Evaluating Usage and Quality of Technical Software Documentation: An Empirical Study

Golara Garousi[1], Vahid Garousi[1,2], Mahmoud Moussavi[1], Guenther Ruhe[1,3], Brian Smith[4]

[1]Department of Computer and Electrical Engineering[1]
[3]Department of Computer Science
University of Calgary, Alberta, Canada
{ggarousi, vgarousi, moussam, ruhe}@ucalgary.ca

[2]Informatics Institute, Middle East Technical University, Ankara, Turkey
vahid@metu.edu.tr

[4]NovAtel Inc.
Calgary, Alberta, Canada
brian.smith@novatel.com

ABSTRACT

Context: Software documentation is an integral part of any software development process. However, software practitioners are often concerned about the lack of usage and quality of documentation in practice. Unfortunately, in many projects, practitioners find that software documentation artifacts are outdated, incomplete and sometimes not beneficial. **Objective:** Motivated by the needs of NovAtel Inc. (NovAtel), a world-leading company of GPS software systems, we propose in this paper an approach to analyze the usage and quality of software documentation in development and maintenance phases. **Method:** The approach incorporates inputs from automated analysis (e.g., mining of project's data) and also experts' opinion extracted from survey-based questionnaire. The approach has been designed based on the "action-research" approach and in close collaboration between industry and academia. **Results:** To evaluate the feasibility and usefulness of the proposed approach, we have applied it in an industrial setting and results are presented in this paper. One of the results is that, in the context of our case-study, usage of documentation for an implementation purpose is higher than the usage for maintenance purposes. **Conclusion:** It is concluded that the usage of documentation differs for various purposes and it depends on the type of the information needs as well as the task to be completed (e.g. development and maintenance). In addition, we identify the most important and relevant quality attributes which are critical to improving documentation quality.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement - *Documentation*

General Terms

Technical software documentation, usage, quality.

Keywords

Empirical software engineering, software documentation, software development, maintenance, action research, case study.

1. INTRODUCTION AND MOTIVATIONS

Software documentation is defined as any artifact that helps to communicate information about a software system among stakeholders (e.g., requirement engineers, and developers) [1]. There are generally two types of software documentation: (1) technical documentation (e.g., design documents), and (2) user manuals. In this paper, we focus on technical documentation.

Technical software documentation is one of the important practices that is generally believed to improve development and maintenance activities [2]. Typical examples of technical software documentation include: requirement specifications, design (architectural) documents, source code comments, test documents and defect (bug) reports. Documents are supposed to help software engineers to comprehend a given program or system and accomplish development and modifications tasks more efficiently.

Deciding about the amount and depth of documentation remains a major challenge for many practitioners. Many teams either develop too much, too little and/or outdated documents. In other words, answering the question "How much documentation is enough?" after many years, since software documentation has been practiced, is still not trivial [1, 3].

Unfortunately, there is no end to the stories of legacy software systems lacking documentation or low-quality documentation. In fact, documentation is often incomplete, inconsistent and out of date in practice [1]. In addition, the other concern about documentation is that it is usually perceived as an expensive activity, sometimes not useful and difficult to maintain in many projects [1].

Since high-quality and useful documentation is believed to be one of the key factors in producing high-quality software [1, 3], we investigate in this study the usage and quality of documentation in a real-world industrial context. To do so, we propose a hybrid approach (based on experts' opinion and project data) to mine and analyze benefit and quality of documentation, and to provide evidence-based guidelines for documentation (process) improvement.

The current study is part of a three year Collaborative Research and Development (CRD) project with an industrial partner in which the ultimate goal is to propose guidelines (i.e., improvement opportunities) for the documentation process and practices exercised in a large Canadian software company, and to improve documentation efficiency (i.e., maximizing the benefits of documentation by putting the least cost).

The main contributions of this paper are:

- A hybrid approach for assessing usage and quality of documentation.
- Applying an action-research approach [4] and empirical evaluation of the usage of documentation in dependence of the type of information sources, the life-cycle phase, the

EASE'13, April 14-16, 2013, Porto de Galinhas, Brazil.

document type, the role of the user, and the degree of experience of the user.

- Evaluation of the different aspects of documentation quality and comparing perceived with desired quality.
- Based on the analyzed case study results, providing guidelines for improvement of the documentation process.

The remainder of the paper is organized as follows. Section 2 discusses the related work. Section 3 presents the goal of this study and the proposed approach. Section 4 presents the results from the industrial case study. Section 5 concludes the paper and discusses future work directions.

2. BACKGROUND AND RELATED WORK

The following sections present the background and the related work in the area of documentation benefit and quality.

2.1 Benefits of Documentation

Andrew Forward defined in [5] the usefulness of a documentation artifact as follows: "*A measure of how well a document's content can provide knowledge, information and/or insight to its reader with respect to other available artifact consider both the activity and value of the artifact"*.

Documentation serves various purposes during the different phases of Software Development Life-Cycle (SDLC) [2, 6, 7]. Generally, software engineers rely on technical documentation as an aid in system understanding and program comprehension. Without documentation, the only reliable and objective information repository about the software system is the source code itself. Nevertheless, often times, it takes a lot of time and effort to explore the source-code itself to find the relevant information and gain an understanding of the system functionality [8]. In addition, documentation can take both textual as well as graphical forms, e.g., using the Unified Modeling Language (UML).

By synthesizing the existing literature on benefits of documentation (e.g., [2, 6, 7]), we formalized the concept as a UML class diagram, as illustrated in Figure 1. In terms of terminology, we should mention that once a given documentation artifact is accessed during a given software engineering task, it may or may not provide benefits. When a given document provides benefits, it will be used for the task in hand.

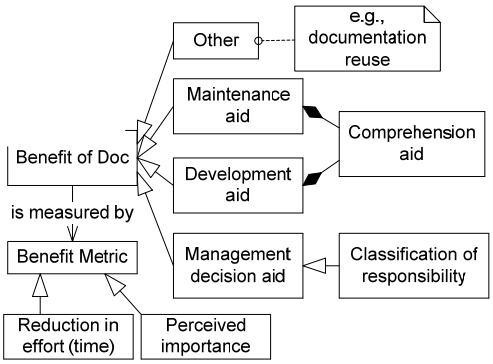

Figure 1. A meta-model for documentation benefit.

In the meta-model of Figure 1, we have classified benefits into four categories: (1) maintenance aid, (2) development aid, (3) management decision aid, and (4) other. For both maintenance aid and development aid, comprehension aid is an important aspect. This is because many aspects of a software under analysis need to be understood (i.e., comprehended) in order to be properly modified, including its "functionality, architecture, and a myriad of design details" [7].

To quantitatively or qualitatively measure benefits of documentation, several works have been reported in the literature which are mostly based on questionnaire-based surveys, e.g., [9, 10]. In [9], the authors presented the results of a survey of 76 software maintainers in Brazil to try to establish what documentation artifacts are the most useful to them. The survey confirmed that source code and comments are the most important artifact to understand a system to be maintained. Data model and requirement description were other important artifacts. Surprisingly, and contrary to what we found in the literature, architectural models and other general view of the system were not found to be very important in [9].

