The Journal of Systems and Software 99 (2015) 175–198

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Cost, benefits and quality of software development documentation: A systematic mapping

Junji Zhi [a], Vahid Garousi-Yusifoğlu [b,c,*], Bo Sun [d,e], Golara Garousi [c,f], Shawn Shahnewaz [c], Guenther Ruhe [c,d]

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

ARTICLE INFO

Article history:
Received 5 September 2012
Received in revised form
28 September 2014
Accepted 28 September 2014
Available online 22 October 2014

Keywords:
Software documentation
Documentation benefit
Systematic mapping

ABSTRACT

Context: Software documentation is an integral part of any software development process. Researchers and practitioners have expressed concerns about costs, benefits and quality of software documentation in practice. On the one hand, there is a lack of a comprehensive model to evaluate the quality of documentation. On the other hand, researchers and practitioners need to assess whether documentation cost outweighs its benefit.
Objectives: In this study, we aim to summarize the existing literature and provide an overview of the field of software documentation cost, benefit and quality.
Method: We use the systematic-mapping methodology to map the existing body of knowledge related to software documentation cost, benefit and quality. To achieve our objectives, 11 Research Questions (RQ) are raised. The primary papers are carefully selected. After applying the inclusion and exclusion criteria, our study pool included a set of 69 papers from 1971 to 2011. A systematic map is developed and refined iteratively.
Results: We present the results of a systematic mapping covering different research aspects related to software documentation cost, benefit and quality (RQ 1–11). Key findings include: (1) validation research papers are dominating (27 papers), followed by solution proposals (21 papers). (2) Most papers (61 out of 69) do not mention the development life-cycle model explicitly. Agile development is only mentioned in 6 papers. (3) Most papers include only one "System under Study" (SUS) which is mostly academic prototype. The average number of participants in survey-based papers is 106, the highest one having approximately 1000 participants. (4) In terms of focus of papers, 50 papers focused on documentation quality, followed by 37 papers on benefit, and 12 papers on documentation cost. (5) The quality attributes of documentation that appear in most papers are, in order: completeness, consistency and accessibility. Additionally, improved meta-models for documentation cost, benefit and quality are also presented. Furthermore, we have created an online paper repository of the primary papers analyzed and mapped during this study.
Conclusion: Our study results show that this research area is emerging but far from mature. Firstly, 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 evidences are still needed to enhance our understanding of this area. In particular, what we expect includes (1) more validation or evaluation studies; (2) studies involving large-scale development projects, or from large number of study participants of various organizations; (3) more industry-academia collaborations; (4) more estimation models or methods to assess documentation quality, benefit and, especially, cost.

* Corresponding author at: System and Software Quality Engineering Research Group (SySoQual), Department of Software Engineering, Atilim University, İncek, Ankara, Turkey. Tel.: +90 (312) 586-8774.
E-mail addresses: zhij@cs.toronto.edu (J. Zhi), vahid.garousi@atilim.edu.tr (V. Garousi-Yusifoğlu), sbo@ucalgary.ca (B. Sun), golara.garousi@gmail.com (G. Garousi), smshahne@ucalgary.ca (S. Shahnewaz), ruhe@ucalgary.ca (G. Ruhe).

http://dx.doi.org/10.1016/j.jss.2014.09.042
0164-1212/

1. Introduction

Software documentation is an integral part of any software development process (Bayer and Muthig, 2006). In fact, software documentation has become a popular sub-domain in software engineering (Cook and Visconti, 1994) to the extent that there are special interest groups such as the ACM Special Interest Group on Design of Communication (SIGDOC).

A literature search in the beginning of this study (Fall 2013) yielded 500+ papers on software documentation. A large portion of this set proposes various types of documentation management systems or formats. Another portion of the paper set focuses on cost, benefits and quality of documentation, the subjects which we focus on in this study.

In our study, we target the documents that are software development related. We call them *technical* and refer to those documents that (1) are produced during the software development lifecycle and (2) whose target audience(s) are software developer(s). The types of documents within the scope of our investigation typically include requirement, design, implementation and test documents as well as code comments. Product or user manuals may also be produced during development lifecycle, but are excluded in our investigation because it violates the second criterion, i.e., their target audiences are not software developers. We define the term *cost* as the value of effort or time that has been used to produce a software artifact (e.g., code, or documentation).

A considerable share of software projects' costs are spent on documentation, e.g., a ratio of 11% was reported in (Sanchez-Rosado et al., 2009). This indicates that the effort consumed in documentation is one significant cost drivers during software development processes. It is natural and expected that, when cost is spent in developing an artifact, that artifact should be used and provides *benefit* at some point in the development or maintenance phase (Mira, 2005; Tilley and Huang, 2003; Lethbridge et al., 2003). The benefits could be reflected in many aspects, e.g., shortened task duration, improved code quality, higher productivity, or any other improvements related to software development. In terms of documentation *quality*, we define it as the character of documents with respect to fineness which is often influenced by how much time/effort is spent on and affects the benefits practitioners get from the documents. Therefore, the aspect of document quality is also included in our scope of study.

