Information and Software Technology 57 (2015) 664–682

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Usage and usefulness of technical software documentation: An industrial case study

Golara Garousi [a,b], Vahid Garousi-Yusifoğlu [c,b,*], Guenther Ruhe [b,d], Junji Zhi [e], Mahmoud Moussavi [b], Brian Smith [f]

[a] geoLOGIC Systems Ltd., Calgary, Alberta, Canada
[b] Department of Electrical and Computer Engineering, Schulich School of Engineering, University of Calgary, Alberta, Canada
[c] System and Software Quality Engineering Research Group (SySoQual), Department of Software Engineering, Atilim University, İncek, Ankara, Turkey
[d] Department of Computer Science, University of Calgary, Calgary, Alberta, Canada
[e] Department of Computer Science, University of Toronto, Toronto, Ontario, Canada
[f] NovAtel Inc., Calgary, Alberta, Canada

ARTICLE INFO

Article history:
Received 19 July 2013
Received in revised form 1 August 2014
Accepted 5 August 2014
Available online 29 August 2014

Keywords:
Technical software documentation
Usage
Usefulness
Industrial context
Case study

ABSTRACT

Context: Software documentation is an integral part of any software development process. However, software practitioners are often concerned about the value, degree of usage and usefulness of documentation during development and maintenance.
Objective: Motivated by the needs of NovAtel Inc. (NovAtel), a world-leading company developing software systems in support of global navigation satellite systems, and based on the results of a former systematic mapping study, we aimed at better understanding of the usage and the usefulness of various technical documents during software development and maintenance.
Method: We utilized the results of a former systematic mapping study and performed an industrial case study at NovAtel. From the joint definition of the analysis goals, the research method incorporates qualitative and quantitative analysis of 55 documents (design, test and process related) and 1630 of their revisions. In addition, we conducted a survey on the usage and usefulness of documents. A total of 25 staff members from the industrial partner, all having a medium to high level of experience, participated in the survey.
Results: In the context of the case study, a number of findings were derived. They include that (1) technical documentation was consulted least frequently for maintenance purpose and most frequently as an information source for development, (2) source code was considered most frequently as the preferred information source during software maintenance, (3) there is no significant difference between the usage of various documentation types during both development and maintenance, and (4) initial hypotheses stating that up-to-date information, accuracy and preciseness have the highest impact on usefulness of technical documentation.
Conclusions: 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 tasks to be completed (e.g., development and maintenance). The results have been confirmed to be helpful for the company under study, and the firm is currently implementing some of the recommendations given.

1. Introduction and motivation

Software documentation is an integral part of the software development and maintenance process. Parnas [1] defines documentation as "... written description that has an official status or authority and may be used as evidence". Such descriptions are expected to provide precise information about the system. In general, software documentation is classified as being technical documentation or non-technical documentation (e.g., user manuals). In this study, our focus is on technical documentation.

Development of technical software documentation is an important practice that is believed to improve development and maintenance activities [2]. It includes all types of documentation related to the different artifacts created during the software life-cycle. Technical software documentation investigated in this paper

* Corresponding author at: Department of Software Engineering, Atilim University, İncek, Ankara, Turkey.

http://dx.doi.org/10.1016/j.infsof.2014.08.003
0950-5849/

include: requirement specifications, design (architectural) documents, comments made on source code, test documents and process descriptions. All of these documents are supposed to help software engineers to comprehend a given system and perform their tasks more effectively and efficiently.

Making decisions about the amount and technical depth of documentation remains a major challenge for many practitioners. Many teams either develop too much or too little documentation. Answering the question "How much documentation is enough?", even after the many years software documentation has been practiced, is still difficult [1,3]. Up to now, there is no end to the stories of legacy software systems lacking documentation or having low-quality documentation. In fact, documentation is often incomplete, inconsistent and out of date in practice [1]. One additional concern with documentation is that it is usually perceived as an expensive activity, sometimes not useful and difficult to maintain in many projects [1].

The results of an industrial case study with NovAtel, a world-leading company of GNSS software systems, are presented in this paper. The focus of the investigation was on the usage and usefulness of technical documentation. In total, 1630 revisions of 55 documents generated in the context of embedded software development and maintenance were studied. The main contributions of this paper are:

- An empirical approach including both quantitative and qualitative evaluations of documents and their revisions to assess technical software documentation usage and usefulness.
- Industrial case study evaluation and confirmative analysis of the usage of documentation and its relationship with other factors such as document type, job function, degree of experience and readability.
- Exploratory analysis of the usefulness of documentation.
- Guidelines for improvement of the documentation and development process based on the analyzed case study results.

This paper extends our former work [4] in both breadth and depth with respect to all the four contribution areas listed above. Compared to the former paper, we have added two more research questions (RQ) related to usefulness of technical documentation and have provided a more precise description of the applied research methodology. We have also performed a more detailed analysis related to some of already existing RQ's (e.g., by application of statistical tests). The discussion of the related work was updated to cover our most recent research results.

The remainder of the paper is organized as follows: Section 2 discusses related work. Section 3 outlines the goal of this research and the proposed research methodology. Section 4 presents the context of the case study and a description of its execution. Results are presented in Section 4.

Discussions and interpretation of findings are presented in Section 5. Section 6 gives recommendations for improvement of documentation process in the industrial context. Section 7 discusses the applicability of the approach and results. Section 8 discusses potential threats to validity. Finally, Section 9 concludes the paper and provides an outlook to future research.

2. Background and related work

We discuss the related work in this section under the following categories.

- Technical documentation
- Usage of technical documentation
- Usefulness of technical documentation

2.1. Technical documentation

To provide a systematic view of the domain of software documentation, we recently conducted a systematic mapping (SM) [4,5] of 69 studies in the areas of documentation cost, usage, usefulness, and quality. The content of our SM study and its underlying classification can be accessed online at [6]. A classification of the main empirical studies is given in Table 1.

In our SM study [5], we summarized the existing work related to the definition of documentation (as a type of artifact). Excerpts from that study are the followings: (1) In terms of focus of papers, 50 papers focused on documentation quality, followed by 37 papers on benefit, and 12 papers on documentation cost, and (2). The quality attributes of documentation that appear in most papers are, in order: completeness, consistency and accessibility. In terms of conclusions, the study [5] showed that the research area of documentation cost, usage, usefulness, and quality is emerging but far from being mature yet. We found that the documentation cost aspect seems to have been neglected in the existing literature and there are no systematic methods or models to measure cost. Also, despite a substantial number of solutions proposed during the last 40 years, more and stronger empirical evidence is still needed to enhance our understanding of this area. In particular, to increase the field's maturity and to fill the gaps, the following types of studies are needed: (1) More validation or evaluation studies; (2) Studies involving large-scale development projects, or including a large number of study participants from various organizations; (3) More industry–academia collaborations; and (4) More estimation models or methods to assess documentation quality, benefit and, especially cost.

In a recent work, Ding et al. [7] conducted a systematic literature review to explore how knowledge-based approaches are employed in software documentation. They investigated quality of software documentation, and the costs and benefits of using knowledge-based approaches. The study classified the knowledge-based approaches into several categories. The authors also concluded that documentation cost mainly includes: document creation, document maintenance and evolution, information retrieval from documents, and finally document distribution.

2.2. Usage of technical documentation

Documentation serves various purposes during the different activities of the software development life cycle [2]. Generally, software professionals rely on internal documentation (e.g., comments embedded in source code) as an aid in system understanding and program comprehension.

During system design activities, design documents (e.g., those including UML models) are prepared to account for the design goals and the software architecture. In this stage, graphical documentation helps to visualize the system architecture. It also helps visualize how classes are interconnected and how they interact [18,32,34].

During software development (i.e. implementation) phase, textual as well as graphical documentation and also design models aid developers in development and also testing activities [35]. UML diagrams such as class and state diagrams aid unit testing and interaction diagrams aid functional testing and regression testing.

During software maintenance, documentation can help in understanding the existing software system (e.g. architecture comprehension and code comprehension) in order to maintain the systems [8,15]. On the other hand, documenting the decisions during design, development and maintenance phases can help newcomers to the project understand the reasons for the decisions and also the solutions. Documentation may also help during

Table 1
A summary of surveys and experiments on the subject of documentation usage, usefulness and benefit.

References	Survey (s)/experiment (e)	# of organizations	# of participants	Type of questions
[9]	(e)	6	47	– Cognitive perspective
[28]	(e)	Not reported	20	– Time saving with usage of UML to change code
[20]	(s)	2	76	– Importance degree of documentation to help system understanding – Percentage of actual usage of documents
[10]	(s)	Not reported	81	– Perceived importance of design rationale
[21]	(s)	18	Not reported	– Ease to understand system situational documentation
[11]	(e)	3	Not reported	– Documentation as an aid to: – Comprehend – Code – Debug – Modify
[12]	(s)	Not reported	150	– Perceived importance
[27]	(e)	Not reported	12	– Documentation as a maintenance and development aid
[32]	(e)	Not reported	34	– Documentation as a maintenance aid
[13]	(s)	2	16	– Documentation as a maintenance aid – Perceived importance of documentation
[16]	(e)	2	Not reported	– Documentation as a maintenance aid – Time saved by using UML
[2]	(s)	Not reported	76	– Perceived importance of documentation – Documentation as a maintenance aid

impact analysis and identifying the location in the code where a change must be performed.

Generally, software engineers rely on technical documentation as an aid in system understanding and program comprehension. Without documentation, the only reliable source of information is source code. As a consequence, it often takes a lot of time and effort to explore the source-code to find the relevant information and gain an understanding of the system functionality [27].

Existing literature has shown that domain-specific knowledge is a critical factor in reading-comprehension [14]. In this context, the information stored in developers' minds (memory) is formed as domain-specific knowledge and has critical impact on understanding the documents. As personal memory is transient and may fade, the information stored in it may have the risk of being lost. For larger teams, sharing personal knowledge with all stakeholders (i.e., developers) is nearly impossible. Creating documentation to archive and disseminate this knowledge is usually the solution to the problem in this case [2,15,16].

Recent research has investigated the impacts of human factors on software documentation [18]. Forward and Lethbridge [19] surveyed software professionals to uncover their perception of software documentation, and the tools and technologies used to maintain, verify and validate documents. In another empirical study, de Souza et al. [20] surveyed software maintainers to reveal the type of documentation artifacts that are the most useful. These studies all focus on the perceptions of practitioners. In this paper, the human aspects that may have impacts on the usage of documentation include not only the perception of practitioners, but also other factors such as job function and degree of experience.