The study in [10] was an experiment conducted on 34 subjects in Norway. The object under study was a system comprising of 2.7K SLOC and around 100 pages of documentation. The subjects recorded the effort spent on different enhancement-maintenance tasks during the experiment. The following empirical findings were reported: (1) The aid of having documentation available during system maintenance reduces the time needed to understand the system and the changes implied by a change request, and (2) It also enables the maintainer with more time and better knowledge so that s/he can make more detailed changes to the system in a restricted amount of time.

2.2 Quality of Documentation

During the SDLC, the quality of documentation artifacts developed in earlier (up-stream) phases, e.g., requirements, will generally have a profound impact on the quality of artifacts developed in the later (down-stream) phases, e.g., implementation.

Similar to software quality which has various quality attributes, quality of documentation consists of various attributes such as: accuracy, completeness, consistency, correctness, information organization, format, readability, spelling and grammar, traceability, trustworthiness, and up-to-date-ness [6, 11].

Quality of documentation has been analyzed in a large number of studies so far (e.g., [5-7, 11-13]) which have based on either experts' opinion (i.e., using questionnaire-based surveys) [5-7, 11, 12], or by mining project data, e.g., using metrics to quantitatively measure documentation quality [13]. We discuss next a few of the above related works.

Arisholm, Dzidek and colleagues reported two studies [7, 12] related to benefit and quality of UML-based documentation. Among the factors that they considered was the impact of documents' up-to-date-ness on the quality of the follow-up (down-stream) artifacts.

Kajko-Mattsson conducted a comprehensive survey across 18 Swedish companies [11] and conducted that: "*Poor system documentation, …, is the primary reason for quick software system quality degradation and aging.*"

The work reported in [13] is an example approach for using metrics to quantitatively measure documentation quality. In an interesting approach, the authors have adapted the "test results" and "test coverage" notions from the software testing domain to the domain of documentation. The study adapts the following heuristics: testing of technical documentation is done manually, e.g., by asking subjects to find information necessary to complete a certain task. The "test" (in the context of documentation) is successful if sufficiently many subjects found the relevant information (in time). Also, the authors of [13] adapt the clone

detection approach from the existing literature as follows: a metrics based on clone detection for technical documentation can provide a measurement of a document's uniqueness, i.e., what fraction of the document is not part of any other?

To provide a systematic view of the domain of software documentation, we have recently conducted a systematic mapping of 69 studies in the area of documentation cost, usage, benefits, and quality which can be accessed online at [14, 15].

Last but not least, since software engineering is a human-intensive field, there have been a large number of "hybrid" approaches (based on experts' opinion and project data) in various sub-fields of software engineering. For example, Freimut et al. proposed such a hybrid approach [16] to determining the cost-effectiveness of inspections. Since software documentation is both a human-intensive and also a technical field, we believe that using "hybrid" approaches in this area would provide better results compared to using each of the approaches in isolation, i.e., surveys or measurements based on automated mining.

2.3 Action-Research in Software Engineering

Last but not least, our work relates to and contributes to the body of industrial evidence and action-research (AR) in software engineering. According to a 2009 survey paper on the extent of AR use in software engineering [4], not many papers follow a formal AR or case-study process, but instead take the form of experience papers. In this study, we intended to take a (semi-) formal AR process (details in Section 3.1).

We have had recent experience with the AR approach in two recent projects in which we used AR to introduce automated configuration testing in an industrial setting [17] and also to contribute another automated testing framework for another industrial partner [18].

3. GOAL AND METHODOLOGY

We start next with our research approach and problem formulation/motivation, inspired through an AR approach. We then present the research goal, research questions, hypotheses, our hybrid approach, and the metrics of our study.

3.1 Research Approach: Action-Research

In terms of our research approach, we followed both the *improving* case study [17] and the AR approaches [18]. As Runeson and Höst [17] point out, these two approaches are inter-related and are encouraged to be utilized jointly in a project between academia and industry. To plan, execute and manage our AR project, we used the recommendations and guidelines provided in [4, 17, 19]. Avison et al. also made suggestions for controlling action research projects [20]. They discussed three aspects of control: (1) the procedures for initiating an AR project, (2) those for determining authority within the project, and (3) the degree of formalization. Our research is inspired by these guidelines.

To put our AR approach in context, the classification of our AR approach using the AR classification map provided by Santos and Travassos [4] is provided in Table 1. The "problem" motivating the need for the AR collaboration was that development and maintenance of documentation for the company under study have incurred a large cost in the past. There is a need to optimize the documentation process and reduce the cost of documentation (both development and maintenance) without adversely impacting the quality and/or delivery schedule of the software products.

In terms of adherence, the classification in [4] proposed three possibilities: (1) Inspired – when the focus of the project is on the researchers learning from a real problem exploring SE research without strictly following the AR principles, (2) Based – when the AR approach is modified or combined with other empirical methods (e.g., case studies), and (3) Genuine – when the full essence of action research approach is present. For our case, it is thus "Based" since we utilized both AR and case study.

Table 1. Classification of our action-research approach based on the classification provided by Santos and Travassos [4].

Attribute		**Value**
Problem		Development and maintenance of documentation for the company under study is expensive. There is a need to optimize the documentation process and optimize its amount.
Action		Development and evaluation of a hybrid approach to mine the projects data and also get expert's opinion to measure and analyze documentation benefit and quality
Adherence		Based
Type		Action research
Length		40 months (18 months have passed so far)
Data collection		Quantitative and qualitative were used. Techniques: Metrics (for quantitative data) and observation (for qualitative data)
Action-research control structures	Initialization	Practitioner
	Authority	Identity
	Formalization	Formal- written contract
Number of action-research cycles		Five (expected)

Our project was of type "action research", rather than action Science or action Learning. The length of our project is planned to be 40 months (3.5 years), from which about 1.5 years has passed as of this writing. We collected both quantitative and qualitative data. Initialization of the project was by the practitioners. Researchers and practitioner had identical authority levels. Before the project start date, we had a formal written contract in place. During the 1.5-year period thus far, we iterated through two major AR cycles in which the initial assessment of documentation cost, usage and quality have been conducted and initial guidelines has been passed to the industrial partner. Details and results are discussed in the remainder of this paper.

3.2 Research Questions

This research is addressing two main research questions (RQ's). Each question is further divided into a number of sub-questions, as discussed below.

RQ 1-What are key factors having an impact on documentation usage?

- **RQ 1.1:** Information Sources

  How does the usage of documentation compare to other information sources, i.e., source-code, communication with team members, and existing (self-) knowledge of developers?
- **RQ 1.2:** Life-cycle phase

  Is the amount of documentation usage different in development and maintenance phases? For example, is documentation used more often in development compared to maintenance?
- **RQ 1.3:** Document type

Does document type have a significant impact on its usage? For example, are requirements documents accessed more often compared to design documents?

- **RQ 1.4:** Role (position)

  Does practitioner's job function have a significant impact on the level of documentation usage? For example, do testers use documentation more compared to developers?

- **RQ 1.5:** Degree of experience

  Does experience have a significant impact on the level of documentation usage? Do engineers with more years of experience use less documentation?

- **RQ 1.6:** Patterns of usage

  What type of patterns, trends and insights in terms of documentation usage can be extracted from the documents access log? This RQ is an investigation with the "exploratory" nature (e.g., similar to studies such as [21]).