On the other hand, the traditional view of software documentation is undergoing the challenge of Agile development methods (Ambler, 2011; Rubin and Rubin, 2011; Rueping, 2003; Stettina and Heijstek, 2011). As the *Agile manifesto* (Beck et al., 2012) points out: "Working software [is valued] over comprehensive documentation". The manifesto also mentions that, while there is value in the items on the right (i.e., documentation), we value the items on the left (i.e., working software) more. Does this mean documentation is no longer important (Stettina and Heijstek, 2011)? Practitioners start to question whether the cost of creating and maintaining documentation outweighs its potential benefit (Rueping, 2003; Stettina and Heijstek, 2011). To answer such a question, one needs to be able to quantitatively measure the cost and benefit of documentation.

During the past three to four decades, researchers, in increasing numbers, have proposed different techniques for analyzing cost, benefit and quality of documentation. As the research area matures and the number of related papers increases, we feel it is important to summarize the current state-of-the-art and provide an overview of the trends in this specialized field. To address that goal, we present in this paper a *systematic mapping* of the literature in this area.

According to Petersen et al. (2008), a systematic mapping (SM) is a method to review, classify, and structure papers related to a specific research field in software engineering. According to Kitchenham et al. (2011): "mapping papers can save time and effort for researchers and provide baselines to assist new research efforts". The goal is to obtain an overview of existing approaches, outlining the coverage of the research field in different facets of the classification scheme that we develop in this paper. Identified gaps in the field serve as a valuable basis for future research directions. Using an empirical study, Kitchenham et al. (2010) reported that SM papers also have educational values and would provide young researchers and students with useful and transferable research skills and are a useful first step for postgraduate PhD candidates.

Unlike a Systematic Literature Review (SLR) (Kitchenham and Charters, 2007), finding evidence for impact of a proposed approach is not the main focus in a systematic mapping (Petersen et al., 2008). However, the two methods have many overlaps and the results of a systematic mapping can be fed into a more rigorous systematic review study to support evidence-based software engineering (Kitchenham and Charters, 2007).

Systematic mapping papers generally consist of five steps including: (1) a definition of research questions, (2) conducting the search for relevant papers, (3) screening of papers, (4) key-wording of abstracts, and (5) data extraction and mapping (Petersen et al., 2008), which we follow in this paper.

As far as we are concerned, we have not been able to find any study to synthesize or to systematically map the existing papers on software documentation cost, benefit and quality. Our study aims to survey the existing literature for purpose of identifying research trends. We hope that this paper contributes a summary of the area that could be useful for follow-up future papers. Also, the need for this SM was motivated in the context of a multi-year industrial collaborative research and development project in which the authors are involved in, which aims to minimize the cost and amount of documentation across the software development life-cycle for one of our industrial partners.

The main questions we intend to answer in this study are:

(1) How do researchers assess the quality of documentation?
(2) What are the cost-related attributes of software documentation?
(3) What benefit does documentation bring to software practitioners?

During our SM study, we have extracted the attributes or metrics to measure these three aspects. For document quality aspect, we extracted more than 13 attributes that cover different aspects of document quality, including up-to-date-ness, completeness, etc. For benefit aspect, we also gathered three main categories (e.g., development aid, maintenance aid, etc.) and two metrics (e.g., task time reduction, etc.). In terms of document cost, we also extracted two main categories (i.e., production or maintenance cost, etc.) and one quantitative metric (i.e., document size). The results are presented in detail in Sections 6.8–6.10.

The main contributions of this paper are two-fold:

- A unified meta-model for documentation quality incorporating and consolidating all the individual and partial parts proposed by previous researchers, and also a meta-model for documentation usage process and benefit (Section 5.2).
- A systematic map (Section 5) developed for the area of documentation cost, benefit, and quality and consequently the systematic mapping of the existing research in this area (Section 6).

Also, we published an online paper repository which has been created during this systematic study (Zhi et al., 2012). Future researchers or practitioners can find related works in the area

of software documentation cost, benefit and quality by using our repository.

The remainder of paper is organized as followed. Section 2 discusses background and related work. In Section 3, we describe our research method, including the overall SM process, the goal and research questions tackled in this study. Section 4 discusses in detail the paper selection process. Section 5 presents the systematic map which has been built through an iterative selection and synthesis process. Section 6 presents the results of the systematic mapping. Validity aspects of this study are discussed in Section 7. Finally, Section 8 summarizes and concludes this study and states the future work directions. The reference section at the end of the paper is divided into two parts: primary papers of the systematic mapping are listed first and then the other references used in this study.