Document quality may have an impact on documentation usage. During the Software Development Life-cycle (SDLC), the quality of documentation artifacts developed in earlier up-stream phases 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 [17], completeness, consistency, correctness, information organization, format, readability, spelling and grammar, traceability, trustworthiness, and up-to-datedness [15,21]. In this paper, we investigate how the documentation type and readability affects its usage.

In a recent article, Falessi et al. [22] analyze the usage and value of a specific type of documentation, i.e., Design Rationale Documentation (DRD). The authors argue that a complete and detailed DRD could support many software development activities, such as an impact analysis or a major redesign. However, this is typically too onerous for systematic industrial use as it is not cost effective to write, maintain, or read. The key idea investigated in that article was that DRD should be developed only to the extent required to support activities particularly difficult to execute or in need of significant improvement in a particular context. The aim of the article was to empirically investigate the customization of the DRD by documenting only the information items that will probably be required for executing an activity. This customization strategy relied on the hypothesis that the value of a specific DRD information item depends on its category (e.g., assumptions, related requirements, etc.) and on the activity it is meant to support. Overall, our study focuses on usage of documentation artifacts in two stages:

- Usage during the development cycle: during requirement, design and coding phases
- Usage after the development (during maintenance): this stage investigates bug fixing (corrective maintenance) and feature addition (additive maintenance) activities.

Documentation usage during the software maintenance phase is mostly concentrated on system understanding (program comprehension) purposes. Once a document has been produced and approved, it is ready to use for different purposes during the software development life-cycle. Generally, usage of documentation depends on the type of "information needs" for each individual who is using the documentation.

2.3. Usefulness of technical documentation

The usefulness of the documentation artifact is described as: "A measure of how well a document's content can provide knowledge, information and/or insight to its reader with respect to other available artifacts considering both the activity and value of the artifact [29]".

There are only a handful of studies conducted on documentation cost, which formed one of NovAtel's main motivations for the whole research project. In an empirical study conducted by Perry et al. [23], the researchers observed that one of the developers spent around 60 min writing documents per day. Their study addresses the questions on how much daily effort is consumed by documentation. The authors targeted the broad range of software production processes in which documentation is a small part. They only present the number of minutes spent on documentation and did not answer the question of how and why these costs are spent or whether the time has been worthy of spending. Sun [25] analyzed 55 documentation artifacts and measured documentation efforts and costs. Eight main cost-drivers were identified and assessed. Sanchez-Rosado et al. [24] also assessed the documentation development effort in software projects.

Generally, from the perspective of Value-based Software Engineering (VBSE) [26], when an artifact consumes cost or effort, the artifact should yield benefits at some point later in the development or maintenance phase [21,27]. To quantitatively or qualitatively measure benefits of documentation, several works have been reported in the literature, which are mostly based on questionnaire-based surveys. de Souza et al. [20] presented the results of a survey of 76 software maintainers in Brazil and determined the documentation artifacts which were the most useful to them. The survey confirmed that source code and comments are the most important types of documentation used to understand a system. Data models and requirement descriptions were other important artifacts. Surprisingly, and contrary to what we found in the literature, architectural models were not found to be very important.

Dzidek et al. [28] reported about an empirical evaluation of the costs and benefits of UML models as a form of documentation during software maintenance. Among the factors that they considered was the impact of the up-to-datedness of documents' on the quality of the follow-up (downstream) artifacts. This will be an analyzed as an important factor for our case study research as well.

Furthermore, we conducted a systematic mapping study [6]. According to that, there are fourteen studies in the software engineering field on the subject of documentation benefit and quality. Six of these studies explore surveys and eight of them investigate experiments.

2.4. Modeling of documentation usage and usefulness

In order to devise a precise methodology to assess documentation usage and usefulness, it is important to clearly understand, characterize and model these two topics. By synthesizing the existing literature on usage and benefits of documentation (e.g., [2,15,16], we have formalized the documentation usage workflow as an UML activity diagram (see Fig. 1). Similarly, documentation usefulness is described as a class diagram in Fig. 2.

When a software engineer intends to work on a development or maintenance task, they may need extra information to perform the task at hand. While software engineers will usually remember the details of the artifacts developed by them, they need to communicate with colleagues and/or refer to documentation for the artifacts that they have not developed themselves. From the experience of the developers at the case study company, information coming from their own experience and knowledge is considered first. Subsequently, if needed, information is retrieved from both communication with colleagues and referring to documentations. The degree of usage of the different sources of information is the main content of RQ 1.

In Fig. 2, we have classified usefulness according to being a maintenance aid, development aid, or management decision aid. For both maintenance aid and development aid, supporting comprehension is an important aspect. This is because many aspects of the software under analysis need to be understood (i.e., comprehended) in order to be properly modified, including "functionality, architecture and a myriad of design details" [16]. In RQ 3, we will use cost effectiveness and perceived usefulness as two measures of usefulness of technical documentation.

3. Research methodology and case-study design

3.1. Overview

The entire research project behind this article was initiated based on the need of the case-study company to optimize their documentation process and to reduce the cost of documentation (both development and maintenance) without adversely impacting the quality and/or delivery schedule of the software products. To conduct the case study, we collected both quantitative and qualitative data. Researchers and practitioners had identical authority levels during the entire project. Before the project start date, we had a formal written contract in place. During the 2.5 year period, we have gone through three major study cycles in which the initial assessment of documentation cost, usage and quality have been conducted and initial guidelines have been passed to the industrial partner.

Our research methodology was a combination of confirmative, descriptive and exploratory case-study approaches. Here, and in what follows for the description of the whole case study, we follow the terminology and methodology proposed by Runeson and Höst [35]. The descriptive portion describes the status quo for the case study company and is applied to the usage of documentation. The explanatory part is seeking impacting factors partially known form literature and considered relevant in the context of the company. Finally, our exploratory research component is looking at what is happening and is seeking new insights. This part of our research was applied for the usefulness of technical documentation. Details and results of this cases study are discussed in the remainder of this paper.

3.2. Research questions

To conduct this study, we raised the following RQs and classified them as follows.

RQ 1 – Characterization of usage of technical software documentation

- RQ 1.1 (Source of information): What is the frequency of use of technical doc compared to that of other sources of information, i.e., communication with team members, reference to code documents, and existing (self-) knowledge of developers?
- RQ 1.2 (Development versus maintenance): Does documentation usage differ between the development and maintenance phases?

RQ 2 – Factors impacting documentation usage

- RQ 2.1 (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 2.2 (Job function): Does a practitioner's role have a significant impact on the level of documentation usage? For example, do testers use documentation more compared to developers?
- RQ 2.3 (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?

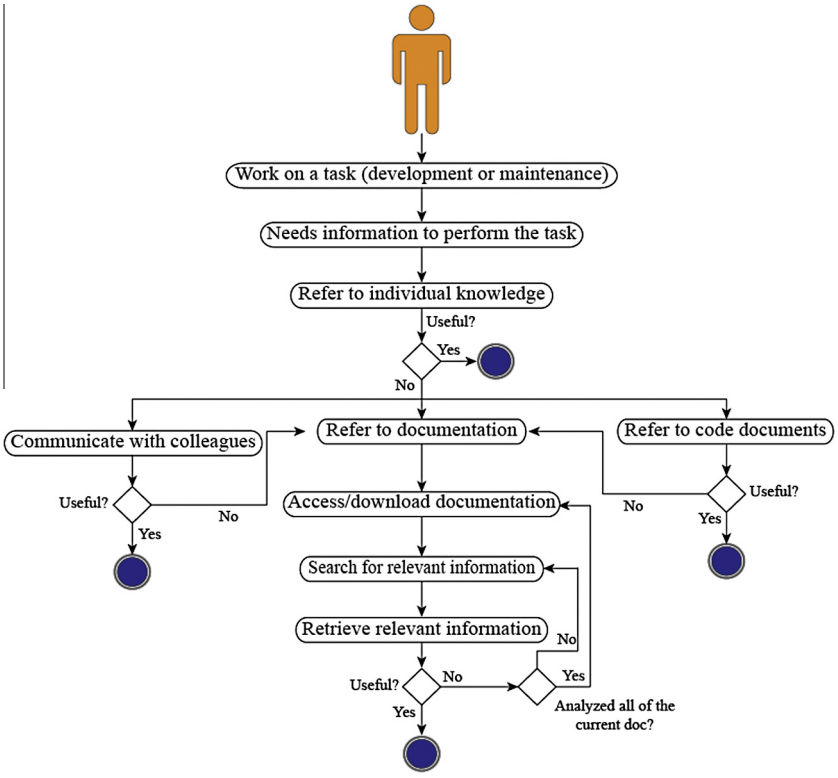

Fig. 1. Workflow of documentation usage.

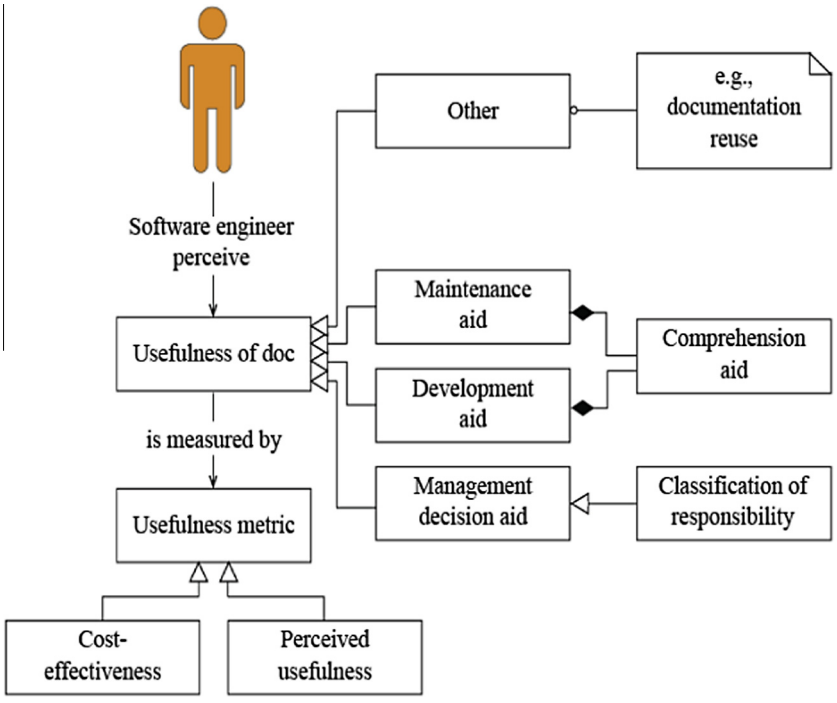

Fig. 2. Diagram for documentation usefulness.