RQ 2- What attributes affect documentation quality? We break down this RQ into the following sub-RQs:

- **RQ 2.1:** What is the impact of quality attributes (such as accuracy, completeness, consistency) on the overall perceived quality of documentation?
- **RQ 2.2:** How do practitioners rate the (perceived) quality of the existing documents versus the expected level of quality?

3.3 Hybrid Approach

An overview of the proposed approach is illustrated as a UML activity diagram in Figure 2. Our hybrid approach is composed of a mining phase which gathers and mines projects data and also a questionnaire-based survey which elicits experts' opinions. Based on results from benefit and quality analysis, a follow-up activity for trend analysis and generalization of the trends for a given project/company context will be conducted and finally the approach will suggest guidelines for documentation (process) improvement. Essentially, outputs from data mining and surveys will be compared with one another and merged to compare what developers say versus what actually was done. For example, a disagreement such as the following might be observed: a team of developers may say that they seldom refer to documents, but the access logs show that they actually read documents quite often.

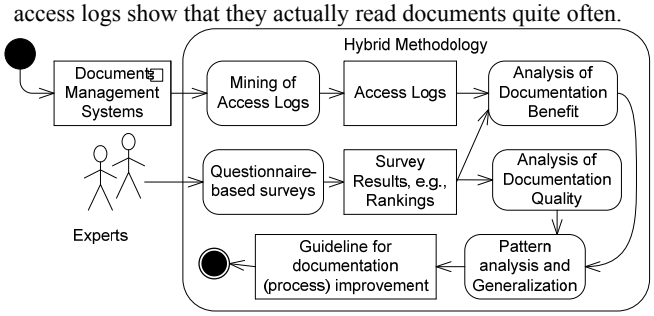

Figure 2. Proposed approach.

3.4 GQM-based Measurement

In order to ensure problem-focused measurement and analysis, we apply the Goal-Question-Metric (GQM) approach [22]. Different metrics have been used in our proposed approach. They are related to the different research components outlined in Figure 2.

For RQ 1, all the different impacting factors were measured in conjunction with a single metric to quantify the number of accesses to a given document in a given time interval (e.g., day).

Definition: Given a document *d*, we define *Document Access-Time Set (DATS)* as the set of time instances that the document has been accessed in the Documentation Management Systems (DMS) that the project under study uses.

Most DMS's (e.g., Wiki's, Microsoft SharePoint, or in-house systems) provide such data. For mining of access logs we were using an existing tool which was developed (as an in house system) in the company. We automatically extracted the Document Access-Time Set (DATS) of each document. Given a document identifier (i.e., number), the tool would list all the time instances that it has been accessed and the user name of the staff member accessing it.

In addition, a set of questionnaire questions with rank-based (i.e., Likert scale) quantitative responses and a final questions with a single descriptive feedback were the key measurement related to RQs 1 and 2. The details of the questionnaire are provided in the Appendix.

4. INDUSTRIAL CASE STUDY

This section describes the industrial case-study's context, design, execution, results, potential threats to its validity, and the steps that we have taken to mitigate them.

4.1 Case-Study Context

The company under study is a Calgary, Canada-based firm called NovAtel Inc., which is a leading provider of Global Navigation Satellite System (GNSS) products. Their OEM (Original Equipment Manufacturer) products heavily depend on the embedded software developed in-house. With strong quality requirements in terms of accuracy, reliability and performance, software development and evolution is a key success factor for the competitiveness and business success of products.

Recently, NovAtel has started the process of introducing lean development processes. To achieve this goal, there has been a need to carefully measure and analyze documentation cost, benefit and quality. The company has more than 20 years of legacy code and documentation. The company is using a comprehensive DMS which has been developed in-house. There are more than 10,000 active technical documents in the system. The company has a comprehensive product-family of GPS/GNSS receivers under the following major types: OEM6, OEMV and OEMStar. At any given time, the company designs and develops new products, and maintain ones. This study is conducted on one specific version of one of the above products. Due to confidentiality, we are unable to disclose identifying data about the system and documents throughout this paper.

Due to similarity of business logic among the products, to save costs, many software-engineering artifacts (e.g., design documents, source code, and test suites) are reused across many products in the product-family. Studying the reuse of documents is an interesting challenge and will be assessed in RQ 1.6.

4.2 Case-Study Design and Execution

For the survey component of our approach, we discuss the survey design next.

4.2.1 Survey Design

By adapting questions from similar studies in the literature, e.g., [5-7, 11, 12], and also adding several additional questions to ensure coverage of our RQs, we developed a questionnaire survey with 11 questions. We applied guidelines for survey design, e.g., [23]. We were also relying on our past experience in designing and executing industry-oriented surveys, e.g. [24]. We listed all the questions in Appendix.

4.2.2 Survey Execution

The online questionnaire which consists of 11 questions was sent to a group of 135 software engineers in the case-study company. Research ethics approval for the survey was obtained from the University of Calgary's Conjoint Faculties Research Ethics Board (CFREB) in April 2012 prior to survey execution. The selection of the subjects and their participation were done on a voluntary and anonymous basis. 25 of the 135 invited staff members responded to our survey, i.e., the response rate (sample ratio) was about 18.5%.

4.2.3 Choice of Statistical Tests

Based on the type of our hypotheses (H1…H5), as discussed in Section 3.4, and also the type of survey questions, we selected appropriate statistical tests [25]. For questions soliciting ranking of factors, they were based on ranked orders (e.g., between 1 and 4) on a Likert scale.

According to the empirical studies literature, non-parametric methods are widely used for studying populations that take on a ranked order [26] (such as movie reviews receiving one to four stars). We thus applied the *Wilcoxon* test (for two treatments) and the *Kruskal-Wallis* test (when having more than two treatments) [25]. These tests do not involve adding, subtracting, dividing or multiplying the numbers in the data and they only use the order of the numbers. They rank the *absolute values* of the difference scores, and can be applied on both ordinal and interval scales [27].

4.3 Analysis of the Case Study Results

The following sections present the results of the research questions that we explained in section 3.2.

4.3.1 RQ 1- Usage of Documentation

The goal of RQ 1 was to address the important attributes affecting usage of documentation. To formally evaluate the results of RQ 1.1-1.5 based on the survey results, we formulate five hypotheses which are presented in the following sections.

4.3.1.1 RQ 1.1 - Usage of Documentation in Comparison to Other Information Sources

Null Hypothesis H1 (for RQ 1.1): There is no significant difference between the usage of documentation versus other sources of information, e.g., source-code, personal communication with team members, and existing knowledge of the developer.

Two of the survey questions asked participants about the percentage (chance) that they consult each of the available resources during development and maintenance: documentation, source-code, communication with team members, and existing (self-) knowledge of developers. Figure 3 shows the box plot of the responses, grouped by development and maintenance.

During development, documentation is used on average in 37% of cases, while code is used with 39% chance. On the other hand, during maintenance, documentation is used on average in only 18% of cases, while code is used with 50% chance. These two sources are only slightly higher than the other two information sources. Thus, in our industrial setting, there is not a significant difference between the usage of documentation versus other sources of information. Since the distributions in Figure 3 have major overlap, there was no need to apply a statistical test to assess hypothesis H1, and it was clear that hypothesis H1 is rejected.