2. Background and related work

Before presenting our SM study and results, we discuss in this section the following background information and related work:

- What is software documentation?
- Systematic mappings and literature-review studies in software engineering.
- Existing work in formalizing software documentation.

2.1. What is software documentation?

In the thesis of Andrew Forward (Forward, 2002), software documentation is defined as "an artifact whose purpose is to communicate information about the software system to which it belongs". In this definition, documentation is stressed for the usage of communication among software engineers. Parnas (2011) defines *document* as "a written description that has an official status or authority and may be used as evidence". Such description is expected to provide precise information about the systems. Parnas points out that the word *document* is used very generally, referring to any information accompanying the software, even if it is imprecise or incomplete. Parnas also classified documents into different types for different purposes. For example, specifications in the forms of "assertions or program functions" may be useful for developers, but might not be significant for other users. One example of "other users" is testers who may only concern the black-box functionalities of the program rather than the detailed implementation. This clarification is helpful for our discussion in this paper since it provides rationale for the attribute scheme to be described in Section 5.

From the above discussion, we can see that the meaning of documentation is many-fold. Thus it is difficult to draw a definition in a single sentence. Instead, we conclude the above-mentioned definitions by listing several aspects of documentation below:

- Documentation is a written description of software systems.
- Documentation is expected to provide precise information about the systems.
- Documentation can refer to the product manual that developers created for non-developer users.
- Development (or technical) documentation is created for the purpose of communication among software engineers.
- Documentation can refer to different artifacts, including requirements, design, code comment, test cases, etc.
- Documentation can be presented in different formats, varying from the traditional written text to graphical models (e.g., those using UML), from static text to dynamic hypertext systems.

Our study is targeting the cost, benefit and quality of documentation in development phases. By 'development phase', we refer to the phases that exist in the process of developing software products, including requirement, design or test, etc. (Pfleeger and Atlee, 2010). As discussed in Section 1, we are only concerned with development (or technical) documentation. Such documentation includes the document types that are commonly used in development phases, such as requirement, design, test, process control documents, etc. In this paper, architecture documentation, which specifies how software systems are structured, is generally considered as a type of design. These types of document either describe the technical details of the system or specify how the system should be built. On the other hand, we excluded those papers only discussing non-development-related documentation (e.g., user manuals).

Some researchers and professionals believe that source code itself can be viewed as one type of documentation. However, in this paper we do not bear this view. We intend to narrow our scope of documentation. Otherwise, the cost or quality of documentation will include the source code quality, which will be too broad to be addressed in one single study.

2.2. Systematic mappings and literature-review studies in software engineering

Research proceeds by learning from and being inspired by existing research works. When a research area grows and owns a large number of existing papers, it requires a substantial effort to read all the literature before conducting new research. Summarizing the existing literature and providing an overview for a certain area is helpful for new researchers (e.g., new Master or PhD students), identifying research trends and shedding lights on future directions.

Like many other research fields, software engineering has its methodologies for conducting secondary studies. Petersen et al. (2008) presented a guideline paper on how to conduct systematic mapping (SM) studies in software engineering. This paper provides insights on building classification schemes and structuring a particular sub-domain of interest in software engineering. The technique described by Petersen et al. was applied in our SM study.

Kitchenham et al. (2009) reported a review on SLR papers in software engineering. 20 relevant papers are collected and analyzed. The authors concluded that the potential value of SLR papers is demonstrated by the series of papers. Yet none of these papers focused on software documentation. As a more 'open' form of SLR (Budgen et al., 2008), SM study method also attracts attention among some researchers. Budgen et al. (2008) conducted an informal review on the SM papers on software engineering and reported a summary of these papers. In total, six SM papers were examined and studied. We also browsed the Software Engineering Evidence Map (Software Engineering Evidence Map, 2012) which listed 41 SLR or SM papers. The areas that these papers cover include Software Testing (Jia and Harman, 2011, 2012; Zhang, 2012), Software Engineering Management, etc. Again, none of these papers concentrate their discussion on documentation issues. Hopefully, our SM study in this paper may address this gap and to lay a foundation for more comprehensive secondary studies, such as Systematic Literature Review (SLR) in the future in the area of software documentation.

2.3. Existing work in formalizing software documentation

Researchers have formalized the documentation process, cost, benefit and quality attributes. We were able to find a few existing works (Arthur and Stevens, 1989; Meng et al., 2006; Palazzolo and Utt, 2000; Visconti and Cook, 1993) in this subject. In this paper, we present a formalized view on the documentation process, cost,

benefit and quality (Section 5.2). We discuss below the existing work and also how our extended models differ from the previous work.

