2012-09-06

A Methodology for Analyzing Cost and Cost-Drivers of Technical Software Documentation

Sun, Bo

UNIVERSITY OF CALGARY

A Methodology for Analyzing Cost and Cost-Drivers of Technical Software Documentation

by

Bo Sun

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

August 2012

UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "A Methodology for Analyzing Cost and Cost-Drivers of Technical Software Documentation" submitted by Bo Sun in partial fulfilment of the requirements for the degree of Master of Science.

Supervisor, Dr. Guenther Ruhe
Department of Computer Science &
Department of Electrical and Computer Engineering

Co-supervisor, Dr. Vahid Garousi
Department of Electrical and Computer Engineering

Dr. Frank Maurer
Department of Computer Science

Dr. Zongpeng Li
Department of Computer Science

Dr. Diwakar Krishnamurthy
Department of Electrical and Computer Engineering

Date

Abstract

Software documentation is an important impact factor to achieve high software maintainability, especially for those large-scale complex changing legacy systems. The question: "how much documentation is enough", is concerned by organizations who are turning their software process to agile development that claims "just enough" documentation. It is therefore important to be able to understand the cost of documentation activities, and what are the underlying cost-drivers, in order to monitor, control and improve documentation practice. However, there is a general lack of such studies dedicated for software documentation cost and cost-drivers.

To address this need, a systematic methodology is proposed to analyze cost and cost-drivers of technical software documentation. The methodology primarily consists of the definition of documentation cost and cost-driver metrics, mining software repositories with tool support for automatic measurements, and cost-driver analysis. The main contributions of this thesis are to provide a practical way to understand documentation cost from the perspectives of single document, one documentation type and each author, and to identify underlying cost-drivers towards documentation process improvement. Results from an initial validation from an industrial case study at NovAtel, a leading provider for a comprehensive line of Global Navigation Satellite System (GNSS) products, are reported.

Acknowledgements

First of all, I would like to thank my supervisors, Dr. Guenther Ruhe and Dr. Vahid Garousi, for sharing their profound knowledge on scientific methods and software engineering research, and for their continuous support, encouragement and guidance throughout this work. Their suggestions and advices always made a lot of difference. Accomplishing this work would have been impossible without their supervision and assistance.

I would also like to thank Dr. Diwakar Krishnamurthy, Dr. Frank Maurer, and Dr. Zongpeng Li, my thesis committee members, for revising this work and providing useful help and feedbacks.

I am also grateful to the support from the technical staff of NovAtel Inc. Special thanks to Brian D. Smith, Eric Tong and other engineers for their commitment and providing the necessary information for the case-study phase of this work.

Table of Contents

List of Figures

List of Tables

List of Acronyms

GNSS	Global Navigation Satellite System
XP	Extreme Programming
RQ	Research Question
MSR	Mining Software Repository
CRD	Collaborative Research and Development
CVS	Concurrent Versions System
COCOMO	Constructive Cost Model
SLOC	Lines of Source Code
DLOC	Lines of Source Code Documentation
SM	Systematic Mapping
UML	Unified Modeling Language
DCCDA	Documentation Cost and Cost-Driver Analyzer
SDLC	Software Development Lifecycle
GLM	Generalized Linear Models
GQM	Goal-Question-Metric
LCE	Lifecycle Effort
LCC	Lifecycle Cost
VIF	Variance Inflation Factor
GEXF	Graph Exchange XML Format
MMRE	Mean Magnitude of Relative Error
RMSE	Root Mean Squared Error

Chapter One - **Introduction**

1.1 Introduction and Motivation

Legacy software organizations are constantly under pressure of modifying their software because of introducing new hardware or changing business environment [1]. There is a general agreement that evolving software is more difficult than developing software from scratch. It adds continuously to the content, size and complexity of software product. Such maintenance activity often requires patching (or fixing), enhancing, or extending the existing software portion of the system. "Maintenance typically consumes 40 to 80 percent (average, 60 percent) of software cost. Therefore, it is probably the most important lifecycle phase of software" Glass [2].

Software maintainability, defined as the ease or simplicity with which a software system can be maintained [3], is therefore a key characteristic of successful software. Lientz et al [4] classified maintenance tasks into four key types: *corrective*, *perfective*, *adaptive* and *preventive* maintenance tasks. For legacy systems, software maintainers were often not involved in the original design of the software system being changed, or they would probably have forgotten some details. Therefore, software maintenance is often costly and error-prone for legacy systems.

Software development documentation, if correct, complete and consistent, is considered as a memory of software evolution, assisting maintainers remain in intellectual control of complex changing software systems [5]. Typical types of such documentation include requirements, specifications, architectural (high-level) design, detailed design, as well as low level information such as source code comments. They are supposed to help maintainers to comprehend the program or system and accomplish subsequent manipulations or modifications.

The attitude against documentation is that documentation is always perceived as a costly activity. Documentation is also difficult to maintain, under the typical time pressure which is

common for software industry [6]. Agile methods claim that the goal is to produce software and documentation is only necessary if it helps to reach the goal. For example, Scott W. Ambler [7] believed “Create documentation only when you need it at the appropriate point in the life cycle” in agile development, and “Update documentation only when it hurts”. Maurer and Martel [8] suggested that the focus of Extreme Programming (XP) for small teams should be “producing executable code and automated test drivers” instead of “paper-based requirements and design documentation” .

To find an appropriate level of documentation, it is important to investigate the cost spent on documentation activities and to understand what the underlying cost-drivers are. This analysis should be done in two levels, considering both documentation artifact lifecycle and each version of it, so that the cost on each document could be further combined with its usage information to determine whether the gained benefits justify the cost. This is particularly true for those organizations that are turning their software process to agile development which claims “just enough” documentation.

To address the above needs, the author has proposed a practical methodology with accompanied tool support. Its main goal is to extract documentation artifacts from relevant software repositories and assess the lifecycle cost of each single or multi-version document based on defined cost metrics. The rationale behind mining software repositories for documentation cost is considering that it is easy to implement in a real time-driven environment and results are based on evidence in data repositories.

A subsequent cost-driver analysis process which utilizes the mined cost information is able to reveal underlying cost-drivers within documentation properties. By combining the output of this

thesis with documentation usefulness information measured by other students, we aim to have a comprehensive evaluation of documentation cost and benefits.

1.2 Goal and Research Questions

This study is part of a three-year Collaborative Research and Development (CRD) project with an industrial partner NovAtel, "Tuning of Artifact and Process Parameters towards Optimized Maintenance" [9]. The overall goal of this project is to lower the cost of software development and maintenance meanwhile without making trade-off to product delivery and quality.

Documentation is considered important for communication and collaborative development in NovAtel. However, the question "how much documentation is enough?" is an issue. Figure 1.1 gives the overview of the planed phases of this CRD project. This thesis focuses on analyzing documentation cost. By combing the results of documentation benefits analysis, other team members plan to evaluate the cost and benefits of documentation and optimize documentation process towards the overall project goal.

Under the umbrella of this CRD project, the goal of this study is to develop a practical methodology to objectively evaluate the cost of documentation artifacts, and to identify what are the underlying cost-drivers. Based on the above goal, the following research questions are raised. To extract detailed information for each of the questions, each question is divided into sub-questions.

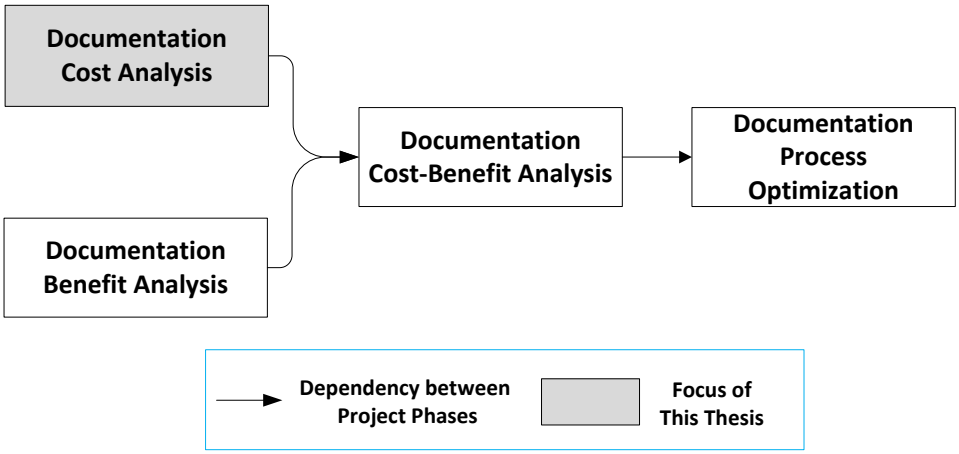

Figure 1.1-Phases of the CRD project

- **RQ1-Documentation cost:** How does mining software repositories (MSR) help to move subjective measurement of documentation cost to objective measurement?

  This question focuses on objectively measuring documentation cost based on data evidence in relevant repositories. Most related works from literature are focused on subjective approaches, e.g., using questionnaires or manually recording.

  - **RQ1.1-Cost per document**: What is the cost spent on each single or multi-version documentation artifact?
  - **RQ1.2-Cost per documentation type**: What does the cost spending vary across different documentation types (e.g., conceptual designs, test plans)?
  - **RQ1.3-Cost distribution over time**: How does the documentation cost distribute over different time periods?
  - **RQ1.4-Cost by person**: How does the documentation cost spending vary across authors?

- **RQ1.5-Time-efficiency on documentation**: how time-efficient are people on writing documentation?

- **RQ2-Documentation Cost-Drivers:** What are the most significant cost-drivers in documentation properties that drive the cost of documentation over time?

  This question aims to identify the causal relationship between documentation cost-drivers and cost.

  - **RQ2.1-Document lifecycle cost-drivers**: What are the main cost-drivers that impact the lifecycle cost of a multi-version documentation artifact?
  - **RQ2.2-Document revision cost-drivers**: What are the main cost-drivers across different versions of a documentation artifact?

1.3 Contributions of this Thesis

The contributions of this work are five-fold: (1) to investigate how existing approaches deal with documentation cost, (2) to propose a practical methodology for assessing documentation cost and cost-drivers by mining software repositories, (3) to automate the fine-grained measurement of cost and cost-driver metrics by providing tool support, (4) to evaluate the feasibility of proposed method by conducting an industrial case study, (5) to contribute initial insights to the body of knowledge in software engineering regarding software documentation cost and cost-drivers.

1.4 Thesis Organization

The remainder of this thesis is structured as follows. Chapter Two introduces background information and Chapter Three analyzes related work. Chapter Four proposes a systematic methodology for how to investigate documentation cost and cost-drivers. 0 describes the tool support for the proposed methodology to make it more practical in real context. In Chapter Six,

an industrial case study is designed to evaluate the feasibility of the methodology. The results from this case study are analyzed in Chapter Seven and Chapter Eight for documentation cost and cost-drivers, respectively. Finally, Chapter Nine concludes the whole thesis and points out some directions for future work.

Chapter Two - **Background Information**

2.1 Mining Software Repository

2.1.1 MSR Data Sources

Mining Software Repositories (MSR) aims to analyze the rich data (artifacts) available in software repositories to uncover interesting and actionable information about software systems and projects [10]. Software repositories contain artifacts that are produced and archived during software evolution, and several examples are listed in Table 2.1.

From a general perspective, they are summarized into three main categories [10, 11].

- **Historical repositories** which record information about the evolution and progress of a project, such as source control repositories (e.g., CVS), bug repositories (e.g., Bugzilla or JIRA), and communication archives (e.g., e-mail). Often these data exist for the entire duration of a project and can represent thousands of versions with years of details about the development.
- **Run-time repositories** contain information about the execution and the usage of an application at a single or multiple deployment sites, such as deployment logs.
- **Code repositories** such as Sourceforge.net and Google Code contain the source code of various applications developed by several developers.

Researchers mine data and metadata from above software repositories to extract pertinent information and therefore guide decision processes in modern software projects. For instance, historical repositories (version history of source code) were mined to capture the hidden dependencies between classes caused by addition or modification of a particular class [12]. Some researchers combined the information from CVS log file with “Bugzilla” to study the question which change properties may lead to problems [13]. Run-time repositories could be used to

detect execution anomaly by identifying dominant execution or usage patterns across deployment.

Table 2.1-Examples of software artifacts

Repository	**Artifacts**	**Description**
Archived documentation	Requirement, Conceptual/detail designs, Test plans, etc.	They document all the requirement specifications, design, test procedures and other standards that regulate a software project.
Source control repositories	Source code with meta-data	They track all the changes to the source code along with meta-data about each change, e.g., the developer who submitted the change, the time the change was performed and a short message describing the change.
Bug repositories	Bug report, Feature request	Bug repositories track the evolution history of bug reports or feature requests that are reported by users and developers of large software projects, e.g., "Bugzilla" and "JIRA"
Archived communications	Mailing lists, Emails, Messages	These repositories track discussions about various aspects of a software project throughout its lifetime.
Deployment repositories	Deployment logs	These repositories record information about the execution of a single deployment of a software application or different deployments of the same applications.
Code repositories	Source code, Code comments	These repositories archive the source code for a large number of projects, such as sourceforge.net and Google code.

2.1.2 MSR Application Areas

A comprehensive literature survey for MSR works prior to 2006 was presented by [14] via four dimensions: the type of software repositories mined (what), the purpose or software engineering task (why), the adopted/invented methodology used (how), and the evaluation

method (quality). Over 80 MSR studies were surveyed by following the four proposed dimensions. Two high-level classes of MSR application areas were utilized in the survey.

- **Market-Basket Question** asks for a set of rules or guidelines describing situations of trends or relationship. It is widely used in describing data mining problems. For example, the occurrence of *A* likely would cause the subsequent occurrence of *B*.
- **Prevalence Question** includes quantitative metrics or boolean queries, such as "how many lines of code are added/deleted/modified?" or "how many and which of functions are reused?"

The methods to address above questions identified by [14] include:

- **Metadata analysis** uses the metadata stored in software repositories for a variety of purposes, e.g., couplings, change patterns and etc.
- **Static source code analysis** extract facts and other information from versions of a software system, e.g., function usage patterns, incomplete refactoring and etc.
- **Source code differencing and analysis** derives and expresses changes between versions of source code in a more fine-grained manner, i.e., syntax and semantic.
- **Software metrics quantitatively** evaluate various aspects of software products, projects and processes.
- **Visualization methods** utilize visual representation of data to support software maintenance and evolution.
- **Clone-detection methods** detect tentative clones existing in source code.
- **Frequent-pattern mining** uncovers software entities which frequently co-change.

- **Information-retrieval methods** apply to textual data (e.g., CVS comments, bug reports and emails) for various purposes, such as change prediction and new developer assistance.
- **Classification with supervised learning** build classification or prediction models for triage bug reports or other purposes**.**
- **Social network analysis** discovers developer roles, contributions and associations in software development.

A more recent survey [15], dedicated on the "Purpose", classified most MSR works after 2006 into the following application domains, which was claimed to represent the state-of-the-art work in MSR.

- Identifying and Predicting Software Quality
- Identifier Analysis and Traceability
- Clone Detection
- Process Improvement and Software Evolution
- Social Aspects in Software Development
- Recommender Systems and Interactive Systems

2.1.3 MSR for Process Improvement and Software Evolution

Since the objective of this thesis is to study documentation cost issue during software evolution and eventually serve for software (documentation) process improvement, the related MSR work for process improvement and understanding software evolution in [15] is listed below in Table 2.2, along with a specific MSR task in second column and mined artifacts in last column.

Table 2.2-Summary of MSR studies for process improvement and software evolution

Ref.	**MSR Tasks for Software Process Improvement**	**Mined Artifacts**
[16]	Who should triage a bug	Bug report
[17, 18]	Identify the trends of code commits	Source code and metadata
[19]	How to file a good bug report	Surveys; Bug report and metadata
[20]	Predict locations of future refactoring	Source code and metadata
Ref.	**MSR Tasks for Understanding Software Evolution**	**Mined Artifacts**
[21]	Study the evolution of software compilations	Source code
[22]	Monitor software process compliance	Source code and metadata, Mailing lists and Bug reports
[23]	Evaluate software readability	Source code
[24]	Identify change couplings	Source code and metadata

2.1.4 In This Thesis

In this thesis, Section 4.4 describes in detail the data mining techniques for modeling the relationship between underlying cost-drivers and documentation cost, and Section 6.3 presents the process of applying them in the case study context. Chapter Six describes the types of documentation artifacts to be mined and date collection process.

To ease the burden of data mining and analysis process, 0 automates the data collection, pre-processing and all needed measurements for analysis, which eliminates the measurement overhead issue.

2.2 Software Documentation

2.2.1 Definition of Software Documentation

Software documentation is a mixed concept. In early work, software documentation refers to end-user documentation, e.g., product manual. Barker defined software documentation as "The design, planning, and implementation of any interface element, written and online, of a software system to enhance the system's usability" [25]. From this definition, documentation refers to software product manuals that were written for guiding end-user for the usage of systems. It does not concern with development documentation.

Andrew Forward defined software documentation as, "Documentation is an artifact whose purpose is to communicate information about the software system to which it belongs. Common examples of such documentation include requirement, specification, detailed design, and architectural documents, as well as low level design information such as comments in the source code" [6]. In this definition, software documentation is expressed for the usage of communication among software engineers, and belongs to development documentation.

In this thesis, we only concerns with software development documentation, excluding the non-relevant technical writing, such as software product manual.

2.2.2 Documentation Usage across Software Development Lifecycle

Software documentation can play a number of roles during the development and maintenance of a software product. Parnas [26] recently summarized the various ways that documentation may be used. For example, the typical usages of documentation are as follows:

- "Design through documentation": writing documents to record and communicate design decisions.

- “Documentation based design reviews”: reviewing design decisions based on design documentation.
- “Documentation based code inspections”: inspecting the actual code based on a document that specifies what the code should do.
- “Documentation based revisions”: relevant documentation aids program comprehension for change tasks in order to reduce cost and delays in maintenance phase.

As part of a systematic mapping of software documentation related studies [27], we proposed a conceptual model to summarize the usage of development documentation. Figure 2.1 summarizes the documentation usage across software development lifecycle (SDLC).

In this model, it is assumed that software engineer needs to perform development tasks. These tasks are classified into two categories: pre-maintenance tasks and maintenance tasks. The former category refers to the tasks performed prior to maintenance phase, including requirement elicitation, system design, implementation and testing. Maintenance tasks are further categorized using the classification proposed by Lientz et al [4]: *corrective*, *perfective*, *adaptive* and *preventive* maintenance tasks. Each maintenance task consists of two steps, including comprehending the program and subsequent manipulation or modification, and testing.

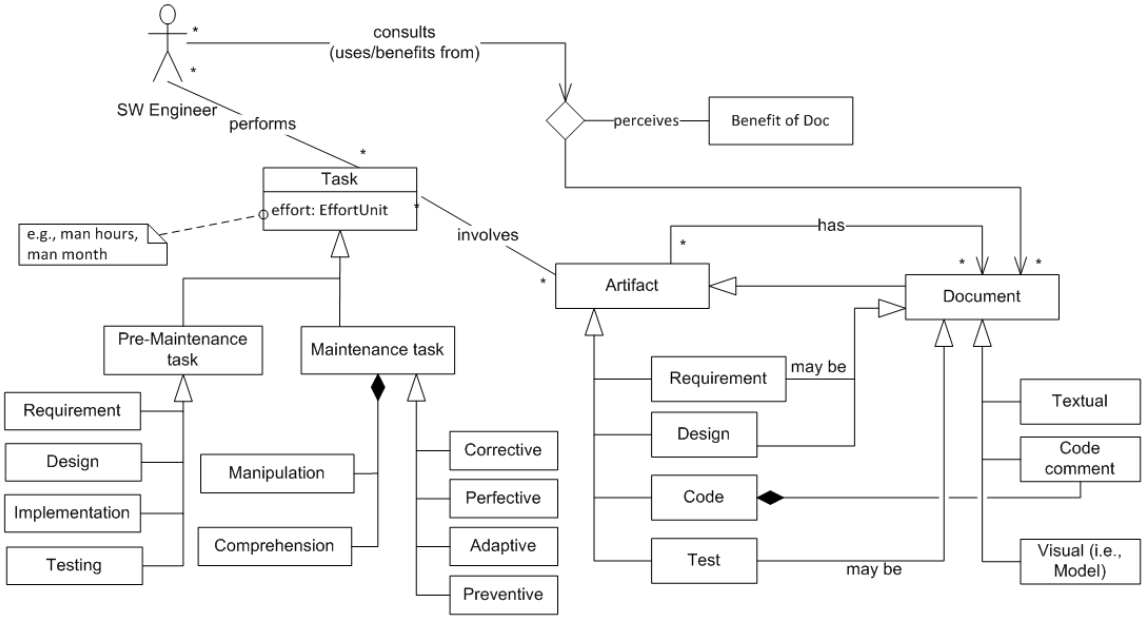

Figure 2.1-Documentation usage model across software development lifecycle [27]

While performing the tasks, Software Engineers need to access to existing artifacts. Documents are modeled as a subclass of artifact. More specifically, important developmental information can be documented in a document artifact. In terms of the format, documents can be presented in pure textual or mixed with visual aid (e.g., graphs charts or in the form of code comments).

2.2.3 Documentation Cost and Benefits

As recommended by value-based software engineering [28], it is necessary to assess software documentation by considering the cost and benefits at the same time, so that we can judge whether the gained benefits outweigh the cost spent on documentation.

In Figure 2.2, a high-level conceptual model takes requirements and design documents for example and demonstrates the cost-benefit aspects of documentation. Documentation cost surrounds the activities in the left two columns, including Creating First Draft, Reviews and Revisions. These iterations may generate multiple draft versions and approved versions of a

document at different time points. Once a version is approved, it can be used to help pre-maintenance tasks or maintenance tasks, illustrated by the right two columns in Figure 2.2. The benefits are measured very often by the reduction of effort (time).

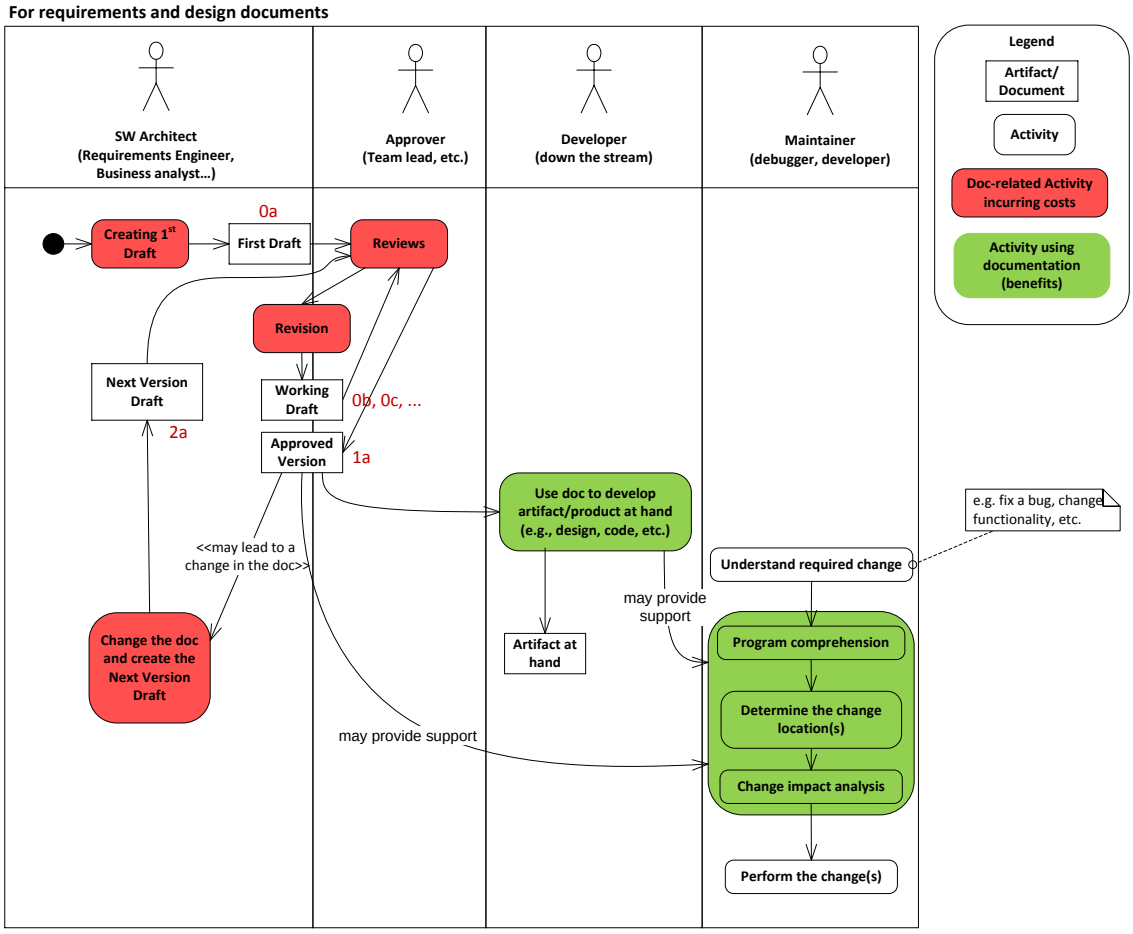

Figure 2.2-Documentation cost and benefit model

2.2.4 In This Thesis

This thesis focuses on studying software documentation cost and underlying cost-drivers. A set of metrics on documentation cost and cost-drivers are defined in Section 4.3. Section 6.2 specifies the types of documentation to study and the process of data collection in case study.

