promising results. First it does not require any additional data other than dates of CR submission and effort data, which makes it a more generalizable approach. We had success in identifying change-points using the likelihood ratio test that has used previously for CR change-point estimation, (Section 4.2.3). We then used a similar approach to the multi-stage approach described in Section 5.3.2 to estimate fit models. These models are different than the models used for CRs, SRGMs. Here we used general linear and non-linear models such as cubic, logarithmic and exponential models to find a fitted model to make predictions of the amount of effort in the future. As shown in Section 8.3.3, the majority of the models provided prediction for up to six months with relatively low RE values. These results are comparable to the results of CR predictions. This method provided us with models that provides with high quality prediction of cumulative effort.

Cumulative multi-stage effort prediction provides a novel way to predict effort using past effort data. It identifies change-points so it can change the selected model according to changing growth rates of cumulative effort data. It allows us to examine the fit among several models and find the most fit model.

8.5 Validity Threats

The results of this study may not be generalizable. This means that the success of prediction using a specific model does not guarantee that the model would be successful on different sets of data. In fact, we have seen that the cubic model make predictions with minimal RE in one part of a release, but big RE values in another part of a release, i.e. predictions of the cubic model in Stage 1 of Release 4 versus predictions of the cubic model for Stage 1 of Release 3. We failed to use COCOMO successfully in effort estimation. This was due to the lack of data. This does not mean COCOMO is not a good effort estimation model. This model has been used widely and successfully over the years. Perhaps by applying the more advanced COCOMO models, we could have seen different results, but the issue we have is that the dataset that is available to us did not have these attributes. Our data was collected by a third party which means the researchers have less control over the quality of the data, which is a threat to internal validity. Construct validity which is concerned with the relation between theory and observation can be compromised by selecting a model among several fit models that might not perform well in predicting effort.

8.6 Conclusion

Effort (cost) estimation is essential to any organization. It is key for management to help in budget planning, including hiring staff, managing resources, etc. The COCOMO model is one of the widely used models that relies mainly on the availability of effort data and Lines of Code. We tried a number of ways to use the basic COCOMO model, but they all failed to provide good fit. Therefore, we did not use this model for effort estimation.

Since we saw a resemblance between cumulative effort growth and CR growth in Section 3.2.3.2, we thought of applying a similar method to how we predicted cumulative CRs for cumulative effort estimation. First, we estimated change-points. Then we divided each release into stages based on changes in cumulative effort growth. Then we applied a multi-stage approach in fitting a model to the existing effort data. Finally, a model is being selected and used for effort prediction. This approach shows predicted effort for up to six months into the future with relatively low RE values. Some models perform better than other models but in general effort prediction has been successful.

Chapter 9

Future Work

This dissertation proposes contributions in two major subjects, CR prediction and effort estimation. We applied our approaches to four releases of a case study and discussed findings. These findings raised more questions that could be addressed in future work.

In future work we would like to extend our research as follows: (1) Applying our curve-fitting methods in CR prediction to various case studies. (2) Investigate generalizability of our effort estimation approach in an evolving software system using more case studies. (3) Improving the quality of SRGM selection approach.

9.1 Application and Comparison of CR Prediction

Regarding CR prediction, we have demonstrated our results for an aerospace software system that has a database of Change Requests (CR) where most CRs are enhancement requests and only a small proportion of CRs are discrepancies. This case study covered model estimation and CR prediction for four releases. It showed an improvement when using a multi-stage approach, specifically the Multi-stage approach with TT over the original curve-fitting approach proposed by Stringfellow

and Andrews [128]. The TT approach was superior to the other approaches in two releases, Release 3 and Release 4. Its performance was equal to the multi-stage approach in Release 2 and it performed worse than the multi-stage approach in Release 1. Therefore, more case studies should be performed to understand the benefits of each approach and the environments where these approaches perform at their best. In addition, the model selection method should be improved to have a more intelligent selection method to select the most fit model. We will elaborate more on these ideas:

9.1.1 Case Study Objectives

This case study could be expanded to test the generalizability of the approaches and the results. In addition to reevaluating the model selection process to improve the predictive ability of these curve-fitting approaches. These approaches are based Stringfellow and Andrews [128] curve-fitting approach.

9.1.2 Case Study Research Question

The research questions derived from the case study objectives and are as follows:

- RQ1: Can CR prediction approaches be used to various CR databases from different systems and application domains?
- RQ2: How accurate are these curve-fitting approaches in predicting CRs for various systems and application domains?
- RQ3: How can we improve the predictive ability of these approaches?

9.1.3 Case Study General Descriptions & Rationale

Various case studies with multiple domains, sizes, multiple software development methodologies and infrastructures should be performed. CR databases from multiple companies with different standards, different business domains and environments should be studied. Different systems have different behaviors in terms of the number of CRs, the frequency of CR submission, the type of changes and the frequency of evolution and change. These case studies should measure and compare the relative error in long-term CR prediction, and the quality of predicted CRs. CR prediction should have a positive relative error to avoid under-estimation of CRs. In fact, the long-term prediction could be extended to provide a year-long prediction instead of only six months to test the effectiveness of the selected models. In fact, from these case studies methodology guidelines could be built for each type of system depending on the system's characteristics.