Arthur and Stevens (1989) reported a case study on assessing the adequacy of project documentation based on a taxonomic structure for documentation characteristics. They defined Document Quality Indicators (DQI) to decompose the quality into factor level and form a hierarchical model. Based on the characteristics they collected, our proposed model incorporates other attributes that are not included in the Arthur and Stevens' model, such as up-to-dateness and content duplication. In other words, our work offers a more comprehensive model of document quality characteristics.

Visconti and Cook (1993) proposed a 4-level Documentation Process Maturity model. Their model aims to evaluate the document process maturity and to address the low quality issues of documentation. In their model, each level is identified with a name, keywords, key process areas, key practice, key indicators, etc. They also proposed an intermediate set of goals toward higher levels of process maturity. However, their maturity model does not solve the problem of measuring the document quality directly. Our study, in contrast, decomposes the document quality concept into different attributes and lays a foundation for further concrete, quantitative measurement of document quality.

Palazzolo and Utt (2000) integrated the documentation development process into the Rational Unified Process (RUP). In their model, they identified the workers (e.g., technical writers, information architects), the artifacts (e.g., concept, task and documentation) and the documentation development work-flow. While they focus their discussion on the process of developing documentation, our proposed model emphasizes the relationships among the entities in the process: software personnel, tasks and documentation, so as to investigate the cost or benefit of software documentation.

Meng et al. (2006) proposed a software comprehension model where document is modeled as "descriptions of the documented artifacts". The authors aim to use such a process model to describe comprehension tasks in different contexts. Their proposed model is similar to the one which we present (see Fig. 2) in this paper in several aspects: (1) Both models consider tasks as the main activity that motivates software engineers to consult documents; (2) Documents are considered as main information sources in both models. However, since their model concentrates on program comprehension, the authors do not elaborate on and types of documents or types of tasks as we do in this paper. For example, we differentiate tasks into two main categories: maintenance tasks and pre-maintenance tasks, and for each category we include corresponding sub-categories (see Fig. 2). Also, types of documentation artifacts are elaborated in our model. A document can be presented in textual or visual form, or as code comments. Such elaborations are necessary to study the documentation usage process in detail and thus to investigate the benefit or cost of documentation.

3. Research method

In the following, an overview of our research method and then the goal and research questions of our study are presented.

3.1. Overview

This SM is carried out in reference to the guidelines provided by Petersen et al. (2008), and Kitchenham and Charters (2007). In designing the methodology for this SM, methods (e.g., deriving concise research questions and research methodology) from several other SMs such as (Ali et al., 2010; Elberzhager et al., 2012; Garousi, 2012) were also incorporated.

Our SM study process is outlined as a UML activity diagram in Fig. 1. The process consists of several phases (activities) which are described throughout Sections 4–6. The Research Questions (RQs) appearing in Fig. 1 are discussed in the next section.

3.2. Goal and research questions

The goal of this study is to systematically review the state-of-the-art in analyzing cost, benefit, and quality of software documentation within the software development lifecycle (SDLC), to classify the papers in this area and to find out the recent trends in this field from the perspective of researchers and practitioners.

In simpler terms, the goal is to understand the software documentation research with regard to the following three attributes: cost, benefit, and quality. We are also interested in understanding how the research area has evolved over time with regards to those attributes.

Based on the above goal, the following research questions are raised. Main topics, question bodies and rationale for each research question (RQ) are presented as below. Classifications used to answer each of the questions are discussed in Section 5.

- **RQ 1 – Number of papers by research facet**: What type of research methods are used in the papers? A paper could be simply a solution proposal, while other papers take experimental research approaches (with various levels of rigor) (Petersen et al., 2008). The rationale behind this RQ is that knowing the breakdown of the research area with respect to (w.r.t.) research facet types will provide us with the maturity of the field and papers in using empirical approaches.
- **RQ 2 – Number of papers by contribution facet:** What types of contributions are made by the papers? How many papers present documentation methods/techniques, tools, models, metrics, or processes? Knowing the breakdown of the papers w.r.t. contribution facet types will provide us with the list of each contribution type (e.g., documentation tools presented in the previous papers) which could be used in follow-up papers.
- **RQ 3 – Types of development life-cycle model:** In what type of development life-cycle model (e.g., waterfall or Agile) is the documentation applied? Knowing the breakdown of the papers with respect to development phases in which documentation is used will provide us with the ratio of the processes using or studying documentation.
- **RQ 4 – Type of artifact:** What are the artifact types for which documentation is made? Artifacts types may include requirement, design, test, process and code. The answer to this RQ can help us understand for what purposes most documentation is developed.
- **RQ 5 – Documentation formats:** In what format is the documentation presented? Possible formats include formatted text, models (e.g., UML), code comments, specific tools, etc. Addressing this RQ will help us gain knowledge about the type of formats most documentation is expressed.
- **RQ 6 – Objects under study:** What are the attributes of the objects under study? Some papers have studied the documentation issues in the context of case-papers, while others have involved participants in surveys to solicit their input in studying the documentation issues. This question intends to analyze the attributes of those objects, e.g., size and scale of case study systems, and number of participants involved in surveys. The rationale behind this RQ is to characterize the size and scale of the case papers in papers which have involved software systems or survey participants.