Chapter Seven analyzes the results of mining documentation cost in the context of the case study project.

2.3 Software Cost and Cost-Drivers

Software development is a costly process, and it is even the case for software maintenance. Accurate estimation of software cost will assist decision-making on resource allocation and project scheduling. On the other hand, it is important to know what are the cost-drivers that impact maintenance activities in order to achieve a high degree of software maintainability.

2.3.1 Software/Documentation Cost Estimation

Software cost/effort estimation is the process of predicting the cost/effort required to develop or maintain software. Cost estimates are used as input to project plans, iteration plans, budgets, and investment analysis etc.

Software researchers have been long trying to estimate software cost. A systematic review [29] on software cost estimation studies until 2007 identified 304 software cost estimation papers in 76 journals. Most of them focused on building formal models to estimate software cost. A high number of model building approaches have been applied, such as regression analysis, case-based reasoning, classification, regression tree, neural network, Bayesian statistics and etc.

Perhaps the most common cost estimation method today is Constructive Cost Model (COCOMO). The initial COCOMO model, Basic COCOMO, uses a basic regression formula with program size and project characteristics. Its current version, COCOMO II [30], has been updated and designed to estimate the software effort with the consideration of cost-drivers from software requirements, design, implementation, and testing phases.

However, software documentation, which has gained significant importance and consumed a large amount of cost in software development, was only considered in a few cost estimation models as presented afterwards.

For early cost estimation models, documentation factor was not considered. As the importance of documentation progressively increased, the effort to build documentation increased as well as the total project effort. NASA reported that documentation might require at least 11% of total project effort [31]. Then, NASA [32] proposed a lineal model to estimate documentation size ($Pages = 34.7*(KLOC)^{0.93}$).

In COCOMO II [30], the factor, software documentation, was presented to have an impacting range of 1.52, which is greater than, for example, programming experience and development tools. However, previous versions, Basic COCOMO and COCOMO 81 did not consider documentation factor.

Rosado et al. [33] conducted an experiment with university students and recorded the actual ratio between documentation effort and total development effort of a software project. Their empirical evidence suggested that COCOMO II should extend and calibrate software documentation variable to three ranges, Nominal (1.00), High (1.11) and Very High (1.23), in order to accurately capture different levels of documentation intensity in reality.

2.3.2 Cost-Driver Analysis

Cost-driver analysis has been widely used in software engineering, to understand the factors that influence the cost of software development, maintenance and evolution.

Briand and Wuest [34] investigated the cost-drivers within design properties that might impact the development cost in Object-Oriented systems. They built a linear regression model

between class size, design metrics and the effort upon developing, testing and maintaining each class. Those metrics that were significantly correlated with cost were concluded as cost-drivers.

Li et al. [35] analyzed 1,400 software defect reports from two software organizations. To understand what led to high corrective maintenance cost, they also looked into defect descriptions and recorded discussions between developers in the course of correcting defects. Several cost-drivers have been identified by statistical methods, such as software size, complexity, and maintainers' experience etc. They also concluded that cost-drivers for corrective maintenance might be different from company to company.

Nguyen et al. [36] conducted a controlled experiment with 23 graduate students to assess effort distributions in three types of maintenance, enhancive, corrective and preventive. Their study suggested three cost-drivers of software maintenance, metrics of SLOC added, modified, and deleted.

To understand the cost-drivers of software evolution, Benestad et al. [37] analyzed 336 change tasks from two software companies. Their quantitative analysis found that "Dispersion of changed code" and "Volatility of the requirements" were two major cost-drivers. In addition, other underlying cost-drivers that were revealed by the analysis of the qualitative interviews were "Difficulties in comprehending dispersed code" and "Difficulties in anticipating side effects of changes".

However, there is no dedicated work in the literature on studying software documentation cost-drivers.

2.3.3 In This Thesis

In this thesis, we adapted previous cost-driver analyses of software (source code) evolution to analyze the cost-drivers of documentation evolution. We use a similar cost-driver analysis

process as proposed by Briand and Wuest [5]. However, our analysis was applied to two different levels in order to capture cost-drivers for both individual documentation artifact lifecycle and each version of it (Section 6.3). Moreover, we referred to the cost-drivers used by Benestad et al [37] for source code evolution, and adapted those that are applicable to documentation artifacts (Section 4.3.1.3).

2.4 Chapter Summary

This chapter introduced the knowledge of the following software research domains, Mining Software Repository, Software Documentation, and Software Cost and Cost-Drivers. In this thesis, we intend to objectively analyze software documentation cost and cost-drivers through mining relevant data repositories.

Chapter Three - **A Systematic Mapping of Cost, Benefit and Quality of Technical Software Documentation**

We conducted a systematic mapping (SM) study [27] to synthesize the existing findings in the literature about software documentation cost/benefit/quality aspects. In this chapter, we briefly present the process of performing this SM study and the results specifically related to documentation cost aspect.

Section 3.1 discusses briefly the research goal and process of performing this SM study. The results regarding focuses of software documentation studies are presented in 3.2. In Section 3.3 we discuss in detail the documentation cost related studies. Section 3.4 concludes this SM study and explains how this thesis helps to fill in the gap in the literature.

3.1 Research Goal and Process of Systematic Mapping

The goal of this SM study is to systematically review the state-of-the-art in analyzing benefit/cost/quality of software documentation within development lifecycle, to identify the weaknesses and strengths, and to find out the recent trends and directions in this field from the view point of researchers and scientists in this area. Documentation artifacts are specifically restricted to development documents, excluding other types of technical documents, e.g., user manual.

This SM is carried out based on the guidelines provided by Petersen et al. [38], and Kitchenham and Charters [39]. In designing the methodology for this SM, methods from several other SMs such as [40-42] were also incorporated.

The process that lies at the basis of this SM is outlined in Figure 3.1. This process has four phases, 1) Article selection, 2) Classification Schema/Map, 3) Systematic mapping, 4) Trends,

Bibliometrics and Demographics. For the details of each phase, please refer to the descriptions in Sections 5-8 in the SM paper [27].

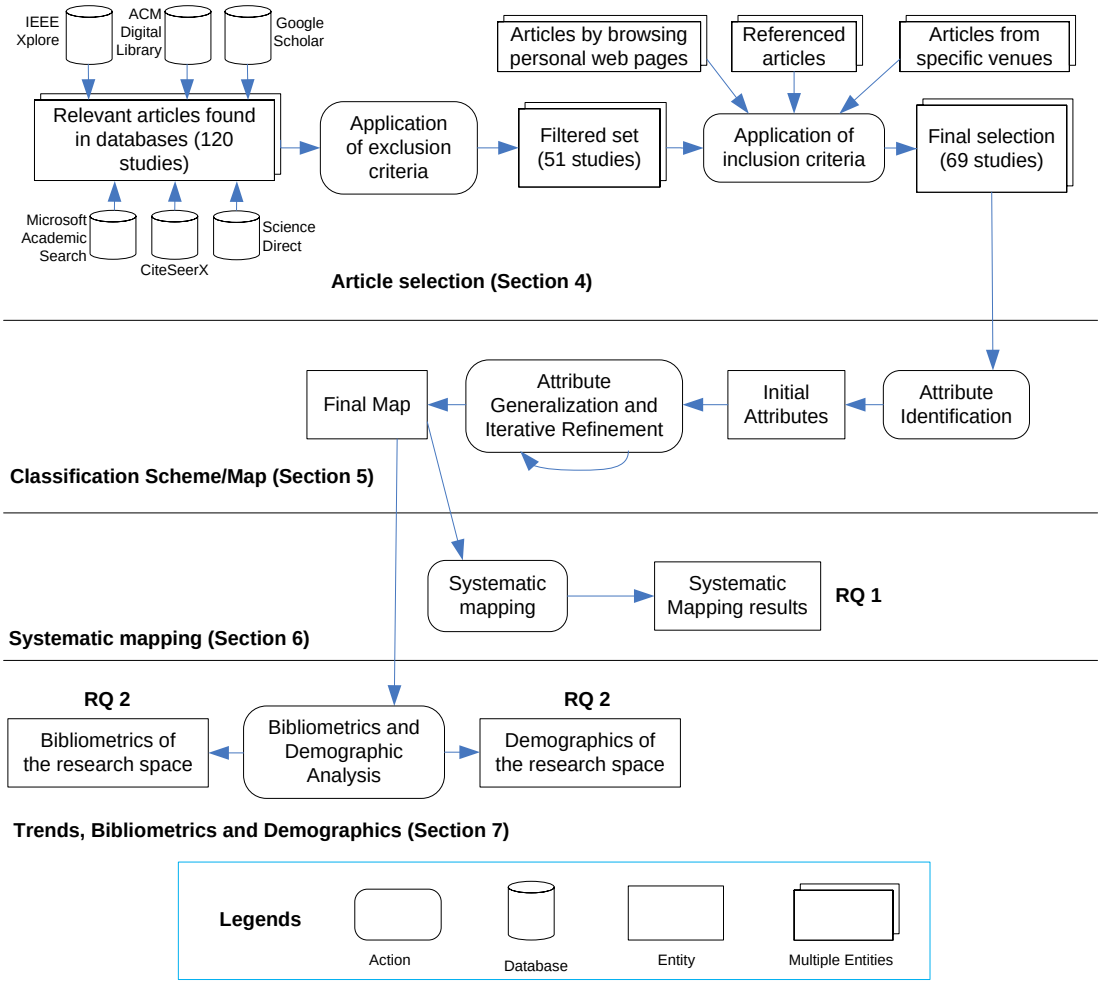

Figure 3.1- Research process of SM [27]

The final pool of selected publications has been published as an online repository using Google Docs, and is accessible publically online at [43]. We plan to update the online repository at least once each year and to add relevant material published in the future.

In this thesis, we only choose to present the findings regarding documentation cost from the SM study.

3.2 Focus on Documentation Cost, Benefit and Quality

The histogram in Figure 3.2 shows the distribution of studies that discuss each documentation aspect we are concerned with. 69 publications on documentation cost/benefit/quality were found, and some of them concerned more than one aspect, e.g., documentation benefit and quality. From the chart we can see that 48 studies (71%) discuss documentation quality, followed by 37 studies (54%) on usage/benefit. Surprisingly, only 12 studies (18%) in total are related to documentation cost.

From Figure 3.2, we can notice that only a very small proportion of research work conducted on documentation cost. This is worth the attention of research community.

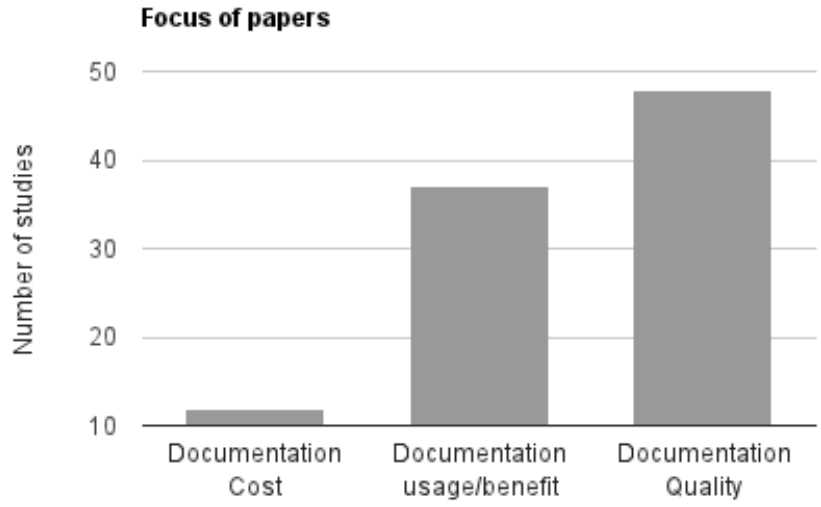

Figure 3.2-Focuses of software documentation studies [27]

3.3 Software Documentation Cost Attributes

One of the research question concerned by this SM study is “what are the cost-related attributes of software documentation?”. This section presents the results of this SM study to this question.

Table 3.1 shows the summary of the 12 works about software documentation cost from the mapping study. The process of extracting data related to documentation cost was iterative. Two rounds of reviews among the authors were conducted.

In Table 3.1, the motivation of each work and the documentation artifacts under study are compared. Then documentation cost attributes, their corresponding metrics/measurements and degree of evidence (evaluation) are compared. If the attribute is discussed in paper together with quantitative evidence support (e.g., survey data, control experimental results, etc.), then the attribute has a degree of two. In contrast, if that attribute is only mentioned or discussed in qualitative manner without any quantitative validation or evaluation, then such attribute is assigned one, which is lower than the degree of two.

Figure 3.3 summarizes the distribution of studies that discuss documentation cost related attributes in terms of cost attributes and degree of evidence. Among our results, only five studies (7%) discuss the development/production cost of software documentation. Four studies (6%) concern documentation maintenance cost. From this statistics we can see that there are a very limited number of researchers conducting studies on documentation cost. In other words, documentation cost-related aspects seemed to be neglected by most researchers.

Table 3.1-Summary of related work

Ref.	**Motivation**	**Doc Artifacts**	**Cost Attributes**	**Metrics/ Measurement**	**Degree of Evidence**
[7]	Strategies of Agile/Lean documentation	Requirement; Design; Process	Development cost		1
[44]	Evaluating cost/benefits of UML	Requirement and Design	Maintenance cost	Time (manually recording)	2

[45]	Identifying factors impacting the size of code comments	Code comments	Document size	Size of source code documentation (DLOC)	2
[33]	Assessing documentation development effort	Requirement; Design; Process	Development cost; Document size	Time (manually recording); Number of characters	2
[46]	Automating documentation quality detection	Requirement; Design; Process	Maintenance cost; Document size	Effort (qualitatively questionnaire); Number of words	2
[47]	Designing documentation to facilitate development	Design; Code comments	Development cost		1
[48]	Measuring the quality of code comments	Code comments	Document size	Number of words	2
[49]	Perceptions of documentation in Agile	Generic	Development cost	Time (qualitatively questionnaire)	2
[50]	Understanding documentation usage in practice	Generic	Maintenance cost; Document size	Time (qualitatively questionnaire)	2
[51]	Guidelines on how to document	Requirement; Design; Comments	Other		1
[52]	Guidelines on how to document	Generic	Development cost; Other		1
[5]	Impact of UML on software maintenance	Requirement and Design	Maintenance cost	Time (manually recording)	2

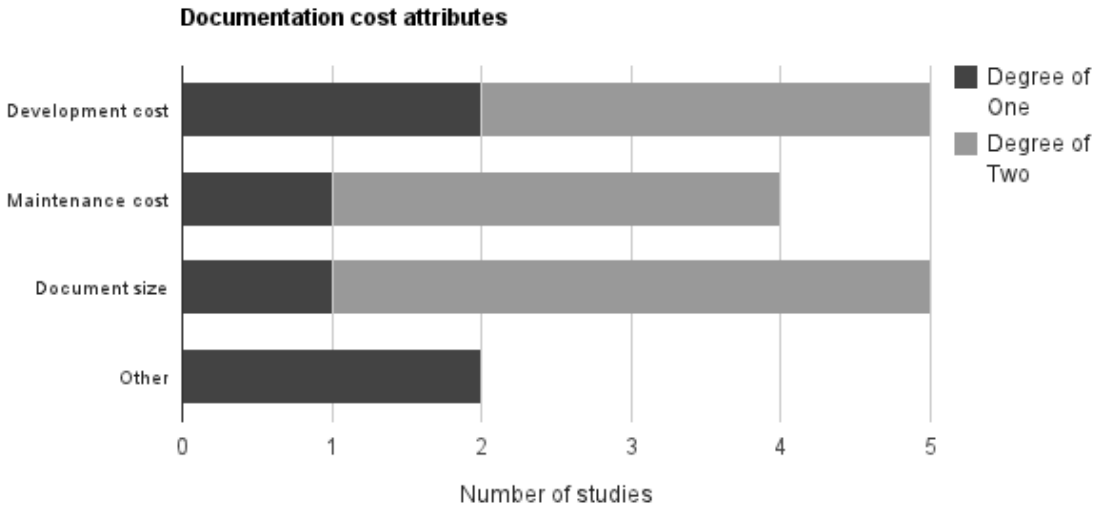

Figure 3.3-Software documentation cost attributes [27]

In terms of cost metrics, nine studies used time/effort to capture the cost. Meanwhile, five studies used document size to measure the cost. Among the two studies that fall into "Other" category, one discussed the "waste of time caused by unstructured docs or incorrect information" and "security risk because of incomplete documentation".

3.4 Conclusion

The results from this SM study conclude that only a few empirical studies exist in the literature to investigate the cost of documentation. Within a few these studies, most of them measured documentation cost via a subjective manner, e.g., questionnaire or manually time recording. Subjective approaches are often risky to introduce bias, and also cannot provide fine-grained measurement on each documentation artifact due to large volume of documentation in a real context. On the other hand, though documentation is realized as one of significant cost-drivers during development process [31, 33], none of the work above further explores underlying cost-drivers of documentation.

It is to the belief of the author that future research on software engineering is to concern with more delicate cost control on development process. Documentation, as an integral part of investment, might need better tuning to optimize the software development process. More research on documentation cost can bring benefit to the industry and the knowledge of academic community.

All of these motivate us to design a more practical method to assess documentation cost and underlying cost-drivers. In this thesis, the proposed methodology objectively measures documentation cost based on data evidence by automatically mining relevant repositories (Chapter Four). This enables the whole process to be easily implemented in a real context, and provides cost measurement on each documentation artifact over time. Based on the data evidence mined from repositories, a subsequent cost-driver analysis helps to uncover cost-drivers in documentation properties.

3.5 Chapter Summary

In this chapter, we presented the results from a systematic mapping study on benefit/cost/quality aspects of software development documentation [27], more precisely, the findings that are related to documentation cost. The works related to documentation cost were further elaborated in Section 3.3, and the conclusions were made in Section 3.4.

Chapter Four - **Research Methodology**

4.1 Overview

A methodology, named DCCDA-"Documentation Cost and Cost-Driver Analyzer", is proposed to assess objectively the cost of documentation artifacts, and to uncover what are the underlying cost-drivers.

DCCDA is aimed to help with understanding documentation process from cost perspectives. Moreover, this methodology applies statistical and machine learning techniques to understand impacting cost-drivers for software evolution and provide evidence-based decision-support [37, 53, 54]. The identified cost-drivers are considered to be a valuable input for documentation cost control and process improvement.

DCCDA consists of three main phases, Phase 1: Design of measurement model, Phase 2: Mining documentation cost, and Phase 3: Identification of documentation cost-drivers.

Phase 1: Design of measurement model

For DCCDA, the first phase includes the definition of a list of documentation cost metrics. Their measurement depends on the availability of data in relevant repositories. For example, three metrics for documentation cost are defined in Section 4.3.1, in order to capture documentation cost from different perspectives.

In addition, a list of candidate cost-drivers and their metrics are defined in Section 4.3.1.3. For different types of documentation, some metrics may need to be tailored. Overall they should be generic and applicable to most textual software documentation, such as requirement specifications, conceptual designs, test plans, etc. Section 4.3 presents the design of measurement model of DCCDA.

Phase 2: Mining documentation cost

Based on defined metrics in Phase 1, this phase automatically mines documentation cost from relevant repositories where documentation artifacts are stored, counting the cost spent on each version. For each revision/version of a multi-version document, the cost and useful metadata (e.g., check-out/check-in times, committer's information, and etc.) for cost-driver analysis should be extracted. The whole data collection and measurement process should be supported by a tool as this process is nearly impossible to implement without automation. Such fine-grained measurement of documentation cost enables to obtain objective answers to RQ1, as well as measurement results of candidate cost-driver metrics for cost-driver analysis afterwards.

Phase 3: Identification of documentation cost-drivers

Taking the measurement results from Phase 2 as input, Phase 3 models documentation cost as response variable with the candidate cost-driver metrics, to explore their relationship. It is applied to two different granularity levels, investigating the cost-drivers for both individual documentation revision/version ("Documentation revision cost-drivers") and total lifecycle cost of a document ("Documentation lifecycle cost-drivers"). Within each level, univariate regression and multivariate regression are applied sequentially to model the relationship between cost-drivers and cost. Univariate regression examines the relationship between each cost-driver metric and cost separately, where multivariate regression examines the compound effect of cost-drivers. The application of the two modeling techniques on different granularity levels enables the identification of a comprehensive and relevant set of cost-drivers.

4.2 DCCDA: Description of the Process

This section provides an operational description of the underlying process of the DCCDA method. The process is subdivided into nine steps. An overview of all steps and their dependencies is shown in Figure 4.1. A description of the individual steps is given afterwards.

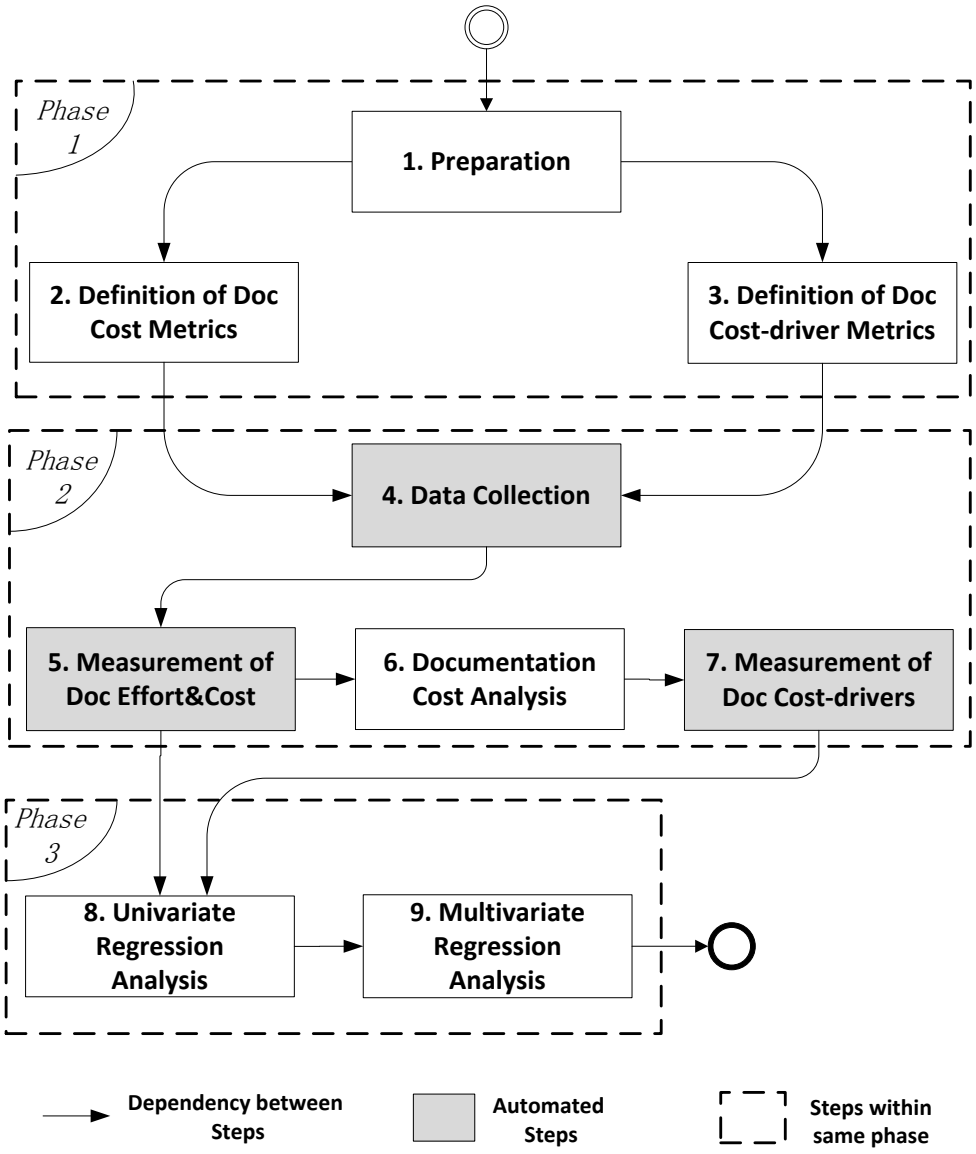

Figure 4.1-Process steps of DCCDA

Step 1: Preparation

The first step is dedicated for the preparation of DCCDA. This step sets up the scope of the study, including the selection of product/project for study and documentation types, the time period to investigate, and the granted access to relevant data repositories. The selection of documentation should consider data availability. To make this methodology work,

documentation artifacts need to be stored in repositories where all different versions of one document can be traced. This step prepares for the actual data collection.