On the other hand we found that many fit models perform poor prediction although they have a high R^2 value and an estimated cumulative number of CRs that is at least as high as the actual. This could be caused by the early selection of a model when the growth of the model was not clear and this requires more data for the model to be useful. Sometimes more than one model meets the acceptance criteria. In the current approach one of these models was selected randomly. This random selection is risky when a model that performs poorly was chosen instead of another model that might have better performance in the future. As future work, the approaches should be modified to deal better with model selection in case there was more than a candidate model. Perhaps the two models could be used for predictions and the model that performs better as time progresses is the model to be used. This reduces the risk of discarding a fitted model that provides good predictions when

more than one model are candidates to be selected. In addition, the approach could be improved to allow re-estimation once the model performs poorly.

9.2 Effort estimation

In terms of effort estimation our preliminary finding shows CR prediction using COCOMO model was not promising, (Chapter 8). The early attempts to predict effort show that we were using the whole data set for the selected model or parts of the data according to its priority, functional area and time. We then discovered using PCA and correlation analysis that none of the data is strongly correlated to effort, which explains why our early attempts to estimate effort using different sets divided according to their priorities, change type, functional areas and duration were not successful. When analyzing the data in Section 3.2.3.2 we found that there are similarities between the cumulative CR data curve and the cumulative effort curve for all releases. They both have change-points that are similar in some weeks. We applied a multi-stage approach for cumulative effort prediction analogous to the multi-stage CR prediction. We were able to predict effort for up to six months into the future.

9.2.1 Case Study Objectives

Since the results of using the basic COCOMO model were not promising we would like to expand on applying more advance version of COCOMO Effort Adjustment Factors (EAF) which requires change information to be available. In addition, since the multi-stage model was successfully applied, we would like to investigate the applicability and the performance of using the TT approach and how it will this affect the results.

9.2.2 Case Study Research Question

The research questions derived from the case study objectives and are as follows:

- RQ1: Can COCOMO model provide better effort estimation when more detail data is available so we can use the intermediate or advanced COCOMO model?
- RQ2: How can we improve the predictive ability of the multi-stage cumulative effort prediction approach?
- RQ3: Can we apply the TT method to cumulative effort prediction?

9.2.3 Case Study General Descriptions & Rationale

When estimating effort we should look for what data should be available vs. what is commonly available COCOMO has several versions and more factors could be incorporated. If more data was available we can apply the more detailed COCOMO models and hopefully obtain more successful results. Effort estimation using methods for function points instead of SLOC is possible, if editing and change data is available. Effort estimation by analogy is possible if information is available related to similarity of CRs. This data can be found in change information in Clear Case. We currently have attribute data from Clear Quest only.

Predictive ability of using the multi-stage model were promising but sometimes several models meet the acceptance criteria for a certain week to be used for future prediction. The choice of selecting the best of the candidates is random. As future work, a more intelligent choice should be performed. This reduces the risk of selecting a model with poor predictions. In addition to improving the model estimation approach to allow re-estimation when the model performs poorly.

In addition, the applicability of using the multi-stage approach with Time Transformation (TT) could be investigated and compared to the performance of the current multi-stage model used in effort prediction. These case studies could provide improvement over the current prediction results.

Chapter 10

Conclusion

Estimating effort or cost is crucial to any software organization. A reliable prediction system with long-term prediction ability is valuable to management who are concerned with planning budget, staffing and so on. To estimate effort we first need to predict the number of CRs. Software Reliability Growth Models (SRGM) have been used to predict future defects in a software release. Modern software engineering databases contain Change Requests (CR), which include both defects and other maintenance requests. Our goal is to use defect prediction methods to help predict CRs in an evolving legacy system. In situations where maintenance and evolution are based on predefined, competitive contracts, accurate effort (and cost) estimation is crucial, since overestimating will reduce competitiveness of a bid while underestimating risks loses the organization money. Some of these systems are decades old and represent major assets.

This work demonstrates the use of curve-fitting defect prediction methods to predict CRs. It focuses on providing a curve-fit solution that deals with evolutionary software changes but yet considers long-term prediction of data in the full release. We then compare three curve-fit solutions in terms of their predictive ability of CRs. Our data show that the Time Transformation approach (TT) provides more

accurate CR predictions and less under-predicted Change Requests than the other curve-fitting methods. In addition to CR prediction, we investigated the possibility of estimating effort as well. We found that changes in Lines of Code of CRs do not necessarily predict the actual effort spent on CR resolution.

First, we propose a method that uses SLOC data in estimating points of evolution, change-points. We validate that method along with other methods proposed in previous research for change-point estimation for the purpose of defect or CR prediction using reliability models. These methods rely on cumulative CR data to predict change. We found that using SLOC for change-point estimation provides reliable predictions of change-points in several software releases, while it requires less computation overhead than the other two methods, the likelihood ratio method and the control charts method.

Then, we investigate the use of three different curve-fitting approaches that have been used in defect prediction for predicting Change Requests (CR). We applied the first approach [128], which is a curve-fitting approach that does not account for change-points. The predictions showed a low Relative Error (RE) at first, but RE increases dramatically which makes the predictions unreliable in the long term. The multi-stage model based on the work presented by Chi et al. [30] uses change-points to split the release into stages and fit each stage with a different SRGM that is used for CR prediction. This approach had proven to give lower relative error in the predicted values but many times the values are underestimated. The Time Transformation method (TT) [98] [97] uses a multi-stage approach but with changes in the time of CR occurrence as if changes were known since the beginning of the release. This method shows predictions with lower RE than both of the other approaches in two of the four releases and is close to the predictive ability to the

multi-staged model in one release, and worse than the multi-staged model in one release. The two later approaches provide low Relative Error (RE) in general.