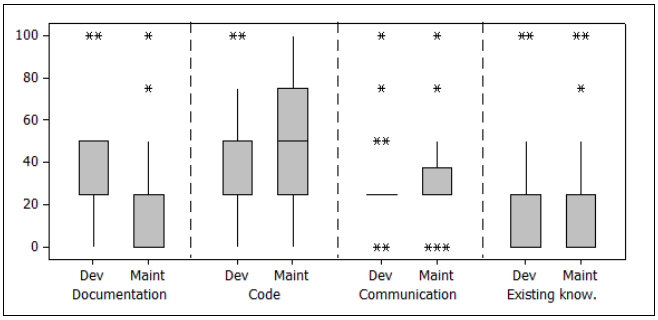

Figure 3. Percentage of usage of documentation (y axis) for development (Dev) vs. maintenance (Maint) in dependence of information sources.

4.3.1.2 RQ 1.2- Usage of Documentation in Development versus Maintenance

Null Hypothesis H2 (for RQ 1.2): There is no significant difference between the benefit of documentation during software development versus maintenance activities.

RQ 1.2 and hypothesis H2 aimed at assessing whether there is a significant difference between the usage of documentation during development versus maintenance. For this, we applied statistical testing to data shown in Figure 3.

We have two treatments: Usage of documentation in development and maintenance. Thus, the Wilcoxon test [28] was applied to evaluate the hypothesis H2. Table 2 presents the result for this test.

Table 2. Evaluating hypothesis H2 using the Wilcoxon test.

Treatments	W	p-value
Usage during development	312.5	0.002
Usage during maintenance		

Since the *p-value* of the Wilcoxon test is less than 0.05, it means that the hypothesis H2 is rejected. Thus, we conclude that there is a statistically significant difference between the usage of documentation in development and usage of documentation in maintenance phases, and it is higher in the latter (as per Figure 3).

There is difference in opinions, but majority of participants agree that there is a chance of 25 to 50 percent that they refer to source code and documentation for development purposes. However, two of them claimed that they will definitely consult source code and two of them mentioned they would refer to documentation.

The personal communication and their own knowledge seems to be the least referred resources. As shown in Figure 3, the source code is the first consulted and useful resource during maintenance activities. Interestingly, documentation is the least useful artifact for the maintenance purposes (e.g., fixing issues). They prefer to refer directly to the source code itself and they rely on source-code to support their information needs for maintenance tasks.

Based on aforementioned discussion, we found that the usage of documentation differs slightly between development and maintenance purposes. Our results are in line with the existing literature, e.g., [9, 10], in that source code is the most important artifact to study during maintenance.

4.3.1.3 RQ 1.3- Usage of Documentation in Dependence of the Document Type

Null Hypothesis H3 (for RQ 1.3): There is no significant difference between the usage of various documentation types

including requirement, design, code comments, test and process documents.

RQ 1.3 and H3 are about the usefulness of different types of documentation in the SDLC. Two of the survey questions were directed towards this RQ. Figure 4 illustrates the analyzed results as a box-plot and the difference in usage of various documentation artifacts.

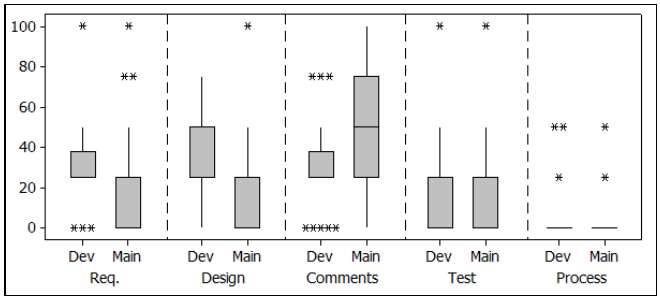

Figure 4. Percentage of usage of documentation (y axis) for development (*Dev*) vs. maintenance (*Maint*) in dependence of document type.

To study the differences carefully, we divided H3 into two sub-hypothesis:

- H3-1: There is a significant difference between the usage of various documentation types during development
- H3-2: There is a significant difference between the usage of various documentation types during maintenance

For both H3-1 and H3-2, there are five treatments which are the types of documentation: requirement/specification, design, code comments, test plan/procedures, and process documents. This time, since we have more than two independent treatments, we applied the Kruskal-Wallis test, and the results are shown in Table 3. In this table, "df" represents the degree of freedom in the statistical test.

Table 3. Evaluating hypothesis H3-1 and H3-2 using Kruskal-Wallis test.

Hypothesis	Chi-square	df	p-value
H3-1	33.37	4	1.003e-06
H3-2	37.40	4	1.486e-07

According to the Table 3, the *p-value* of the test shows that both H3-1 and H3-2 are accepted. Note that, as per using Kruskal-Wallis result interpretation, this does not mean that every group differs from every other group, only that at least one group differs from the others. By a close examination of the distributions in Figure 4, we can see that, during development, the distribution for test documents differs from the others. During maintenance, the distribution for comments differs from the others.

According to the results shown in Figure 4, the Usage of different types of documentation artifacts varies for both development and maintenance purposes. The most useful documentation artifacts during development phase is design documents, however code comments play the role of the most useful documentation artifacts for maintenance purposes. Interestingly, according to the results, design documents do not provide that much benefit during maintenance.

4.3.1.4 RQ 1.4- Usage of Documentation in Dependence of the Job Function

Null Hypothesis H4 (for RQ 1.4): The practitioners' role (position) does not have any significant impact on the amount of documentation they use.

RQ 1.4 aimed at assessing whether practitioner's roles (positions) have a significant impact on the level of documentation usage. To analyze this, we grouped the results by job function (position types) and the results are shown in Figure 5.

Job Function		Job Function	
Entry Developer	ED	Principle Architect	PA
Intermediate Developer	ID	Entry Tester	ET
Senior Developer	SD	Intermediate Tester	IT
Intermediate Architect	IA	Senior Tester	ST
Senior Architect	SA	Manager	M

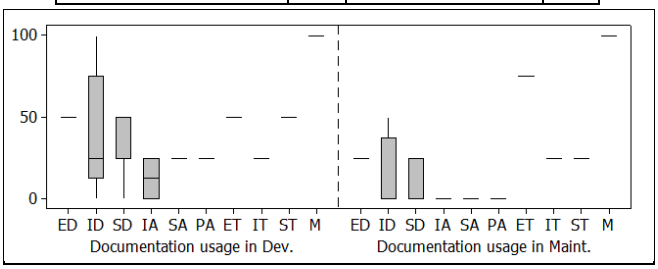

Figure 5. Percentage of usage of documentation (y axis) for development (*Dev*) vs. maintenance (*Maint*) by job functions.

As we can visually observe in Figure 5, since all distributions (except SD and IA) have major overlap, without applying statistical tests, we can say that job functions do not have a significant impact on the amount of documentation that practitioners use in our context in all cases, except the case of Senior Developers (SD) and Intermediate Architects (IA). Thus, H4 is rejected. However, we can see from the box plots that, to some extent, Intermediate Developers (ID) use more documentation compared to Senior Developers (SD), which is as expected.

4.3.1.5 RQ 1.5- Usage of Documentation in Dependence of the Degree of Experience

Null Hypothesis H5 (for RQ 1.5): The practitioners' experience does not have any significant impact on the amount of documentation they use.