Step 2: Definition of Documentation Cost Metrics

This step defines a set of cost metrics which can be applied to the documentation artifacts under study. Cost metrics are supposed to capture documentation cost from different perspectives, e.g., effort/time or change size. Their measurement depends on data availability in a real context. In this thesis, we propose three generic documentation cost metrics in Section 4.3.1, to capture documentation cost in terms of time spent, fine-grained change size (formulated as Document Churn) and relative change degree.

Step 3: Definition of Documentation Cost-driver Metrics

This step will define a set of metrics for each quantifiable candidate cost-driver. These cost-driver metrics should be generic and easily tailored to different documentation types. For example, size metric may be number of words for textual design documents, but lines of code comments for source code documentation.

DCCDA investigates eight candidate cost-drivers in documentation properties. The selection of candidate cost-drivers refers to a similar study by Benestad et al [37], where applicable cost-drivers for source code evolution are adapted to candidate documentation cost-drivers. 18 corresponding metrics are defined to capture these candidate cost-drivers (Section 4.3.1.3). The selection of cost-drivers and metrics is open to be extended in different contexts.

Step 4: Data Collection

Based on defined metrics by Step 2 and Step 3, this step extracts documentation artifacts and metadata from relevant data repositories. For each multi-version document, all its versions should be collected and ordered by time. Meanwhile, metadata along with each documentation

revision/version check-in, e.g., check-out/check-out times and committer, needs to be collected as well. This step is automated with tool support (described in 0), and its output is raw documentation artifacts and metadata for measurement.

Step 5: Measurement of Documentation Effort and Cost

This step mines the effort spent on each document version, and the lifecycle effort of a document counting for all its versions over time. The effort information should be associated with committer and time stamp, so that we would able to convert the effort to real cost. Since this step is computationally expensive and time-consuming, it is supported by a tool (described in 0). The output of this step is fine-grained measurement of documentation cost.

Step 6: Documentation Cost Analysis

Based on the results of Step 5, this step aggregates and analyzes documentation cost from the perspectives of RQ1.1 to RQ1.5. Moreover, important findings and patterns regarding documentation cost will be summarized, e.g., unstable documents, cost trend over time, time-efficiency on writing documentation and etc. The output of this step provides answers to the research questions, RQ1.1 to RQ1.5.

Step 7: Measurement of Documentation Cost-Drivers

Once candidate cost-drivers and their metrics are defined by Step 3, this step aims to conduct automatic measurement of them. This step is highly computationally expensive and nearly impossible to be manually implemented in reality. It is also eased with tool support (described in 0). The output of this step is measurement of candidate cost-drivers for cost-driver analysis (Steps 8 and 9).

Step 8: Univariate Regression Analysis

This step takes the measurement of documentation cost and cost-drivers from Step 5 and Step 7 as input, and examines the relationship between each candidate cost-driver metric and cost separately. This step will output what types of cost-driver metrics are significantly related to documentation cost, and how they impact the cost. As previous cost-driver studies [34, 37], Generalized Linear Models (GLM) [55] will be used for modeling the relationship between documentation cost-drivers and cost. The output of this step is a set of identified cost-drivers and their impacts on documentation cost.

Step 9: Multivariate Regression Analysis

This step also takes the output from Step 5 and Step 7 as input, but looks at the compound effect of cost-driver metrics on documentation cost. It is a complementary of Step 8 as univariate regression may neglect the couplings among cost-driver metrics. A combination of GLM and a forward stepwise variable selection procedure [56] will be used to achieve high modeling accuracy on cost. It outputs which cost-driver metrics play a more predominant role on determining documentation cost.

4.3 Design of Measurement Model

This section provides for the details on the design of documentation cost metrics and cost-driver metrics, which are Step 2 and Step 3 of DCCDA, respectively. The design of cost and cost-driver metrics is following Goal-Question-Metric (GQM) paradigm [57].

4.3.1 Measurement of Documentation Effort and Cost

4.3.1.1 Model Definition

Based on the software evolution model proposed by [37], an adjusted model of documentation evolution is given in Figure 4.2. The perspective of this study is that documentation creation and evolution is originated from a *Change Request*. A *Change Request*

on documentation is manifested in a sequence of *Revisions* on one or several documents. Each revision will create a new *Version* of a document (based on either a pre-existing version or it can be the initial creation of a document), which belongs to a *Documentation Type* of the *System*. Finally, only one version of a document will be released for usage after successfully going through *Review*.

According to the conceptual model defined in Figure 4.2, an evolution process for a multi-version document is depicted in Figure 4.3. It contains all draft versions and released versions of a documentation artifact. Its cost is spent upon revisions (resulting in different versions), including the effort on accomplishing changes and the effort on reviewing for approval.

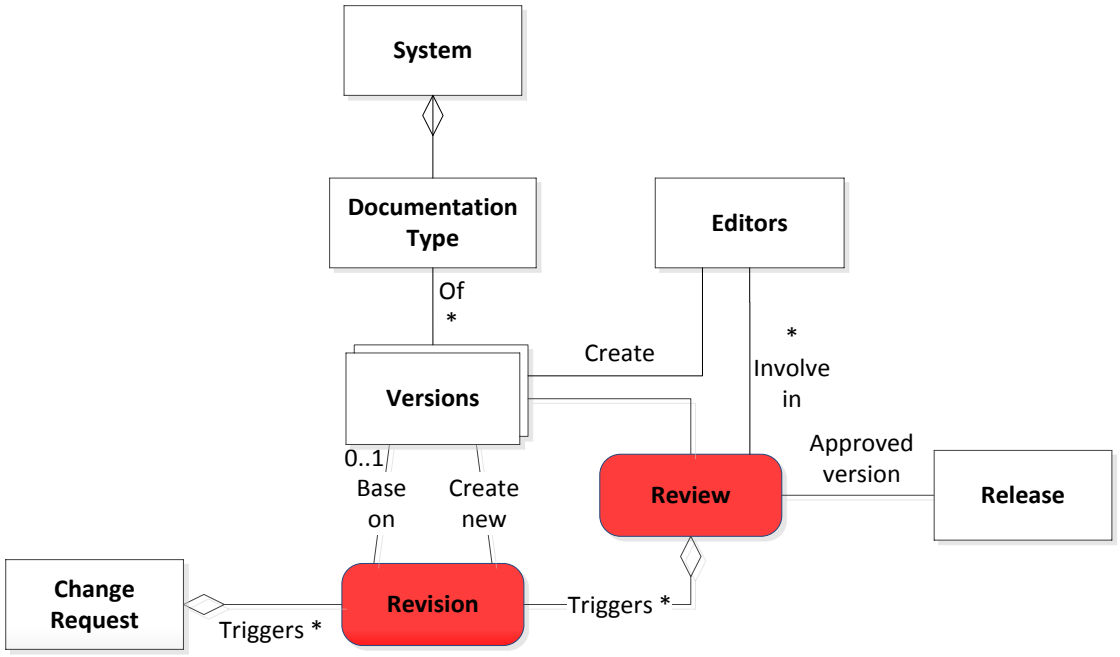

Figure 4.2-Conceptual model for software documentation evolution

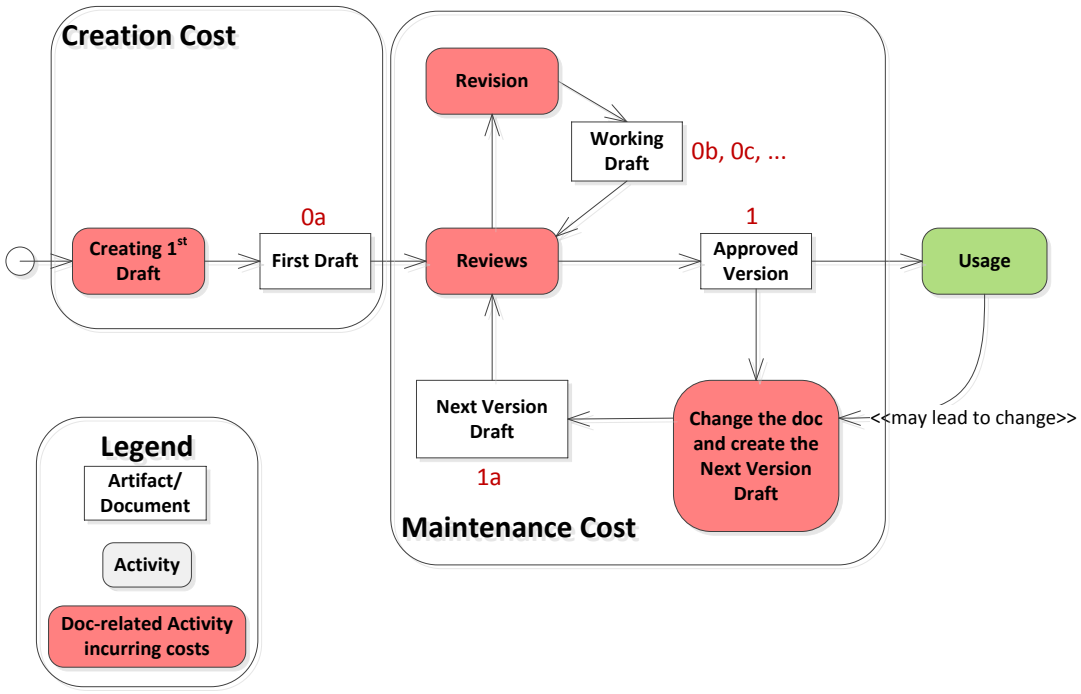

Figure 4.3-Conceptual evolution process for a multi-version document

4.3.1.2 Documentation Effort and Cost Metrics

The notation to describe documentation cost metrics is introduced as follows:

Notation 1: $Hist(d) = \{d_0, d_1, \dots, d_n\}$ denotes the history of n versions of a given document d as stored in data repository. In particular, d_0 denotes the initial creation of this document.

In order to capture cost from different perspectives, three metrics for the lifecycle effort *LCE(d)* of a documentation artifact d are proposed:

LCE(d) Metric 1: Time Expenditure

For a document, time expenditure measures: (1) the effort on editing the document, (2) the effort on designing the artifacts besides editing the document (e.g., using another UML tool), and (3) the effort on reviewing the document (e.g., peer review or normal meeting), defined by

$T_{Edition}$, T_{Design} and T_{Review}, respectively. It aims to capture the overall effort of all people involved in the development of a multi-version document.

The time expenditure on documentation can be defined as follows.

Definition 1: $LCE_{Time}(d, n)$ denotes the time expenditure (in hours) of document *d* having a history of *n* versions, and is defined as:

$$LCE_{Time}(d, n) = \sum_{i=0}^{n} (T_{Edition}(d_i) + T_{design}(d_i) + T_{Review}(d_i))$$

LCE(d) Metric 2: Document Churn

Code churn has been widely used to assess the overall change to a software system in terms of accumulated source code change [58, 59]. Similarly, Document churn is proposed to measure the overall change to a document through its evolution.

Document churn reflects the amount of accumulative changes, which is measured by summing up the number of words changed (added and deleted) between each two consecutive revisions (Note the movement of words is not counted). According to defined lifecycle cost model, the word expenditure on documentation can be defined as follows.

Definition 2: $LCE_{DocChurn}(d, n)$ denotes the document churn of document *d* having *n* versions, and is defined as:

$$LCE_{DocChurn}(d, n) = \sum_{i=0}^{n} (|Delta_{added}(d_i, d_{i+1})| + |Delta_{deled}(d_i, d_{i+1})|)$$

The function $Delta$ takes two versions of a given document and computes their absolute difference, in terms of number of words added and deleted.

LCE(d) Metric 3: Change Degree

Change degree measures how much a given version document has evolved from previous one. They apply on two consecutive versions of the same document. According to defined lifecycle cost model, the total accumulative change degree on documentation can be defined as follows.

Notation 2: $sizeof(d_i)$ denotes the number of words of d_i.

Definition 3: $LCE_{CD}(d, n)$ denotes the total change degree of document *d* having *n* versions, and is defined as:

$$LCE_{CD}(d, n) = \sum_{i=0}^{n} \left(\frac{|Delta_{added}(d_i, d_{i+1})| + |Delta_{deled}(d_i, d_{i+1})|}{sizeof(d_{i+1})} * 100\% \right)$$

4.3.1.3 Converting Effort to Cost

Time expenditure measures the overall effort spending on a multi-version document, which may involve the participation of multiple people. By dividing the overall effort with all involved authors, we would be able to convert their effort (time expenditure) to the real money cost. The lifecycle cost *LCC(d)* of document *d* is defined as:

Definition 4: Let $LCE(d, P_i)$ be the time expenditure (effort) of person P_i spending on a document *d* and $HWage_{P_i}$ for the average hourly wage of P_i. $LCC(d)$, which denotes the amount of cost spending on *d* by the involved people set *P* ($P_i \in P$), is defined as:

$$LCC(d) = \sum_{\substack{i=0 \\ P_i \in P}}^{m} LCE(d, P_i) * HWage_{P_i}$$

4.3.2 Measurement of Documentation Cost-Drivers

Table 4.1 summarizes the documentation cost-drivers and their metrics. Since there is no work in the literature regarding documentation cost-drivers, the selection of candidate cost-drivers refers to the relevant cost-drivers studies on source code evolution in the literature.

In Table 4.1, all documentation cost-drivers are adapted from the literature [34-37], but the metrics of each cost-driver are tailored for documentation artifacts. Each cost-driver is quantified by one or several metrics, which will be used as explanatory variables in quantitative models to investigate their impact on documentation cost.

Table 4.1-Summary of candidate cost-drivers in documentation properties and corresponding metrics

Cost-Driver	**Metric**	**Explanation of Metric**
Document Type	*Type*	Type of documents in SDLC;
Document Size	*InitialSize*	Size after initial creation;
	FianlSize	Size of current version;
	AvgSize	Average size considering all versions;
Change Size	*ChangeWord*	Number of words modified;
	AddWord	Number of words added;
	DelWord	Number of words deleted;
	ChangeBlock	Number of changed word blocks [60];
Change Volatility	*StdevSize*	Standard deviation on the size of a multi-version document;
	StdevAdd	Standard deviation on the number of added words through all versions;
	StdevDel	Standard deviation on the number of deleted words through all versions;
Change Frequency	*MinorRevs*	Number of revisions resulting in non-released versions;
	ReleaseRevs	Number of revisions resulting in released versions;
Document Quality (Readability)	*Readability_1*	Measured by Flesch Reading Ease[61];
	Readability_2	Measured by Flesch-Kincaid Grade[61];
Coupling	*InRefs*	Number of references coming in a given document;
	OutRefs	Number of documents that a given document refers to;
Editor	*NumEditors*	Number of editors involved;
	AvgExp	Weighted average previous check-ins by editors;

4.3.2.1 Document Type

Document type refers to various types of documents generated during software development lifecycle, such as requirement, design, test plan etc.

Notation 3: *DocType* denotes the type of documentation artifacts.

4.3.2.2 Document Size

The number of words is determined as the measurement unit of document size, because others seem less convenient and reliable [33]:

1. The number of pages, paragraphs or lines directly depends on the specific presentation format. And quite normal, different companies or contexts use different formats.
2. The number of sections or subsections cannot be relied because it is hardly to assess the relative importance of different sections with in a document.

Since each document might have gone through a few revisions/versions, its size is measured by:

Notation 4: *InitialSize* denotes the number of words after the first creation.

Notation 5: *AvgSize* denotes the average number of words through all versions.

Notation 6: *FinalSize* denotes the number of words of the latest version.

4.3.2.3 Change Size

The change size captures the differences between each two adjacent versions of a document. Similar to source code change studies, which apply text difference algorithms [62] to measure the number of SLOC added or deleted, but more fine-grained, the number of words added and deleted, are computed at each version by comparing to previous version. The summation of these two metrics equals to the Document Churn metric defined in Section 4.3.1.2.

In addition, a coarse-grained metric of number of change units (a block of adjacent words that are changed together) is used as well as a change size measurement.

If a document has gone through a few revisions, the measurements should be summed up to achieve the total change size.

Notation 7: *AddWord* denotes the accumulated number of word added through all versions.

Notation 8: *DelWord* denotes the accumulated number of word deleted through all versions.

Notation 9: *ChangeWord* denotes the summation of *addWord* and *DelWord*.

Notation 10: *ChangeBlock* denotes the accumulated number of change units (or blocks) [60] through all versions.

4.3.2.4 Change Volatility

Change Volatility is proposed to capture how volatile a document is during its evolution, which is similar to the concept of source code volatility [37]. It can be measured by the movements of document size, and the number of words added and deleted at different versions. Standard deviation is a statistical measurement of volatility, which captures the amount of change dispersion around an average. The larger this dispersion is, the higher the standard deviation and volatility.

Notation 11: *StdevSize* denotes the standard deviation on the size of a multi-version document.

Notation 12: *StdevAddWord* denotes the standard deviation on the number of added words at different revisions.

Notation 13: *StdevDelWord* denotes the standard deviation on the number of deleted words at different revisions.

4.3.2.5 Change Frequency

The direct measurement of change frequency on a document is by simply counting the number of revision/version check-ins during its lifecycle. To differentiate different versions types, the metrics are defined as follows.

Notation 14: *MinorRevs* denotes the total number of non-release versions through its lifecycle, the ones between each two consecutive released versions.

Notation 15: *ReleaseRevs* denotes the total number of released versions.

4.3.2.6 Quality of Documentation

Documentation quality is a complex issue and contains many aspects. Unfortunately, most of them do not have established metrics.

Automated readability metrics are used to represent the quality of documentation. The most widely used, tested and reliable formulas [61] for readability are "Flesch Reading Ease" and "Flesch-Kincaid Grade". They are named as *Readability_1* and *Readability_2*, respectively in this thesis. The automatic computations for them are given bellow:

Definition 5: *Readability_1* denotes the "Flesch Reading Ease" for documentation readability, and is defined as [61]:

$$Readability\_1 = 206.835 - \left(1.015 * \frac{\#word}{\#sentences}\right) - (84.6 * \frac{\#syllables}{\#Words})$$

Definition 6: *Readability_2* denotes the "Flesch-Kincaid Grade" for documentation readability, and is defined as [61]:

$$Readability\_2 = 0.39 * \frac{\#word}{\#sentences} - 11.8 * \frac{\#syllables}{\#Words} - 15.59$$

Where $\frac{\#word}{\#sentences}$ measures the average sentence length, and $\frac{\#syllables}{\#Word}$ measures the average number of syllables per word.

4.3.2.7 Coupling

Similar to the Object-Oriented principles cohesion and coupling for source code, documentation artifacts also have couplings. One direct measurement on this point is the reference relationship. The hypothesis is that the changes on one artifact probably will trigger follow-up changes on other artifact(s) which directly refers to it. The reference mining idea is illustrated in Figure 4.4.

Reference *InRefs* and *OutRefs* are defined as how many references coming in or going out from a given document. A tool to automate the two metrics has been implemented and will be discussed in next chapter.

Notation 16: *InRefs* denotes the number of references coming in a given document.

Notation 17: *OutRefs* denotes the number of documents that a given document refers to.

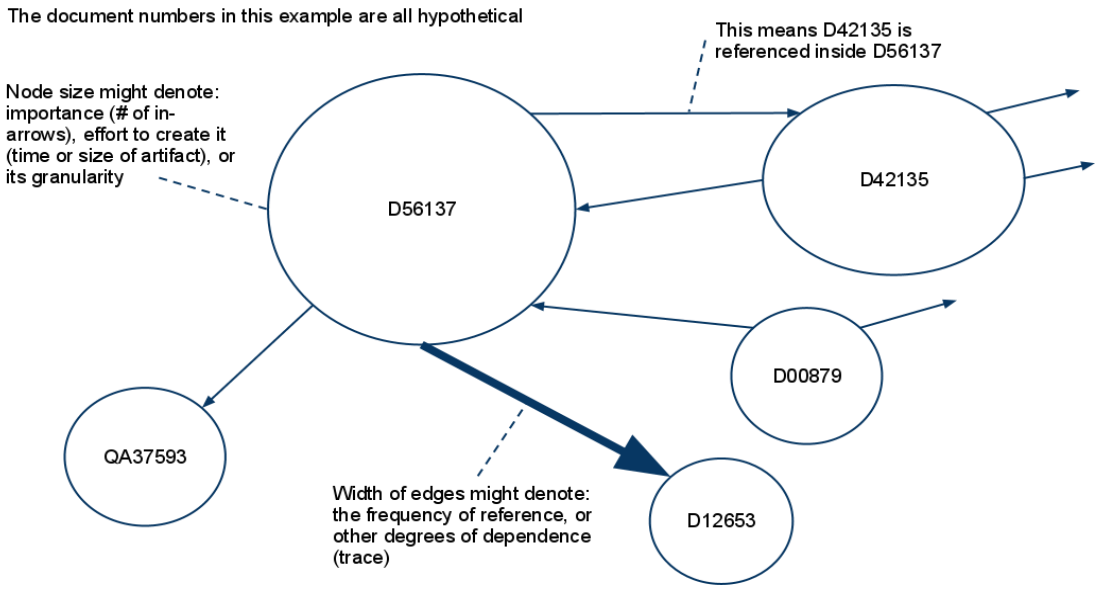

Figure 4.4-An example of mined coupling/reference relationship among documentation artifacts

As an example, document *D56137* in Figure 4.4 is measured by *InRefs* and *OutRefs* with values two and three respectively, indicated by the number of references coming in and going out.

4.3.2.8 Editor

If one document has gone through a few versions over a long time period, multiple editors might be involved. Even for single version, it might involve the effort of several people, especially for those critical documents which require consensus from different stakeholders. One direct measurement of human factor is to count how many editors have been involved.

Notation 18: *NumEditors* denotes the number of editors involved in evolving a given document.

Empirical evidence [63] has shown that productivity between individual developers can be very different. This conclusion also applies to people who write documentation. In order to capture individual difference, change experience is introduced instead of directly assessing individuals. Similar to measuring developer's expertise [1, 37], the experience of all editors involved in a multi-version document is measured by the average number of check-ins per editor. If several people are involved in the evolution of a document, the average experience is calculated by:

Definition 7: *AvgExp* denotes the average experience of involved editors, and is defined as:

$$AvgExp = (1 + MinorRevs + ReleaseRevs)/NumEditors$$

4.4 Generalized Linear Regression for Cost-Driver Analysis

This section introduces the technique used by Step 8 and Step 9 of our methodology for cost-driver analysis, Generalized Linear Model (GLM). The selection of GLM instead of ordinary

regression models is because the cost data is always positive and non-normally distributed (See Section 6.3).

4.4.1 Generalized Linear Model

Regression models are widely used in software engineering for both cost estimation and cost-driver analysis [33, 34, 37]. An ordinary linear regression model is built as:

Definition 8: $\overline{\mathrm{Y}} = \mathrm{E(Y)} = \mathrm{x_1} * \beta_1 + \mathrm{x_2} * \beta_2 + \cdots + \mathrm{x_n} * \beta_\mathrm{n}$

In this model, $\overline{Y} = E(\mathrm{Y})$ is a series of estimated cost variable Y, $X = (x_1, x_2, \cdots, x_n)$ is a set of candidate cost-driver variables, and $\beta = (\beta_1, \beta_2, \cdots, \beta_n)$ is a set of parameters to be estimated. Hence, variable x_i has no influence on Y if and only if β_i is zero. This ordinary regression model is applicable based on the assumption that the response variable Y should have a normal distribution [55].

If the distribution of Y is non-normal, which is often the case for cost prediction and cost-driver analysis in software engineering [55], a generalized linear model (GLM) can be used as a replacement. GLM is loose enough to encompass a wide class of models useful in statistical practice, but tight enough to allow the development of a unified methodology of estimation and inference, at least approximately.