Afterwards we investigated the most suitable approach for effort estimation. The basic COCOMO model uses Lines of Code to estimate effort. The model did not fit the data very well, it did not meet the minimum threshold for a model to be considered fit according to our threshold. Therefore, we did not use this model for effort estimation.

We then used cumulative effort to predict future cumulative effort similar to the methods used in CR prediction. First, we estimated change-points, then we divided each release into stages based on changes in effort growth. Then we applied a multi-stage approach in fitting a model to the existing effort data. Finally, a model is being selected and used for effort prediction. This approach shows predicted effort for up to six months into the future with relatively low RE values. Some models perform better than other models but in general effort prediction has been successful.

Finally, we emphasize the importance of early planning for managers of operational software. We find that early model selection risks poor long-term predictions but in general they give managers an idea of how their systems are performing, which assists them in planning. From our case study we found that five data-points (weeks) are enough to model but having at least 10 data-points (weeks) makes CR predictions much more reliable. For further investigation we find that besides having a minimum number of weeks to select a model, there should be a controlled method to select the best among several fit models to perform the best predictions of either cumulative CRs or cumulative effort.

Bibliography

[1] Walid AbdelMoez, Mohamed Kholief, and Fayrouz M Elsalmy. Improving Bug Fix-time Prediction Model by Filtering Out Outliers. In *2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*, pages 359–364. IEEE, 2013.

[2] Anu G Aggarwal, Vikas Dhaka, and Nidhi Nijhawan. Reliability Analysis for Multi-release Open-source Software Systems with Change Point and Exponentiated Weibull Fault Reduction Factor. *Life Cycle Reliability and Safety Engineering*, 6(1):3–14, 2017.

[3] Herman Aguinis and Erika E Harden. *Sample Size Rules of Thumb*. Routledge: New York, 2009.

[4] Mahdieh Ahmadi, Iraj Mahdavi, and AHS Garmabaki. Multi Up-gradation Reliability Model for Open Source Software. In *Current Trends in Reliability, Availability, Maintainability and Safety*, pages 691–702. Springer, 2016.

[5] Allan J Albrecht. Measuring Application Development Productivity. In *Proc. Joint Share, Guide, and IBM Application Development Symposium*, 1979.

[6] Allan J. Albrecht and John E Gaffney. Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. *IEEE Transactions on Software Engineering*, (6):639–648, 1983.

[7] L Alhazzaa and A Amschler Andrews. A Systematic Mapping Study on Software Reliability Growth Models that Consider Evolution. In *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, pages 83–90. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2019.

[8] L Alhazzaa and A Amschler Andrews. Trade-offs Between Early Software Defect Prediction Versus Prediction Accuracy. In *The 2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2019.

[9] L Alhazzaa and A Amschler Andrews. Change Request Prediction in an Evolving Legacy System: A Comparison. In *Transactions on Computational Science and Computational Intelligence*. Springer, 2020.

[10] Lamees Alhazzaa and Anneliese Amschler Andrews. Estimating Changepoints based on Defect Data. In *The 2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 878–883. IEEE, 2018.

[11] Shaukat Ali, Lionel C Briand, Hadi Hemmati, and Rajwinder Kaur Panesar-Walawege. A Systematic Review of the Application and Empirical Investigation of Search-based Test Case Generation. *IEEE Transactions on Software Engineering*, 36(6):742–762, 2010.

[12] Amirhossein Amiri and Saeed Allahyari. Change Point Estimation Methods for Control Chart Postsignal Diagnostics: A Literature Review. *Quality and Reliability Engineering International*, 28(7):673–685, 2012.

[13] Adarsh Anand, Subhrata Das, Deepti Aggrawal, and PK Kapur. Reliability Analysis for Upgraded Software with Updates. In *Quality, IT and Business Operations*, pages 323–333. Springer, 2018.

[14] Carina Andersson. A Replicated Empirical Study of a Selection Method for Software Reliability Growth Models. *Empirical Software Engineering*, 12(2):161, 2007.

[15] Anneliese Andrews and Joseph Lucente. Predicting Incident Reports for it Help Desk. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 678–683. IEEE, 2014.

[16] Anneliese Amschler Andrews, Philip Beaver, and Joseph Lucente. Towards Better Help Desk Planning: Predicting Incidents and Required Effort. *Journal of Systems and Software*, 117:426–449, 2016.

[17] Sara Abbaspour Asadollah, Daniel Sundmark, Sigrid Eldh, Hans Hansson, and Wasif Afzal. 10 Years of Research on Debugging Concurrent and Multicore Software: A Systematic Mapping Study. *Software Quality Journal*, 25(1):49–82, 2017.

[18] Cheng-Gang Bai, Kai-Yuan Cai, Qing-Pei Hu, and Szu-Hui Ng. On the Trend of Remaining Software Defect Estimation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 38(5):1129–1142, 2008.

[19] Néstor Ruben Barraza. Software Reliability Analysis of Multistage Projects. In *2019 Amity International Conference on Artificial Intelligence (AICAI)*, pages 67–73. IEEE, 2019.

[20] Boehm Barry et al. *Software Engineering Economics*. Prentice Hall, New York, 1981.

[21] Keith Bennett. Legacy Systems: Coping with Success. *IEEE Software*, 12(1):19–23, 1995.

[22] Peter M Bentler. *Theory and Implementation of EQS: A Structural Equations Program*. BMDP Statistical Software, 1985.