- RQ 2.4 (Readability): Does readability of the document have a significant impact on the level of documentation usage?

RQ 3 – Usefulness of technical documentation

- RQ 3.1: How could the cost-effectiveness of documents be characterized?
- RQ 3.2: What is the perceived usefulness of technical documentation?

3.3. Research approach

An overview of the approach is illustrated in Fig. 3. The problem area was defined as the result of a project (NSERC ENGAGE Collaboration grant) performed over a period of six months at the case study company. Concurrently with the above additional

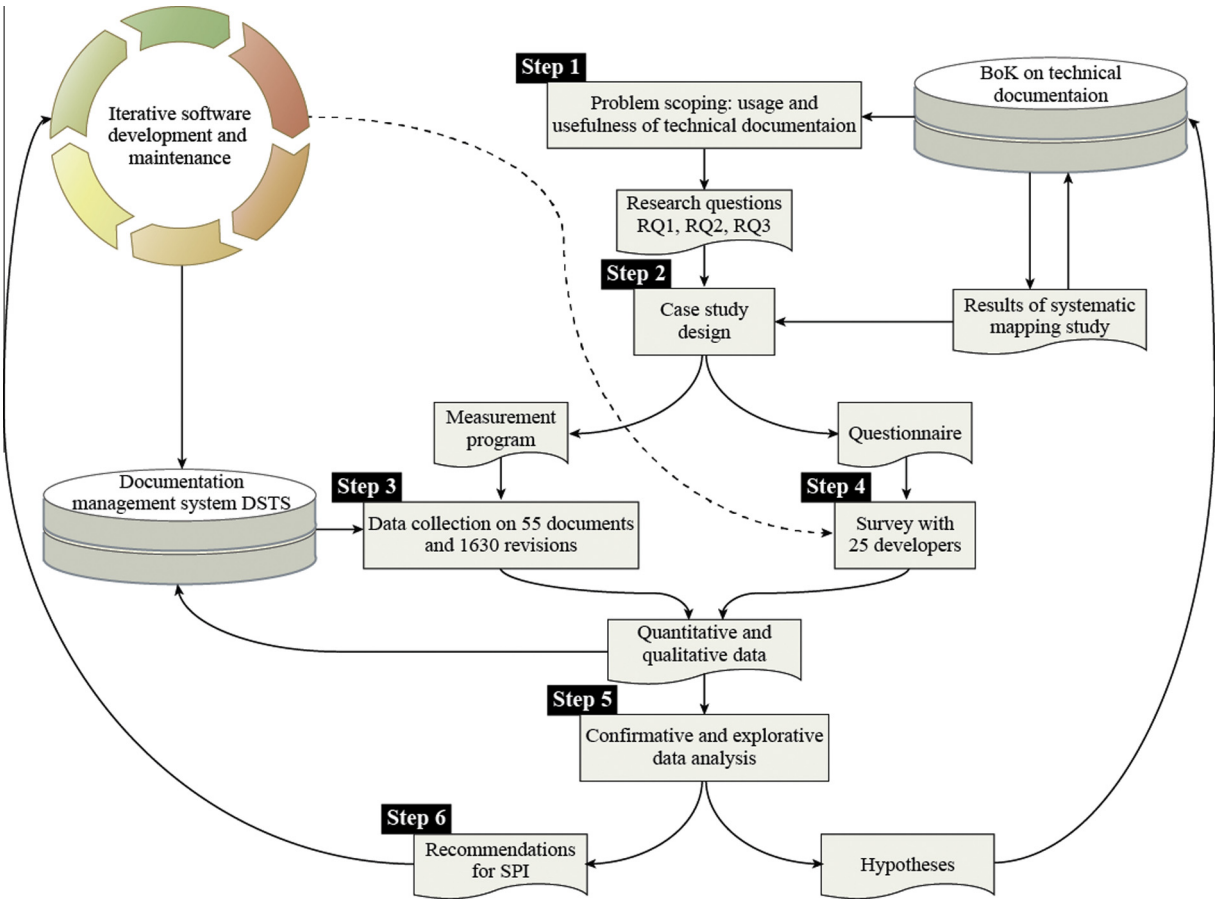

Fig. 3. Main steps and artifacts of the case study approach.

work was completed investigating the existing body of knowledge (BoK) in the area of technical documentation, a systematic mapping study [5] was conducted by Zhi et al. The objective of the study was to analyze the current state-of-the art in terms of cost, benefit and quality of software development documentation. As the result of these two efforts, the RQs were formulated for the collaboration project.

Case studies are an established empirical method aimed at investigating contemporary phenomena in their context. The phenomena studied are the usage and usefulness of technical documentation during the software development and maintenance process. Our actual case study can be described as consisting of six main steps:

Step 1: Definition of scope and context
Step 2: Case study design
Step 3: Data collection
Step 4: Survey
Step 5: Data analysis
Step 6: Reporting and conclusions

The main steps and artifacts are described in Fig. 3. Section 4 describes the Steps 1–4. The subsequent Steps 5 and 6 are the content of Section 5. Data collection was done utilizing the company's existing documentation management system DSTS. From there, 55 documents were selected for the case study analysis. The selection was done by the domain experts, reflecting the importance of the documents for the product under development. The evolution of the documents was taken into account by analyzing a total of 1630 of their revisions. The survey conducted as Step 4 included 25 developers from the case study company (dotted arc to the actual software development and maintenance process).

As the result of the whole process, the main contributions from conducting various types of analysis based on case study data were recommendations for software process improvement and validation of stated hypotheses or derivation of new ones.

Following the Runeson and Höst's guidelines on reporting case-study results [35], we have structured this paper as being *linear-analytic*, which is the standard research report structure for case-study papers (problem, related work, methods, analysis and conclusions).

3.4. Case-study design and data collection

3.4.1. Definition of scope and context

NovAtel Inc. is a leading provider of Global Navigation Satellite System (GNSS) products. Their OEM (Original Equipment Manufacturer) products depend heavily 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. Shorter cycles of iterations were introduced. Also, with a large number of legacy products, the overall maintenance cost was increasing continuously. Looking into the usage and usefulness of documentation was considered one means to improve this situation. This was found as a result from conducting

an initial survey that included 43 managers and developers from the company.

Taken from both literature and the knowledge and experience within the company, 64 impacting factors were ranked in terms of the likelihood of occurrence and the perceived impact on maintenance. The factors were grouped into

- Programming and implementation
- System requirements
- Design
- Documentation
- Verification and validation
- Personnel resources
- Process and project management

3.4.2. Data collection

The company has more than 20 years of legacy code and documentation. An in-house developed document management system named DSTS is used to manage the documentation repository in the company. DSTS is essentially a storage, retrieval and versioning system for documentation files (in formats such as Word and PDF). In DSTS, documentation creation and revision are organized by check-ins, similar to version control system for source code. Note that, in the company, code comments are not stored in the DSTS, but rather embedded in source-code files stored in source-code version control systems.

At the time of our study, the DSTS system had a repository of more than 60,000 documents and their revisions. Generally, the number of both released and draft documents increased dramatically after the year 2004 because of introducing new products to the OEM family. Since the number of documents increases with time and proportionally to the number of new products, their maintenance is also getting more difficult and costly.

For the assessment of the usage and usefulness of documentation artifacts, three types of OEM6 related documents were studied: conceptual design, test plan (procedures), and process documents (QA).

3.4.3. Questionnaire-based survey

By adapting questions from similar studies in the literature, e.g., [15], 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., [30]. We were also relying on our past experience in designing and executing industry-oriented surveys, e.g. [31].

The high-level documentation assessment questionnaire was developed as an online survey to investigate the usefulness, usage and quality aspects from the software professionals' view. The goal of this questionnaire was to have a better understanding of the role of documentation in practice. This survey was answered by software practitioners involved with real-world software projects.

The high-level questionnaire targets individuals involved with software development and maintenance activities. The selection of the subjects was done by using the contact person provided by the company. The subjects of the questionnaire are categorized based on their position (job function) in the company and based on their experience.

The online questionnaire was sent to a group of 135 software engineers in the 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 was done on a voluntary and anonymous basis. Before the execution of the survey, a briefing session was held among the engineering team (survey respondents) and researchers to ensure that everyone had the same understanding of all the questions. 25 of the 135 invited staff members who were involved with OEM6 product, responded to our survey, i.e., the response rate (sample ratio) was about 18.5%. The first two questions of the survey (see the Appendix A) queried the subjects' profile and demographics. A large number of participants (52%) had more than 10 years of experience, and just one of the participants has less than one year experience in the software engineering field.

Question 2 of the questionnaire asked the participants about their current job function. This question helps to address the relationship and correlation between the documentation usage and each individual's job function. Table 2 presents all the job functions involved. Most of the participants (40%) are classified as *senior developers*, and twenty percent of them were *intermediate developers*.

The complete version of the questionnaire is presented in the Appendix A. The main aspects of the survey relate to:

- Participant demographics and level of experience-related questions.
- Demographic questions designed to screen the respondents and to help separate individuals based on their position and software field experience. These are questions about the respondent's characteristics and circumstances.
- Perceived quality of documentation versus potential (expected) quality of documentation.
- The role of documentation in the software development and maintenance process.
- Current use of documentation artifacts for development and maintenance purposes.
- Perceived benefit of documentation during development and maintenance activities.
- The participant's personal use and preference for different types of documentation during development and maintenance activities, as well as opinions concerning the effectiveness of these documentation artifacts.

3.4.4. Selection of the product under study

To conduct our study and evaluate the methodology, one of the latest released products (OEM6), which is still under development and maintenance, was selected as the system under study. The decision in favor of the ongoing product OEM6 was done by the case study company. The rational was to incorporate results of the research into the ongoing development process. OEM6 is a medium-size project with extensive change activities on both source code and documentation.