Definition 9: $\overline{\mathrm{Y}} = E(Y) = g^{-1}(\mathrm{x_1} * \beta_1 + \mathrm{x_2} * \beta_2 + \cdots + \mathrm{x_n} * \beta_\mathrm{n})$

Here $\overline{\mathrm{Y}} = E(Y)$ is the expected value of Y defined by probability theory, and g is the link function. The actual selection of link function g and error distribution depends on the specific application, as shown bellow. The rationale for each choice was thoroughly discussed in [55].

- **Gaussian**: a Gaussian (Normal) distribution
- **Binomial**: a binomial distribution for proportions
- **Poisson**: a Poisson distribution for counts

- **Gamma**: a gamma distribution for positive continuous data
- **Inverse.Gaussian**: an inverse Gaussian distribution for positive continuous data

For each candidate cost-driver metric, its p-value and coefficient are used to interpret its impact on the models. The significance level (p-value) is often set to 0.05. This means that any variable having p-value less than 0.05 is considered as a "significant" cost-driver metric.

4.4.2 Planned Evaluation and Validation

Mean Magnitude of Relative Error (MMRE) and Root Mean Squared Error (RMSE) are widely used to assess cost estimation models [34, 37]. In this study, MMRE and RMSE are used to evaluate our models, as they are independent of the modeling techniques and allowing for straightforward comparisons between models [34].

For a testing data set having *n* documents, let y_i be its actual measured cost for a document d_i and $\bar{y}_i$ for the estimated cost by a GLM. Then, MMRE and RMSE are defined as follows.

Definition 10: For a given model that is built by *n* documents, MMRE is calculated by:

$$MMRE = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \frac{|y_i - \bar{y}_\mathrm{i}|}{y_i}$$

Definition 11: For a given model that is built by *n* documents, RMSE is calculated by:

$$RMSE = \sqrt{\frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{y_i} - \bar{y}_\mathrm{i})^2}$$

MMRE and RMSE are calculated by n-fold cross-validation. This procedure models a given data set (*n* data points) with *n* iterations. In each iteration, one model fitted by the subset of *n-1* data points will predict the last data point. The MMRE and RMSE are calculated on the results of *n*-fold cross-validation.

It is difficult to interpret coefficients of candidate cost-driver metrics when high degree of multicollinearity [64] exists between independent variables for a regression model. Variance inflation factor (VIF) proposed by Fox and Monette [64] is used to check the degree of multicollinearity. If VIF is or close to 1, there is no or negligible multicollinearity. A common rule of thumb is that multicollinearity is high if VIF is greater than 5 as the threshold.

4.5 Applicability

The application of this methodology in another context is a practical process supported by measurement automation and tool support, which will be introduced in 0. Qualitative approaches to measure documentation cost (e.g., surveying people [49, 50] or recording effort [33, 44]) are time-consuming and the results usually contain high risk of bias. The proposed methodology is on the basis of data evidence in repositories, and the tool support makes it practical that the measurement of documentation cost and the cost-drivers can be conducted at any point in time during software evolution.

The conceptual cost model on software documentation evolution is proposed in Section 4.3.1. This model is generally defined in order to be applicable to another software development context. The measurement completeness of defined metrics depends on the availability of data from relevant repositories in different contexts. The more supports that the documentation process is able to provide (e.g., documentation is organized by a centralized system which tracks its evolution along with metadata), the more accurate results will be achieved. For certain cases that the data might be not able to be directly collected from repositories, expert estimations through carefully elicitation may be utilized as complementation.

In this study, the quantitative cost-driver analysis investigates the relationship between underlying cost-drivers and documentation cost, with the purpose of understanding

documentation cost and improving documentation practice. A follow-up qualitative analysis can also be introduced, aiming at further refining the quantitative results. For example, if this study indicated that documentation cost had a high correlation with certain cost-driver, interviews would be taken to explore the root-cause of this factor. This enables appropriate use of the study results towards software process improvement.

4.6 Chapter Summary

This chapter overviewed the methodology DCCDA (Section 4.1), described its operational process (Section 4.2), defined metrics for documentation cost and cost-drivers (Section 4.3), introduced the technique applied to cost-driver analysis (Section 4.4), and finally discussed the applicability of proposed methodology (Section 4.5).

The data collection and measurement process of DCCDA was automated with tool support through mining relevant software repositories (described in 0). It enables this methodology a practical process which can be easily replicated in another context.

Chapter Five - **DCCDA Tool Support**

This chapter discusses tool support for DCCDA. Steps 4, 5 and 7 of DCCDA (described in Section 4.2) are automated as they are computationally expensive and time-consuming because of the high volume of data in repositories.

5.1 Requirements of Tool Support

While relatively easy to compare two versions of a document by Unix/Linux *diff* or *compare* feature in MS Word for fine-grained measurements of its evolution, analyzing a large amount of documentation in which each one may have multiple versions, e.g., *500 documents * 10 versions * 5000 words per version*, is considerably time-consuming and nearly impossible to accomplish without automation. For each document under study, accumulated cost over its multiple versions at different time points needs to computed and summed up, as well as the measurements of candidate cost-drivers.

Though lots of tools support for directory comparison to ease the burden, such as *diff*, *WinMerge* [65], and *WordDocDiff* [66], none of them provides any convenient feature to work with a large number of multi-version documents and conducts comparisons on each two adjacent versions by chronological order. Moreover, candidate cost-drivers should also be automatically measured to eliminate the measurement overhead issue.

To automate the measurement process (Steps 4, 5 and 7) of DCCDA, the functional requirements of tool support are derived and listed as follows:

- **Requirement 1** (derived from Step 4): Support for various formats of textual documentation, including MS Word/Excel, PDF, Rich Text, Text, XML or HTML, and cross-format comparison (e.g., a MS Word against a PDF document);

- **Requirement 2** (derived from Step 5): Compare every two consecutive versions of a given multiple-version document and compute the cost spent at each version based on cost metrics in Section 4.3.1.2;
- **Requirement 3** (derived from Step 7): Support for computing cost-driver metrics in Section 4.3.1.3 for a given multi-version document or a single version;
- **Requirement 4**: Support for concurrent processing of a large number of documents in which each may have multiple versions over time. It is important to notice that we are dealing with large amount of documentation and requiring fine-grained measurements. The computation process should be done within an acceptable time frame.

5.2 Architecture

The architecture of the tool support consists of two layers, called “Data Preparation” and “Computation”. As illustrated in Figure 5.1, the “Data Preparation” layer provides support for Step 4 of DCCDA on data collection and preprocessing. The “Computation” layer eases the computational burden of Steps 5 and 7 of DCCDA.

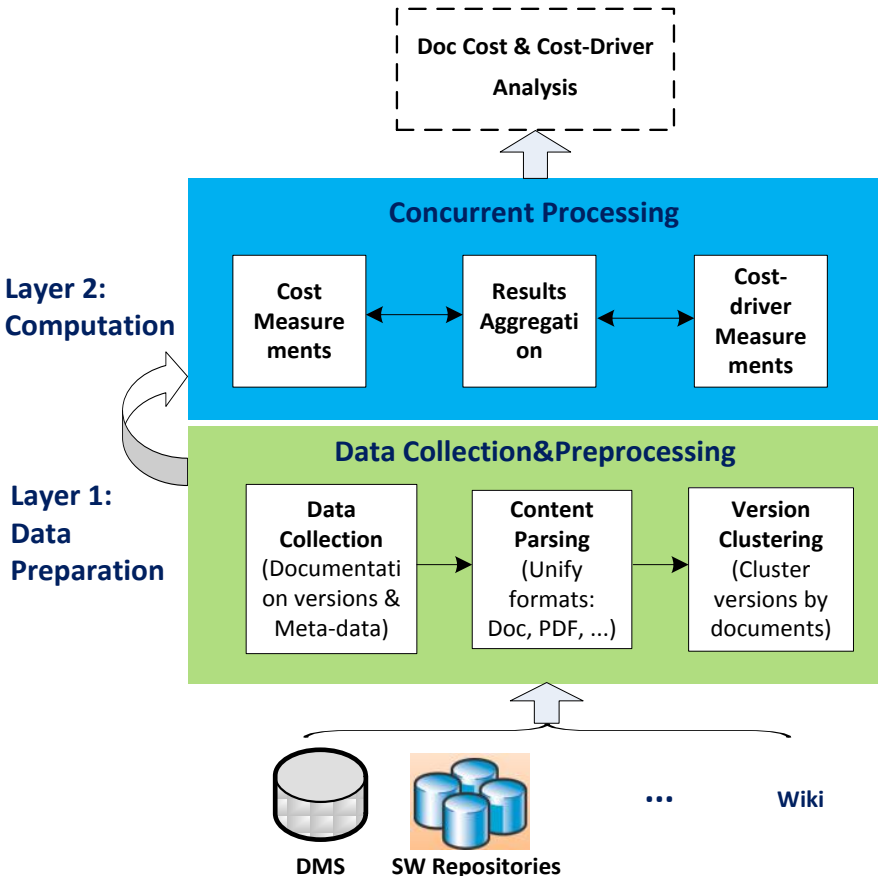

Figure 5.1-A layered structure of tool support on DCCDA

Layer 1: Data Preparation

- "Data Collection". It aims to select the target documentation set for analysis. The data access to target source, such as documentation management system, software repositories or online Wiki, should grant retrieval of all versions of studied documents, as well as useful metadata, such as authors' information, check-in and check-out records etc.
- "Content Parsing". The contents of collected documents may have different formats, all of which need to be parsed into a unified format (textual information) for analysis.

- "Version Clustering". All versions of the same document need to be grouped together by a chronological order, with the purpose of enabling pair comparison of two adjacent versions and concurrent processing on all documents (a set of multi-version documents).

Layer 2: Computation

Computation layer is where the actual measurement of documentation cost and cost-drivers happens. It takes in the data prepared by Layer 1 and generates measurement results.

Parallel programming [67], a subset of the broader concept of multithreading, was applied to speed up the intensive computation by making true parallelization taking place on multiple processors. This process is illustrated in Figure 5.2, which demonstrates the internal mechanism of "Computation" layer in Figure 5.1.

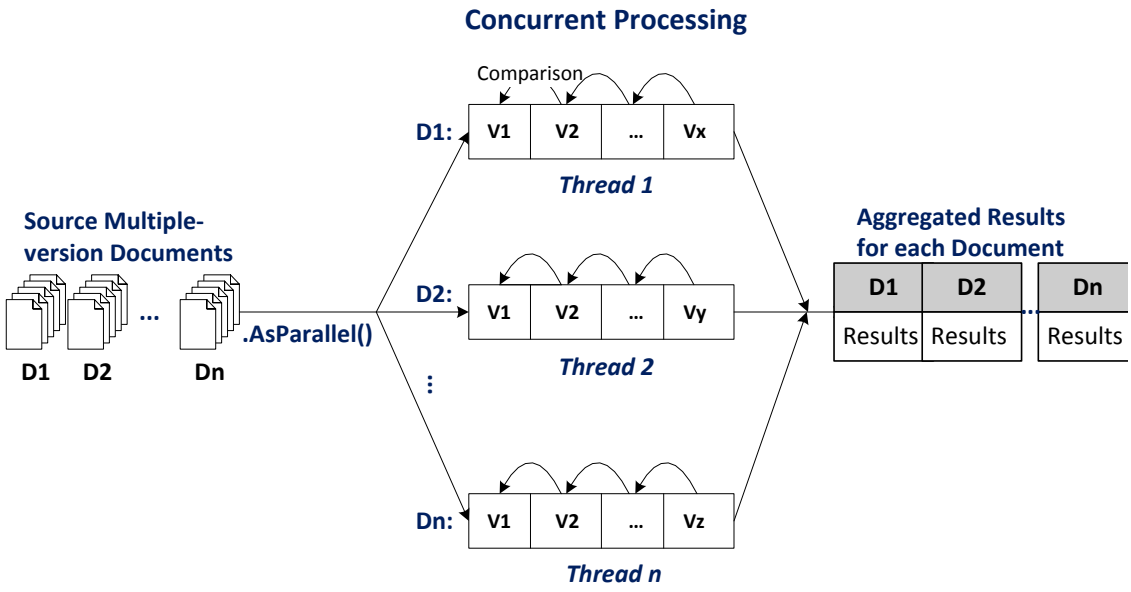

Figure 5.2-Mechanism of concurrent processing on a large number of multi-version documents

In Figure 5.2, each cluster, a document having multiple versions, will be assigned to one thread, so that all clusters can be processed in parallel. Within any cluster, the cost on each

version will be computed by comparing to its previous version, and the total cost is computed by summing up the cost on each version. Similarly, the cost-driver metrics of each document can be achieved by using the metadata upon every single version.

We tested the capability of the tool with case study data on an Intel Xeon CPU E5620 with two processors and six GB RAM machine. The results indicates the application of this mechanism enables the processing of 1,000+ documents (average size: 4,950 words) within 20 minutes, down from more than one hour (>67%).

5.3 Usage for Cost Measurement

To demonstrate the usage of tool support for documentation cost measurement, some sample results based on case study data (described in Chapter Six) are reported in this section.

The cost metrics defined in Section 4.3.1 are automatically computed for each multi-version document. Figure 5.3 illustrates an exemplary collection of cost measurement results on partial documents from case study. In addition to the cost metrics, the tool also computes the number of total revisions/versions, lifespan (from creation to last update) and number of authors involved.

A more detailed view on each version of a given document from case study is also presented, as shown in Figure 5.4. Similarly, some metadata of each version, such as revision number, committer, check-in/check-out times, and fine-grained change size metrics are also extracted along with cost measurements.

DocNo.	#Revs	LifeSpan(days)	#Editors	TotalEditTime(hours)	DocChurn(words)	ChangeDegree
D09633	13	1221	5	195.1	22709	396.7%
D11064	3	147	1	26.9	1545	54.4%
D11162	11	554	2	618.0	85880	1126.2%
D11308	1	252	1	191.1	1424	38.8%
D11376	5	670	2	2207.3	54433	362.3%
D11452	27	1565	3	352.9	37393	922.3%
D11471	1	57	1	66.5	6134	79.9%
D11551	4	1534	3	989.5	21381	367.4%
D11562	2	151	1	678.8	18015	42.1%
D12218	6	394	1	97.6	22907	98.6%
D12224	0	6	1	125.5	1160	0.0%
D12694	9	253	7	248.6	16377	118.4%
D12810	4	210	1	30.1	9327	59.8%
D12974	6	163	2	351.2	10552	195.7%

Figure 5.3-Snapshot of cost measurement results: an aggregated view

DocNo.	Revision	Author	Check-outTime	Check-inTime	#Words	#ChangeBlock	#WordAdded	#WordDeled	TotalEditTime
D11162	R0A	######	###########	###########	6735	237	6735	0	110.0
D11162	R0B	######	###########	###########	6753	2	179	150	0.2
D11162	R0C	######	###########	###########	8100	125	6569	2847	17.2
D11162	R0D	######	###########	###########	7855	105	5527	11694	43.9
D11162	R0E	######	###########	###########	9281	221	4220	2641	108.6
D11162	R0F	######	###########	###########	8843	299	2759	3113	21.6
D11162	R0G	######	###########	###########	5557	434	5513	9708	72.0
D11162	R0H	######	###########	###########	6171	62	4189	2144	28.9
D11162	R0I	######	###########	###########	5741	122	3152	3943	46.2
D11162	R0J	######	###########	###########	7663	106	4030	1175	18.5
D11162	R0K	######	###########	###########	7779	119	516	365	39.3
D11162	R0L	######	###########	###########	7800	3	35	0	0.1

Figure 5.4-Snapshot of cost measurement results: a single document view

5.4 Usage for Cost-Driver Measurement

Most of candidate cost-drivers are automatically measured while analyzing each version for computing documentation cost, such as *Document Size*, *Change Size*, *Change Volatility* and *Editor*. However, measuring documentation *Quality (Readability)* and *Coupling* is not straightforward, and they have to be achieved with extra tool supports.

5.4.1 Document Quality

As aforementioned in Section 4.3.2.6, documentation quality is a multifaceted issue. In this study documentation readability is selected as a representative, since readability has well-defined metrics in literature [61].

To automate the measurements of documentation readability, the *ReadabilityStatistics* API of MS Office 2010 [68] is used in the program to compute *Flesch Reading Ease* and *Flesch-Kincaid Grade Level*. For those documents which are not stored in MS Word, their contents would be extracted and transferred to MS Word files for readability measurements.

5.4.2 Coupling

An automatic reference mining tool was developed to capture the couplings/references among documents. Some sample data from case study context (described in Chapter Six) was used to demonstrate the capability of the tool. Given certain number of source documents as input, the expected output is a directed graph visualizing input documents and their reference relationship. The work flow is illustrated by Figure 5.5, including reference mining, graph data preparation and visualization as the three main steps.

At the beginning, the documents to be analyzed are selected as input. In principle, they can be any type of textual artifacts, requirement specifications, design documents or test procedures etc. The reference lists will be transferred to a special data format for visualization. Finally, all input documents and those detected references will be visualized in the framework (Gephi [69]).

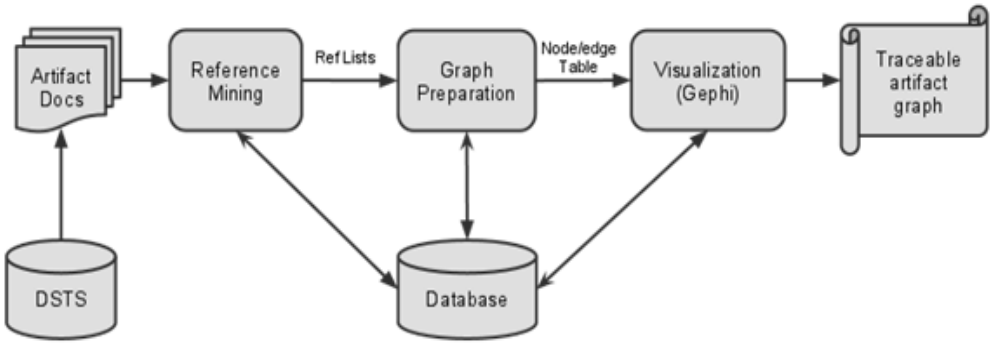

Figure 5.5-Automatic mining and visualizing documentation coupling/reference relationship

"Reference Mining" in Figure 5.5 needs to first parse documents written in different formats and extract textual contents (Apache POI [70] for Microsoft Word/Excel, and Apache PDFBox [71] for PDF), and then search for references (e.g., searching for terms "*D******" and "*QA*********" which are the unique identifiers for documents in case study organization). A list of references for each document will be generated.

"Graph Preparation" selects GEXF (Graph Exchange XML Format) for Gephi [69] visualization because of its capability for elaborating complex network structures, and their associated meta-data. It is an XML-based language, supporting for hierarchy structure. Figure 5.6 is a simple example for a minimal graph containing two nodes and one edge between them. In this context, all documents are represented as nodes, and reference relationship is expressed by an arrow pointing from source document to target document. In addition, *InRefs* and *OutRefs* which measure the number of edges coming in and out are calculated. The width of each edge is used to indicate the frequency of reference.

```
<gexf xmlns="http://www.gexf.net/1.2draft" version="1.2">
    <graph mode="static" defaultedgetype="directed">
        <nodes>
            <node id="0" label="Hello" />
            <node id="1" label="Word" />
        </nodes>
        <edges>
            <edge id="0" source="0" target="1" />
        </edges>
    </graph>
</gexf>
```

Figure 5.6-GEXF file format for visualization

"Visualization" utilizes the popular Gephi framework to demonstrate couplings among documents. As an example, a single document view is able to provide a clear *InRefs* and *OutRefs* perspective, as shown in Figure 5.7. From this view, we can easily find the values for metrics *InRefs* and *OutRefs*.

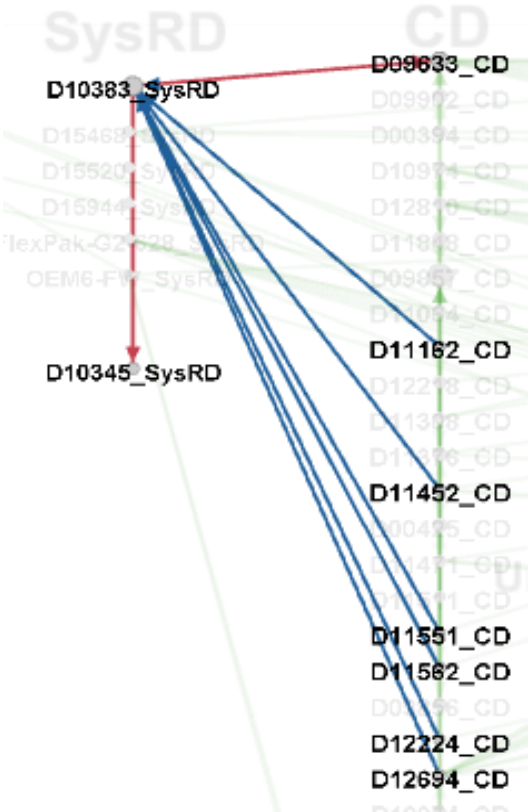

Figure 5.7-Reference visualization for single document

5.5 Chapter Summary

This chapter discussed the requirements of measurement automation and tool support for DCCDA (Steps 4, 5 and 7). A two-layer structure of automation framework was introduced, and fine-grained computations on each revision/version, were applied to enable accurate measurements of cost and cost-drivers. Various formats of documentation are supported, and the usage of Parallel Computing by [67] speeds up the whole process (about 67%) in an acceptable time frame. In addition, an implementation for capturing coupling/reference among documentation was also demonstrated.

Chapter Six - **Evaluation: An Industrial Case Study**

This case study is conducted by following the structure and guidelines recommended by Runeson and Höst [72]. This chapter discusses in detail the case study design, data collection and data analysis procedure. The results of this case study are presented in Chapter Seven and Chapter Eight.

6.1 Case Study Design

6.1.1 Case Study Context

NovAtel is a leading provider for a comprehensive line of Global Navigation Satellite System (GNSS) products, and its OEM products heavily depend on embedded software. With strong quality requirements in terms of accuracy, reliability and performance, software development and evolution is a key success factor. At present, NovAtel is in the process to introduce more flexible and lean development processes with shorter iterations and earlier feedback cycles. The question at hand is how to tune the development process and artifact parameters to achieve best results. Documentation has been considered important for communication and collaborative development, and how much documentation is enough to ensure “optimal” cost is an issue of concern for NovAtel.

6.1.2 Case Process Selection

The case in this study is the software documentation process in NovAtel, which will be investigated from cost perspective. NovAtel has more than 20 years of legacy code and documentation. Figure 6.1 summarizes the volume history of documentation in recent 10 years from an in-house-made documentation management system, called DSTS. In DSTS, documentation creation and revision are organized by check-ins, similar to version control system for source code. In Figure 6.1, y-axis denotes the number of documentation

revision/version check-ins in DSTS. At NovAtel, DSTS stores most types of documentation produced during software development (84,237 documents in total), such as requirement specifications, conceptual/detail designs, test plans, test results, process regulations and etc.

As the volume of documentation increased at each year in general, the cost on documentation is also considerably high. Documentation process, as an integral part of investment, needs better cost control to optimize the software development process.

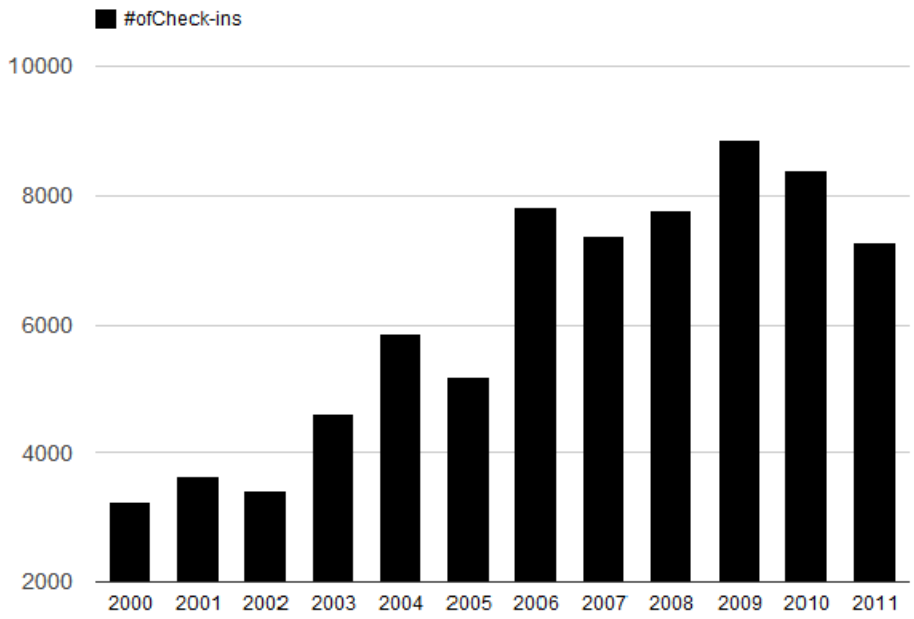

Figure 6.1-Volume history of documentation in DSTS

6.1.3 Objectives

The objective of the case study in NovAtel is to evaluate documentation cost and to investigate underlying cost-drivers by applying the methodology DCCDA (described in Chapter Four). It requires a careful analysis of real project data. Since there is no evidence in the literature about the possible cost-drivers in documentation, it can only be tackled via empirical studies. This case study aims to explore and provide initial answers to above questions.