[23] Peter M Bentler and Chih-Ping Chou. Practical Issues in Structural Modeling. *Sociological Methods & Research*, 16(1):78–117, 1987.

[24] Barry Boehm, Bradford Clark, Ellis Horowitz, Chris Westland, Ray Madachy, and Richard Selby. Cost Models for Future Software Life Cycle Processes: COCOMO 2.0. *Annals of Software Engineering*, 1(1):57–94, 1995.

[25] Barry W Boehm. Software Engineering Economics. *IEEE Transactions on Software Engineering*, (1):4–21, 1984.

[26] George EP Box and George C Tiao. Intervention Analysis with Applications to Economic and Environmental Problems. *Journal of the American Statistical Association*, 70(349):70–79, 1975.

[27] F Calzolari, Paolo Tonella, and Giuliano Antoniol. Dynamic Model for Maintenance and Testing Effort. In *International Conference on Software Maintenance (Cat. No. 98CB36272)*, pages 104–112. IEEE, 1998.

[28] IBM Knowledge Center. SPSS Statistics V24. 0 Documentation. *Retrieved October*, 10:2019, 2016.

[29] Yen-Chang Chang and Lii-Yuh Leu. A State Space Model for Software Reliability. *Annals of the Institute of Statistical Mathematics*, 50(4):789–799, 1998.

[30] Jieming Chi, Kiyoshi Honda, Hironori Washizaki, Yoshiaki Fukazawa, Kazuki Munakata, Sumie Morita, Tadahiro Uehara, and Rieko Yamamoto. Defect Analysis and Prediction by Applying the Multistage Software Reliability Growth Model. In *2017 8th International Workshop on Empirical Software Engineering in Practice (IWESEP)*, pages 7–11. IEEE, 2017.

[31] Kuei-Chen Chiu and Shulan Hsieh. A Discussion of Multiple Learning Effects and Unconscious Behavior in the Software Debugging Process with Variable Potential Errors and Change-points. In *2013 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1656–1660. IEEE, 2013.

[32] Kuei-Chen Chiu and Shulan Hsieh. An Application of Learning Effects for Assessing Work Performance using a Software Reliability Growth Model with Multiple Change-points. In *2013 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1711–1715. IEEE, 2013.

[33] James J Cusick. The First 50 Years of Software Reliability Engineering: A History of SRE with First Person Accounts. *ArXiv Preprint ArXiv:1902.06140*, 2019.

[34] Andrea De Lucia, Aldo Persico, Eugenio Pompella, and Silvio Stefanucci. Improving Corrective Maintenance Effort Prediction: An Empirical Study.

In *Seventh IEEE Workshop on Empirical Studies of Software Maintenance, Florence, Italy*, 2001.

[35] Andrea De Lucia, Eugenio Pompella, and Silvio Stefanucci. Effort Estimation for Corrective Software Maintenance. In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, pages 409–416. ACM, 2002.

[36] Andrea De Lucia, Eugenio Pompella, and Silvio Stefanucci. Assessing Effort Estimation Models for Corrective Maintenance Through Empirical Studies. *Information and Software Technology*, 47(1):3–15, 2005.

[37] Ali Dehghan, Kelly Blincoe, and Daniela Damian. A Hybrid Model for Task Completion Effort Estimation. In *Proceedings of the 2nd International Workshop on Software Analytics*, pages 22–28. ACM, 2016.

[38] L Sandamali Dharmasena and P Zeephongsekul. Fitting Software Reliability Growth Curves using Nonparametric Regression Methods. *Statistical Methodology*, 7(2):109–120, 2010.

[39] William M Evanco. Modeling the Effort to Correct Faults. *Journal of Systems and Software*, 29(1):75–84, 1995.

[40] Felipe Febrero, Coral Calero, and M$^{\underline{a}}$ Ángeles Moraga. A Systematic Mapping Study of Software Reliability Modeling. *Information and Software Technology*, 56(8):839–849, 2014.

[41] Pedro Galeano. The Use of Cumulative Sums for Detection of Changepoints in the Rate Parameter of a Poisson Process. *Computational Statistics & Data Analysis*, 51(12):6151–6165, 2007.

[42] Neha Gandhi, Anu G Aggarwal, Abhishek Tandon, et al. Estimating Reliability for OSS: An Approach with Change-point in Operational Phase. In *2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, pages 248–253. IEEE, 2017.

[43] Amir Hossein Soleiman Garmabaki, Abbas Barabadi, F Yuan, J Lu, and Yonas Zewdu Ayele. Reliability Modeling of Successive Release of Software using NHPP. In *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 761–766. IEEE, 2015.

[44] Amir HS Garmabaki, Anu G Aggarwal, and PK Kapur. Multi Up-gradation Software Reliability Growth Model with Faults of Different Severity. In *2011 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1539–1543. IEEE, 2011.

[45] Amrit L Goel and Kazu Okumoto. Time-dependent Error-detection Rate Model for Software Reliability and Other Performance Measures. *IEEE Transactions on Reliability*, 28(3):206–211, 1979.

[46] Swapna S Gokhale and Robert Mullen. Software Defect Repair Times: A Multiplicative Model. In *Proceedings of the 4th International Workshop on Predictor Models in Software Engineering*, pages 93–100. ACM, 2008.

[47] Swapna S Gokhale and Robert E Mullen. A Multiplicative Model of Software Defect Repair Times. *Empirical Software Engineering*, 15(3):296–319, 2010.