The choice of OEM6 as the software product under study narrowed the pool of 60,000 documents stored in the DSTS document management system to a few hundred. We then applied cluster and random sampling to select the most representative documents from all stages of the SDLC for OEM6. After a few systematic sampling iterations and careful discussions among the researchers and practitioners, we selected, for the case study, 20 conceptual design documents covering major modules of the OEM6 embedded software, 15 test documents and 20 software process documents. Moreover, for each individual document, all revisions were collected for documentation benefit and quality analysis.

Table 2
Job functions studied and their abbreviations used in Fig. 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

Table 3 presents a summary of the documents collected for this study. Each access log document for each revision was stored individually. Since there are a large number of revisions for some of documents (e.g. 1036 for test documents), we automated the process of collecting all accesses of various revisions using a developed Visual Basic macro. Each individual document has multiple revisions over time. All revisions were studied and analyzed for measurement purposes. Design documents are entirely for OEM6, so that they have relatively fewer revisions, compared to process documents shared by different past projects. Due to the fact that most of the test documents were reused from previous product lines (e.g., OEM4, OEM5), they had a long history that includes a large number of revisions created over time.

3.4.5. Data analysis methods

As part of the case-study design, it is important to decide about the type of analysis as well. An overview of all the RQs, sub RQs, goals of analysis, corresponding hypotheses and the analysis approach applied for each RQ (shown in Table 4). The formulation of each hypothesis also includes the different treatments studied in the respective RQ. To properly address RQ 1.1, six different pairs of corresponding treatments were identified, as shown in Table 4. In this particular context, the six cases correspond to the number of permutations on the factors involved, i.e., sources of information being technical documentation, source code, personal communication or individual knowledge; and also the two SDLC phases of development and maintenance.

For the statistical inference analysis, we applied non-parametric tests (Mann–Whitney, Kruskal–Wallis) as we could not confirm that the underlying populations follow a normal distribution. The data source for all of the RQs in Table 4 was the data from the survey, except for RQ 2.5 which we used the automatically-mined access logs.

4. Results

In what follows, we report the results from analyzing the three RQ's and the subsequent aggregation of the results for providing recommendations for process improvement.

4.1. RQ 1 – analysis of usage

This RQ is devoted to the analysis of usage in dependence of the source of information (RQ 1.1) and the development phase (RQ 1.2). 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. Fig. 4 shows the mean value confidence intervals for usage of technical documentation in development versus maintenance for the different sources of information.

4.1.1. RQ 1.1 – usage of documentation in comparison of other sources of information

To address RQ 1.1, we posed six hypotheses. In what follows, we report the results of the formal analysis in terms of the p-values received from the application of the Mann–Whitney test and the conclusion (based at a 0.01 significance level) that can be made out of it. Here and for the rest of the paper, we intend to compare source code as a source of information with non-source-code-type technical documentation (e.g., design documents, models).

(a) **Null Hypothesis H_0(1.1a)**: According to the Mann–Whitney U-test results ($p = 0.7262$), there is no significant difference between the usage of technical documentation versus usage of source-code during development.
(b) **Null Hypothesis H_0(1.1b)**: According to the Mann–Whitney U-test results ($p = 0.0$), there is significant difference between the usage of technical documentation versus usage of source-code during maintenance.
(c) **Null Hypothesis H_0(1.1c)**: According to the Mann–Whitney U-test results ($p = 0.1610$), there is no significant difference between the usage of technical documentation versus personal communication during development.
(d) **Null Hypothesis H_0(1.1d)**: According to the Mann–Whitney U-test results ($p = 0.0085$), there is significant difference between the usage of technical documentation versus personal communication during maintenance.
(e) **Null Hypothesis H_0(1.1e)**: According to the Mann–Whitney U-test results ($p = 0.0022$), there is significant difference between the usage of technical documentation versus existing individual knowledge during development.
(f) **Null Hypothesis H_0(1.1f)**: According to the Mann–Whitney U-test results ($p = 0.1030$), there is no significant difference between the usage of technical documentation versus existing individual knowledge during maintenance.

As shown in Fig. 4, the source code is the first consulted and useful resource during maintenance activities. Interestingly, documentation is among 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 the above-mentioned 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., [20,32], in that source code is the mostly used artifact to study during maintenance.

4.1.2. RQ 1.2 – usage of documentation in development versus maintenance

In this section, we present the results of the RQ 1.2 being "Is the amount of documentation usage different between development and maintenance phases?"

Table 3
Summary of collected data for analysis.

Type of documentation	Design	Test plans/procedures	Process (QA)
Total number of documents	20	262	20
# of collected sample documents	20	15	20
# of all revisions	140	1036	454
Selection criteria	All software conceptual designs of OEM6	Test plans and test procedures of OEM6 (ensuring full coverage of OEM6 modules)	All software related process documents
Coverage (sampling ratio) (%)	100	5.7	100

Table 4
Design of the case study.

RQ	Sub-RQ	Goals of analysis, treatments and corresponding hypotheses	Type of analysis (and visualization)
RQ 1 – Analysis of usage of technical documentation	RQ 1.1 (Usage per source of information)	– H_0(1,1a): Usage of technical documentation versus usage of source code during development – H_0(1,1b): Usage of technical documentation versus usage of source code during maintenance – H_0(1,1c): Usage of technical documentation versus personal communication during development – H_0(1,1d): Usage of technical documentation versus personal communication during maintenance – H_0(1,1e): Usage of technical documentation versus individual knowledge during development – H_0(1,1f): Usage of technical documentation versus individual knowledge during maintenance	Confirmatory (box plots and Mann–Whitney *U* test)
	RQ 1.2 (Usage per phase)	– H_0(1,2): Usage of technical documentation during development versus maintenance	Confirmatory (box plots and Mann–Whitney *U* test)
RQ 2 – Factors impacting documentation usage	RQ 2.1 (Document type)	– H_0(2,1a): Usage of the various types of technical documentation during development – H_0(2,1b): Usage of the various types of technical documentation during maintenance	Confirmatory (box plots and Kruskal–Wallis test)
	RQ 2.2 (Job function)	– H_0(2,2a): Usage of technical documentation during development for varying job functions – H_0(2,2b): Usage of technical documentation during maintenance for varying job functions	Exploratory (box plots)
	RQ 2.3 (Information usage for varying degree of experience)	– H_0(2,3a): Usage of types of information for varying degree of experience during development – H_0(2,3b): Usage of types of information for varying degree of experience during maintenance	Confirmatory (box plots and Mann–Whitney test)
	RQ 2.4 (Documentation type usage for varying degree of experience)	– H_0(2,4a): Usage of documentation type for varying degree of experience during development – H_0(2,4b): Usage of documentation type for varying degree of experience during maintenance	Confirmatory (box plots and Kruskal–Wallis test)
	RQ 2.5 (Readability)	Impact of the readability of documents on their usage	Exploratory (regression analysis)
RQ 3 – Analysis of usefulness	RQ 3.1 (Cost effectiveness)	Analysis of correlation of factors potentially impacting the cost-efficiency index and the number of accesses to documents.	Exploratory (Spearman correlation, clustering)
	RO 3.2 (Perceived usefulness)	Perceived usefulness of technical documentation.	Exploratory (bar charts)

Null Hypothesis H_0(1.2): According to the Mann–Whitney *U*-test results ($p = 0.002$), there is no significant difference between the usage of documentation during software development versus maintenance activities.

4.2. RQ 2 – factors influencing documentation usage

This RQ looks into the impact of factors supposed to influence usage of technical documentation. The results of both descriptive and inference statistics are reported below.

4.2.1. RQ 2.1 – usage of documentation for varying document types

For both H_0(2.1a) and H_0(2.1b), 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, and we applied the non-parametric Kruskal–Wallis test. For the degree of freedom, with five treatments, we applied df = 4.

Null Hypothesis H_0(2.1a): According to the Kruskal–Wallis test results ($p = 1.003e{-}06$), there is no significant difference between the usage of various documentation types *during development.*
Null Hypothesis H_0(2.1b): According to the Kruskal–Wallis test results ($p = 1.486e{-}07$), there is no significant difference between the usage of various documentation types *during maintenance.*

According to the results shown in Fig. 5, the usage of different types of documentation artifacts varies for both development and maintenance purposes. The most intensively used documentation artifacts during development phase are the design documents. Similarly, code comments are the most intensively used documentation artifacts for maintenance purposes.

4.2.2. RQ 2.2 – usage of documentation in dependence of job function

RQ 2.2 aimed at assessing whether the developers role (position) has a significant impact on the level of documentation usage during development and maintenance. We formulate two null hypotheses:

Null Hypothesis H_0(2.2a): The practitioners' job function does not have any significant impact on the amount of documentation they use during development.
Null Hypothesis H_0(2.2b): The practitioners' job function does not have any significant impact on the amount of documentation they use during maintenance.

The results of the related descriptive statistics are shown in Fig. 6. We could not apply statistical tests because of lack of sufficient data. As we observe, all distributions, except the ones related to SD and IA, have major overlap. We do not have enough evidence to conclude that job functions do have a significant impact on the extent of documentation usage. However, we can see from the box