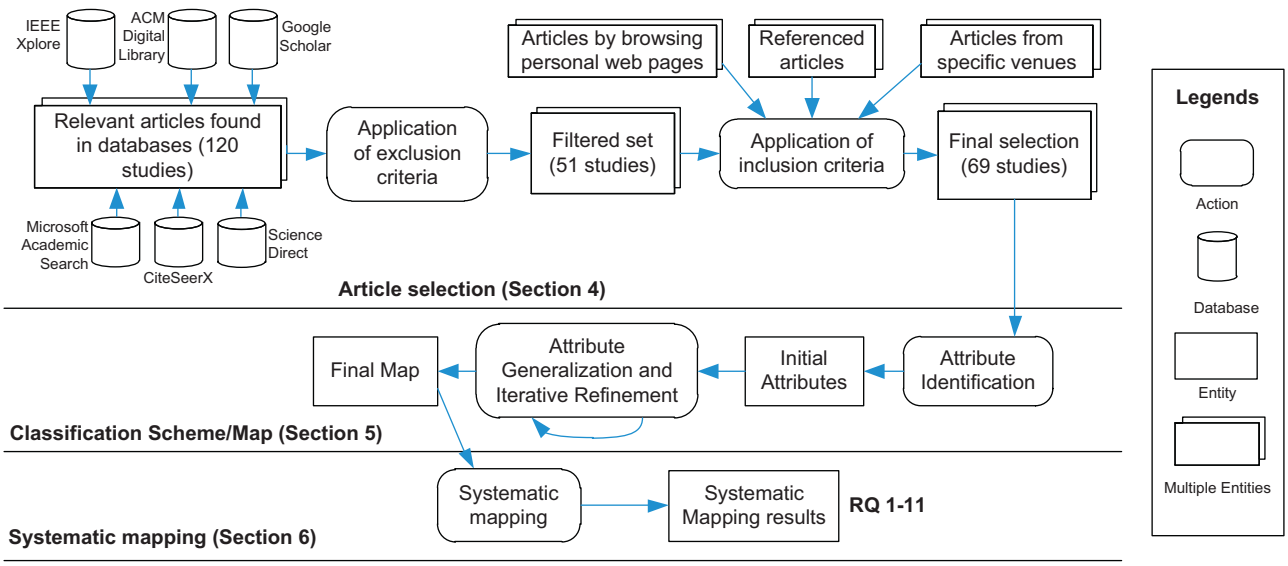


Fig. 1. Our research process.

- **RQ 7 – Focus of papers:** Among all papers devoted to documentation, what is the percentage of papers that focus on these aspects of interest: cost, benefit and quality? Addressing this RQ will reveal which of the above aspects have received most of the attention in the community, thus, enabling us to find the less-developed areas for conducting new research.
- **RQ 8 – Software documentation cost attributes and metrics:** What attributes and metrics related to the cost of documentation have been studied? Addressing this RQ will provide the list of most-popular attributes and metrics, along with the level of research activity on each, which could be used for future papers.
- **RQ 9 – Software documentation usage and benefit attributes and metrics:** What attributes and metrics related to the usage/benefit of documentation have been studied? Addressing this RQ will provide same benefits as RQ 8.
- **RQ 10 – Software documentation quality attributes and metrics:** What attributes and metrics related to the quality of documentation have been studied? Addressing this RQ will provide same benefits as RQ 8 and RQ 9.
- **RQ 11 – Industry's involvement:** What are industry's involvement related to software documentation cost, benefit and quality? Addressing this RQ will assess the degree of software industry's involvement in and its contribution to the body of knowledge in this field.

4. Paper selection

Recall from Fig. 1 that the first phase of our SM process is selection of relevant papers. For the paper selection phase of our SM, we explain the following steps in order:

- Source selection and search keywords (Section 4.1).
- Exclusion criteria (Section 4.2).
- Inclusion criteria (Section 4.3).
- Final pool of papers and the online repository (Section 4.4).