[48] B Gompertz. (1825). On the Nature of the Function Expressive of the Law of Human Mortality, and on a New Mode of Determining the Value of Life Contingencies. *Philos Trans. R. Soc. London, Ser. A*, 115(513):252–253.

[49] DN Goswami, Sunil K Khatri, and Reecha Kapur. Discrete Software Reliability Growth Modeling for Errors of Different Severity Incorporating Change-point Concept. *International Journal of Automation and Computing*, 4(4):396–405, 2007.

[50] Varuvel Antony Gratus and Xavier Pruno Pratibha. Multi-release Software: An Approach for Assessment of Reliability Metrics From Field Data. In *Mining Intelligence and Knowledge Exploration*, pages 475–486. Springer, 2013.

[51] Alaa Hassouna and Ladan Tahvildari. An Effort Prediction Framework for Software Defect Correction. *Information and Software Technology*, 52(2):197–209, 2010.

[52] Jane Huffman Hayes, Sandip C Patel, and Liming Zhao. A Metrics-based Software Maintenance Effort Model. In *Eighth European Conference on Software Maintenance and Reengineering (CSMR), 2004*, pages 254–258. IEEE, 2004.

[53] QP Hu, Rui Peng, Min Xie, Szu Hui Ng, and Gregory Levitin. Software Reliability Modelling and Optimization for Multi-release Software Development Processes. In *2011 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 1534–1538. IEEE, 2011.

[54] Chin-Yu Huang. Performance Analysis of Software Reliability Growth Models with Testing-effort and Change-point. *Journal of Systems and Software*, 76(2):181–194, 2005.

[55] Chin-Yu Huang and Tsui-Ying Hung. Software Reliability Analysis and Assessment using Queueing Models with Multiple Change-points. *Computers & Mathematics with Applications*, 60(7):2015–2030, 2010.

[56] Chin-Yu Huang, Tsui-Ying Hung, and Chao-Jung Hsu. Software Reliability Prediction and Analysis using Queueing Models with Multiple Change-points. In *Third IEEE International Conference on Secure Software Integration and Reliability Improvement, 2009. SSIRI 2009*, pages 212–221. IEEE, 2009.

[57] Chin-Yu Huang and Chu-Ti Lin. Reliability Prediction and Assessment of Fielded Software based on Multiple Change-point Models. In *11th Pacific Rim International Symposium on Dependable Computing, 2005.*, pages 8–pp. IEEE, 2005.

[58] Chin-Yu Huang and Chu-Ti Lin. Analysis of Software Reliability Modeling Considering Testing Compression Factor and Failure-to-fault Relationship. *IEEE Transactions on Computers*, 59(2), 2010.

[59] Chin-Yu Huang, Chu-Ti Lin, and Chuan-Ching Sue. Software Reliability Prediction and Analysis During Operational Use. In *3rd International Conference on Information Technology: Research and Education, ITRE 2005*, pages 317–321. IEEE, 2005.

[60] Chin-Yu Huang and Michael R Lyu. Estimation and Analysis of Some Generalized Multiple Change-point Software Reliability Models. *IEEE Transactions on Reliability*, 60(2):498–514, 2011.

[61] GR Hudson. Program Errors as a Birth and Death Process. *System Development Corporation. Report SP-3011, Santa Monica, CA*, pages 465–484, 1967.

[62] IBM Corp. IBM SPSS Statistics for Windows. `https://www.ibm.com/products/spss-statistics`, 2017.

[63] Ali Idri, Mohamed Hosni, and Alain Abran. Systematic Literature Review of Ensemble Effort Estimation. *Journal of Systems and Software*, 118:151–175, 2016.

[64] Shinji Inoue, Shiho Hayashida, and Shigeru Yamada. Toward Practical Software Reliability Assessment with Change-point Based on Hazard Rate Models. In *2013 IEEE 37th Annual Computer Software and Applications Conference (COMPSAC)*, pages 268–273. IEEE, 2013.

[65] Shinji Inoue and Shigeru Yamada. Optimal Software Release Policy with Change-point. In *IEEE International Conference on Industrial Engineering and Engineering Management, 2008. IEEM 2008*, pages 531–535. IEEE, 2008.

[66] Shinji Inoue and Shigeru Yamada. Change-point Modeling for Software Reliability Assessment Depending on Two-types of Reliability Growth Factors. In *2010 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 616–620. IEEE, 2010.

[67] Shinji Inoue and Shigeru Yamada. Software Reliability Growth Modeling with Change-Point and Its Goodness-of-Fit Comparisons. In *International Conference on Advanced Software Engineering and Its Applications*, pages 354–361. Springer, Berlin, Heidelberg, 2011.

[68] Shinji Inoue and Shigeru Yamada. Software Reliability Measurement with Effect of Change-point: Modeling and Application. *International Journal of System Assurance Engineering and Management*, 2(2):155–162, 2011.

[69] Shinji Inoue and Shigeru Yamada. Software Hazard Rate Modeling with Multiple Change-point Occurrences. In *2014 IEEE International Conference on*

Industrial Engineering and Engineering Management (IEEM), pages 1151–1155. IEEE, 2014.

[70] Madhu Jain, T Manjula, and TR Gulati. Software Reliability Growth Model (SRGM) with Imperfect Debugging, Fault Reduction Factor and Multiple Change-point. In *Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) December 20-22, 2011*, pages 1027–1037. Springer, India, 2012.

[71] Zygmunt Jelinski and P Moranda. Software Reliability Research. In *Statistical Computer Performance Evaluation*, pages 465–484. Elsevier, 1972.