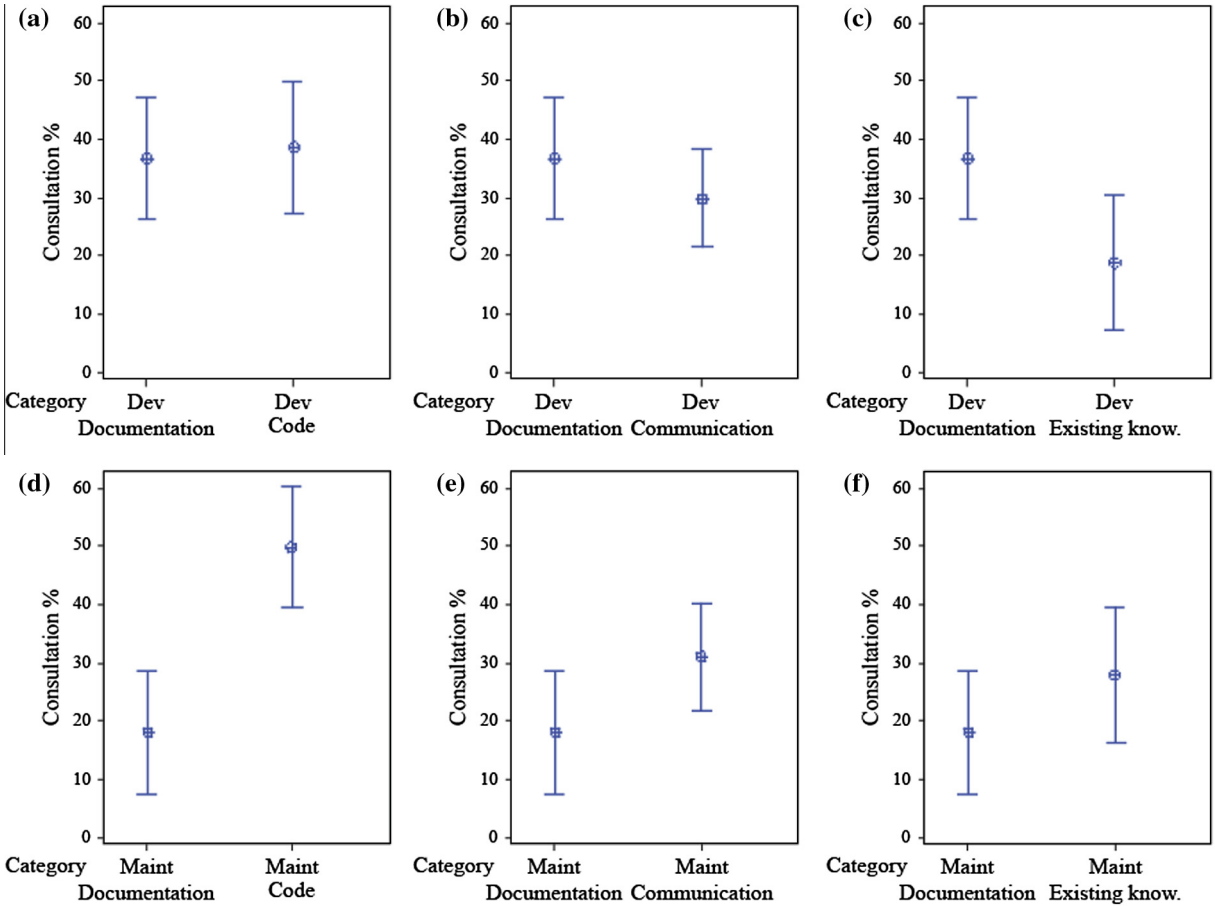

Fig. 4. 95% confidence interval of sample means for usage ratio of documentation versus usage of code (left), communication (middle) and existing knowledge (right).

plots that, to some extent, intermediate developers (ID) use more documentation compared to senior developers (SD).

4.2.3. RQ 2.3 – usage of source of information in dependence of the degree of experience

RQ 2.3 aimed at assessing whether the degree of developers' experience has a significant impact on *the level of documentation usage* during development and maintenance. We formulate two null hypotheses:

Null Hypothesis H_0(2.3a): According to the Kruskal–Wallis test results ($p = 0.171$, 0.484, 0.826 and 0.324 for sets of distributions for documentation, source-code, communication and personal knowledge, respectively), there is no significant influence of job experience on their source of information usage during development.

Null Hypothesis H_0(2.3b): According to the Kruskal–Wallis test results ($p = 0.496$, 0.861, 0.130 and 0.057 for sets of distributions for documentation, source-code, communication and personal knowledge, respectively), there is no significant influence of job experience on their source of information usage during maintenance.

Fig. 7 illustrates the usage of technical documentation and the degree of consultation from various sources of information during development (top) and maintenance (bottom) for varying degrees of experience. As there was just one participant with less than one year experience, we have partitioned into three ranges: up to 5 years, 6–10 years and more than 10 years of experience.

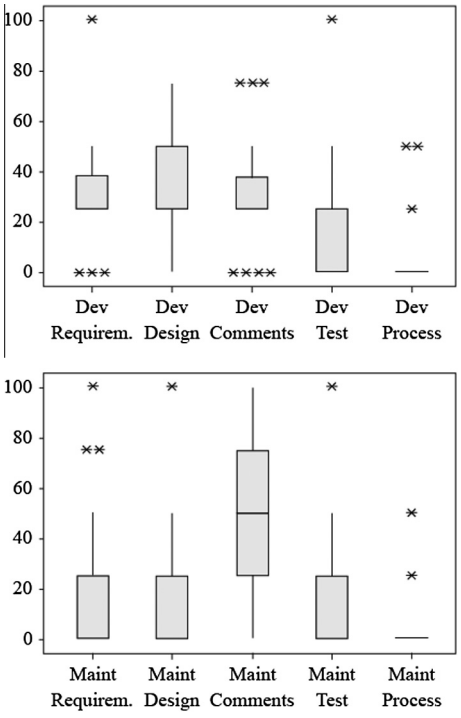

Fig. 5. Percentage of usage of documentation (*y*-axis) for development (top) versus maintenance (bottom) in dependence of document type.

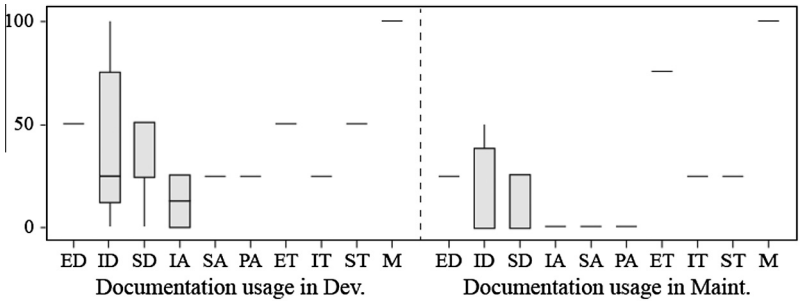

Fig. 6. Percentage of usage of documentation (y axis) for development (left) versus maintenance (right) classified by job functions.

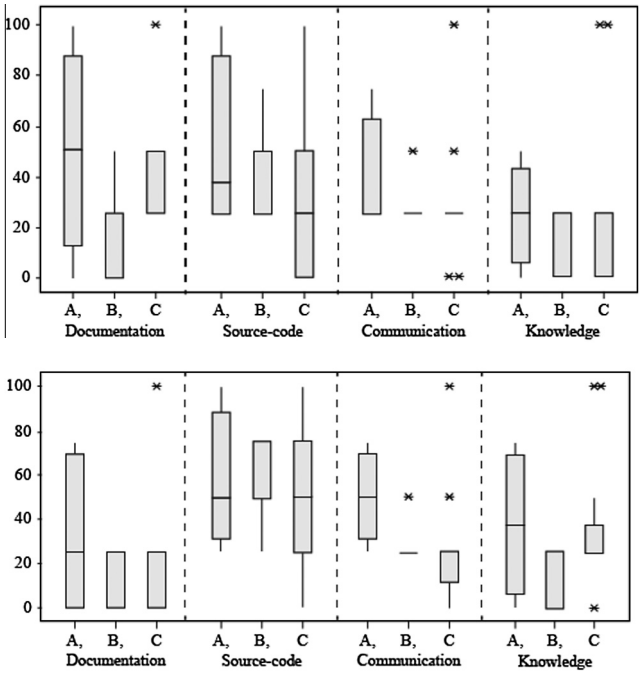

Fig. 7. Information usage for development (top) and maintenance (bottom) in dependence and years of experience for up to 5 years (A), 6 to 10 years (B) and more than 10 years (C).

4.2.4. RQ 2.4 – usage of documentation type in dependence of the degree of experience

RQ 2.4 aimed at assessing whether the developers degree of experience has a significant impact on the *type of technical documentation* used during development and maintenance.

Null Hypothesis H_0(2.4a): According to the Mann–Whitney U-test results (p = 0.179, 0.338, 0.920, 0.412 and 0.103 for sets of distributions for requirements, design, comments, test documents and process documents, respectively), there is no significant influence of job experience on the usage profile during development.

Null Hypothesis H_0(2.4b): According to the Mann–Whitney U-test results ($p = p$ = 0.664, 0,184, 0.820, 0.716 and 0.572) for sets of distributions for requirements, design, comments, test documents and process documents, respectively), there is no significant influence of job experience on the usage profile during development.

Fig. 8 describes the preferred source of information 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.

4.2.5. RQ 2.5 – impact of readability and usage

We also evaluated the relationship between the readability of documentation and its usage by combining the automated readability scores and document access log records. This analysis has been performed on design documents as these were the ones most frequently used by developers.

We used linear regression analysis to address the relationship between the usage and readability of documentation artifacts. The regression line $y = 2.9x - 77.8$ for the aforementioned relationships is visible in Fig. 9. With an R-squared value equal to 0.21, we conclude a weak positive correlation between the readability of design documents and their usage.

As per our experience and discussions with the team members, we predict that a major contributing factor to increasing usage (reading frequency) of a document would be the extent to which the software developers need the information written inside that document. In the scope of this project, we did not have the chance to systematically measure/analyze this prediction. No significant correlation was found from similar analysis of test and process (QA) documents.

4.3. RQ 3 – analysis of usefulness

Usefulness of technical documentation is defined in [29] as "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." As part of the case study approach, our main focus is the documentation usefulness to support practitioners' information needs in an efficient way. We approach the usefulness characterization from two angles. Firstly, and based an automated measurement, we study cost-effectiveness as one of the important aspects of usefulness. Secondly, we look at various aspects that are supposed to impact usefulness and analyze existing gaps for the context of the case study project.