4.1. Source selection and search keywords

To find the relevant primary papers, we searched the following six major online search academic paper search engines: (1) IEEE Xplore,[1] (2) ACM Digital Library,[2] (3) Google Scholar,[3] (4) Microsoft Academic Search,[4] (5) CiteSeerX,[5] and (6) Science Direct.[6]

To ensure that we include as many relevant papers as possible in the pool of selected papers, we followed a rigorous procedure to construct the search string. First, we searched the *IEEE Standard Glossary of Software Engineering Terminology* (IEEE Standard 610.12-1990) for the definitions of *document* and *documentation*. Based on the definitions of those terms, we identified the following keywords: *project plan, specification, test plan, user manual, technical documentation, design documentation, architecture document, architecture documentation, requirement document*, and *design document*. Since *user manual* is not our focus in this study, we excluded this term. These terms form our noun set S_1. Next, since our aspects of interest are *cost*, *benefit*, and *quality*, which forms our second set of terms, S_2. Also, in order to narrow our results related to software or computer systems, we identified a set of modifiers S_3 including *software*, *computer*, and *system*. Finally, each search string is constructed with pattern $\{String = s_3 s_1 s_2 \mid s_1 \in S_1,\ s_2 \in S_2, s_3 \in S_3\}$. For example, the following search strings were generated with this method: *software document cost*, *software document benefit*, and *software document quality*.

In our search phase, we benefited from the guidelines presented by Zhang et al. (2011). With these search strings, we found 120 papers as our initial pool of potentially-relevant papers. Only papers written in English language were considered. Most papers' full-text PDF files were electronically available. For those not available online, we ordered them via the University of Calgary's Interlibrary Loan system. We received a dozen of those cases, but five papers (Bayer, 2004; Comer, 1983; Prause et al., 2007; Robles et al., 2006; Tausworthe, 1976) could not be accessed even after we ordered them. Finally, only papers published/available by the end of March 2012 were included in our pool.

[1] http://ieeexplore.ieee.org.
[2] http://dl.acm.org.
[3] http://scholar.google.com.
[4] http://academic.research.microsoft.com.
[5] http://citeseerx.ist.psu.edu.
[6] http://www.sciencedirect.com.

4.2. Exclusion of unrelated papers

The next step in our paper selection process was to exclude the irrelevant papers. Specifically, our focus was on papers that discussed cost, benefit and quality of software documentation applied during any software development phase (e.g., requirement, design, etc.).

The study s will be excluded if the report or paper p of s meets all the criteria as below:

- p has no content that explicitly discusses any known form of cost related to technical software documentation, which can use (1) quantitative metrics such as money, time, or (2) subjective metrics, such as survey responses, perceived mental efforts, etc. or (3) any other metrics associated with costs, such as document size;
- p has no content that explicitly discusses any known form of benefit related to technical software documentation, which can use (1) quantitative metrics such as money or time, or (2) subjective metrics, such as survey responses which are commonly seen in empirical studies to provide subjective measurements from the perspective of practitioners, (3) any other metrics associated with documentation benefit;
- p has no content that explicitly discusses any known form of quality aspects related to technical software documentation. The target quality aspects can be any characteristic related to documents that are discussed in the context of software development, such as document structure and document's up-to-date-ness.

To apply this exclusion criterion to the initial pool (120 papers), a voting phase was conducted among the first four authors of this paper. They inspected the papers in the initial pool and assigned a vote on a 9-point scale to each paper, with '9' indicating a strong opinion in favor of including a paper, and '1' indicating a strong opinion in favor of excluding a paper. Thus, the maximum vote on a paper could be 36 marks. We decided to use a threshold of average score of 5 out of 9, i.e., vote value of $4 \times 5 = 20$ in total, for the decision on paper inclusion/exclusion.

For each paper inspected, we reviewed its title, abstract and keywords. If a vote could not be made based on this information, a more in-depth evaluation was conducted. In case of a wide variance value among the votes for a study, authors carefully discussed such cases to ensure quality of the votes. Based on the results of the joint voting, the size of the pool of selected papers decreased from 120 to 51. The excluded papers are Alan and Roger (1983), Alberto et al. (2005), Ambler (2002), Ashley (2004), Azmi and Ibrahim (2011), Bauer and Parnas (1995), Benson and Tom (1988), Berglund (2000), Berglund and Priestley (2001), Bethke et al. (1991), Bill et al. (1988), Thomas and Tilley (2001), Clark et al. (1989), Clear (2003), Clements et al. (2003), Courtois and Parnas (1993), Parnas et al. (1988), Parnas (2005), Delanghe (2000), Forward and Lethbridge (2014), French et al. (1997), Galdo et al. (1986), Gerth et al. (2014), Guillemette (1986), Haneef (1998), Hargis (2000), Hargis et al. (2004), Horowitz and Williamson (1985), Hyland-Wood et al. (2008), Jazzar and Scacchi (1995), Johnson and Rey (1994), Kalus and Torsten (1996), Landes et al. (1999), Landis et al. (1988), Liu (2001), Lobato et al. (2010), Luqi et al. (2003), Lutsky et al. (2000), Magyar (2000), Marcus and Maletic (2005), Marovac (1994), Marovac (1998), Medina (1984), Van der Meij (2000), Metzger et al. (2007), Moallem (2003), MacKinnon and Murphy (2004), Odenthal and Quibeldey-Cirkel (1997), Peters and Parnas (1998), Phoha (1997), Powell et al. (1996), Price (1984), Price and Korman (1993), Rob (2003), de Rosis et al. (1999), Ruth (1997), Sabou (2004), Tilley (1993), Tang et al. (2011), Taulavuori et al. (2004), Visconti and Cook (2002), Walters and Beck (1992), Wang et al. (2009), Wingkvist et al. (2010b), Wise (1993), Briand (2003)

4.3. Inclusion of additional papers

To minimize the risk of missing relevant papers, similar to previous SM papers and SLRs, we included additional papers manually via:

- Personal web pages of active researchers in the field of interest.
- References found in papers already in the pool.
- Specific venues.