[72] Magne Jørgensen. A Review of Studies on Expert Estimation of Software Development Effort. *Journal of Systems and Software*, 70(1-2):37–60, 2004.

[73] Magne Jørgensen. Forecasting of Software Development Work Effort: Evidence on Expert Judgement and Formal Models. *International Journal of Forecasting*, 23(3):449–462, 2007.

[74] Magne Jørgensen, Ulf Indahl, and Dag Sjøberg. Software Effort Estimation by Analogy and "Regression Toward the Mean". *Journal of Systems and Software*, 68(3):253–262, 2003.

[75] Karama Kanoun, Marta Rettelbusch de Bastos Martini, and Jorge Moreira De Souza. A Method for Software Reliability Analysis and Prediction Application to the TROPICO-R Switching System. *IEEE Transactions on Software Engineering*, 17(4):334–344, 1991.

[76] Karama Kanoun and Jean-Claude Laprie. Software Reliability Trend Analyses from Theoretical to Practical Considerations. *IEEE Transactions on Software Engineering*, 20(9):740–747, 1994.

[77] PK Kapur, RB Garg, Udayan Chanda, and Abhishek Tandon. Development of Software Reliability Growth Model Incorporating Enhancement of Features and Related Release Policy. *International Journal of Systems Assurance Engineering and Management*, 1(1):52–58, 2010.

[78] PK Kapur, Gaurav Mishra, and AK Shrivastava. A Generalized Framework for Modelling Multi Up-gradations of a Software with Testing Effort and Change Point. In *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, pages 129–134. IEEE, 2016.

[79] PK Kapur, Prabhanjan Mishra, AK Shrivastava, and Sunil K Khatri. Multi Release Modeling of a Software with Testing Effort and Fault Reduction Factor. In *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, pages 54–59. IEEE, 2016.

[80] PK Kapur, Hoang Pham, Anu G Aggarwal, and Gurjeet Kaur. Two Dimensional Multi-release Software Reliability Modeling and Optimal Release Planning. *IEEE Transactions on Reliability*, 61(3):758–768, 2012.

[81] PK Kapur, Abhishek Tandon, and Gurjeet Kaur. Multi up-gradation software reliability model. In *2010 2nd International Conference on Reliability, Safety and Hazard (ICRESH)*, pages 468–474. IEEE, 2010.

[82] Syuan-Zao Ke and Chin-Yu Huang. Software reliability prediction and management: A multiple change-point model approach. *Quality and Reliability Engineering International*, 36(5):1678–1707, 2020.

[83] Syuan-Zao Ke, Chin-Yu Huang, and Kuan-Li Peng. Software Reliability Analysis Considering the Variation of Testing-effort and Change-point. In *Proceedings of the International Workshop on Innovative Software Development Methodologies and Practices*, pages 30–39. ACM, 2014.

[84] Dimitri Kececioglu. *Reliability Engineering Handbook*, volume 1. DEStech Publications, Inc, 2002.

[85] Paul Luo Li, James Herbsleb, Mary Shaw, and Brian Robinson. Experiences and Results from Initiating Field Defect Prediction and Product Test Prioritization Efforts at ABB Inc. In *Proceedings of the 28th International Conference on Software Engineering*, pages 413–422. ACM, 2006.

[86] Chu-Ti Lin and Chin-Yu Huang. Software Reliability Modeling with Weibull-type Testing-Effort and Multiple Change-Points. In *TENCON 2005-2005 IEEE Region 10 Conference*, 2005.

[87] Chu-Ti Lin and Chin-Yu Huang. Software Release Time Management: How to Use Reliability Growth Models to Make Better Decisions. In *2006 IEEE International Conference on Management of Innovation and Technology*, volume 2, pages 658–662. IEEE, 2006.

[88] Chu-Ti Lin and Chin-Yu Huang. Enhancing and Measuring the Predictive Capabilities of Testing-effort Dependent Software Reliability Models. *Journal of Systems and Software*, 81(6):1025–1038, 2008.

[89] Chu-Ti Lin, Chin-Yu Huang, and Jun-Ru Chang. Integrating Generalized Weibull-type Testing-effort Function and Multiple Change-points into Software Reliability Growth Models. In *2005 12th Asia-Pacific Software Engineering Conference. APSEC'05*, pages 8–pp. IEEE, 2005.

[90] Michael R Lyu et al. *Handbook of Software Reliability Engineering*, volume 222. IEEE Computer Society Press CA, 1996.

[91] MB Bastos Martini, Karama Kanoun, and J Moreira de Souza. Software-reliability Evaluation of the TROPICO-R Switching System. *IEEE Transactions on Reliability*, 39(3):369–379, 1990.

[92] Microsoft Corporation. Microsoft Excel.

[93] Gaurav Mishra, PK Kapur, and AK Shrivastava. A General Framework for Modeling of Multiple-Version Software with Change-Point. In *Quality, IT and Business Operations*, pages 17–32. Springer, 2018.

[94] Prabhanjan Mishra, AK Shrivastava, PK Kapur, and Sunil K Khatri. Modeling Fault Detection Phenomenon in Multiple Sprints for Agile Software Environment. In *Quality, IT and Business Operations*, pages 251–263. Springer, 2018.

[95] Kjetil Moløkken-Østvold and Magne Jørgensen. A Review of Surveys on Software Effort Estimation. In *2003 ACM-IEEE International Symposium on Empirical Software Engineering (ISESE 2003)*, pages 220–230, 2003.