4.3.1. RQ 3.1 – characterize the cost-effectiveness of documents?

Measuring usefulness by cost-effectiveness, we defined the Cost-Effectiveness-Index (CEI) for a given document *doc* as

$$CEI(doc) = Access\#(doc)/EditingTime(doc) \quad (1)$$

where Access#*(doc)* denotes the total number of accesses or downloads and *EditingTime(doc)* refers to the time spent on editing the document file (in hours). *CEI(doc)* is defined based on the assumptions that (1) the access/download operations indicate the developers are browsing the documents and thus are gaining benefits from them and (2) the time spent on editing a document is an indication

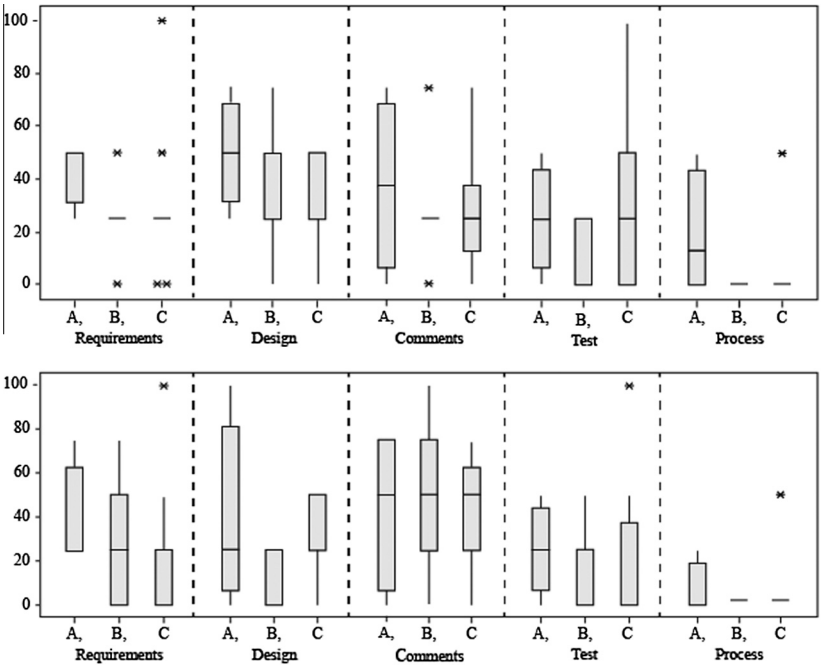

Fig. 8. Percentage of documentation usage (*y* axis) in dependence of documentation type and years of experience for up to 5 years (A), 6 to 10 years (B) and more than 10 years (C).

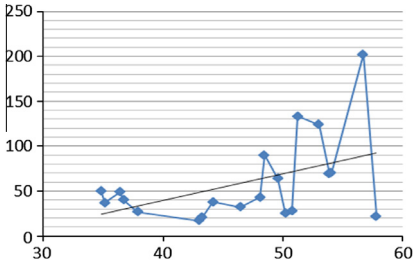

Fig. 9. Relationship between readability (*x*-axis) and usage of design documents (*y*-axis).

of cost, denoting the developer efforts invested on writing or maintaining the documents.

For each hour spent on editing, the CEI value expresses how many download times we can get from that one-hour investment. In other words, the investment is counted by editing time and the outcome is reflected by the download times. CEI can be seen as an indicator of Return of Investment (ROI) for the given document. The higher the *CEI(doc)* value, the more cost-effective (and thus potentially useful) that document is. The metric was discussed with and considered meaningful by the case study company.

In order to better understand the root causes for usefulness expressed by CEI, we performed a correlation analysis on all the sampled documents. The purpose of correlation analysis is to investigate quantitatively the correlation relationship among factors potentially related to CEI and Access#. The factors studied include document Author#, Revision#, Visitor#, Access#, and Size. The factors were selected together with the industry partner and were based on their assumed degree of importance. The definition of these factors is given in Table 5.

For all the selected relevant pairs, and as we could not ensure the data is normally distributed, we computed the Spearman correlation coefficient. In that case, the rank orders are compared between all the pairs of factor values. The results are presented in Table 6.

Correlations of value 0.8 or higher (positive or negative) are highlighted. Except for test documents, there is no significant correlation relationship between CEI and any other attributes. For the strongest one (between Size and CEI), the correlation is negative. That means, the larger the document (in terms of number of words), the lower the CEI. This aspect was found to be useful by the case study company. We also observe that visitor# and access# are highly correlated across all types of documents.

In addition to the above, we conducted a cluster analysis to find the degree of similarity between Author#, Revision# and their CEI. We selected the single-linkage clustering method [33] which is a variant of hierarchical clustering. Incrementally applying nearest neighbor search, a dendrogram is created describing the similarity between objects (documents). We applied the linkage method to "Single-linkage" and selected distance to be "Euclidean".

The results from running the cluster analysis using Minitab[1] are shown in Fig. 10. We can see that there are two clusters and an outlier Doc#6 ((Boot Code Design). Cluster #2 has five documents (Doc# 1, 3, 18, 19 and 12). The biggest cluster (Cluster #3) contains 13 documents. A closer check confirms that Cluster #3 is a group of documents with relatively low CEI value, similar Author# and Revision#.

Our exploratory clustering analysis reveals a possibility that documents may form certain clusters based on our measured attributes. Each cluster can be treated as a conceptual management unit. If a document falls within a particular cluster whose characteristics are well studied or known, then it may help managers to reason or predict the benefits or usefulness of such a document.

4.3.2. RQ 3.2 – perceived usefulness of technical documentation

RQ 3.2 aimed at measuring the impact of various attributes to the usefulness of documentation. As an exploratory RQ, we wanted to better understand which factors were dominant and which of them offered the biggest potential for improvement based on the perception of the survey participants.

[1] http://www.minitab.com/en-us/.

Table 5
Independent factors for explaining usefulness.

Factor	Description
Author#	The number of authors editing one document
Revision#	The number of revisions on a document. Both Engineering Change Order (ECO) and draft revision are counted as one revision
Visitor#	The number of unique visitors on a document. This variable is intended to investigate whether some documents are accessed by a small group of users only
Access#	The number of access/download times of a document
Size	The document size, measured by the number of words

Table 6
Spearman rank correlation coefficients among pairs of variables including CEI and Access#.

Variable pairs	Design Doc	Test Doc	QA Doc	All Doc
Corr(Author#, CEI)	-0.13	-0.57	0.29	0.04
Corr(Revision#, CEI)	-0.16	-0.59	-0.06	-0.06
Corr(Visitor#, CEI)	-0.24	0.24	-0.34	0.00
Corr(Size, CEI)	-0.18	-0.83	-0.23	-0.32
Corr(Author#, Access#)	0.58	0.29	0.69	0.65
Corr(Revision#, Access#)	0.80	0.32	0.82	0.74
Corr(Size, Access#)	0.30	0.27	0.76	0.21
Corr(Visitor#, Access#)	0.90	0.96	0.80	0.90

Question #10 of the survey asked the participants to rate the impact of each of the following attributes on the overall quality of a document: organization, inclusion of visual models, relevance of the content, preciseness, readability, completeness, accuracy, consistency, being up-to-date, documentation technology, and use of examples (as part of the documentation). While these attributes can be considered quality factors in general, they are expected to impact usefulness in particular. The participants rated each of the attributes between *very low*, *low*, *high*, and *very high*. The results are summarized in Fig. 11.

From the participants' point of view, the most relevant factors that contribute to high-quality documentation include documentation up-to-datedness, relevance of content, accuracy and documentation preciseness. Other important factors consist of documentation completeness, visual models and accuracy. Conversely, documentation technologies (Word, HTML, and Doxygen[2] for C/C++ code) were considered the least relevant factor.

In addition to the analysis of impacting factors, 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 histogram in Fig. 12. For simplicity reasons, we treated the ordinal scaled measures like being on an interval scale. It is interesting to see that expectations are higher than the perceived actual values for all the seven attributes. The biggest gap was found in the most important criterion of being up-to-date. Combining the two results from Figs. 11 and 12 resulted in Recommendation 7 described in subsequent Section 5.4.

5. Discussions and Interpretation of findings

5.1. Summary of the findings and interpretations of the findings

Based on the results of RQ 1–RQ 3, interpretations of the findings are discussed next.

In results of RQ 1.1, we found that usage of source-code during maintenance is significantly more than usage of technical documentation. Other than that case, we observed no significant differences between usage of documentation in comparison of other sources of information, i.e., source-code, communication with team members, and existing (self-) knowledge of developers during development and maintenance. Results of RQ 1.2 revealed that usage of documentation during software development is more than that during the maintenance phase.

RQ 2 explored various factors influencing documentation usage during development versus maintenance phases. Results of RQ 2.1 revealed that the most intensively used documentation artifacts during the development phase is design documents, however code comments play the role of the most intensively used documentation artifacts for maintenance purposes.

According to results of RQ 2.2, although we do not have enough evidence to conclude that job functions do have a significant impact on the extent of documentation usage, we saw (from the box plots) that, to some extent, intermediate developers use more documentation compared to senior developers.

According to results of RQ 2.3, we observed that practitioners' experience does not have any significant impact on documentation usage. However we observed (from the box plots) that, to some extent, developers with more experience tend to somewhat use less documentation which is as one would expect.

RQ 2.4 explored the usage of various documentation types (requirements, design, comments, test documents and process documents) by the degree of practitioners' experience. According to results, we observed that developers experience does not have any significant impact on the type of documentation used during development or maintenance.

RQ 2.5 explored, in an exploratory fashion and using regression analysis, the impact of the readability of documents on their usage. We found that there is a weak positive correlation between the readability of design documents and their usage. No significant correlation was found from similar analysis of test and process (QA) documents.

RQ 3 explored and analyzed the usefulness of documents. RQ 3.1 analyzed the cost effectiveness aspect by correlation analysis of factors potentially impacting the cost-efficiency index and the number of accesses to documents. Among the findings were that the larger the document (in terms of number of words), the lower the CEI (cost effectiveness), denoting that developing (extra) large documents may not be that cost effective. Our exploratory clustering analysis also revealed a possibility that documents may form certain clusters based on our measured attributes. Each cluster can be treated as a conceptual management unit. If a document falls within a particular cluster whose characteristics are well studied or known, then it may help managers to reason or predict the benefits or usefulness of such a document.

As an exploratory analysis, RO 3.2 helped us to understand the factors with the highest impact on the overall quality of a document. The firm's software engineers rated the impact of each of the following attributes on the overall quality of a document: organization, inclusion of visual models, relevance of the content, preciseness, readability, completeness, accuracy, consistency, being up-to-date documentation technology, and use of examples (as part of the documentation). From the participants' point of view, the most relevant factors that contribute to high-quality

[2] http://www.stack.nl/~dimitri/doxygen/.

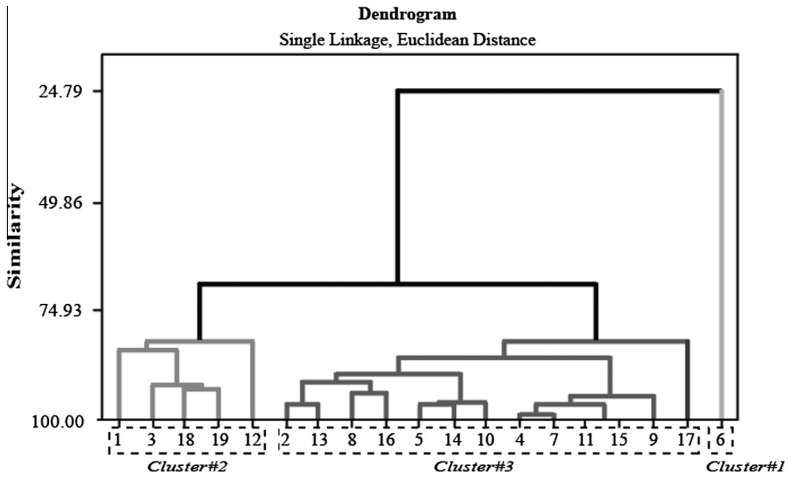

Fig. 10. Dendrogram between design documents based on similarity defined by Author#, Revision# and CEI.