To identify the additional personal web pages, we ranked the author names by the number of times they appear in the author list. The authors ranked top 10% were identified as active authors and then we browsed their personal webpages to look for additional papers.

Also, we examined the reference lists of the 51 papers that we obtained in the search phase and looked for new relevant papers. We did so to include those relevant papers that we might have missed when using search engines.

To identify the specific venues, we ranked the venues by the number of papers belonging to them. The top 10% venues are International Conference on Design of communication (SIGDOC), International Conference on Software Maintenance (ICSM), Journal of Systems and Software (JSS) and IEEE Transactions on Software Engineering (TSE). However, we found that the only venue focusing on the area of software documentation was the SIGDOC.

All papers found in the additional venues that were not yet in the pool of selected papers but seemed to be candidates for inclusion were fed into the voting mechanism described in Section 4.2, and went through the same procedure as explained above. After this phase, we believe that we have included the relevant papers that we missed using search engines. By *relevant papers*, we refer to those papers that have a certain number of citations and thus should appear in the reference list of at least one of included papers.

4.4. Final pool of papers and the online repository

After the above-mentioned phrases, the pool was finalized with 69 papers (Tryggeseth, 1997; Sametinger, 1991; Forward, 2002; Dzidek et al., 2008; Arisholm et al., 2006; Soloway et al., 1988; Schreck et al., 2007; Sheppard et al., 1981; Aguiar and David, 2005; Ambler, 2011; Arthur et al., 1988; Arthur and Stevens, 1989; Bachmann et al., 2000; Bayer and Muthig, 2006; Blum, 1988; de Boer and van Vliet, 2009; Cioch et al., 1996; Cook and Visconti, 1994; Correia, 2008; Correia et al., 2009; Curtis et al., 1989; Das et al., 2007; Dautovic et al., 2011; Ericsson et al., 2011; Fletton and Munro, 1988; Huang and Tilley, 2003; Jansen et al., 2009, 2008; Kanter et al., 2008; Katzenelson, 1971; Kylmkoski, 2003; Lehner, 1993; Maarek and Benyfj, 1989; MacKinnon and Murphy, 2003; Meng et al., 2006; Mira, 2005; Palazzolo and Utt, 2000; Parnas, 2011; Parnas et al., 1994; Parnas and Vilkomir, 2007; Pemberton et al., 1996; Pendharkar and Rodger, 2002; Poshyvanyk and Marcus, 2007; Robert, 1985; Rubin and Rubin, 2011; Rueping, 2003; Sametinger, 1994; Sanchez-Rosado et al., 2009; Schoeffel, 1976; Schoonewille et al., 2011; Tilley and Huang, 2003; Sheppard et al., 1982; Sommerville, 2001; de Souza et al., 2005, 2006; Stettina and Heijstek, 2011; Su, 2010; Tang et al., 2005; Tilley, 2009; Tilley and Hausi, 1991; Tilley et al., 1992; Lethbridge et al., 2003; Trese and Tilley, 2007; Visconti and Cook, 1993, 2004; Wayne and Carolyn, 2007; Wingkvist et al., 2010a, 2011; Wong and Tilley, 2002). The reader can refer to full reference list of all 69 primary papers. The final pool of selected papers has been published as an online repository using Google Docs. It has been analyzed through our systematic

mapping study, and is accessible publically online (Zhi et al., 2012). We plan to update the online repository at least once a year in the future and to add new relevant papers as published.

5. Development of the systematic map

Iterative development of our systematic map (classification scheme) is discussed in Section 5.1. Section 5.2 presents the meta-models summarizing and formalizing the concepts related to documentation cost, benefit and quality. Those meta-models have helped us to carefully develop our final systematic map, as reported in Section 5.3. Section 5.4 presents our data extraction approach.

5.1. Iterative map development

To develop our systematic map, we followed the process described in Fig. 1. We analyzed the papers in the pool and identified the initial list of attributes related to cost, benefit and quality of software documentation. We used iterative refinement to derive the final map. To increase the preciseness of our classification scheme, we utilized the "observer triangulation" method (Runeson et al., 2012) in designing the systematic map.