[96] John D Musa. A Theory of Software Reliability and Its Application. *IEEE Transactions on Software Engineering*, (3):312–327, 1975.

[97] John D Musa and Anthony Iannino. Software Reliability Modeling: Accounting for Program Size Variation Due to Integration or Design Changes. *ACM SIGMETRICS Performance Evaluation Review*, 10(2):16–25, 1981.

[98] John D Musa, Anthony Iannino, and Kazuhira Okumoto. *Software Reliability: Measurement, Prediction, Application*. McGraw-Hill, New York, 1987.

[99] MyAssays Ltd. MyCurveFit Online Curve Fitting. https://mycurvefit.com/.

[100] Vidhyashree Nagaraju and Lance Fiondella. A Single Changepoint Software Reliability Growth Model with Heterogeneous Fault Detection Processes. In *2017 Annual Reliability and Maintainability Symposium (RAMS)*, pages 1–6. IEEE, 2017.

[101] Vidhyashree Nagaraju, Lance Fiondella, and Thierry Wandji. A Heterogeneous Single Changepoint Software Reliability Growth Model Framework. *Software Testing, Verification and Reliability*, 29(8):e1717, 2019.

[102] Frank Niessink and Hans Van Vliet. Predicting Maintenance Effort with Function Points. In *1997 Proceedings International Conference on Software Maintenance*, pages 32–39. IEEE, 1997.

[103] Nidhi Nijhawan and Anu G Aggarwal. On Development of Change Point based Generalized SRGM for Software with Multiple Releases. In *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions)*, pages 1–6. IEEE, 2015.

[104] Jinhee Park, Nakwon Lee, and Jongmoon Baik. On the Long-term Predictive Capability of Data-driven Software Reliability Model: An Empirical Evaluation. In *2014 IEEE 25th International Symposium on Software Reliability Engineering*, pages 45–54. IEEE, 2014.

[105] Karl Pearson. LIII. On Lines and Planes of Closest Fit to Systems of Points in Space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[106] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. Guidelines for Conducting Systematic Mapping Studies in Software Engineering: An Update. *Information and Software Technology*, 64:1–18, 2015.

[107] Dorian Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann, 1999.

[108] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016.

[109] Rakesh Rana, Miroslaw Staron, Christian Berger, Jörgen Hansson, Martin Nilsson, and Fredrik Törner. Evaluating Long-term Predictive Power of Standard Reliability Growth Models on Automotive Systems. In *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*, pages 228–237. IEEE, 2013.

[110] Himani Rastogi, Swati Dhankhar, and Misha Kakkar. A Survey on Software Effort Estimation Techniques. In *2014 5th International Conference-Confluence The Next Generation Information Technology Summit (Confluence)*, pages 826–830. IEEE, 2014.

[111] Per Runeson and Martin Höst. Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering*, 14(2):131, 2009.

[112] Federica Sarro and Alessio Petrozziello. Linear Programming as a Baseline for Software Effort Estimation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 27(3):12, 2018.

[113] SAS Institute Inc. JMP. `https://www.ibm.com/products/spss-statistics`, 2017.

[114] Sumeet Kaur Sehra, Yadwinder Singh Brar, Navdeep Kaur, and Sukhjit Singh Sehra. Research Patterns and Trends in Software Effort Estimation. *Information and Software Technology*, 91:1–21, 2017.

[115] Jalal Shah and Nazri Kama. Extending Function Point Analysis Effort Estimation Method for Software Development Phase. In *Proceedings of the 2018 7th International Conference on Software and Computer Applications*, pages 77–81. ACM, 2018.

[116] Martin Shepperd and Chris Schofield. Estimating Software Project Effort using Analogies. *IEEE Transactions on Software Engineering*, 23(11):736–743, 1997.

[117] Walter A Shewhart. Quality Control Charts 1. *Bell System Technical Journal*, 5(4):593–603, 1926.

[118] Emad Shihab, Yasutaka Kamei, Bram Adams, and Ahmed E Hassan. Is Lines of Code a Good Measure of Effort in Effort-aware Models? *Information and Software Technology*, 55(11):1981–1993, 2013.

[119] Yasuo Shimazu, Kozo Sugiyama, Takashi Kojima, and Eishi Tomida. Some Problems in Ecology Oriented Environmentology, Terrestrial Environmentology II. *The Journal of Earth Sciences, Nagoya University*, 20:31–89, 1972.

[120] Ruchi Shukla and Arun Kumar Misra. Multi-variate Principal Component Analysis of Software Maintenance Effort Drivers. In *International Conference on Computational Science and Its Applications*, pages 273–284. Springer, 2010.

[121] Jagvinder Singh, Ompal Singh, and PK Kapur. Multi Up-gradation Software Reliability Growth Model with Learning Effect and Severity of Faults using

SDE. *International Journal of System Assurance Engineering and Management*, 6(1):18–25, 2015.

[122] Ompal Singh, Deepti Aggrawal, Adarsh Anand, and PK Kapur. Fault Severity Based Multi-release SRGM with Testing Resources. *International Journal of System Assurance Engineering and Management*, 6(1):36–43, 2015.

[123] Ompal Singh, Adarsh Anand, Deepti Aggrawal, and Jagvinder Singh. Modeling Multi Up-gradations of Software with Fault Severity and Measuring Reliability for Each Release. *International Journal of System Assurance Engineering and Management*, 5(2):195–203, 2014.