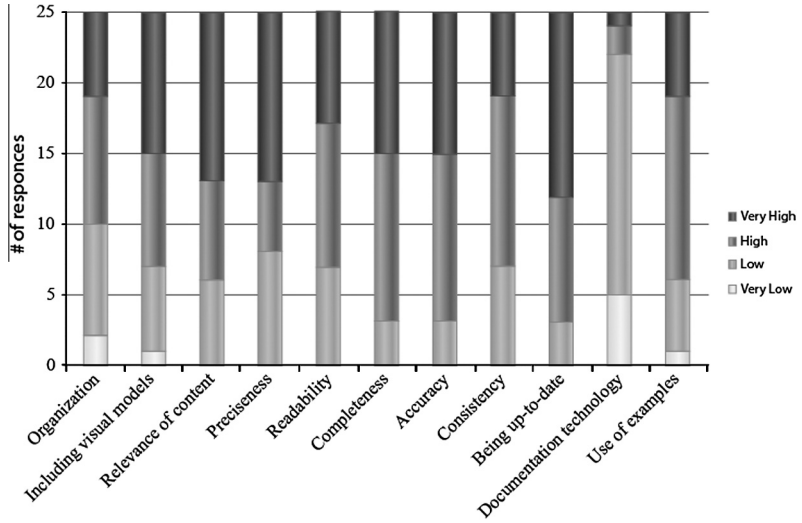

Fig. 11. Impacting factors on perceived quality.

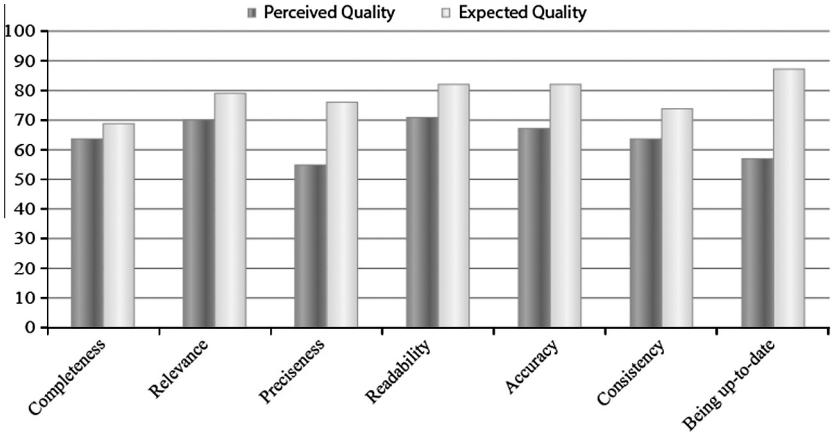

Fig. 12. Expected versus perceived quality level of attributes impacting quality of documentation.

documentation include documentation up-to-datedness, relevance of content, accuracy and documentation preciseness. Other important factors consist of documentation completeness, visual models and accuracy. Conversely, documentation technology was considered the least relevant factor.

5.2. Challenges and lessons learned

Similar to many previously-reported industrial case-studies in various areas of software engineering, we faced numerous challenges during this project which we carefully devised solution strategies for. We also learned several lessons which we discuss next.

We can group the challenges into technical and non-technical challenges. The main technical challenge in the project was the following: It was not trivial to objectively measure the usage of documentation in a quantitative manner. We measured the usage using any possible quantitative and qualitative metric we could but, since usage of documentation is a human-intensive task, further more sophisticated measurement instruments for this purpose (e.g., PC screen video recording of how the software engineers use documents) could be considered.

The major non-technical challenges in the project were the followings:

- Ensuring the buy-in of all the stakeholders and keeping their motivation level for the study, especially the large team of software engineers in the case-study company, was not easy. Unfortunately, documentation has not historically been a very popular topic among software engineers in the software engineering industry. Thus, we had to constantly and explicitly demonstrate the usefulness of the results and outcomes of the study to ensure the buy-in of the large team of software engineers involved as the subjects of the study. Such a situation and phenomenon have also been observed and reported by earlier studies, e.g., [39].
- We observed slight differences in terms of short- and long-term research plans. The practitioners were in preference of "quick" ways of doing things and fixing the issues in short term. However, academic researchers preferred developing systematic solutions that will work effectively in long term. We as a team found a good balance and a middle ground solution for this challenge to satisfy both views.
- Academic researchers and practitioners were using slightly different terminologies, e, g., the practitioners used to refer to process-type of documents as "QA" documents. This caused challenges in communications in initial phases of the projects.

In terms of lessons learned, we observed and learnt that industry–academia collaborations could be a win–win situation. We learned that once industry and academia focus on real problems which would provide long- and short-term benefit, systematic approaches from the software engineering domain could be tailored and developed to help solve real-world challenges.

6. Recommendations for improvement of documentation process in the Industrial Context

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 (i.e., recommendations) for the documentation process in the company under study. In each case, the explicit traces to the specific RQ(s) and the corresponding results that have triggered each recommendation have been mentioned.

- Based on the output from RQ 1.1, we found that usage of source-code itself and code comments during maintenance is significantly more than usage of documentation. Thus, we recommended to the software engineering team to ensure providing suitable code comments to be used during maintenance. Also, to prevent overhead and promote efficient usage of time, clear guidelines on the usage of each of the four sources of information (source-code, communication with team members, and existing self-knowledge of developers) should be implemented (when, for whom, in which order).
- Since the results of RQ 1.2 revealed that usage of documentation during software maintenance is less than that during the development phase, we thus recommended documentation to be mostly prepared and targeted for development purposes than maintenance, and that the software engineering team prepares documentation accordingly. We also recommended the idea of "opportunistic" and "situational" documentation [34], i.e., preparing documents only for the purposes which they will be used.
- Results of RQ 2.1 revealed that the most intensively used documentation artifacts during development phase is design documents. We thus recommended to the software engineering team to ensure availability of design documents on time and with high quality. Results of RQ 2.1 also revealed that code comments play the role of the most intensively used documentation artifacts for maintenance purposes. We thus recommended having good quality code comments.
- In the analysis of RQ 2.2, we saw that, to some extent, intermediate developers use more documentation compared to senior developers. We thus recommended the needs for intermediate developers be considered in preparing documents.
- In analysis of RQ 2.3, we observed that, to some extent, developers with more experience tend to use less documentation which is as one would expect. We did not provide any recommendation based on this RQ nor RQ 2.4, as the findings were not that surprising.
- Results of RQ 2.5 revealed a weak positive correlation between the readability of design documents and their usage. We thus recommended increasing the readability of documents.
- Among the findings of RQ 3.1 was that: the larger the document (in terms of number of words), the lower the CEI (cost effectiveness), denoting that developing (extra) large documents may not be that cost effective. We thus recommended avoiding creating extra-large documents.
- Results of RQ 3.2 revealed that, from the participants' point of view, the important quality attributes that would impact the quality of a document were: documentation up-to-datedness, relevance of content, accuracy and documentation preciseness. We thus recommended to the software engineering team that the improvement opportunities for increasing quality of document should focus on those attributes. Keeping documents up-to-date is the attribute that is supposed to have the highest impact. At the same time, it is the attribute with the biggest gap between expected and perceived (current) level of quality.

7. Applicability of the results

The case-study approach and its results achieved so far (answers to RQ's and the above recommendations for improvement) have shown to be useful for the practitioners at the company. They have

started to assess the potential improvements opportunities, discussed above. The overhead of the approach applied was minimal in that the mining of access log was automated and the survey had brief questions which tool about 20 min to complete.

For the case study company, the initial investigation pointed out that they were missing documents in several areas. Currently they are working on improving their requirements process and coverage. While doing so they are keeping in mind the results from the research tied to out of date information being a key contributor to poor perceived documentation quality [4] and making sure that they identify the user of the information and ensure that the requirements have an owner that is responsible for their upkeep. In addition, they were looking at the research associated with costing as well. Currently, they are trying to ensure that each section of requirements has a single focus and to ensure that all features are broken out separately and assigned to a particular owner. Having documents that are too large and covering many topics was identified as a key cost factor.

All other research outcomes are influencing the decision making process toward improving the overall process. Highlights include: identifying the audience for the document and its lifetime (during design, testing, maintenance or just one of them, etc.) and identifying the high level documentation tree and creating smaller subordinate documents underneath for specific topics so each document can be as small as possible (modular documentation).

The applicability of the approach in other industrial contexts will be discussed in the external validity aspects (Section 8.2).

8. Threats to validity

We discuss in the following four types of threats to validity with the classification and terminology following [37].

8.1. Construct validity

The construct validity issue in this case study is related to what extent the selected documentation usage and usefulness 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 usefulness attributes. Similar to the general concept of software quality, documentation usefulness is a multi-faceted phenomenon and should be measured by a portfolio of related metrics. We addressed this potential threat to the validity by incorporating quality attributes taken from the literature, e.g. [15,36].

Finally, we understand that our CEI formula only concerns the document usage logs that are the outcome, and not the process of how developers or maintainers use documents. It does not concern how the readers read the documents, whether they found them useful or whether they get the useful information from them.

8.2. External validity

The issue of external validity concerns whether the results of the case study can be generalized beyond this specific study context. External validity is not the focus of the case study. Three issues limit the generalization of the results from this case 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.

Although our industry setting was a typical setting which can be expected to be quite common, it is impossible 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 ability to replicate the findings in this context. To facilitate that, we offered as many details as possible under the given confidentiality constraints.

8.3. Internal validity

Internal validity reflects the extent to which a causal conclusion based on a study is warranted. On the selection of a factor, we utilized the guidance obtained from our systematic mapping study. In addition, from interacting with the case study team, we aimed at applying their practical experience about their impact on usage and usefulness. Still, there is existing risk that other factors (such as pressure from project schedule) have influenced the results on usage of documentation.