We would expect that as a practitioner gets more senior, s/he would read less documentation. Figure 6 illustrates the ratio of consultation from various information sources during development and maintenance by years of experience, partitioned into three ranges: 1-5 years, 6-10 years and more than 10 years. In almost all of the eight clusters, more senior practitioners tend to refer less to documentation, code comments and the two other sources. However, since the three distributions in each of the eight clusters are partially overlapping, without applying statistical tests, we can conclude is no statistical significance for supporting the above trend.

As a follow-up, Figure 7 illustrates the ratio of consultation from different document types by years of experience during development and maintenance. As we can observe, design documents are mostly used by practitioners early in their careers (up to 5 years of experience). The same is true for code comments, process documents and also requirements documents during development.

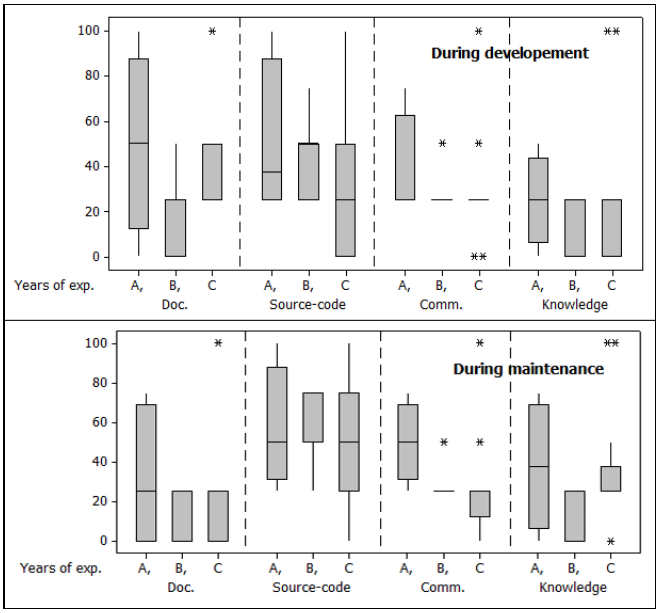

Figure 6. Percentage of usage (y axis) for development vs. maintenance in dependence information source and years of experience for 1 to 5 years (A), 6 to 10 years (B) and more than 10 years (C).

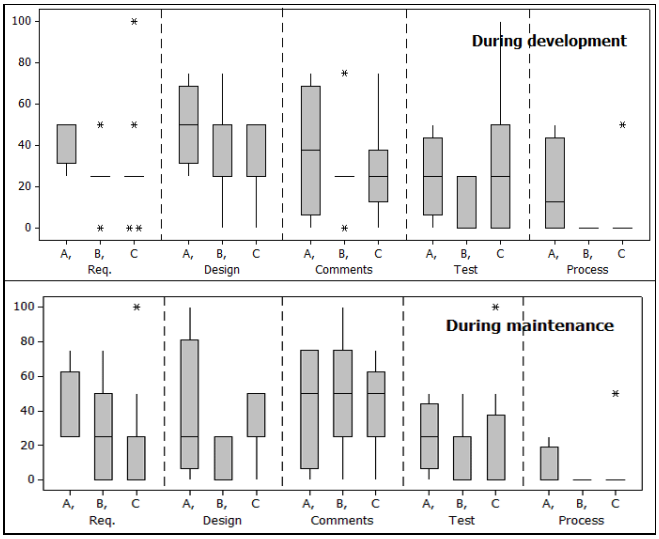

Figure 7. Percentage of usage (y axis) for development vs. maintenance in dependence of documentation type and years of experience for 1 to 5 years (A), 6 to 10 years (B) and more than 10 years (C).

Test documents, on the hand other, are accessed by all practitioners in the same magnitude quite independent of their years of experience. This is perhaps since everyone is doing some certain level of testing and they need to know the test procedure and test suites.

4.3.1.6 RQ 1.6- Patterns analyzed from Access Logs

Figure 8 shows the access log of design and test documents during the last five years (2007-2012) in the project. These data were automatically extracted using a specific tool we developed to gather the Document Access-Time Set (DATS) of each document. For the software product under study, there are 20 (detailed) design documents, where each document belongs to a software component, e.g., GNSS signal tracking module.

Due to confidentiality, the actual document names have been replaced with sequential numbers, e.g., 1…20. The X-axis is the timeline. The number of accesses for each document (i.e., the number of points in each horizontal line) is specified in the right side of the graph. Also, the major milestones of three products (inside one product family/generation) are overlaid on the graph. To analyze the visual information in Figure 8 and derive insights, we took an exploratory approach (similar to one of our previous studies [21]) and conducted root-cause analysis. The overlay of major milestone show that three consecutive products (called "x", "y" and "z") have been under development and maintenance in somewhat intertwined manner.

We can see that documents #6, 12 and 19 are the three most accessed documents. By a careful discussion with the development team, we realized that those three documents correspond to the three most important components in the system in terms of functionality, e.g., signal tracking module. Thus, it seems that, as one would expect, the more important a software component is, the more its documents would be expected to be accessed during the SDLC process.

The other observation is regarding the documents #15-20. It is interesting to see that they have been accessed starting from the maintenance phase of the product "y". We also looked at the creations date of each of these documents. It was clear that these additional design documents were created in the maintenance phase. We identified the root cause as follows: in the maintenance phase, the larger design documents were broken down into more modular pieces, thus creating several additional new documents. During design phases of products "y" and "z", earlier design documents have been reused to create new documents.

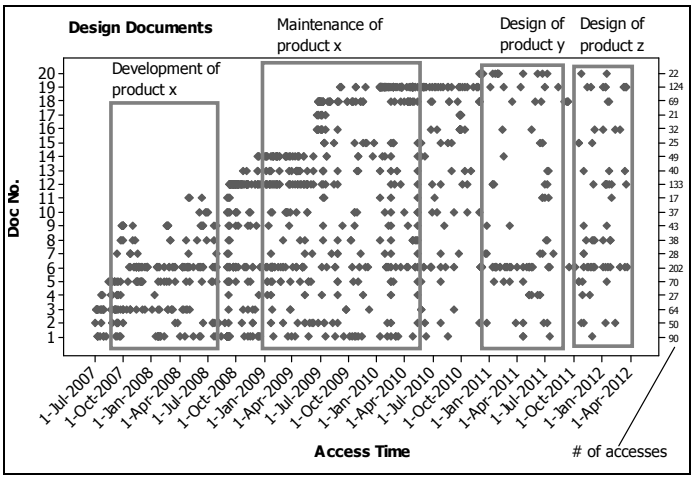

Figure 8. Access log of design documents during the last five years.

4.3.2 RQ 2- Quality of Documentation

The goal of RQ 2 was to address the important attributes affecting documentation quality in practice. To addresses this research question, the following sub-sections are presented.

4.3.2.1 RQ 2.1-Important Quality Attributes

RQ 2.1 aimed at measuring the impacts of each quality attribute (e.g., accuracy, completeness, consistency) in the overall quality of documentation, as perceived by the survey respondents.

Question #10 of the survey asked the participants to rate the impact of each of the following 10 attributes on the overall quality of a document: completeness, organization, including visual models,

relevance of content, preciseness, readability, accuracy, consistency, being up-to-date and use of examples.

For each attributes, the participants rated between (1) very low, (2) low, (3) high, and (4) very high. The results are summarized in Figure 9. When the responses were encoded using the above numbering scheme, the average values were mostly in the range of 2-3 (low to high). Attributes: readability, relevance of content, and organization were rated the top three.