[124] Ompal Singh, Amir HS Garmabaki, and PK Kapur. Unified Framework for Developing Two Dimensional Software Reliability Growth Models with Change Point. In *2011 IEEE International Conference on Quality and Reliability (ICQR)*, pages 570–574. IEEE, 2011.

[125] VB Singh, Meera Sharma, and Hoang Pham. Entropy Based Software Reliability Analysis of Multi-version Open Source Software. *IEEE Transactions on Software Engineering*, 44(12):1207–1223, 2018.

[126] Abhishek Srivastava, PK Kapur, and Deepti Mehrotra. Modelling Fault Detection with Change-point in Agile Software Development Environment. In *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions)(ICTUS)*, pages 303–308. IEEE, 2017.

[127] Stat-Aids. Control Chart Template. `https://asq.org/quality-resources/control-chart`.

[128] Catherine Stringfellow and A Amschler Andrews. An Empirical Method for Selecting Software Reliability Growth Models. *Empirical Software Engineering*, 7(4):319–343, 2002.

[129] Ushio Sumita and J George Shanthikumar. A Software Reliability Model with Multiple-error Introduction & Removal. *IEEE Transactions on Reliability*, 35(4):459–462, 1986.

[130] Ferdian Thung. Automatic Prediction of Bug Fixing Effort Measured by Code Churn Size. In *Proceedings of the 5th International Workshop on Software Mining*, pages 18–23. ACM, 2016.

[131] Anshul Tickoo, PK Kapur, and SK Khatri. Developing Software Reliability Growth Model for Multi Up Gradations with Faults of Different Severity and Related Release Time Problem. In *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, pages 376–382. IEEE, 2015.

[132] Marvin D Troutt. 10 k Rule of Thumb for Regression. *Encyclopedia of Statistical Sciences*, 2004.

[133] John W Tukey. The Future of Data Analysis. *The Annals of Mathematical Statistics*, 33(1):1–67, 1962.

[134] K Usharani, V Vignaraj Ananth, and D Velmurugan. A Survey on Software Effort Estimation. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 505–509. IEEE, 2016.

[135] Cathrin Weiss, Rahul Premraj, Thomas Zimmermann, and Andreas Zeller. How Long Will it Take to Fix This Bug? In *Fourth International Workshop*

on Mining Software Repositories (MSR'07: ICSE Workshops 2007), pages 1–1. IEEE, 2007.

[136] Peter A Whigham, Caitlin A Owen, and Stephen G Macdonell. A Baseline Model for Software Effort Estimation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24(3):20, 2015.

[137] Peter Wildenauer and Leonhard Eisner. Adaptive Model for a Software Test to Calculate a Residual Error Forecast. *Computing*, 42(2-3):141–158, 1989.

[138] J.W. Wittwer. Box Plot Template. `https://www.vertex42.com/ExcelTemplates/box-whisker-plot.htm`.

[139] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering*. Springer Science & Business Media, 2012.

[140] Alan Wood. Predicting Software Reliability. *Computer*, 29(11):69–77, 1996.

[141] Shigeru Yamada, Mitsuru Ohba, and Shunji Osaki. S-shaped Reliability Growth Modeling for Software Error Detection. *IEEE Transactions on Reliability*, 32(5):475–484, 1983.

[142] Shigeru Yamada, Hiroshi Ohtera, and Hiroyuki Narihisa. Software Reliability Growth Models with Testing-effort. *IEEE Transactions on Reliability*, 35(1):19–23, 1986.

[143] Jianfeng Yang, Jing Chen, and Xibin Wang. Em algorithm for estimating reliability of multi-release open source software based on general masked data. *IEEE Access*, 9:18890–18903, 2021.

[144] Jianfeng Yang, Xibin Wang, Yujia Huo, and Jing Cai. Change point reliability modelling for open source software with masked data using expectation maximization algorithm. In *2020 Global Reliability and Prognostics and Health Management (PHM-Shanghai)*, pages 1–6. IEEE, 2020.

[145] Dianqi Yao and Nan Zhang. A Queue Theory-Based Approach to Software Reliability Assessment. In *2018 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, pages 244–250. IEEE, 2018.

[146] Hongyu Zhang, Liang Gong, and Steve Versteeg. Predicting Bug-fixing Time: An Empirical Study of Commercial Software Projects. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 1042–1051. IEEE Press, 2013.

[147] Jing Zhao, Hong-Wei Liu, Gang Cui, and Xiao-Zong Yang. Software Reliability Growth Model with Change-point and Environmental Function. *Journal of Systems and Software*, 79(11):1578–1587, 2006.

[148] Yuxin Zhao, Chengcheng Wan, Feng Gao, and Shuai Chang. Change Points Estimation and Simulation for Software Reliability Model. In *2013 International Conference on Measurement, Information and Control (ICMIC)*, volume 1, pages 434–438. IEEE, 2013.

[149] Mengmeng Zhu and Hoang Pham. A Multi-release Software Reliability Modeling for Open Source Software Incorporating Dependent Fault Detection Process. *Annals of Operations Research*, pages 1–18, 2017.

[150] Mengmeng Zhu and Hoang Pham. Environmental Factors Analysis and Comparison Affecting Software Reliability in Development of Multi-release Software. *Journal of Systems and Software*, 132:72–84, 2017.

[151] Feng-Zhong Zou. A Change-point Perspective on the Software Failure Process. *Software Testing, Verification and Reliability*, 13(2):85–93, 2003.