8.4. Conclusion validity

Threats to conclusion validity refer to the ability to draw correct conclusions about the relations between the treatment and the outcome of an experiment [37]. Having 25 participants was not easy to achieve, but does not provide sufficient statistical power [38]. The subjects, however, mainly had intermediate or senior level of experience. The threat is somehow mitigated also by the fact these developers in total worked on more than 10,000 documents across the company.

Furthermore, we have not conducted an effect size analysis on the data and results, thus this is another potential threat to the conclusion validity. Finally, we applied the statistical tests to make conclusions based on the data sample sets, even without full approval that all assumptions are fulfilled. We found this approach reasonable for the nature of (case study) research we have performed.

9. Conclusion and future work

We presented a case study approach to evaluate documentation usage and usefulness. The proposed approach consists of a combination of quantitative and qualitative evaluations of technical documentation usage and usefulness. The proposed approach was evaluated using an industrial case study, conducted in the context of a large corporation developing embedded GNSS software systems. We statistically validated a number of hypotheses which were studied based on conducting surveys of software professionals.

In the context of the case study, we found that (1) source code was considered most frequently as documentation source during maintenance tasks, (2) technical documentation was considered least frequently for use during maintenance activities and most frequently as an information source for development activities, (3) there is no significant difference between the usage of various documentation types being requirement, design, code comments, test and process documents during both development and maintenance, (4) low impact of readability on the usage of technical documentation, and (5) some initial hypotheses stating that up-to-datedness and preciseness have the highest impact on usefulness of technical documentation. In addition, from exploratory analysis on documentation usefulness, we formulated new hypotheses to be studied in future work.

The proposed approach can be utilized and applied to other software development companies. The proposed suggestions and guidelines are expected to be useful in improving the 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 the software development and maintenance phases. In addition, they will be able to compare the existing documentation usefulness attributes with the expected levels.

Future work is suggested also in doing analysis based on additional characteristics of the:

- Technical documents (such as size, level of formality applied),
- Software products being considered (degree of legacy, target quality criteria, size and complexity of the document), and
- Underlying development and organizational processes (in terms of their level of maturity).

We expect more specific results for the combination of instances of these characteristics. Another rather unexplored area of research is related to reuse of documentation. To what degree is reuse planned for and how much of that is actually applied?

While useful for the case study company and conclusive in terms of formal hypothesis testing, we consider the results as one step in the long way to increase the level of evidence in understanding the usage and usefulness of technical documentation. Different types of empirical investigations for transitional introduction of improvements may be used, e.g., pilot projects and controlled experiments. We are planning to conduct pilot projects in the context of small teams in the company and apply each of the above improvement opportunities and assess the benefits of each solution. This will complete the steps of our improving case-study research.

Acknowledgements

First of all, we would like to sincerely thank all the software engineers at NovAtel who participated in the survey and for their continuous support during this project. This work was supported by the NSERC CRD Grant #CRDPJ414157-11, and NSERC ENGAGE Grant #EGP-413039. Vahid Garousi was also additionally supported by Atilim University and the Visiting Scientist Fellowship Program (#2221) of the Scientific and Technological Research Council of Turkey (TÜBİTAK). We appreciate the technical support provided by Kornelia Streb in visualizing the analysis results. Finally, the authors would like to thank the anonymous reviewers and the special issue editors for their valuable comments.

Appendix A

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	☐ Senior Developer
☐ Entry Designer (Architect)	☐ Principle Developer
☐ Intermediate Designer (Architect)	☐ Entry Tester/QA
☐ Senior Designer (Architect)	☐ Intermediate Tester/QA
☐ Principle Designer (Architect)	☐ Senior Tester/QA
☐ Entry Developer	☐ Principal Tester/QA
☐ Intermediate Developer	☐ 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 (corrective maintenance) or adding a feature (enhancive maintenance)), 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 are the following items 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				

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

11. Please provide any descriptive comment about the above issues that you think we should know of.

References

[1] D.L. Parnas, Precise documentation: the key to better software, in: S. Nanz (Ed.), The Future of Software Engineering, Springer, 2011, pp. 125–138.

[2] S. de Souza, N. Anquetil, K.M.d. Oliveira, Which documentation for software maintenance?, J Braz. Comput. Soc. 12 (2006) 31–44.

[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] G. Garousi, V. Garousi, M. Moussavi, G. Ruhe, B. Smith, Evaluating Usage and quality of technical software documentation: an empirical study, in: Proceedings EASE'13, 2013, pp. 24–35.

[5] J. Zhi, V. Garousi, B. Sun, G. Garousi, S. Shahnewaz, G. Ruhe, Cost, benefits and quality of software development documentation: a systematic mapping (under second revision), J. Syst. Software.

[6] J. Zhi, V. Garousi, B. Sun, G. Garousi, S. Shahnewaz, 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: April 2014).
[7] W. Ding, P. Liang, A. Tang, H.v. Vliet, Knowledge-based approaches in software documentation: a systematic literature review, Inf. Softw. Technol. 56 (2014) 545–567.
[8] A. von Mayrhauser, Program comprehension during software maintenance and evolution, IEEE Comput. 28 (1995) 44–55.
[9] H. Schoonewille, W. Heijstek, M.R.V. Chaudron, T. Kühn, A cognitive perspective on developer comprehension of software design documentation, in: ACM Conference on Design of communication (SIGDOC), New York, 2011.
[10] A. Tang, M.A. Babar, I. Gorton, J. Han, A survey of the use and documentation of architecture design rationale, in: Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA), Washington, DC, 2005.
[11] B. Curtis, S.B. Sheppard, E. Kruesi-Bailey, J. Bailey, D.A. Boehm-Davis, Experimental evaluation of software documentation formats, J. Syst. Software Arch. 9 (1989) 167–207.
[12] C.J. Stettina, 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, 2011.
[13] S. Das, W. Lutters, C.B. Seaman, Understanding documentation value in software maintenance, in: Proceedings of the 2007 Symposium on Computer Human Interaction for the Management of Information Technology, 2007.
[14] J.-Y. Mao, I. Benbasat, The use of explanations in knowledge-based systems: cognitive perspectives and a process-tracing analysis, J. Manage. Inform. Syst. 17 (2000) 153–180.
[15] M. Visconti, C.R. Cook, Assessing the state of software documentation practices, in: Proc. Conf. on Product Focused Software Process Improvement, 2004, pp. 485–496.
[16] E. Arisholm, L.C. Briand, S.E. Hove, Y. Labiche, The impact of UML documentation on software maintenance: an experimental evaluation, IEEE Trans. Software Eng. 32 (2006) 365–381.
[17] R.Y. Wang, D.M. Strong, Beyond accuracy: what data quality means to data consumers, J. Manage. Inf. Syst. 12 (1996) 5–34.
[18] M. John, F. Maurer, B. Tessem, Human and social factors of software engineering: workshop summary, ACM SIGSOFT Software Eng. 30 (2005) 1–6.
[19] A. Forward, T.C. Lethbridge, The relevance of software documentation, tools and technologies: a survey, in: Proceedings of the 2002 ACM Symposium on Document Engineering, 2002, pp. 26–33.
[20] S. de Souza, N. Anquetil, K.M. Oliveira, A study of the documentation essential to software maintenance, in: Proceedings of International Conference on Design of Communication, 2005, pp. 68–75.
[21] M. Kajko-Mattsson, A survey of documentation practice within corrective maintenance, Emp. Software Eng. 10 (2005) 31–55.
[22] D. Falessi, L.C. Briand, G. Cantone, R. Capilla, P. Kruchten, The value of design rationale information, ACM Trans. Softw. Eng. Methodol. 22 (2013) 1–32.
[23] D.E. Perry, N.A. Staudenmayer, L.G. Votta, People, organizations, and process improvement, IEEE Softw. 11 (1994) 36–45.
[24] I. Sanchez-Rosado, P. Rodrguez-Soria, B. Martn-Herrera, J. JosCuadrado-Gallego, J. Martínez-Herráiz, A. González, Assessing the documentation development effort in software projects, in: International Conferences on Software Process and Product Measurement, 2009.
[25] B. Sun, A Methodology for Analyzing Cost and Cost-Drivers of Technical Software Documentation (Master's Thesis), Department of Computer Science, University of Calgary, 2012.
[26] S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, P. Grünbacher, Value-Based Software Engineering, Springer, 2006.
[27] T. Lethbridge, A. Forward, How software engineers use documentation: the state of the practice, IEEE Softw. (2003) 35–39.
[28] W.J. Dzidek, E. Arisholm, L.C. Briand, A realistic empirical evaluation of the costs and benefits of UML in software maintenance, IEEE Trans. Software Eng. 34 (2008) 407–432.
[29] A. Forward, Software Documentation: Building and Maintaining Artefacts of Communication, MSc Thesis, University of Ottawa, 2002.
[30] O. Laitenberger, M. Ciolkowski, S. Vegas, S. Biffl, Practical experiences in the design and conduct of surveys in empirical software engineering, Empirical Methods Stud. Software Eng. (2003) 104–128.
[31] V. Garousi, T. Varma, A replicated survey of software testing practices in the canadian province of Alberta: what has changed from 2004 to 2009?, J Syst. Softw. 83 (2010) (2004) 2251–2262.
[32] E. Tryggeseth, Report from an experiment: impact of documentation on maintenance, Emp. Software Eng. 2 (1997) 201–207.
[33] R. Sibson, SLINK: an optimally efficient algorithm for the single-link cluster method, Comput. J. 16 (1973) 30–34.
[34] S. Tilley, S. Huang, A qualitative assessment of the efficacy of UML diagrams as a form of graphical documentation in aiding program understanding, Proc. Int. Conf. Doc. (2003) 184–191.
[35] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical Software Eng. 14 (2009) 131–164.
[36] A. Wingkvist, M. Ericsson, R. Lincke, W. Lowe, A metrics-based approach to technical documentation quality, Proc. Int. Conf. Q. Inform. Commun. Technol. (2010) 476–481.
[37] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, Experimentation in Software Engineering, Springer, 2012.
[38] J. Cohen, Statistical Power Analysis for the Behavioral Sciences, second ed., Lawrence Erlbaum Associates, 1988.
[39] D. Falessi, M.A. Babar, G. Cantone, P. Kruchten, Applying empirical software engineering to software architecture: challenges and lessons learned, Empirical Software Eng. 15 (2010) 250–276.