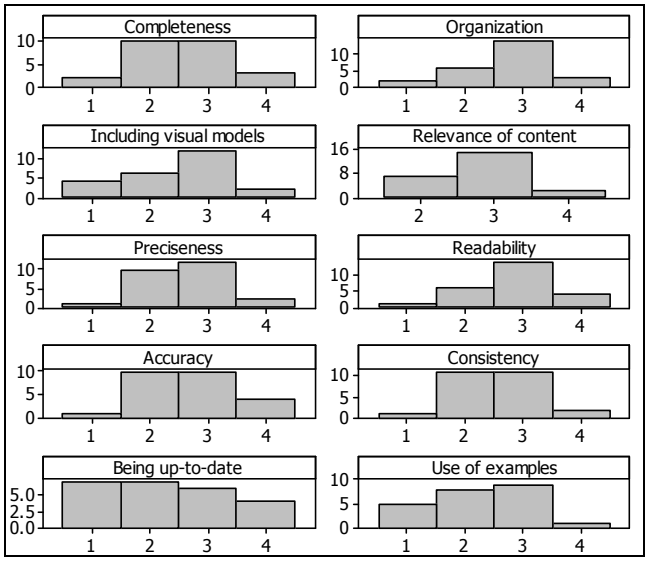

Figure 9. Impact of individual quality attributes on perceived total quality of documentation.

4.3.2.2 RQ 2.2- Perceived versus Expected Quality

Next, respondents ranked their perceived quality of the existing documents versus the expected level of quality that they would like to see. The results are shown as a scatter plot in Figure 10. The x- and y-axis denote the perceived and expected quality of the existing documents in the company, on a percentage ratio. Each point is the average of the 25 responses received. For brevity, only the data ranges where the points are located are shown. Also, a diameter line with 45 degree angle is shown.

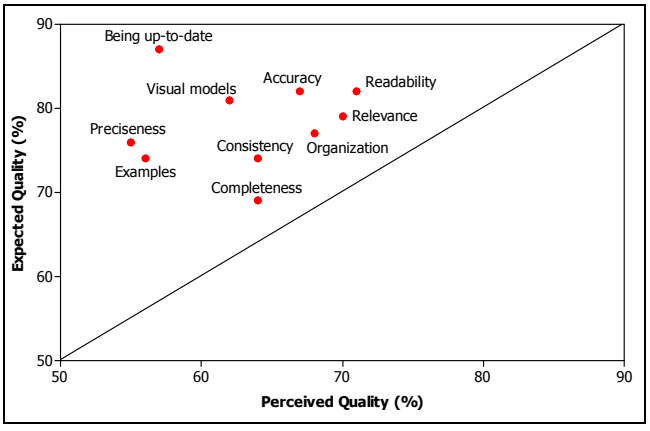

Figure 10. Perceived versus expected levels for quality aspects of documentation.

It is surprising to see that, for all 10 quality attributes, perceived quality is less than the expected quality. The most critical (lacking) quality attributes are: up-to-date-ness, preciseness, and use of examples. Organization relevance and readability have been rated decent, and their perceived quality levels are quite close to their expected quality levels.

4.4 Guidelines for Improvement of Documentation Process and its Efficiency

In this section, we adapt ideas from the literature on evidence- and measurement-based Software Process Improvement (SPI), e.g., [29] for improvement of documentation process and its efficiency based on the findings and the analyses that we have conducted in previous sections. Note that the current study intends to only present our initial ideas in this direction and more work is yet to be done in future. Furthermore, since our entire project is based on action-research (AR), as discussed in Section 3.1, close interactions between the researchers and practitioners are always our top priority. We are reviewing carefully the literature on how to ensure the success of SPI, e.g., [30, 31], to ensure that our documentation improvement and optimization approach is as successful as possible.

4.4.1 Identification of Improvement Opportunities

Based on findings and the analyses presented in the previous sections, in the context of our case study, we have proposed the following initial improvement opportunities for documentation process. In each case, the explicit traces to the specific RQ(s) and the corresponding results which have triggered each guideline have been mentioned.

1. Based on output from RQ 1.1 and 1.2, we found that using documentation was more in development compared to maintenance. However, code comments were reviewed more in maintenance compared to documentation. This finding encourages the idea of "opportunistic" and "situational" documentation [32], i.e., creating documents which are only going to be used during development and then expecting developers to embed, in the code, useful and precise code comments for the usage during maintenance. Also, to prevent overhead and efficient usage of time, clear guidelines on usage of each of the four types of information sources should be implemented (i.e., documentation, source-code, communication with team members, and existing knowledge of developers).
2. Output from RQ 1.3 measured the usage of each type of documentations: requirement/specification, design, code comments, test plan/procedures, and process documents. Again, we found that during maintenance, earlier documentations types (requirement and design) are used less than code comments. Thus again, we see "opportunistic" and "situational" documentation [32] as a promising improvement opportunity.
3. Output from RQ 1.4 confirmed that obviously people in different roles refer to different types of documents on a "need basis". Thus, each type of documents should be tailored to the style/need of its intended users (readers) and it is not a good idea to develop documents following the "one size fits all" style.
4. Results of RQ 1.5 confirmed that more senior practitioners tend to refer less to documentation. Thus, when preparing documents, the specific information needs of young practitioners and new comers should be considered.
5. Based on the exploratory study in RQ 1.6, the team should continue to identify the least accessed documents and compare that information to the amount of effort (cost) spent to develop and maintain those documents. If the usage does not justify the costs, the document may be decided to be retired (archived).

6. Responses to RQ 2.1 identified the important quality attributes that would impact the quality of a document, the top three of which were: readability, relevance of content, and organization. The improvement opportunities for increasing quality of document should thus focus on those attributes.
7. Last, but not least, RQ 2.2 identified the perceived versus expected standing of each quality attribute. That RQ also identified important improvement opportunities in that the management should tailor the documentation process towards aligning expected and the actual perceived levels of each quality attribute. The most critical (lacking) quality attributes are: up-to-date-ness, preciseness, and use of examples.

4.4.2 Assessing Potential Solutions

Once deficiencies and improvement opportunities for the documentation process in a given organization have been clearly identified, the search for sound and economically viable solutions starts. New technologies and methods should be introduced with care in an organization. Most likely, a new technology will need to be adapted and will have a transition impact until the learning phase is completed [29].

Different types of empirical investigations for transitional introduction of improvements may be used, e.g., pilot projects and controlled experiments [29]. We are planning to conduct pilot projects in the context of small teams in the company and apply each of the above seven improvement opportunities and assess the benefits of each solution. This will complete the steps (i.e., feedback loop) of our AR-based improving case-study (Section 3.1).

4.5 Discussions and Benefit of the Approach

Based on our action-research (AR) approach, we summarize the benefits of our hybrid approach in the context of our case study. Number of AR cycles is expected to be five, and we have had two cycles so far in our project. In each AR cycle, we are conducting technology transfer and dissemination of results from academia to industry and get intense feedback on the progress of the project.

The hybrid approach and its outputs (results of RQ 1 and 2, and also the improvement opportunities) have shown to be useful for the practitioners in the company. We have started to assess the potential improvements opportunities, discussed above. The overhead of the hybrid approach was minimal in that the mining of access log was automated and the survey had brief questions which tool about 20 minutes to complete.