By this case study, we first intend to reveal documentation cost from different perspectives (i.e., RQ1.1 to RQ1.5), and then analyze the relationship between candidate cost-drivers and the cost.

The overall business goal of this CRD project [9] is to improve software documentation cost-effectiveness and maintenance efficiency in NovAtel. By combining the results from this study with documentation usefulness information measured by other students, an evaluation of documentation cost-effectiveness would be achieved. We believe such evidence is able to provide decision support on documentation process improvement.

6.1.4 Unit of Analysis

NovAtel's OEM product line has evolved from OEM1 to OEM6, while previous products are either discontinued or less maintained. Figure 6.2 shows the volume of software documentation over each product from DSTS. Each check-in denotes the creation or revision of one document.

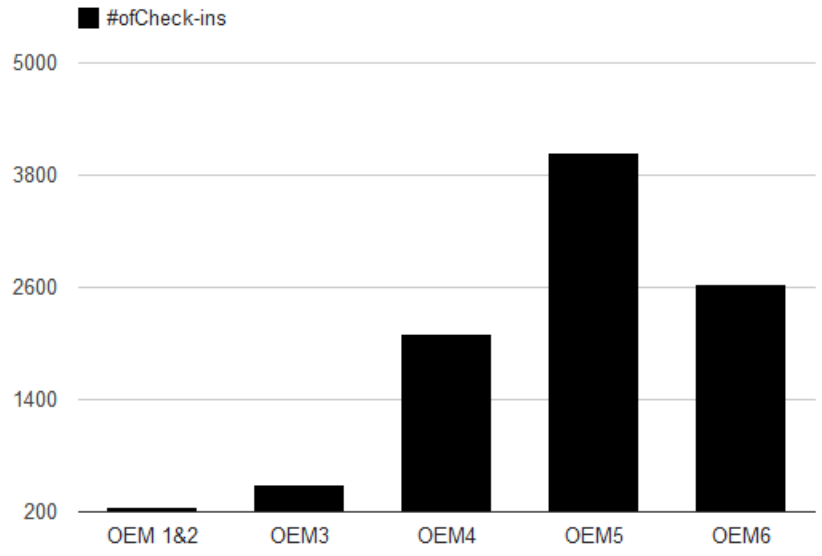

Figure 6.2-Volume of documentation over OEM products in DSTS

The unit of analysis chosen by this case study is the recent product OEM6. It was launched from 2007 and first released in 2010. OEM6 embedded software is a medium-size project (20 functional components and around 100,000 SLOC) with extensive change activities on both source code and documentation.

For each document under study, the lifecycle cost should be summarized from all its possessed versions. The cost spent on other aspects, for example the cost on storing or locating a document, will not be taken into account, because they would be extremely difficult to obtain.

The relationship between cost-drivers and cost will be examined in two levels of granularity. We will first look into the cost-drivers impacting the lifecycle cost of a document, and then investigate the cost-drivers that determine the effort spent on each revision or version. By this way, we intend to identify a more comprehensive and more reliable set of cost-drivers of documentation.

6.2 Collecting Data

This section discusses in detail the data collection process. Three types of high-level documentation of OEM6 (Conceptual Design, Test Plan, and Process Regulation documentation) were first selected to study their cost. These three documentation types (named Design, Test and Process for short) are available in DSTS, which keeps traces of all versions of a document and the metadata of each revision check-in.

Table 6.1 provides a summary of collected documents (Note: Test Plan documents were selected by the software manager in NovAtel with consideration of their representativeness and availability.). Most documents have multiple revisions/versions over time. Due to the fact that most Test documents of OEM6 are reused from previous products (as far as the former OEM2),

they have an extremely long history with a large number of revisions/versions over time. Design documents are newly developed for OEM6, so that they have fewer revisions.

Table 6.1-Summary of collected data for case study

Type	**Design**	**Test**	**Process**
#**ofDocuments** (Total: 55)	20	15	20
#**ofVersions** (Total: 1630)	140	1,036	454
#**ofEditors** (Total: 125)	23	58	52
Period of Collected data	Apr 2006 – Dec 2011	Jul 1996 – Nov 2011	Jun 1999 – Apr 2012
Sample Ratio	100%	5.7% (15 out of 262)	100%
Selection Criterion	All conceptual designs on OEM6 software	Test plans on OEM6 software	All process regulations on software development and evolution

Each documentation revision check-in needs to be classified into a group of previous revisions owned a multiple-version document, where all its revisions/versions are ordered by check-in sequence. In addition to collect documents, the metadata along with each check-in is extracted as well, such as check-out and check-in times, committer and check-in number etc.

6.3 Data Analysis Procedure

This section discusses the data analysis procedure of this case study, which corresponds to Steps 6, 8 and 9 of the methodology DCCDA. The results of data analysis are reported in Chapter Seven and Chapter Eight.

At a high level, the data analysis of this case study will proceed as follows:

6.3.1 Documentation Cost Analysis

The effort (measured by *Time Expenditure*) of each document version is collected through reading the built-in property "Total Editing Time[1]" of MS Word files with tool support, since most of studied documents was written in MS Word (> 99.4%). This measurement captures the effort on editing (or creating) a MS Word document, and the effort on designing or reviewing whenever people were conducting these activities within the same Word file. Though alternative measure of cost can be used, for example by counting the time interval between each documentation revision check-out and check-in, we believe this measurement is more reliable for capturing substantial cost on developing documentation.

In this case study, the effort measured by *Time Expenditure*, is used as response variable for cost-driver analysis, because people's wage information is considered as confidential.

By following Step 6 of DCCDA, we analyze the collected documents from the perspectives of RQ1.1 to RQ1.5. A few findings based on evidence are summarized to assist the understanding of documentation effort in NovAtel. The results of documentation effort analysis are presented in Chapter Seven.

6.3.2 Documentation Cost-Driver Analysis

This analysis (described by Steps 8 and 9 of DCCDA in Chapter Four) is conducted from two different granularity levels. At coarse-grained level, a document with single or multiple versions is treated as one sample. Therefore, the identified cost-drivers are the ones that influence the lifecycle effort of a documentation artifact. At fine-grained level, each individual

[1] "When you open a Microsoft Word document, a behind-the-scenes timer starts. As long as your document is open and its window is in front of all other windows, the timer continues, whether you're actively changing the document, scrolling around, or just thinking. If you save your changes, the additional time is added to the document's Total Editing Time. If you close without saving, the additional time is discarded.", MS Office.

revision/version is considered as one sample, and the cost-drivers that determine the effort spent on individual documentation revision can be identified.

Table 6.2 provides the descriptive statistics for document effort data (i.e., the response variable *Y* in our GLM) at both levels. Based on Table 6.2, we notice the effort data is always positive and right skewed (non-normal distribution). According to the recommendations provided by Myers et al. [55], a GLM with a *gamma* response variable and a *log* link-function is appropriate to apply between candidate cost-driver metrics and documentation effort. The results of documentation cost-driver analysis are presented in Chapter Eight.

Table 6.2-Descriptive statistics for documentation effort (in hours)

Metric	**Document Lifecycle Effort**	**Document Version Effort**
Mean	261.4	22.8
StdDev	441.3	32.5
Min	1.0	1.7
25% Percentile	30.1	3.1
Median	165.7	7.0
75% Percentile	352.9	24.7
Max	2,207.3	166.4

6.4 Chapter Summary

In this chapter, we detailed the design of an industrial case study at NovAtel. The statistical results from Section 6.1 have shown a burdensome documentation practice and the necessity for cost control and cost-driver analysis. To this end, three types of documentation were selected for cost analysis. Moreover, a two-tier analysis procedure was proposed to investigate the nature of documentation cost-drivers on both individual artifact and each documentation revision. The case study results are presented in Chapter Seven and Chapter Eight.

Chapter Seven – **Case Study Results from Documentation Cost Analysis (RQ1)**

This chapter presents the results from documentation cost analysis in this case study, organized by RQ1.1 to RQ1.5, respectively.

7.1 Cost per Document (RQ1.1)

RQ1.1 concerns the lifecycle cost spent on each single or multi-version documentation artifact. This section presents the lifecycle cost of each document from case study through mining its evolution history.

The versions belonging to one document are grouped together and ordered by time as one row, and each version is mapped onto a 2-D presentation as one dot, as shown in Figure 7.1. Here, y-axis is a collection of case study documents, separated by documentation types, and x-axis is the timeline for version check-ins.

For each document under study, all its versions (data points within the same column in Figure 7.1) are computed for effort, so that the lifecycle effort (LCE) can be obtained by summing up all of them. Due to space limitation, Table 7.1 summarizes the measurement results of top 20 costly documents from case study, ranked in descending order by *Time Expenditure*. A more detailed view on individual document is presented afterwards in Table 7.2.

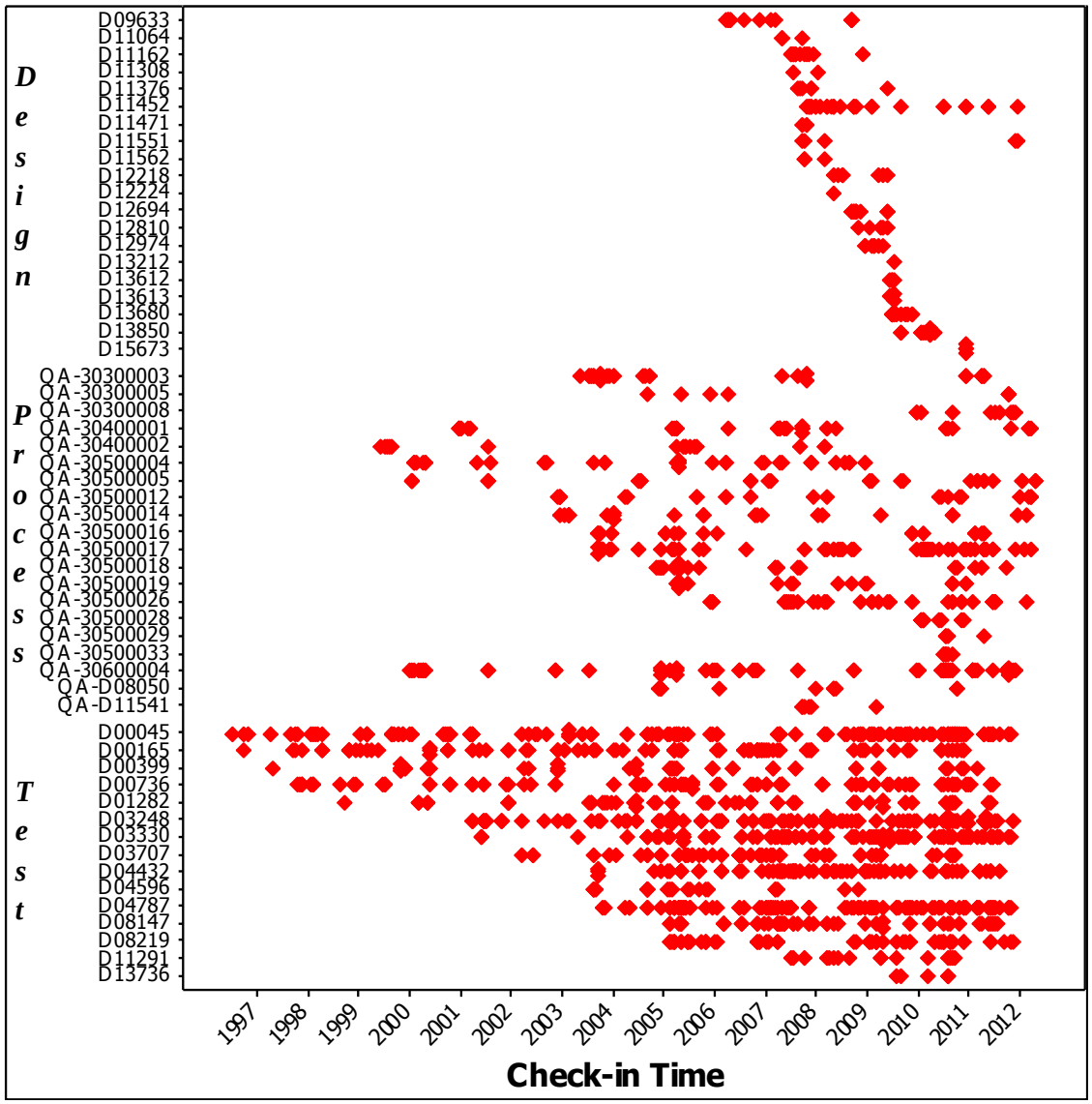

Figure 7.1-Visualization of version history of case study documents

Table 7.1-Measurement results of top 20 costly documents

Rank	**DocNo.**	**DocType**	**Time Expenditure** (hours(man-months))	**Document Churn** (words)	**Change Degree**
1	D11376	Design	2,207 (9.2)	54,433	362%
2	D00736	Test	1,954 (8.1)	215,905	1,461%
3	D00165	Test	1,075 (4.5)	488,724	2,163%
4	D04787	Test	1,031 (4.3)	180,067	3,769%
5	D11551	Design	989 (4.1)	21,381	367%
6	D03330	Test	817 (3.4)	125,666	3,643%
7	D11562	Design	679 (2.8)	18,015	42%
8	D03248	Test	618 (2.6)	283,821	5,608%
9	D11162	Design	618 (2.6)	85,880	1,126%
10	QA-30400001	Process	496 (2.1)	41,043	1,209%
11	D13850	Design	493 (2.1)	6,060	151%
12	D04432	Test	463 (1.9)	137,656	2,825%
13	D00045	Test	434 (1.8)	196,865	5,669%
14	QA-11541	Process	377 (1.6)	6,540	50%
15	D03707	Test	372 (1.6)	51,930	1,127%
16	D11452	Design	353 (1.5)	37,393	922%
17	D12974	Design	351 (1.5)	10,552	196%
18	QA-08050	Process	326 (1.4)	29,779	606%
19	QA-30500004	Process	322 (1.3)	18,709	1,325%
20	D11376	Design	175 (0.7)	54,433	362%

Table 7.2 takes a multi-version document (*D11162*) from case study for example and presents the detailed measurement results. This document has been through 12 revisions/versions by two people after initial creation over a 554 day's period. Eight consecutive revisions had relatively large-scale changes happened (more than 65% measured by *Change Degree* metric). In total, this document consumed about 618 hours (2.6 man-months, one man-month is considered to have 30 days and each day has eight hours) through its evolution.

Table 7.2-Detailed measurement results on document "*D11162*"

Metrics	**Results**
#Versions	12
#Editors	2
AvgSize	7,356 (words)
Lifespan	554 (days)
Document Churn	81,204 (words)
AddWord	43,424 (words)
DelWord	37,780 (words)
ChangeDegree (at each version)	5.2%; 116.2%; 223.9%; 74.7%; 89.8%; 293.3%; 117.0%; 125.2%; 69.3%; 11.3%; 0.4%
ChangeDegree	1,126%
TimeExpenditure	618 hours (2.6 man-months)

In addition to the measurement results of each document, we observed a few interesting findings based on documentation version history in Figure 7.1.

- **Finding 1: Unstable documentation artifacts**

  Unstable documentation artifacts refer to documentation artifacts that have been frequently changed during their lifecycle. They are indicated by a dense horizontal line in Figure 7.1. For example, the document with the most revisions can be identified by the amount of dots from rows, which is *D000045* (the first row of Test documents) having 136 revision check-ins. By consulting the software manager in NovAtel, two main root-causes for unstable artifacts are found: (1) poorly written documents which demand frequent changes, and (2) the functionality contained therein is considered strategic and needs to be updated from time to time.

- **Finding 2: Life span of documentation artifacts**

  Long life span documentation artifacts refer to those documents which have long evolution history and been reused many times since initial creation. For example, the

document with the longest time span (from initial creation to last update) is also *D000045*, followed by *D00165*, by observing the length of each row (timeline).

- **Finding 3: Revision frequency of different documentation types**

  The revision frequency of different documentation types is different. From Figure 7.1, we can notice that Test documents were most frequently revised during evolution, followed by Process and Design documents. Process documents, for example, were more discretely changed than the other two. The root-cause is "Once software processes have been practiced for a certain time period, they were supposed to be reviewed and modified for improvement. Consequently, version history of Process documents discretely distributed over timeline.", as explained by the software manager in NovAtel.

7.2 Cost per Documentation Type (RQ1.2)

RQ1.2 helps to understand how cost spending across different documentation types varies and to reveal what is the most costly documentation type that is worth our attention.

Hypothesis 1: The rankings between different documentation types from the same organization are same based on three types of effort metrics (i.e., *Time Expenditure*, *Document Churn*, and *Change Degree*).

To test this hypothesis, documents from case study were classified into three types, i.e., Design, Test and Process, in order to compare different documentation types by each effort metric. All three documentation types were statistically summarized by each effort metric, as shown by box-plots in Figure 7.2, Figure 7.3 and Figure 7.4. Each box-plot provides effort distributions of three documentation types measured by a specific effort metrics, as well as the mean value.

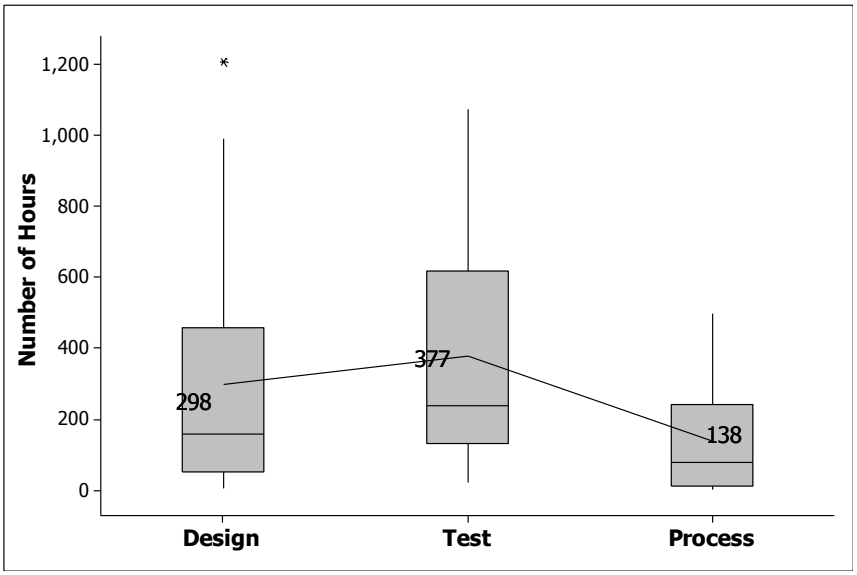

Figure 7.2-Box-plot of the three documentation types by *Time Expenditure*

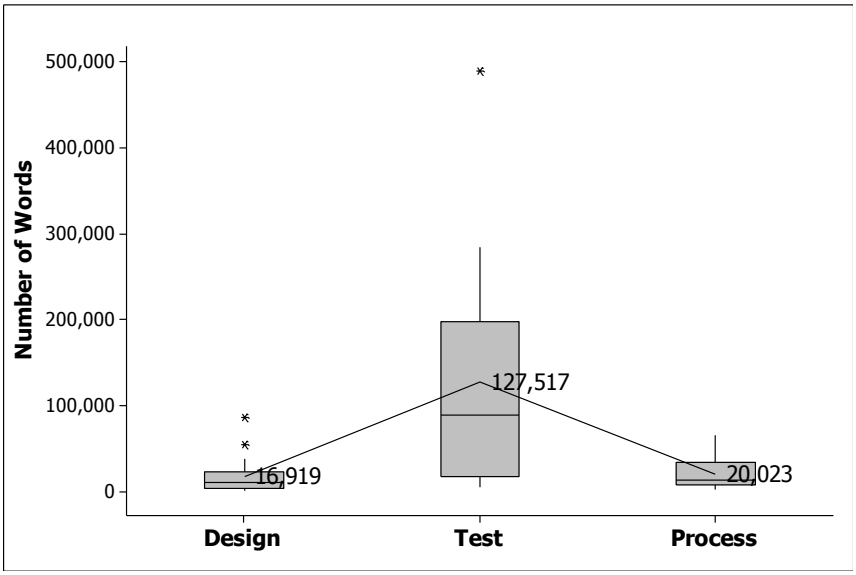

Figure 7.3-Box-plot of the three documentation types by *Document Churn*

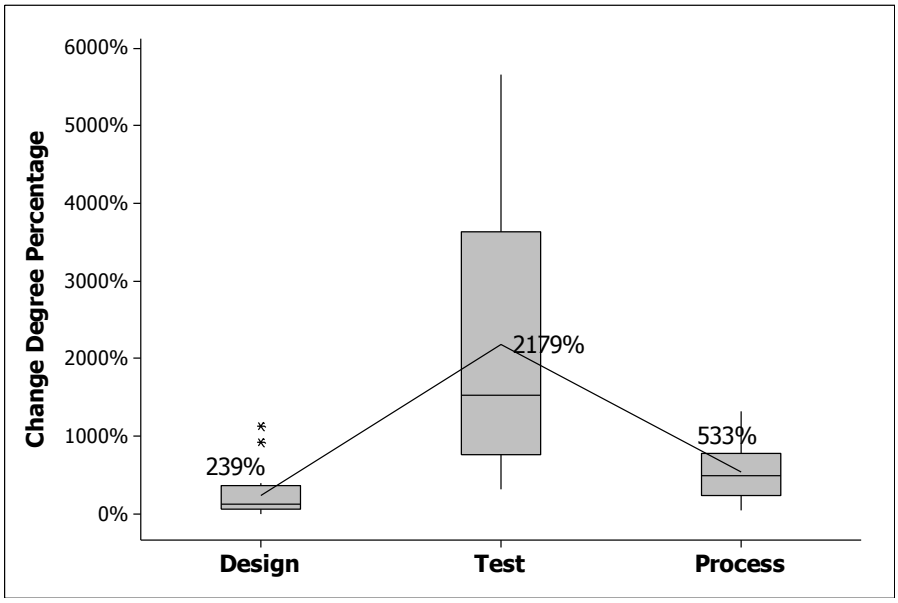

Figure 7.4-Box-plot of the three documentation types by *Change Degree*

Based on the mean value of each documentation type, the rankings by the three effort metrics are shown below:

- **Ranking by *Time Expenditure*** (in hours):

$$Test(377) > Design(298) > Process(138)$$

- **Ranking by *Document Churn*** (in words):

$$Test(127{,}517) > Process(20{,}023) > Design(16{,}919)$$

- **Ranking by *Change Degree***:

$$Test(2{,}179\%) > Process(533\%) > Design(239\%)$$

The results above reject the hypothesis, since the ranking varies when using different effort metrics. This is caused by the fact that these three metrics are designed to capture effort from different perspectives, and not necessarily equal with each other. For example, Figure 7.2 reveals that the time spent on Design and Test documentation far outweighs Process documentation,

while Figure 7.3 and Figure 7.4 tell that Test documentation far outweighs the other two in terms of *Document Churn* and *Change Degree*.

7.3 Cost Distribution over Time (RQ1.3)

RQ1.3 concerns documentation effort distributions over different time periods. It helps to compare development activities with documentation support. It is applicable to different levels of granularity, single document, one documentation type or all documentation in general, assisting to understand documentation behaviour on time basis. This section presents the documentation effort distributions over time from case study.

First, the overall documentation effort over recent 11 years (from 2001 to 2011) is shown in Figure 7.5. As we can notice, the effort on Design documentation originated from 2006 and reached a peak at 2007. This is because that product OEM6 was launched around this time period, when most of initial conceptual designs were accomplished. After 2007, the phenomenon that certain amount of "Design" effort still exists indicates that the new product was going through design changes. In addition, the new product also triggered the modifications on old test plans to incorporate new features. This explains the increase of "Test" cost since 2007. The "Process" effort overall shows a slightly increasing trend, which indicates that NovAtel has been focusing on software process improvement. All above explanations were validated by the software manager in NovAtel.

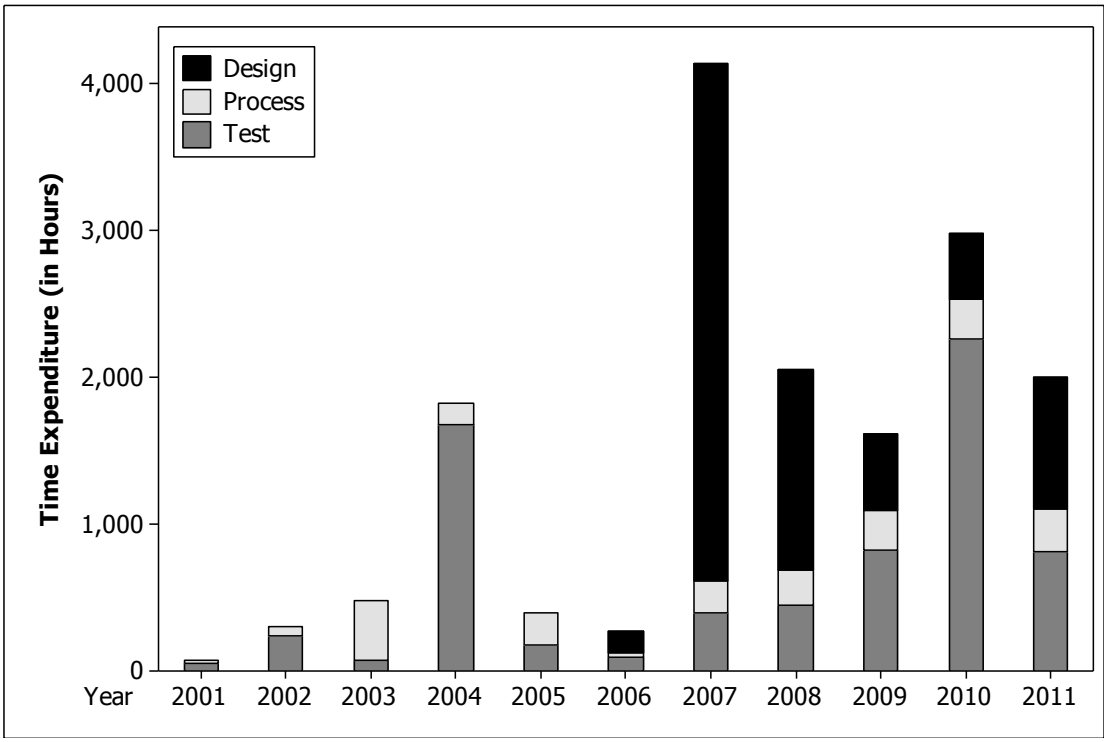

Figure 7.5-Total documentation effort over timeline from case study

As another example, the document *QA-30300003* with total 30 revisions is illustrated in Figure 7.6. For each revision check-in, its cost (time) is mapped into one point, where its x-axis is the revision check-in time and y-axis denotes the amount of effort.

From Figure 7.6, we can observe that most of documenting activities were relatively centralized within three time periods, separated by the gap from 2005 to 2007 and the gap from 2008 to late 2010. Most effort was introduced in the first period, including the extremely high case, the initial creation. Overall, there were lots of revisions which did not consume much effort (the dots locate closely to *x*-axis).

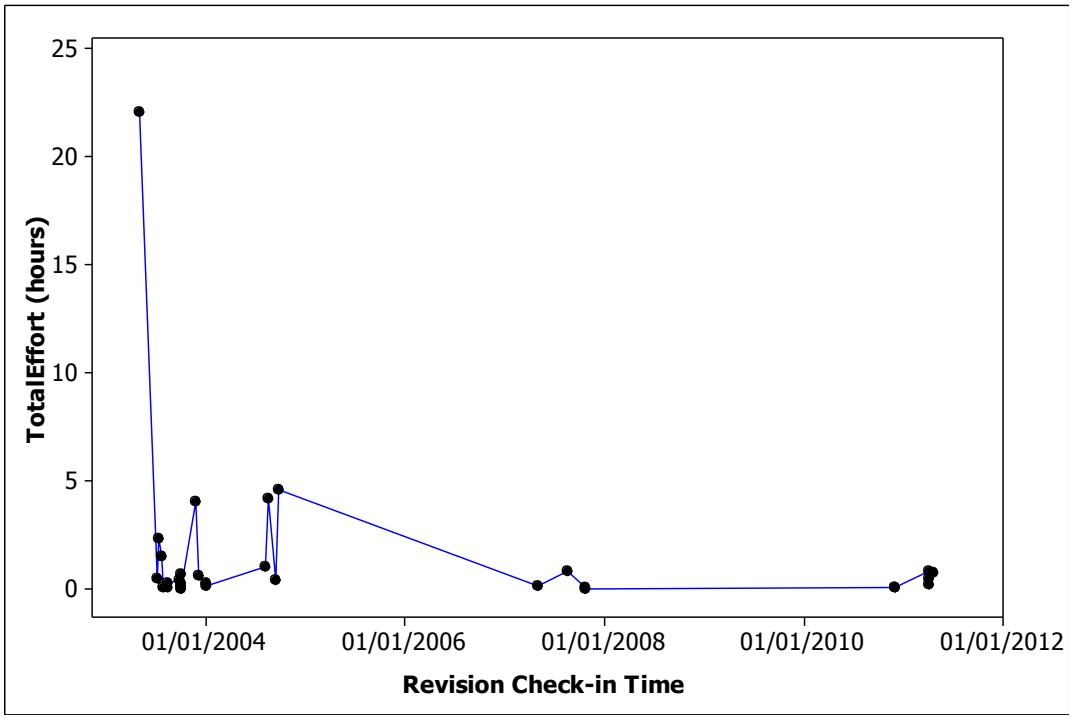

Figure 7.6-Effort distribution of document *QA-30300003* over time

The explanation to those effortless revisions is that only minor modifications were made for these revisions, such as changing only a few words. To verity this explanation, the fine-grained measurements of each revision, in terms of number of words added, deleted and unchanged, are shown in Figure 7.7. In Figure 7.7, x-axis is a time series of revision (denoted by version ID), and y-axis denotes number of words.

Figure 7.7 shows that revisions *ROA*, *ROC*, *ROD*, *R2A* and *R2B* have relative higher change size, in terms of word added and deleted, which is compatible with their corresponding cost shown in Figure 7.6. In contrast, most of the other revisions only have small change size. All of these indicate that documentation change size is highly correlated with effort, which will be statistically tested in documentation cost-driver analysis (Chapter Eight).

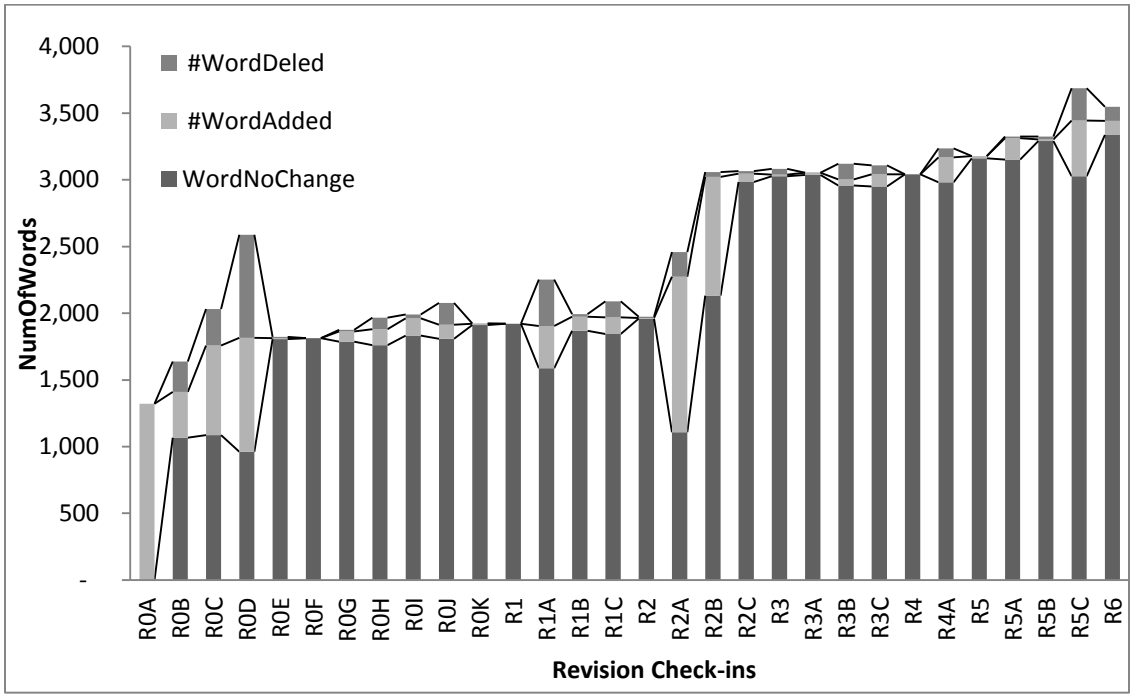

Figure 7.7-*Document Churn* measurements of *QA-30300003* over time

7.4 Cost by Person (RQ1.4)

RQ1.4 focuses on associating documentation effort with each person, so that we can better understand their participation and performance level on documentation.

Top 10 people who committed most revisions (named "C1") versus who spent most effort (named "C2"):

Table 7.3 compares the people with most documentation revision check-ins versus people who spent most time, ranking by descending order. For confidential reason, each person is represented by a capital letter. People who show up in both categories are in ***bold***.

Table 7.3-Top 10 people ("C1") with most revision check-ins versus the top 10 people ("C2") spent most effort (in hours (man-months))

Rank	People within "C1"		People within "C2"		
	Person	**#ofCheck-ins**	**Person**	**#ofCheck-ins**	**Time Expenditure**
1	A	124	K	6	1,328 (5.5)
2	B	76	***D***	61	1,226 (5.1)
3	***C***	64	L	26	916 (3.8)
4	***D***	61	M	19	809 (3.4)
5	E	58	N	18	780 (3.3)
6	F	56	O	37	701 (2.9)
7	G	55	P	39	580 (2.4)
8	***H***	46	***C***	64	540 (2.2)
9	I	44	***H***	46	531 (2.2)
10	J	42	Q	7	464 (1.9)

Top 15 people in "C1" and "C2" across each documentation type:

Since most people write only one type of documentation based on their roles, more detailed views need to be obtained across different documentation types. This information assists to understand people's activities within each documentation type.

To this end, the top five people recognized by both categories across different documentation types are listed in Table 7.4. Comparing to Table 7.3, eight additional people *(#ofCheck-ins<42*) are recognized by "C1", five from Design and three from Process. Similarly, "C2" has five such cases from all categories.

Overall, time spending on documentation across authors varies from 22 minutes to 1,328 hours/5.5 man-months (standard deviation is 225 hours/0.9 man-month).

Table 7.4-Top 15 people in both categories across documentation types

DocType	People within "C1"		People within "C2"	
	Person	**#ofCheck-ins**	**Person**	**Time Expenditure** (hours(man-months))
Design	***I***	39	K	1,328 (5.5)
	M	19	***M***	809 (3.4)
	R	15	N	706 (2.9)
	S	9	***Q***	464 (1.9)
	Q	7	***I***	418 (1.7)
Test	A	124	***D***	1,226 (5.1)
	E	58	O	701 (2.9)
	D	58	P	580 (2.4)
	F	56	H	531 (2.2)
	G	55	J	362 (1.5)
Process	B	76	L	915 (3.8)
	C	64	***C***	540 (2.2)
	T	37	W	198 (0.8)
	U	31	***U***	141 (0.6)
	V	28	X	140 (0.6)

In Table 7.3 and Table 7.4, there are people (in ***bold***) existing in both categories, "C1" and "C2". It is interesting to know if *#ofCheck-ins* correlates to *Time Expenditure*.

Hypothesis 2: Higher number of documentation revision check-ins leads to more effort spending by each person.

The Pearson correlation was applied between *#ofCheck-ins* and *Time Expenditure* for all 125 people from case study. The correlation coefficient is 0.27 with a p-value 0.004 (less than 0.05 is considered as significant). It indicates that *#ofCheck-ins* and *Time Expenditure* have low-level linear correlation. Other factors may also influence the actual time spending across authors.

7.5 Time-efficiency on Documentation (RQ1.5)

On the basis of results from RQ1.4, RQ1.5 aims to provide preliminary evaluation on the variation of people's time-efficiency on writing software documentation. This section introduces our metric on defining documentation time-efficiency, presents the corresponding results from case study, and discusses the limitation. In addition, an interesting hypothesis about documentation time-efficiency is formulated and tested.

Programmer productivity in software engineering is often defined as the ratio between code size and programming time [73]. Similarly, the productivity or efficiency of people on writing documentation can be defined as follows:

Definition 12: Efficiency (X) denotes the time-efficiency of person X on writing textual software documentation, and is defined as:

$$\text{Efficiency } (X) = \frac{Document\ Churn}{writing\ time}$$

$Document\ Churn$ denotes the number of changed words committed by X. *Efficiency*, therefore, captures people's productivity on writing textual software documentation (words) in a time unit (hour). Of course this definition does not apply to all types of documentation, such as visual documentation UML.

The top 15 people (out of 125) who spent most effort in Table 7.4 are tested by *Efficiency*, as shown by the last column.

Finding 4: The overall results reveal the fact that people wrote documentation with highly different *Efficiency* levels (variation from minimal 15.6 to maximal 2,592.6). This variation is greater than coding variation that was originally found between 1 and 20 by Sackman et al. [74]. In general, the *Efficiency* on writing Design documents is the lowest (456.6), followed by Test

(1,194.6) and Process (1,417.2). This is caused by higher complexity of software design task itself.

Table 7.5-*Efficiency* of top 15 people in terms of cost spending on documentation

DocType	**Person**	**#Check-ins**	**Document Churn**(#words)	**Time** (#hours)	**Efficiency** (#words/hour)
Design	K	6	56,148	1,328	42
	M	19	101,304	809	125
	N	7	19,560	706	28
	Q	7	8,170	464	18
	I	39	38,277	418	92
Test	D	58	165,662	1,226	135
	O	37	34,405	701	49
	P	*39*	135,044	580	233
	H	46	49,206	531	93
	J	42	86,836	362	240
Process	L	24	42,508	915	46
	C	64	41,338	540	77
	W	5	4,175	198	21
	U	31	17,725	141	126
	X	6	15,749	140	112

Limitation: Comparing to programming productivity which has been widely studied [73, 75, 76], the measurement of the time-efficiency on writing documentation is so far rarely studied. The *Efficiency* metric defined in this context only captures the average speed of individual writing documentation, regardless the quality of documentation.

It is possible that a person who writes documentation fast (high *Efficiency*) might neglect the quality of documentation. Consequently, it results in more additional revisions to leverage the quality later on. A hypothesis based on this assumption was formulated and tested.

Hypothesis 3: Higher *Efficiency* results in more documentation revisions committed by each person.

To test this hypothesis, the Pearson correlation was applied between *#ofCheck-ins* and *Efficiency* for all the people involved in case study. The correlation coefficient is 0.51 with a p-value 0.06 (less than 0.05 is considered as significant). It indicates that revision numbers and writing speed are not significantly correlated.

7.6 Threats to Validity

Construct Validity: One threat to construct validity is the *Efficiency* metric defined in Section 7.5. It can only capture the speed of individuals on writing documentation. However, time-efficiency on writing documentation has to be jointly considered with quality and complexity of documentation. These two factors may influence individual's time-efficiency on writing documentation and are considered as threats to construct validity.

External Validity: Though we have selected all Design and Process documents of OEM6, only 15 Test documents were selected by the software manager in NovAtel to represent overall Test documentation. The selection of Test documents might introduce bias and is considered as a threat to external validity.

7.7 Chapter Summary

The chapter summarized the documentation cost analysis results on RQ1.1 to RQ1.5 from the case study. In addition, four findings were concluded and three hypotheses were tested.

A preliminary metric for the "Time-efficiency on writing software documentation" was proposed and measured in the case study. Moreover, we tested a hypothesis whether there is a positive correlation between high "Time-efficiency" and high number of revision check-ins,

considering people who wrote documentation fast may result in low quality documentation and therefore more revisions. The results indicated that they are not significantly correlated.

Chapter Eight – **Case Study Results from Documentation Cost-Driver Analysis (RQ2)**

This chapter presents the cost-driver analysis results of the case study, organized by RQ2.1 and RQ2.2. The analysis has been done through two different levels of granularity. The coarse-grained level relates to the cost-drivers on the lifecycle cost of a multi-version document (named "Document Lifecycle Cost-Drivers"). The fine-grained level targets the cost-drivers that influence each individual revision or version of a document (named "Document Revision Cost-Drivers"). The analysis at each level has gone through the same procedure, as described by Step 8 and Step 9 of the methodology DCCDA (Figure 4.1).

In this chapter, the results for RQ2.1 and RQ2.2 are presented in Section 8.1 and Section 8.2, respectively. The overall results are discussed in Section 8.3. Section 8.5 discusses possible generalization of case study results, followed by discussing of the threats of validity.

8.1 Document Lifecycle Cost-Drivers (RQ2.1)

8.1.1 Descriptive Statistics

It is important to understand the characteristics of current system under study and explain why results of future replications of this study could be same or different from this study. The distributions statistics can tell whether the system under study is representative or has unusual patterns, e.g., extremely long size, high revision numbers or many people involved in evolving one document.

Figure 8.1 shows the distribution statistics of cost-driver metrics, where its y-axis denotes the number of documents.

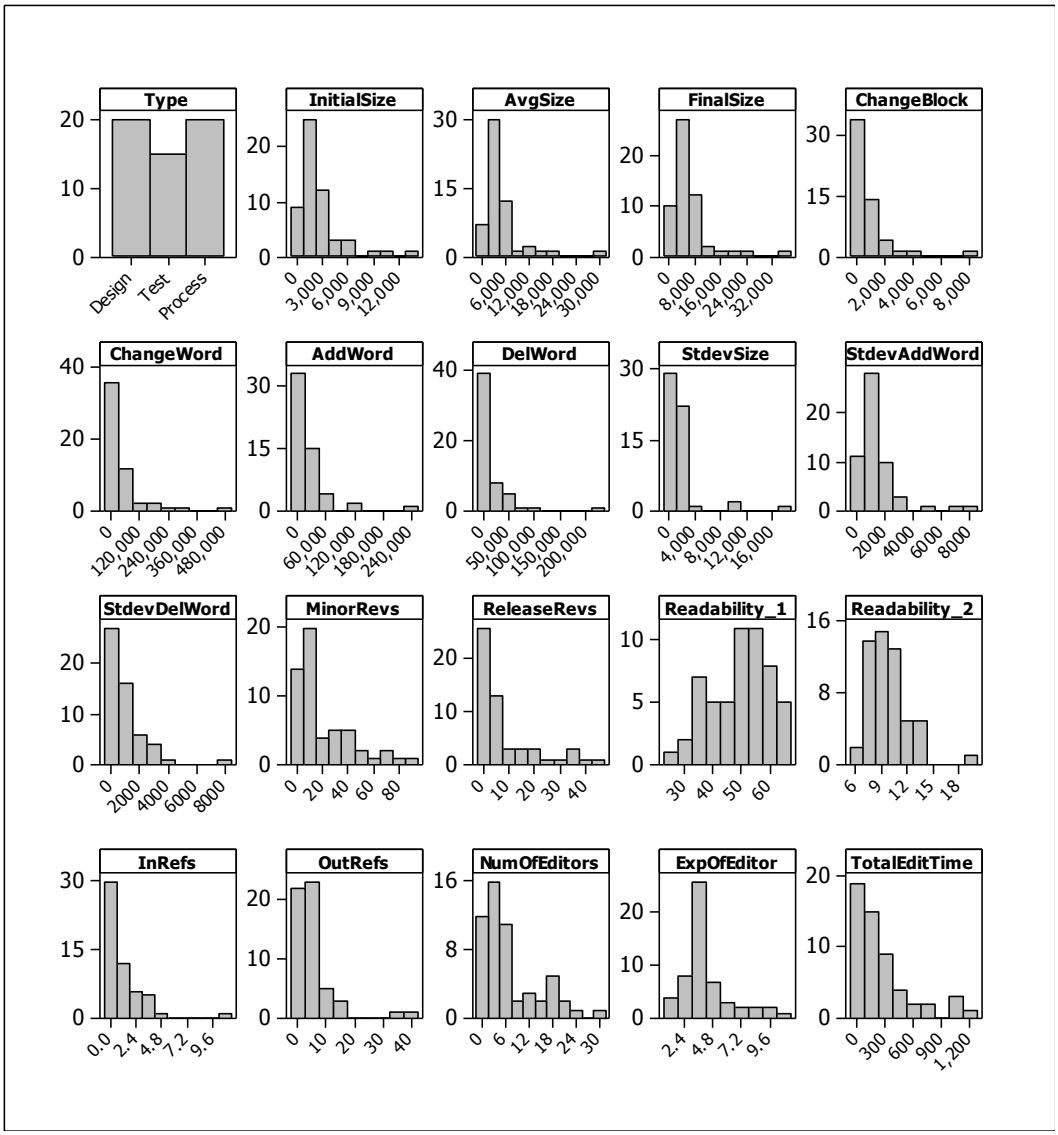

Figure 8.1-Frequency distributions of cost-driver metrics and cost for "Document Lifecycle Cost-Drivers"

From Figure 8.1, we can observe a few characteristics on case study data.

- Most of documents size (84%) values are less than 8,000 words based on all three metrics *InitialSize*, *AvgSize* and *FinalSize*. The distributions of these three metrics indicate that documentation size is increasing over time, from *InitialSize* (average 2,551 words per document) to *FinalSize* (average 5,705 words per document)..

- *MinorRevs* and *ReleaseRevs* indicate that most of documents have multiple versions, but no patterns on this point have been observed.
- Most of documents under study involve multiple people. But the number of people and their average experience varies, based on the distributions of *NumOfEditors* and *ExpOfEditor*.
- For most of documents under study, the number of outgoing references outweighs the number of incoming references.

In summary, all of these observations indicate that the organization has been practicing intensive documentation with highly number of documentation revisions (average: 30 versions/document) and spent cost over time (average: 313.5 hours/document). This motivates us to investigate the underlying cost-drivers of documentation.

8.1.2 Univariate Regression Analysis

The results of univariate analysis are provided in Table 8.1. Columns "Cost-Driver" and "Metric" indicates the name of each candidate cost-driver and its metrics, and columns "Coefficient", "StdErr", "T-Test" and "P(Coef.)" denote the estimated regression coefficient, its standard error, t-test value and p-value. A p-value bellow than 0.05 (the chosen significance level for regression analysis), 0.01 and 0.001 are marked with one, two and three asterisks, respectively. All the tables in this chapter reporting GLM results will follow the same structure.

Based on the modeling results in Table 8.1, the ranking of significant cost-drivers ("Document Type" is excluded as it is not continuous numeric data), from the highest to the lowest, is given as follows. The ranking of each cost-driver is based the average coefficient (absolute value) of its significant metrics (Note the measures of all cost-driver metrics in

different scales were normalized between 0 and 1 before regression). The explanation of each cost-driver is given afterwards.

$$[Change\ Size(4.3), Document\ Size(3.6), Change\ Volatility(2.7),$$
$$Coupling(-2.0)] \rightarrow Document\ Lifecycle\ Cost$$

Table 8.1-Results of univariate regression modeling for "Document Lifecycle Cost-Drivers"

Cost-Driver	**Metric**	**Coefficient**	**StdErr**	**T-Test**	**P(Coef.)**
Doc Type	*Type*	-0.9253	0.4139	-2.238	0.0294*
Doc Size	*InitialSize*	1.3409	0.2757	1.406	0.0166*
	AvgSize	4.6875	0.8875	5.319	<0.0001***
	FinalSize	4.6679	0.7769	6.002	<0.0001***
Change Size	*ChangeBlock*	4.2310	1.1741	3.477	0.0010**
	ChangeWord	3.9676	1.0632	3.732	0.0005***
	AddWord	5.0427	0.9909	5.061	<0.0001***
	DelWord	3.8526	1.2770	2.906	0.0053**
Change Volatility	*StdevSize*	2.6621	1.0286	2.588	0.0124*
	StdevAddWord	2.4523	0.9379	2.615	0.0116*
	StdevDelWord	2.958	1.187	2.492	0.0159*
Change Frequency	*MinorRevs*	1.5069	0.8791	1.733	0.089
	ReleaseRevs	1.2552	0.7830	1.598	0.116
Doc Quality	*Readability_1*	0.7254	0.6673	0.845	0.402
	Readability_2	-0.5810	1.2527	-1.667	0.645
Coupling	*InRefs*	-1.1208	1.1932	-0.963	0.352
	OutRefs	-1.9833	0.9529	-2.102	0.0403*
Editor	*NumEditors*	0.0474	0.0289	1.641	0.107
	AvgExp	0.0102	0.0881	0.115	0.909

- With respect to **change size** all its metrics are significant. Similar to previous change-based studies on source code [37, 77, 78], which confirmed the correlation between change size and cost, documentation change size also affects cost. However, *ChangeWord* and *AddWord* metrics are more significant than *DelWord* and *ChangeBlock* Our interpretation is that the two metrics capture more time-consuming activities,

comparing to the coarse-grained metric *ChangeBlock* and the *DelWord* metric for relatively effortless deleting words. But we have no evidence to verify this assumption.