The proposed approach can be utilized and applied to any software development company. The proposed suggestions and guidelines are expected to be useful in improving documentation process. Using a similar approach, practitioners will be able to notice which parts of their documentation is useful and which parts need to be improved to be useful during software development and maintenance phases. In addition, they will be able to compare the existing documentation quality attributes with the expected levels.

Note that due to space constraints, only a selected subset of our study's results is presented in this paper. For further details, the reader is referred to a recent thesis dedicated for this study [33].

4.6 Threats to Validity

This section discusses potential threats to the validity of our study, the steps that we have taken to minimize or mitigate them, the strengths and limitations of this study. The following validity aspects are applicable in our study: construct, external, and (statistical) conclusion validity.

4.6.1 Construct Validity

The construct validity issue in this case study is related to what extent the selected documentation usage and quality metrics really represent what we intended to measure. The access log data were automatically mined from a documentation management system, and all the qualitative/quantitative survey data were captured from a limited number of the software professionals in the company, the metrics may not perfectly capture all the factors to be investigated. We attempted to mitigate this threat by involving different level of experienced software professionals with various roles and positions.

Another threat to construct validity is the measurement of documentation quality attributes. Similar to general concept of software quality, documentation quality is a multi-facet phenomenon and should be measured by all of its related metrics, such as readability, consistency, accuracy, up-to-date etc. We addressed this potential threat to the validity by incorporating 11 quality attributes which were taken from the literature, e.g., [5-7, 11-13].

4.6.2 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: (1) representativeness of the case-study company/systems, (2) the number of documentation artifacts under study, and (3) the number of practitioners who participated in the survey. This case study was conducted on a large company which develops embedded software systems. According to our experience in research collaborations with many other companies, the development and documentation processes and practices in this firm are quite similar to other companies. The software product-family and systems under study were legacy embedded software systems which were non-trivial large-scale industrial systems. The access log study was done on a selected list of 20 design documents and test documents. The survey received input from 25 practitioners who have been working with 10,000 documents across the company.

Although our industry setting was a typical setting which would be quite typical across many companies, similar to other empirical studies, it is not easy to generalize the outcomes from this study to other industrial contexts. More companies and more software systems should be examined in future studies in order to determine the replicability of the findings in this context.

4.6.3 Conclusion Validity

Conclusion validity denotes whether the treatments/changes that we introduced have a statistically significant effect on the outcomes that we measured. Several of our hypotheses could be accepted with statistically significant outcomes from the statistical tests. However, in few cases, the hypotheses were rejected, because the treatments did not result in outcomes with statistically significant differences. For example, we observed in RQ 1.2 that, in our industrial setting, there is not a significant difference between the usage of documentation versus other sources of information. This observation does not denote a problem with the empirical design, but rather the usual reality in term of the above issue (the so called "face validity" [25]), i.e., some practitioner prefer to use documentation while others use other sources of information.

5. CONCLUSIONS AND FUTURE WORK

We presented a practical approach to evaluate documentation quality and usage from different perspectives and identified the

relevant impacting factors towards documentation efficiency improvement. The proposed hybrid approach consists of both quantitative and qualitative evaluations to assess software documentation and quality.

Within the approach, we formulated two hypotheses to determine the usage of documentation artifacts in practice. Then we validated and assessed the hypotheses by conducting surveys of software professionals. The proposed approach was developed in an action-research-based project and was evaluated using an industrial case study, conducted in the context of a large corporation developing GPS/GNSS embedded software systems.

As the main results of our study, we found that the usage of documentation differs for various purposes. For example, the usage of documentation for implementation purposes is more than its usage for maintenance. Also, when we consider different types of documentation artifacts, usage is not same in development and maintenance activities. For instance, software design (architectural) artifacts are useful for implementation; however, code-comments artifacts are beneficial for software maintenance purposes.

Furthermore, we identified the most important and relevant quality attributes which are critical to improving documentation efficiency. According to our results, the most important quality factors affecting overall documentation quality include documentation artifacts up-to-date-ness, accuracy and completeness. In terms of the expected versus perceived usage, the biggest gaps were related to up-to-date-ness and preciseness.

The results of our work will be combined with the results extracted from the cost measurement of documentation which was done by another member of our team [34]. Then both results will be merged for a follow-up cost-benefit study and analysis. At the end, we are aiming at optimizing processes towards documentation cost-benefit.

In this paper, documentation artifacts were found to be lacking in the various quality attributes such as up-to-date-ness, accuracy, completeness, use of visual models and examples. It is important to solve these shortcomings, investigate more the cause-effect relationship between these quality attributes and usage factors.

In addition, the survey could be replicated, preferably with more subjects that use more sophisticated documentation artifacts. We plan to investigate the root-cause analysis of results including:

1. Why is the usage of documentation in development less than that in maintenance?
2. Why are design documents more beneficial for development purpose, and code-comments are useful for maintenance purposes?

The study has been done for embedded type of software. The question “how much documentation in enough?” and “optimization of documentation process” are among our future plans, but need to be studied also outside the embedded software domain. There are plans by the authors to increase the automation of data mining for measuring and benefit of documents to provide more comprehensive data for future follow-up studies.

Acknowledgements

This work was supported by the NSERC CRD grant #CRDPJ414157-11, and NSERC ENGAGE grant #EGP-413039. Vahid Garousi was also additionally supported by the Visiting Scientist Fellowship Program (#2221) of the Scientific and Technological Research Council of Turkey (TÜBİTAK). We would like to sincerely thank all the software engineers in NovAtel who participated in the survey and for their support so far during this action-research project.

References

[1] D. L. Parnas, "Precise Documentation: The Key To Better Software," in *The Future of Software Engineering*, S. Nanz, Ed.: Springer, 2011.

[2] S. Cozzetti, B. d. Souza, N. Anquetil, and K. M. d. Oliveira, "Which documentation for software maintenance?," *Journal of the Brazilian Computer Society,* vol. 12, 2006.

[3] L. C. Briand., "Software documentation: How much is enough?," in *Proceedings of the European Conference on Software Maintenance and Reengineering*, 2003, pp. 13–17.

[4] P. S. M. d. Santos and G. H. Travassos, "Action research use in software engineering: An initial survey," in *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, 2009

[5] A. Forward, "Software Documentation: Building and Maintaining Artefacts of Communication," University of Ottawa, MSc Thesis, 2002.

[6] M. Visconti and C. R. Cook, "Assessing the State of Software Documentation Practices," in *Conf. on Product Focused Software Process Improvement*, 2004, pp. 485-496.

[7] 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, 2006

[8] A. Von Mayrhauser, "Program comprehension during software maintenance and evolution," *IEEE Computer,* vol. 28, pp. 44-55, 1995.

[9] S. C. B. d. Souza, N. Anquetil, and K. M. d. Oliveira, "A study of the documentation essential to software maintenance," in *Proceedings of the annual international conference on Design of communication*, 2005.

[10] E. Tryggeseth, "Report from an Experiment: Impact of Documentation on Maintenance," *Empirical Software Engineering,* pp. 201-207, 1997.

[11] M. Kajko-Mattsson, "A Survey of Documentation Practice within Corrective Maintenance," *Empirical Software Engineering,* vol. 10, pp. 31-55, 2005.

[12] 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.

[13] A. Wingkvist, M. Ericsson, R. Lincke, and W. Lowe, "A Metrics-Based Approach to Technical Documentation Quality," in *Proceedings of the International Conference on the Quality of Information and Communications Technology*, 2010, pp. 476 - 481

[14] 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, Under Review,* 2012.

[15] J. Zhi, V. Garousi, B. Sun, G. Garousi, S. Shahnewaz, and G. Ruhe, "Online paper repository for: Cost, Benefits and Quality of Technical Software Documentation: A Systematic Mapping " http://www.softqual.ucalgary.ca/projects/SM/doc_cost_benefit_usage_quality, Last accessed: Nov. 2012.

[16] B. G. Freimut, L. C. Briand, and F. Vollei, "Determining Inspection Cost-Effectiveness by Combining Project Data and Expert Opinion," *IEEE Trans. Software Eng.,* vol. 31, pp. 1074-1092, 2005.

[17] 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.
[18] F. Shull, J. Singer, and D. I. K. Sjøberg, *Guide to Advanced Empirical Software Engineering*: Springer, 2007.
[19] D. E. Avison, F. Lau, M. D. Myers, and P. A. Nielsen, "Action Research," *Communications of the ACM,* vol. 42, pp. 94-97, 1999.
[20] D. Avison, R. Baskerville, and M. Myers, "Controlling action research projects," *Information Technology & People,* vol. 14, pp. 28-45, 2001.
[21] V. Garousi, "Evidence-based Insights about Issue Management Processes: An Exploratory Study," in *Proceedings of the International Conference on Software Process (ICSP)*, 2009, pp. 112-123.
[22] V. Basili, G. Caldiera , H. D Rombach , and R. v. Solingen, *The Goal Question Metric Approach (Encyclopedia of Software Engineering 1)*: John Wiley and Sons, 2001.
[23] O. L. Marcus Ciolkowski, Sira Vegas, Stefan Biffl, "Practical Experiences in the Design and Conduct of Surveys in Empirical Software Engineering," in *Empirical Methods and Studies in Software Engineering*, 2003, pp. 104-128.
[24] V. Garousi and T. Varma, "A Replicated Survey of Software Testing Practices in the Canadian Province of Alberta: What has Changed from 2004 to 2009?," *Journal of Systems and Software,* vol. 83, pp. 2251-2262, 2010.
[25] F. Shull, J. Singer, and D. I. K. Sjoberg, *Guide To Advanced Empirical Software Engineering*: Springer, 2008.
[26] L. Wasserman, *All of nonparametric statistics*: Springer, 2007.
[27] E. McCrum-Gardner, "Which is the correct statistical test to use?," *British Journal of Oral and Maxillofacial Surgery,* pp. 38–41, 2007.
[28] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin,* pp. 80–83, 1945.
[29] L. C. Briand, C. M. Differding, and H. D. Rombach, "Practical guidelines for measurement-based process improvement," *Software Process Improvement and Practice Journal,* vol. 2, 1997.
[30] M. Umarji and C. Seaman, "Predicting acceptance of Software Process Improvement," *ACM SIGSOFT Software Engineering Notes,* vol. 30, pp. 1-6, 2005
[31] A. Rainer and T. Hall, "Key success factors for implementing software process improvement: a maturity-based analysis," *Journal of Systems and Software,* vol. 62, pp. 71–84, 2002.
[32] D. J. Gilmore, R. L. Winder, and F. Detienne, *User-Centred Requirements for Software Engineering Environments*: Springer, 1994.
[33] G. Garousi, "A Hybrid Methodology for Analyzing Software Documentation Quality and Usage," *MSc Thesis, University of Calgary,* Oct.. 2012.
[34] B. Sun, "A Methodology for Analyzing Cost and Cost-Drivers of Technical Software Documentation," *MSc Thesis, University of Calgary,* 2012.

Appendix

1. How long have you been working in the software industry? Please select one of the following options:

- < 1 year
- 1 - 5 years
- 6-10 years
- 10 years

2. What is your current job function?

- Manager
- Entry Designer (Architect)
- Intermediate Designer (Architect)
- Senior Designer (Architect)
- Principle Designer (Architect)
- Entry Developer
- Intermediate Developer
- Senior Developer
- Principle Developer
- Entry Tester / QA
- Intermediate Tester / QA
- Senior Tester / QA
- Principle Tester / QA
- Other. Please specify:

3. Based on your experience, during the development phase (not sustaining), what is the percentage (chance) that you consult each of the following resources to complete the task? (Please make sure the sum adds up to 100%)

	0	**25**	**50**	**75**	**100**
Documentation					
Source-code itself					
Personal communication					
Existing knowledge (no need to search for information)					
Other. Please specify					

4. As a follow-up to the above question, during the development phase (not sustaining), in case when you refer to documentation, how often do you refer to each type of documentation? (Please make sure the sum adds up to 100%)

	0	**25**	**50**	**75**	**100**
Requirements and specifications documents					
Design and detailed design documents					
Code comments					
Test documents					
Process-related ("QAx") documents					
Other. Please specify					

5. Based on your experience, to perform a given software sustaining task (fixing a defect (Maintenance) or adding a feature (Enhancement)), what is the percentage (chance) that you consult each of the following resources to complete the task? (Please make sure the sum adds up to 100%)

	0	25	50	75	100
Documentation					
Source-code itself					
Personal communication					
Existing knowledge (no need to search for information)					
Other. Please specify					

6. As a follow-up to the above question, to perform a given software maintenance task (sustaining), in case when you refer to documentation, how often do you refer to each type of documentation? (Please make sure the sum adds up to 100%)

	0	25	50	75	100
Requirements and specifications documents					
Design and detailed design documents					
Code comments					
Test documents					
Process-related ("QAx") documents					
Other. Please specify					

7. Regardless of how many times you refer to documentation, how useful do you find the existing documentations? (1 as "very low" and 4 as "very high")

	1	2	3	4
For learning the existing software code-base (for newly-hired staff)				
For the development phase				
For the testing/QA phase				
For the maintenance phase				

8. How useful have you found each of the following documentation styles? (1 as "very low" and 4 as "very high")

	1	2	3	4
Textual descriptions				
Visual diagrams (such as Visio diagrams)				

9. Please rate the quality of the documents you refer to, based on the following aspects: (1 as "very low" and 4 as "very high")

	1	2	3	4
Organization (table of contents, sections, sub-sections, etc.)				
Including visual models, if applicable (such as Visio diagrams)				
Relevance of Content (document's internal information)				
Documentation preciseness				
Documentation readability				
Documentation completeness				
Documentation accuracy				
Documentation consistency				
Being up-to-date				
Use of examples in document				

10. Independent of the above question, how important do you think each of the following items are to increase the quality of a document. (1 as "very low" and 4 as "very high")

	1	2	3	4
Organization (table of contents, sections, sub-sections, etc.)				
Including visual models, if applicable (such as Visio diagrams)				
Relevance of Content (document's internal information)				
Documentation preciseness				
Documentation readability				
Documentation completeness				
Documentation accuracy				
Documentation consistency				
Being up-to-date				
Use of examples in document				

11. If you would like to make any descriptive feedback on the benefit and quality of documentation in your projects, please discuss it below.