- For **document size**, the two metrics *AvgSize* and *FinalSize* indicate a strong positive correlation with cost, while *InitialSize* is considered as moderate significant. One possible explanation is that *AvgSize* and *FinalSize* are able to, to some extent, mirror the revision effort spent after initial creation for a multi-version document. However, *InitialSize* is a precise indictor of total cost, since an initially short document, for example, may grow to become a very large document, thus consuming large amount of cost overall.
- **Change volatility** metrics are also identified as significant, which indicate that high change volatility causes more cost on documentation. This may be explained by the presence of redundant movements of words from one version to next version, including both addition and deletion, which normally costs more effort than a document undertaking smooth evolution. The underlying reason for high change volatility could be multifaceted. For example, it could be caused by the fact that software system has gone through significant changes from time to time. Therefore, the corresponding documentation would need the same revisions to ensure consistency. However, in order to reveal the root-cause, we recommend that further qualitative and quantitative studies should be applied on this aspect.
- **Coupling** metric *OutRefs* is significant and indicates that having more references to other documents reduce cost. This phenomenon can be explained by referring to the Object-Oriented principles, cohesion and coupling. On one hand, redundant details are eliminated by appropriately using reference. Consequently, the significant cost-driver, documentation size, is able to be controlled, so as the cost. On the other, if we try to push

too far on reducing coupling/reference, we will end up with large size documents at which point we lose the quality of cohesion. Therefore, there must be a trade-off for documentation to achieve loose coupling and high cohesion.

- Other candidate cost-drivers, **change frequency**, **documentation quality** (measured by readability) and **editor**, are not considered as significant factors. Change frequency, measuring the number of revision check-ins, surprisingly doesn't show significant correlation with cost. The explanation is that most of revisions (89.8%) had only a few changes and consumed the cost less than one hour, which means that number of versions does not significantly determine the cost.

8.1.3 Multivariate Regression Analysis

Multivariate regression looks at the compound impact of the cost-driver metrics on documentation cost. All cost-driver metrics are chosen to build the GLM model with a stepwise variable selection process. The results of multivariate regression are shown in Table 8.2.

Table 8.2-Results of multivariate regression modeling for "Document Lifecycle Cost-Drivers"

Cost-Driver	**Metric**	**Coefficient**	**StdErr**	**T-Test**	**P(Coef.)**
Doc Size	*AvgSize*	4.2995	0.7294	3.811	<0.0001***
Change Size	*ChangeWord*	1.8552	0.7282	1.807	0.0146*
Change Volatility	*StdevSize*	2.1220	0.7830	1.850	0.0081**
	StdevDelWord	2.6337	1.2893	1.237	0.0461*
Coupling	*OutRefs*	-2.6610	0.8762	-1.902	0.0038**

The multivariate regression model in Table 8.2 shows:

- The ranking of documentation cost-drivers, from the highest to the lowest, is:

$$[Document\ Size(4.3), Coupling(-2.7), Change\ Volatility(2.4),$$
$$Change\ Size(1.9)] \rightarrow Document\ Lifecycle\ Cost$$

- Several cost-driver metrics that were previously identified by univariate analysis are not considered as significant, such as *InitialSize*, *FinalSize*, *ChangeBlock*, *AddWord and DelWord*. This is because that these metrics are to some extent coupled with the identified ones, as they are capturing similar aspects of documentation (e.g., *InialSize*, *AvgSize* and *FinalSize* are measuring the size of a document). Therefore, coupled metrics (also called "Multicollinearity") had gone through a step-wise selection process which results in eliminating less important, duplicated metrics. This process generates a set of less correlated and significant cost-driver metrics.

Based on the identified cost-drivers in Table 8.2, we can conclude that document size, change size, change volatility and coupling play an important role on determining the lifecycle cost. These cost-drivers have been dually confirmed by both univariate analysis and multivariate analysis. Controlling these factors may achieve a higher chance to reduce documentation cost. For example, coupling metric *OutRefs* has a negative significant correlation with cost. It indicates empirically that high number of references of documentation saved the cost in the case study context. But we cannot conclude how much references are suitable for each document or each type of documentation, since there must be a trade-off between document's cohesion and coupling.

8.1.4 Summary of "Document Lifecycle Cost-Drivers"

Table 8.3 summarizes the significant cost-drivers and their metrics identified by both univariate regression and multivariate regression in this section. Column "Effect" indicates the impact direction of cost-drivers, "↑" for positive and "↓" for negative. Column "Significance"

denotes the influence power of each factor, where an p-value bellow than 0.05, 0.01 and 0.001 are marked with one, two and three asterisks respectively. If one metric has been identified twice, its significance for both analyses is separated by a "/" mark.

Table 8.3-Summary of "Document Lifecycle Cost-Drivers" with their effect (positive or negative) and significant level (indicated by p-value)

Cost-Drivers		Univariate Regression	Multivariate Regression	Effect	Significance (by P-Value)
Name	**Metric**				
DocType	*Type*	✓		↓	*
DocSize	*InitialSize*	✓		↑	*
	AvgSize	✓	✓	↑	***/***
	FinalSize	✓		↑	***
Change Size	*ChangeBlock*	✓		↑	**
	ChangeWord	✓	✓	↑	***/*
	AddWord	✓		↑	***
	DelWord	✓		↑	**
Change Volatility	*StdevSize*	✓	✓	↑	*/**
	StdevAddWord	✓		↑	*
	StdevDelWord	✓	✓	↑	*/*
Coupling	*OutRefs*	✓	✓	↓	*/**

8.2 Document Revision Cost-Drivers (RQ2.2)

The cost-driver analysis follows the same procedure as Section 8.1. However, individual revision/version, instead of the whole multi-version document, is considered as one sample. It is

aimed to investigate cost-drivers for each documentation revision (named "Document Revision Cost-Drivers").

Since the observation unit is each revision, some cost-drivers metrics that are previously defined are not applicable, such as change volatility metrics which capture lifecycle behaviours of a document. An adapted summary of investigated cost-driver metrics is listed in Table 8.4, where new metric or metrics with new definitions are highlighted in ***bold***. Another possible cost-driver "Revision" is added in Table 8.4, to investigate the influence of revision type and number.

Table 8.4-Updated summary of candidate cost-drivers in documentation properties and corresponding metrics

Cost-Driver	**Metric**	**Explanation of Metric**
Document Type	*Type*	Type of documents in SDLC;
Revision	***IsRelease***	Whether the revision result in a released version or not;
	RevNum	How many revisions have been submitted, including the current one;
Document Size	*FinalSize*	Size after current revision;
Change Size	*ChangeWord*	Number of word modified;
	AddWord	Number of word added;
	DelWord	Number of word deleted;
	ChangeBlock	Number of changed word blocks;
Document Quality	*Readability_1*	Measured by Flesch Reading Ease[61];
	Readability_2	Measured by Flesch-Kincaid Grade[61];
Coupling	*InRefs*	Number of references coming in a given document;
	OutRefs	Number of documents that a given document refers to;
Editor	*NumEditors*	Number of editors involved in this revision;
	AvgExp	Number of previous check-ins on the same document submitted by the author(s);

8.2.1 Descriptive Statistics

The case study data (55 multi-version documents) has total 1,630 different revision/version check-ins. Please note the focus here is to find out the cost-drivers that cause high cost on single revision. Therefore, the revision check-ins with cost less than one hour are excluded, resulting in 366 revisions/versions as samples for the analysis.

The frequency distributions of cost-driver metrics on these 366 revisions are shown in Figure 8.2.

From their distributions in Figure 8.2, we can observe:

- Most of selected revisions are "non-release" revisions from "Test" documentation, followed by "Process" and "Design" documentation.
- The distribution of *RevNum* indicates that the selected revisions nearly randomly locate, and neither early revisions nor later revisions are predominant in terms of cost.
- Most of selected revisions are from documents under size 8,000 words.
- Most documents have less incoming references than outgoing references.

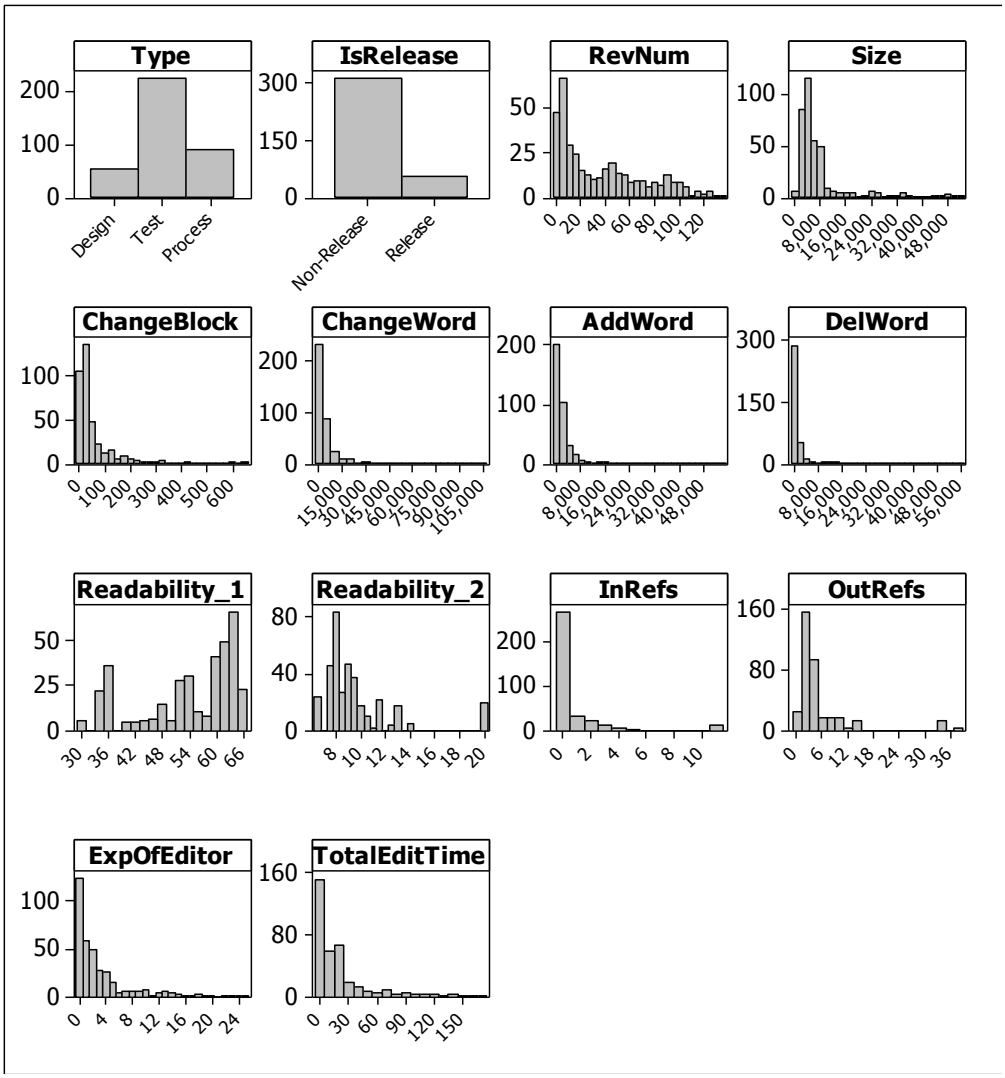

Figure 8.2-Frequency distributions of cost-driver metrics and cost for "Document Revision Cost-Drivers"

8.2.2 Univariate Regression Analysis

The results from univariate regression modeling are provided in Table 8.5, which has the same structure as previous tables demonstrating GLM modeling results.

Table 8.5-Results of univariate regression modeling for "Document Revision Cost-Drivers"

Cost-Driver	Metric	Coefficient	StdErr	T-Test	P(Coef.)
DocType	*Type*	-0.7561	0.2727	-2.731	0.0059**
Revision	*IsRelease*	0.8634	0.2085	3.748	0.0002***
	RevNum	-0.3245	0.3135	-1.041	0.301
Doc Size	*FinalSize*	0.0728	0.4664	0.156	0.876
Change Size	*ChangeBlock*	1.4652	0.7096	2.0865	0.0397 *
	ChangeWord	1.9408	1.0539	1.713	0.0664
	AddWord	2.6545	0.9246	2.865	0.0043**
	DelWord	-0.0323	0.9608	-0.034	0.973
Doc Quality	*Readability_1*	-0.0630	0.0278	-0.227	0.821
	Readability_2	-0.1171	0.3597	-0.325	0.745
Coupling	*InRefs*	-0.4163	0.3889	-1.0071	0.285
	OutRefs	-0.9624	0.4066	-2.293	0.0224*
Editor	*AvgExp*	-1.0527	0.4071	-2.561	0.0108*

From Table 8.5, we find that cost-drivers "Document Type", "Change Size" and "Coupling" are consistent with coarse-grained analysis (Section 8.1), while "Revision" and "Editor" are newly revealed by this analysis.

The ranking of documentation cost-drivers ("Document Type" is excluded as it is not continuous numeric data), from the highest to the lowest, is:

$$[Change\ Size(2.1), Editor(-1.1), Coupling(-1.0), Revision(0.9)] \rightarrow Document\ Revision\ Cost$$

- For **change size**, metrics *ChangeBlock* and *AddWord* are identified as significant, while all change size metrics are significant in previous analysis (Section 8.1.2). We do not have evidence to explain this variance. But for both analyses, metrics *ChangeBlock* and *AddWord* are confirmed as significant cost-drivers on documentation.
- **Editor** (indicated by *AvgExp*) shows significant negative correlation with documentation cost. It reveals that people who have prior revision experience on one document would

spend less effort on upcoming revisions. It suggests that having the same person maintain the evolution of one document is a tangible way to control documentation cost.

- **Coupling** metric *OutRefs* is again significant and indicates that having more references to other documents reduce cost for individual revision check-in. Therefore, it implies the usage of appropriate reference among documentation helps to reduce the cost.
- **Revision type** also plays a significant role on determining documentation cost. The metric, *IsRelease*, shows a strong positive correlation with cost, which indicates that, the revision check-ins which result in a released version would cost more effort. One natural explanation is that "release" revision often requires a relatively stable version for directing downstream development or maintenance. Consequently, it would demand more carefulness and effort from authors.
- Again, documentation **quality** (measured by readability) metrics are not considered significant.

8.2.3 Multivariate Regression Analysis

The results of multivariate modeling for "Document Revision Cost-Drivers" are shown in Table 8.6.

Table 8.6- Results of multivariate regression modeling for "Document Revision Cost-Drivers"

Cost-Driver	Metric	Coefficient	StdErr	T-Test	P(Coef.)
DocType	*Type*	-0.9084	0.3174	-2.547	0.0112*
Revision	*IsRelease*	0.7725	0.2523	-2.062	0.0024**
Editor	*AvgExp*	-0.7060	0.011	-1.095	0.0098**
Coupling	*OutRefs*	-2.6277	0.7827	-2.357	0.0009***

Based on the identified cost-drivers in Table 8.6, we can conclude that document type, revision type, coupling and editor's experience are significantly related to the cost spent on individual documentation revision.

The multivariate regression model in Table 8.6 shows:

- The ranking of documentation cost-drivers, from the highest to the lowest, is:

$$[Coupling(-2.6), Revision(0.8), Editor(-0.7)] \rightarrow Document\ Revision\ Cost$$

- Document size, change size, document quality are not identified as significant cost-drivers.

8.2.4 Summary of "Document Revision Cost-Drivers"

Table 8.7 summarizes the significant cost-drivers and their metrics identified by the analysis in this section. Again, column "Effect" indicates the impact direction of cost factors, "↑" for positive and "↓" for negative. Column "Significance" denotes the influence power of each cost-driver, where an p-value bellow than 0.05, 0.01 and 0.001 are marked with one, two and three asterisks respectively. If one metric has been identified twice, its significances for both analyses are separated by a "/" mark.

Table 8.7- Summary of "Document Revision Cost-Drivers" with their effect (positive or negative) and significant level (indicated by p-value)

Cost-Drivers		Univariate Regression	Multivariate Regression	Effect	Significance (by P-Value)
Name	**Metric**				
DocType	*Type*	✓	✓	↓	**/*
Revision	*IsRelease*	✓	✓	↑	***/**
Change Size	*ChangeBlock*	✓		↑	*
	AddWord	✓		↑	**
Coupling	*OutRefs*	✓	✓	↓	*/***
Editor	*AvgExp*	✓	✓	↓	*/**

8.3 Model Evaluation and Validation

Plots of actual versus fitted effort of the two multivariate regression models are provides in Figure 8.3 and Figure 8.4, respectively. Table 8.8 provides the performance (evaluated by MMRE and RMSE) of the two GLMs, and the maximal variance inflation factors (VIF) for detecting multicollinearity issue. In Table 8.8, the two GLMs are also compared to ordinary linear regression models (LRs) in terms of MMRE and RMSE.

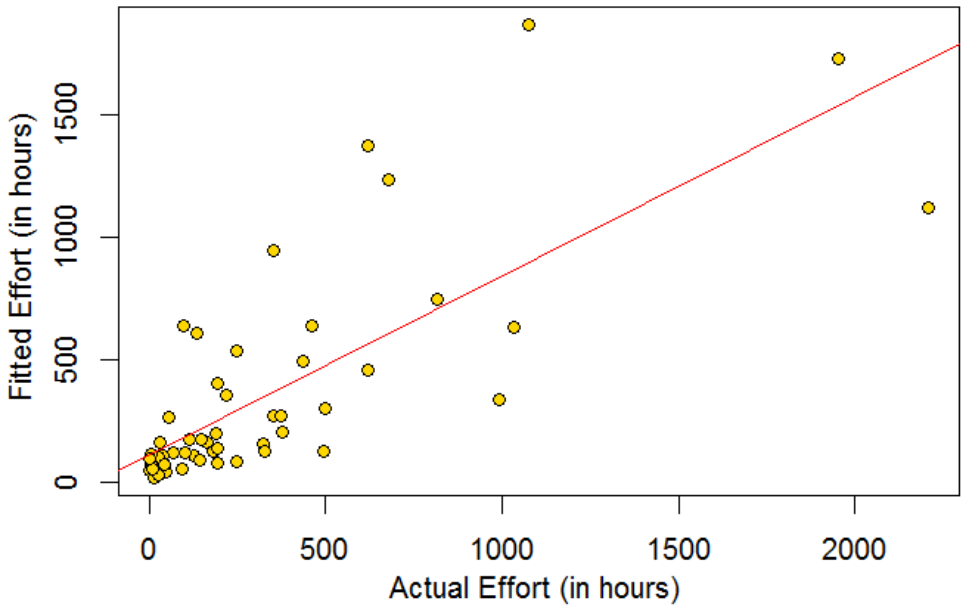

Figure 8.3-Fitted vs. actual effort of "Document Lifecycle Cost-Drivers" model

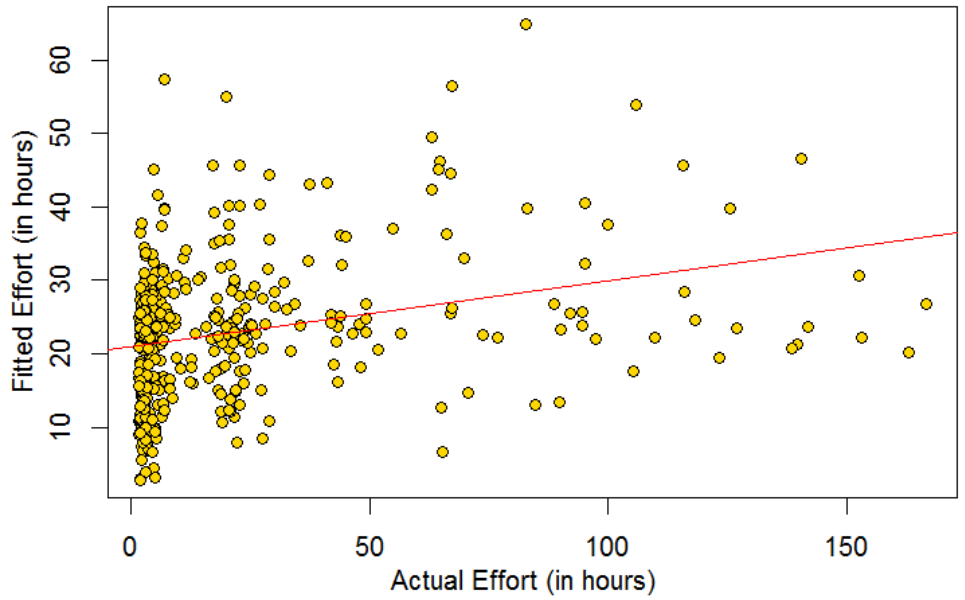

Figure 8.4-Fitted vs. actual effort of "Document Revision Cost-Drivers" model

Table 8.8-Evaluation of multivariate regression models

Metric	Document Lifecycle Cost-Drivers		Document Revision Cost-Drivers	
	LR	**GLM**	**LR**	**GLM**
MMRE	3.21	1.37	3.75	1.83
RMSE	19.73	10.27	10.65	5.25
Max.VIF	1.37		1.58	

From Table 8.8, we notice that our GLMs perform better than ordinary linear models. By comparing the LR and GLM on modeling "Document Lifecycle Cost-Drivers", for instance, MMRE and RMSE decreased from 3.21 and 19.73 to 1.37 and 10.27, respectively. It shows that GLMs are more capable of capturing documentation cost-drivers. However, Conte et al. [79] suggest $MMRE \leq 0.25$ for accepting a model as "good" for effort estimation. Judged by this standard, the performance of both GLMs is relatively poor. This is probably caused by the issue of data quality and quantity. As future work, we should aim at improving the data quality by investigating more and better predictor variables (cost-drivers). Meanwhile, we should pay more attention to gather more and better data to improve data quantity.

For both GLMs, the maximal variance inflation factors (VIF) are 1.37 and 1.58, respectively. They are both less than the threshold 5, which indicates multicollinearity issue is not a threat. Hence, the interpretation of correlation coefficients of cost-drivers is valid for both models.

8.4 Joint Results and Discussions

8.4.1 Implications for the Project under Case Study

The organization under study has spent a substantial amount of cost as indicated by results from Chapter Seven. Most of documentation artifacts have gone through multiple revisions (1630 versions/55 documents≈30 versions/document) over time, which is common for legacy

systems. It is necessary to pay more attention to documentation cost control and process improvement.

We do not plan to give concrete instructions on how to improve documentation process solely based on the evidence from documentation cost, as this issue has to be jointly considered with documentation usefulness and quality. But on the basis of our results, we are able to provide the following recommendations:

- Change Volatility (refer to Section 4.3.2.4 for detailed definition) on documentation artifacts should be controlled to reduce the cost. Actions should be taken to investigate the root-cause of high change volatility. For example, it could be caused by the misconceptions between business and software domains, so that software and software documentation have to been frequently changed to capture business needs. Or it might be, as pointed out by the software manager in NovAtel, people did not follow documentation process regulation since documents in DSTS are supposed to be updated only if major changes exist.
- Appropriate usage of references can reduce documentation cost. Evidence shows that documentation artifacts with more references consumed less lifecycle cost. Reasonable references are considered as good “coupling” and able to reduce documentation redundancy and subsequent maintenance cost.
- Having more experienced people evolving single document may reduce documentation cost. Our results show that the person who has prior experience editing on one document usually spent less effort on upcoming revisions. Though sometimes it is difficult to do so for legacy systems because of employee turnover, we can still try to maintain the same person or group of people in charge of evolving one document.

- Not surprisingly, documentation size shows a significant positive correlation with the cost based on evidence. It recommends that writing concise documentation may reduce documentation cost. Of course we have to find a trade-off between level of details and the cost, and it can only be achieved by combining the results of documentation usage/benefit analysis.
- Our results confirmed that there is strong positive correlations between change size and documentation cost. This is consistent with most change-based studies that tried to identify cost-drivers for source code evolution [36, 37]. Therefore, controlling documentation change size seems to have a higher change to reduce the cost if applicable.

8.4.2 Implications for the Software Engineering Literature

Our systematic mapping of software documentation related studies (Chapter Three) has revealed that documentation cost-related aspects seemed to be neglected by most works. Within the limited number of cost-related studies (12 out of 69), none of them are dedicated to give a practical solution to objectively evaluate documentation cost and investigate underlying cost-drivers. This is the first exploratory case study on this topic.

An important contribution of this study is that it proposed an objective way (by mining software repositories) to understand documentation cost and identify cost-drivers. The results of this study contribute initial insights to the body of knowledge in software engineering regarding documentation cost and cost-drivers.

- The results confirmed a group of cost-drivers on documentation evolution that are consistent with common sense, i.e. Document Type, Document Size, Change Size and Editor’s experience.

- On the other hand, a set of implicit cost-drivers, e.g., Change Volatility and Documentation Coupling (Reference), also had a large effect on documentation cost.

Though we are not aiming to generalize our results across different contexts, the proposed methodology (Chapter Four) with clearly defined processes and metrics can be easily replicated in other contexts. To further assess the generalizability of the approach and results, more case studies across different contexts need to be conducted.

8.5 Generalizability of the Case-Study Results

Stemming from just one case study, the particular models built for documentation cost-drivers do not intend to have any general validity outside of this case study context. The results of this case study are inevitably influenced by context factors pertaining to the development organizations, as the data comes from one system to assess its documentation practice. In addition, additional factors, e.g., application domain or stability of development process, would have likely influence the results we have found here.

However, it is important to notice that our goal is to evaluate the feasibility and applicability of the proposed methodology DCCDA in a real industrial context for assessing documentation cost and cost-drivers. Since no prior work in the literature has provided to identify documentation cost-drivers, the application of the proposed methodology on this case study allows us to provide initial answers to these questions.

8.6 Threats to Validity

This section discusses the threats to validity which are important to in assessing the strengths and limitations of this study.

8.6.1 Construct Validity

The construct validity issue in this case study is related to what extent the selected cost-driver metrics really represent what we intended to measure. Since all quantitative metrics were based on the data automatically mined from a documentation management system, the metrics may not perfectly capture the factors to be investigated. We attempted to mitigate this threat by proposing multiple measures for each cost factor.

One threat to construct validity was the potential bias introduced by the measurement of documentation effort, Step 5 of DCCDA. In the case study, most documents were written in MS Word (>99.4%). The effort upon each version was extracted through reading the API "Total Edit Time" of a MS Word file. This effort would cover most of effort spent on editing and self reviewing a document, but it cannot capture the effort on designing the document beyond the Word file, and the review cost in forms of peer review or formal meeting if any. To include the cost of these effort, further qualitative approached could be applied, such as expert estimations via questionnaire. But we believe that the measured effort in the case study already constituted a substantial amount of overall documentation effort, and the missing of measurement on a relative small portion of effort would not be a serious threat to conclusion validity.

Documentation quality contains many aspects and cannot be fully measured by one or a few metrics, such as readability, consistency, accuracy, up-to-date etc. Therefore it is hardly to judge whether a document has "higher quality" than another without specific criteria. To make sure documentation quality comparison in a meaningful and repeatable manner, "readability" with well defined metrics in literature was selected, as the measurements of other aspects are extremely expensive. This was an obvious construct validity threat to measure documentation quality.

Another threat to construct validity is the measurement of human factors, editors. Two simplified metrics *NumOfEditors* and *AvgExp* were defined to capture people's impact on documentation cost. The number of previous revision check-ins on the same document was used to measure editor's experience. It did not capture the experience working on similar documents or overall documenting experience. But *AvgExp* was still identified as a significant cost factor. In addition, we did not consider their roles as different role players may have different efficiency on writing documentation. All these threats might impact construct validity on measuring human factors.

8.6.2 Internal Validity

The internal validity concerns the degree to which the causal relationship between cost-drivers and cost can be claimed, especially when analysis units cannot be controlled in groups.

One issue to internal validity is the possibility of multicollinearity problem between cost-driver metrics. All identified cost-driver metrics were used in multivariate regression analysis, and some of them may capture similar behaviors of a document. This increased the possibility that significant metrics were identified by chance but not the true underlying effect. To measure the severity of this threat, the variance inflation factor (VIF) was computed for each multivariate regression model. The results show that the VIF for multicollinearity was very low and would not be a serious threat to internal validity.

8.6.3 External Validity

The issue of external validity concerns whether the results of case study can be generalized beyond this specific study context. Three issues limit the generalization of the results from this cause study.

The first issue is the representativeness of the system under study. This case study was conducted on a legacy embedded software system, which has been practicing intensive documentation over time. It is hardly to guarantee the results from this system would be applicable to another context, such as open-source systems. More software systems should be examined in future studies in order to determine the replicability of the findings in this context.

The second threat to external validity is the subject (documentation types) representativeness of this case study. In this case study three high-level documentation types were selected to study documentation cost and cost-drivers. More documentation types, for example requirements or code comments, should also be considered in the future. The extended selection of documentation types enables the comparisons across different types of documentation, to be able to achieve a more comprehensive analysis of documentation in general.

The third issue is the number of documentation artifacts under study, as only limited number of documentation artifacts (Conceptual Design, Test Plan and Process Regulation) exist for the system under study (OEM6). However, the analysis was performed in two different levels of granularity, where in the more fine-grained level (1,630 versions) each version was treated as one sample to eliminate the threat from sample size.

8.7 Chapter Summary

In this chapter, we presented the case study results on documentation cost-driver analysis. Cost-driver analysis was conducted through two different levels, coarse-grained level to capture the cost-drivers on the lifecycle cost of a documentation artifact (Section8.1), and fine-grained level to investigate the cost-drivers for individual documentation revision (Section 8.2). The identified cost-drivers from both analyses were jointly discussed, from the perspectives of implications for the project and the implications for software engineering (Section 8.3). The

potential threats to the validity of this case study were discussed, in terms of Conclusion Validity, Construct Validity, Internal Validity and External Validity (Section 8.6).

The results of cost-driver analysis pointed out the underlying cost-drivers that are worth the attention of the organization for documentation cost control. In addition, they brought initial answers to the body of knowledge in software engineering regarding documentation cost-drivers, though we are not aiming to generalize these results across different contexts.

Chapter Nine - **Conclusions and Future Work**

9.1 Summary

Software documentation is considered as an important factor on maintainability for legacy systems, but meanwhile is an expensive activity to practice. Documentation practices can be improved if we can better understand documentation cost and identify cost-drivers that have been shown empirically to affect the cost. Due to the lack of such study in the literature, this thesis presented a systematic methodology DCCDA to do so.

DCCDA provided an objective way to evaluate documentation cost with defined metrics and identify underlying cost-drivers via mining relevant repositories (Chapter Four). Regression models were built to identify cost-drivers in documentation properties that correlated with the cost. To make the method more practical and operational, the data preparation and measurement process of DCCDA were automated with tool support (0). It was applied to an industrial case study to help assess documentation cost and identify underlying cost-drivers, and prepare us to improve documentation process cost-effectiveness (Chapter Six). Two central results from the case study are:

- Documentation effort from various perspectives (Chapter Seven), i.e., single document, one documentation type, individual person and over timeline. These distributions help to understand documentation cost from different aspects and granularities. For example, the results help to reveal which documentation artifact has gone through most revisions versus consumed most effort, and who committed most revisions or spent most time on documentation, etc.
- Documentation cost-drivers (Chapter Eight). The results, on one hand, confirmed the cost-drivers that are consistent with common sense, i.e., Document Type, Document Size,

Change Size and Editor's Experience. On the other hand, a set of context-specific cost-drivers, for example Change Volatility and Coupling/Reference, also had a large and consistent effect on documentation cost. These factors should be paid enough attention to control documentation cost and eventually improve its cost-effectiveness.

9.2 Future Work Directions

9.2.1 For Research Methodology

As a future work on the method DCCDA, we suggest conducting more case studies from different development processes, e.g., Waterfall, Iterative and Agile, to validate our methodology and identify context-specific documentation cost-drivers. These results would help to establish the initial basis for future confirmatory studies. In addition, more documentation types should be investigated to evaluate DCCDA, such as requirement and code comments, in order to reveal cost-drivers for a specific documentation type. Due to the availability of data, only three types of documentation (Conceptual Design, Test Plan and Process Regulation) were studied in this case study.

We also suggest refining and extending the selection and measurement of candidate cost-drivers in documentation properties, as it is inevitably influenced by the types of documentation to study and the availability of data. To investigate the correlation between documentation quality and cost, for instance, we can further consider the impact of documentation accuracy and up-to-dateness as quality indicators. However, we should notice that they can only be measured with qualitative approaches, instead of the easy data in repositories. Hence, people should consider the trade-off between measurement completeness and measurement effort before conducting such study.

9.2.2 For the CRD Project under Study

As a future work on the CRD project, we plan to convert the measured documentation effort to real cost. By referring to people's payroll information, we would be able to calculate the cost of each document version.

Our method has found a set of cost-drivers existing in the case study context. These cost-drivers should be paid enough attention to control documentation cost in the future. But we cannot reveal the root-causes of these factors. For example, Change Volatility was identified as a significant cost-driver. This phenomenon might be introduced by requirement volatility so that people had to make subsequent changes on source code and documentation, or lack of suitable regulation on documentation process. Therefore, follow-up qualitative studies should be focusing on the root-causes of these cost-drivers.

As part of a three-year CRD project, our ultimate goal is to improve software documentation cost-effectiveness and to achieve software maintainability for our industrial partner, NovAtel. The results on documentation cost from this thesis should be combined with quantitative measurement of documentation usage/benefit accomplished by others. Then we are able to address the question whether the cost outweighs the gained benefit of documentation. Moreover, a fine-grained evaluation on cost-benefit aspects of documentation artifacts would help to ease documentation maintenance effort, by prioritizing documentation with high benefit and relatively low cost. Our ultimate goal is to find the answer to the question "What/How should we document to enable optimal subsequent development and maintenance activities?" in NovAtel.

References

[1] A. Mockus, S. G. Eick, T. L. Graves, and A. F. Karr, "On Measurement and Analysis of Software Changes," National Institute of Statistical Sciences BL0113590-990401-06TM, 1999.

[2] R. Glass, *Facts and Fallacies of Software Engineering*: Addison WesLey, 2002.

[3] M. Ramage and K. Bennett, "Maintaining Maintainability," in *IEEE International Conference on Software Maintenance*, Bethesda, Maryland, 1998.

[4] B. P. Lientz, E. B. Swanson, and G. E. Tompkins, "Characteristics of application software maintenance," *Communications of ACM*, vol. 21, pp. 466-471, 1978.

[5] E. Arisholm, L. C. Briand, S. E. Hove, and Y. Labiche, "The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation," *IEEE Transactions on Software Engineering*, vol. 32, pp. 365-381, 2006.

[6] A. Forward, "Software Documentation – Building and Maintaining Artefacts of Communication," in *Ottawa-Carleton Institute for Computer Science*. vol. Master in Computer Science: University of Ottawa, 2002.

[7] S. Ambler, "Agile/Lean Documentation: Strategies for Agile Software Development," in *http://www.agilemodeling.com/essays/agileDocumentation.htm*, Last accessed: June 2012.

[8] F. Maurer and S. Martel, "Extreme programming: Rapid development for Web-based applications," *IEEE Internet Computing*, vol. 6, pp. 86-90, 2002.

[9] G. Ruhe and V. Garousi, "Tuning of Artifact and Process Parameters towards Optimized Maintenance," NSERC CRD Project #CRDPJ414157-11, 2011.

[10] A. E. Hassan, "The road ahead for Mining Software Repositories," in *Frontiers of Software Maintenance*, 2008.

[11] A. E. Hassan and T. Xie, "Software intelligence: the future of mining software engineering data," in *Proceedings of the FSE/SDP workshop on Future of software engineering research*, Santa Fe, New Mexico, USA, 2010.

[12] H. Gall, M. Jazayeri, and J. Krajewski, "CVS release history data for detecting logical couplings," in *Proceedings 6th International Workshop on Principles of Software Evolution*, Los Alamitos CA, 2003.

[13] J. Sliwerski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?," in *Proceedings 2nd International Workshop on Mining Software Repositories*, New York, 2005.

[14] H. Kagdi, M. L. Collard, and J. I. Maletic, "A survey and taxonomy of approaches for mining software repositories in the context of software evolution," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 19, pp. 77-131, 2007.

[15] I. Kwan and D. Damian, "A Survey of Techniques in Software Repository Mining," University of Victoria Technical Report DCS-340-IR, 2011.

[16] A. John, H. Lyndon, and C. M. Gail, "Who should fix this bug?," in *Proceedings of the 28th international conference on Software engineering*, Shanghai, China, 2006.

[17] A. Hindle, M. W. Godfrey, and R. C. Holt, "What's hot and what's not: Windowed developer topic analysis," in *IEEE International Conference on Software Maintenance*, 2009.

[18] A. Hindle, N. A. Ernst, M. W. Godfrey, and J. Mylopoulos, "Automated topic naming to support cross-project analysis of software maintenance activities," in *Proceedings of the*

8th Working Conference on Mining Software Repositories Waikiki, Honolulu, HI, USA, 2011.

[19] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What Makes a Good Bug Report?," *IEEE Transactions on Software Engineering*, vol. 36, pp. 618-643, 2010.

[20] J. Ratzinger, T. Sigmund, P. Vorburger, and H. Gall, "Mining Software Evolution to Predict Refactoring," in *First International Symposium on Empirical Software Engineering and Measurement*, 2007.

[21] G. Robles, J. M. Gonzalez-Barahona, M. Michlmayr, and J. J. Amor, "Mining large software compilations over time: another perspective of software evolution," in *Proceedings of the 2006 international workshop on mining software repositories*, Shanghai, China, 2006.

[22] A. Hindle, M. W. Godfrey, and R. C. Holt, "Software process recovery using Recovered Unified Process Views," in *IEEE International Conference on Software Maintenance*, 2010.

[23] P. Daryl, H. Abram, and D. Premkumar, "A simpler model of software readability," in *Proceedings of the 8th Working Conference on Mining Software Repositories* Waikiki, Honolulu, HI, USA, 2011.

[24] C. Gerardo, C. Michele, C. Luigi, and P. Massimiliano Di, "Using multivariate time series and association rules to detect logical change coupling: An empirical study," in *Proceedings of the 2010 IEEE International Conference on Software Maintenance*, 2010.

[25] T. T. BARKER, "Software documentation: from instruction to integration," *IEEE Transactions on Professional Communication*, vol. 33, pp. 172-177, 1990.

[26] D. L. Parnas and S. Nanz, *Precise Documentation: The Key to Better Software*: Springer Berlin Heidelberg, 2011.

[27] J. Zhi, V. Garousi, B. Sun, G. Garousi, S. Shahnewaz, and G. Ruhe, "Cost, Benefits and Quality of Technical Software Documentation: A Systematic Mapping," *submitted to Journal of Systems and Software*, 2012.

[28] B. Boehm, "Value-based software engineering: reinventing," *SIGSOFT Software Engineering Notes*, vol. 28, p. 3, 2003.

[29] M. Jorgensen and M. Shepperd, "A Systematic Review of Software Development Cost Estimation Studies," *IEEE Transactions on Software Engineering*, vol. 33, pp. 33-53, 2007.

[30] S. Chulani, B. Boehm, and B. Steece, "Bayesian analysis of empirical software engineering cost models," *IEEE Transactions on Software Engineering*, vol. 25, pp. 573-583, 1999.

[31] NASA, *Software Measurement Guide Book. Revision 1.*: Software Engineering Laboratory Series. NASA-GB-001-94, 1995.

[32] NASA, *Software Process Improvement Book. Revision 1.*: Software Engineering Laboratory Series. NASA-GB-001-95, 1996.

[33] I. Sánchez-Rosado, P. Rodríguez-Soria, B. Martín-Herrera, J. Cuadrado-Gallego, J. Martínez-Herráiz, and A. González, "Assessing the Documentation Development Effort in Software Projects," in *Proceedings of the International Conferences on Software Process and Product Measurement*, Amsterdam, The Netherlands, 2009.

[34] L. C. Briand and J. Wüst, "The Impact of Design Properties on Development Cost in Object-Oriented Systems," in *Proceedings of the 7th International Symposium on Software Metrics*, 2001.

[35] J. Li, T. Stålhane, J. M. W. Kristiansen, and R. Conradi, "Cost drivers of software corrective maintenance: An empirical study in two companies," in *2010 IEEE International Conference on Software Maintenance*, 2010.

[36] V. Nguyen, B. Boehm, and P. Danphitsanuphan, "A controlled experiment in assessing and estimating software maintenance tasks," *Information and Software Technology,* vol. 53, pp. 682-691, 2011.

[37] H. Benestad, B. Anda, and E. Arisholm, "Understanding cost drivers of software evolution: a quantitative and qualitative investigation of change effort in two evolving software systems," *Empirical Software Engineering,* vol. 15, pp. 166-203, 2010.

[38] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *12th International Conference on Evaluation and Assessment in Software Engineering*, 2008.

[39] B. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software engineering," *Evidence-Based Software Engineering,* 2007.

[40] S. Ali, L. C. Briand, H. Hemmati, and R. K. Panesar-Walawege, "A Systematic Review of the Application and Empirical Investigation of Search-based Test-Case Generation," *IEEE Transactions on Software Engineering,* vol. 36, pp. 742-762, 2010.

[41] F. Elberzhager, J. Münch, and V. T. N. Nha, "A systematic mapping study on the combination of static and dynamic quality assurance techniques," *Information and Software Technology*, vol. 54, pp. 1-15, 2012.

[42] V. Garousi, "Classification and trend analysis of UML books (1997-2009)," *Journal on Software & System Modeling,* vol. 11, pp. 273-285, 2011.

[43] "Cost, Benefits, Usage, and Quality of Technical Software Documentation System Mapping Repository (Online)," 2012.

[44] W. J. Dzidek, E. Arisholm, and L. C. Briand, "A Realistic Empirical Evaluation of the Costs and Benefits of UML in Software Maintenance," *IEEE Transactions on Software Engineering*, vol. 34, pp. 407-432, 2008.

[45] P. C. Pendharkar and J. A. Rodger, "An empirical study of factors impacting the size of object-oriented component code documentation," in *Proceedings of the 20th annual international conference on Computer documentation*, Toronto, Ontario, Canada, 2002.

[46] A. Dautovic, R. Plösch, and M. Saft, "Automated Quality Defect Detection in Software Development Documents," in *Proceedings of Fifth International Workshop on Software Quality and Maintainabilit*, Carl Von Ossietzky Universität in Oldenburg, Germany, 2011.

[47] E. Soloway, R. Lampert, S. Letovsky, D. Littman, and J. Pinto, "Designing documentation to compensate for delocalized plans," *Communications of the ACM,* vol. 31, pp. 1259-1267, 1988.

[48] D. Schreck, V. Dallmeier, and T. Zimmermann, "How documentation evolves over time," in *9th international workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting*, Dubrovnik, Croatia, 2007.

[49] C. J. Stettina and W. Heijstek, "Necessary and neglected?: an empirical study of internal documentation in agile software development teams," in *Proceedings of the 29th ACM international conference on Design of communication*, Pisa, Italy, 2011.

[50] T. C. Lethbridge, J. Singer, and A. Forward, "How Software Engineers Use Documentation: The State of the Practice," *IEEE Software*, vol. 20, pp. 35-39, 2003.

[51] D. L. Parnas and Sebastian Nanz, "Precise Documentation: The Key to Better Software " in *THE FUTURE OF SOFTWARE ENGINEERING*: Springer Berlin Heidelberg, 2011, pp. 125-148.

[52] R. M. Posten, "Selecting Software Documentation Standards," *IEEE Software*, vol. 2, pp. 90-91, 1985.

[53] G. Du and G. Ruhe, "Two machine-learning techniques for mining solutions of the ReleasePlaner decision support system," *Information Sciences*, 2009, In Process.

[54] E. Paikari, G. Ruhe, B. Sun, and E. Livani, "Customization support for CBR-based defect prediction," in *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*, Banff, Alberta, Canada, 2011.

[55] R. H. Myers, D. C. Montgomery, and G. G. Vining, *Generalized linear models: with applications in engineering and the sciences*: J. Wiley, 2002.

[56] L. Fahrmeir, W. Hennevogl, and G. Tutz, *Multivariate statistical modelling based on generalized linear models*, 2 ed.: New York: Springer, 2001.

[57] V. S. Rini and E. Berghout, *The Goal/Question/Metric Method*: McGraw-Hill Education, 1999.

[58] G. A. Hall and J. C. Munson, "Software evolution: code delta and code churn," *Systems and Software*, vol. 54, pp. 111-118, 2000.

[59] S. A. Ajila and R. T. Dumitrescu, "Experimental use of code delta, code churn, and rate of change to understand software product line evolution," *Journal of Systems and Software*, vol. 80, pp. 74-91, 2007.

[60] Microsoft, "Range.Revisions Property (Word)," in *http://msdn.microsoft.com/en-us/library/ff838481.aspx*, Last accessed: June 2012.

[61] W. H. Dubay, "The Principles of Readability," *Costa Mesa, CA: Impact Information*, 2004.

[62] J. W. Hunt and M. D. McIlroy, "An algorithm for differential file comparison," Computing Science Technical Report 41, 1975.

[63] T. DeMarco and T. Lister, "Programmer performance and the effects of the workplace," in *Proceedings of the 8th international conference on Software engineering*, London, England, 1985.

[64] J. Fox and G. Monette, "Generalized Collinearity Diagnostics," *Journal of the American Statistical Association*, vol. 87, pp. 178-183, 1992.

[65] C. List, D. Grimm, G. Hammer, J. Tucht, and K. Varis, "WinMerge 2.12.4," in *http://winmerge.org/*, Last accessed: June 2012.

[66] I. SoftInterface, "WordDocDiff," in *http://www.softinterface.com/wdd/wdd.htm*, Last accessed: June 2012.

[67] Microsoft, "Parallel Programming in the .NET Framework 4," in *http://msdn.microsoft.com/en-us/library/dd460693.aspx*, Last accessed: June 2012.

[68] Microsoft, "ReadabilityStatistics Interface," in *http://msdn.microsoft.com/en-us/library/ms264649.aspx*, Last accessed: June 2012.

[69] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: an open source software for exploring and manipulating networks," in *International AAAI Conference on Weblogs and Social Media*, 2009.

[70] A. C. Oliver, G. Stampoultzis, A. Sengupta, R. Klute, and D. Fisher, "Apache POI - the Java API for Microsoft Documents," in *http://poi.apache.org/*, Last accessed: May 2011.

[71] Apache, "PDFBox," in *http://pdfbox.apache.org/*, Last accessed: March 10 2012.

[72] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, pp. 131-164, 2009.

[73] C. Kaner and W. P. Bond, "Software engineering metrics: What do they measure and how do we know? ," in *10th International Software Metrics Symposium*, 2004.

[74] H. Sackman, W. J. Erikson, and E. E. Grant, "Exploratory experimental studies comparing online and offline programming performance," *Communications of ACM*, vol. 11, pp. 3-11, 1968.

[75] R. W. Numrich, L. Hochstein, and V. R. Basili, "A metric space for productivity measurement in software development," in *Proceedings of the second international workshop on Software engineering for high performance computing system applications*, St. Louis, Missouri, 2005.

[76] G. Gousios, E. Kalliamvakou, and D. Spinellis, "Measuring developer contribution from software repository data," in *Proceedings of the 2008 international working conference on Mining software repositories*, Leipzig, Germany, 2008.

[77] T. L. Graves and A. Mockus, "Inferring Change Effort from Configuration Management Databases," in *Proceedings of the 5th International Symposium on Software Metrics*, 1998.

[78] W. M. Evanco, "Analyzing Change Effort in Software during Development," in *Proceedings of the 6th International Symposium on Software Metrics*, 1999.

[79] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software engineering metrics and models*: Benjamin-Cummings Publishing Co., Inc., 1986.

List of Publications

1) Accepted:

- A. Niknafs, B. Sun, M. M. Richter, and G. Ruhe, "Comparative analysis of three techniques for predictions in time series with repetitive patterns, " in *Proceedings of the 13th International Conference on Enterprise Information Systems*, Beijing, China, 2011.
- E. Paikari, G. Ruhe, B. Sun, and E. Livani, "Customization support for CBR-based defect prediction", in *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*, Banff, Canada, 2011.

Submitted:

- J. Zhi, V. Garousi, B. Sun, G. Garousi, S. Shahnewaz, and G. Ruhe, "Cost, Benefits and Quality of Technical Software Documentation: A Systematic Mapping," *Journal of Systems and Software*, 2012.