We recorded the primary papers in a shared spreadsheet hosted at online Google Docs system to facilitate further analysis. The following information was recorded for each paper: (1) paper title, (3) authors, (2) paper venue, (4) year of paper, (5) authors' country of affiliation and (6) authors affiliation (i.e., government, academia, industry or a combination).

With the relevant papers identified and recorded, our next step was to categorize the papers in order to begin building a complete picture of the research that has been conducted to investigate cost, benefit and quality of documentation. Though we did not a priori develop a categorization scheme for this project, we were broadly interested in the attributes or metrics related to cost, benefit and quality of software documentation reported on by the papers.

We refined these broad interests into a categorization scheme using an iterative approach that involved all six of the authors of this paper. The first author of this paper conducted an initial pass over the data, and based on (at least) the title, abstract and introduction of the papers created a set of initial categories and assigned papers to those categories. As a group, we then discussed and reviewed the results of this first analytic pass and refined the categorization. Next the rest of the researchers conducted a second pass over the data, to revisit the categorization. When the assignment of papers to categories could not be clearly determined just based on the title, abstract and introduction, further details of the paper were considered. In this process, both the categories and the assignment of papers to categories, were further discussed and refined. At the end, every paper was reviewed by at least two researchers.

5.2. Formalizing cost, benefit and quality of documentation

In this section, we present the meta-models incorporating all the extracted attributes related to documentation cost, benefit and quality which were developed during our SM process. As discussed in Section 2.3, our three meta-models consolidate and extend the existing documentation models in the literature (Arthur and Stevens, 1989; Meng et al., 2006; Palazzolo and Utt, 2000; Visconti and Cook, 1993).

The model presented in this section provides a unified description of the documentation process. Also, the framework is helpful to capture and illustrate the concepts that will be discussed in later sections. We view it as an independent contribution of this paper in addition to the SM results.

5.2.1. Documentation cost and benefit

In order to investigate the cost or benefit of documentation, we constructed the relationship model between each entity during the documentation process. In the design of this meta-model, we have benefitted from the UML meta-model infrastructure (UML meta-model Infrastructure specification, 2012). Fig. 2 shows the documentation development usage, benefit and cost meta-model.

In our model, a *software practitioner* needs to perform development (called "pre-maintenance" in Fig. 2) or maintenance tasks. Each *task* consumes certain number of effort units (e.g., man-month) and thus *effort unit* is modeled as an attribute of *task*. These *tasks* are classified into two categories: *pre-maintenance* tasks and *maintenance* tasks. The former category refers to the tasks performed prior to maintenance phase, including *requirement*, *design*, *implementation*, and *testing*. Note that *architecture* is considered a part of *design* and it is not illustrated in Fig. 2.

Maintenance tasks are further categorized using the classification proposed by Lientz et al. (1978): *corrective*, *perfective*, *adaptive* and *preventive* maintenance tasks. In our model, all maintenance tasks consist of two steps: *comprehension* of the program and subsequent *manipulation* or modification. The software practitioner creates or maintains a *document* entity, which is a *task* incurring *effort (cost)*. This is where the cost of documentation is incurred.

While performing the tasks, the *software practitioner* needs the involvement of existing *artifacts*. The usual types of *artifacts* include *requirements*, *design*, *code* and *test suites*. *Documents* are modeled as a subclass of artifact. In terms of the format, documents can be presented in pure *textual* or in combination with *visual* models (e.g., UML models), or in the form of *code comments*.

Besides, the *software practitioner* might also need to communicate with other *team members* and/or consult with documents to get the task done. In other words, documents might serve as a communication aid among developers or maintainers.

During the documentation usage phase, the typical *software practitioner* uses the documentation and may perceive the *benefits* and *quality* of documentation. In other words, documentation benefits come during the usage process. Also, in the usage process, a certain amount of costs are incurred, including the time and efforts of reading documents. Figs. 3 and 4 present the meta-model for documentation benefit and quality. Note that most elements in these two meta-models have their correspondents in their data scheme that we derived in Section 5.1. For example, the benefit metric *Perceived Importance* in Fig. 3 is also one benefit attribute that is included in Table 1.

In the meta-model of Fig. 3, we have classified benefits into four categories: (1) maintenance aid, (2) development aid, (3) management decision aid, and (4) other. For both *maintenance aid* and *development aid*, *comprehension aid* is an integral aspect. This is because many aspects of a software need to be understood (i.e., comprehended) in order to be properly modified, including its "functionality, architecture, and a myriad of design details" (Arisholm et al., 2006). We classified those aspects as *architecture comprehension*, and *code comprehension*. For instance, 19 papers in the repository considered documentation as an aid to comprehend code or system structure (see Section 6.9 for details).

Several researchers have argued and shown that documentation can also play an important role in management decision-making process (Katzenelson, 1971). To include this aspect into our mapping, we created a category named *management decision aid*. In particular, some researchers proposed that some documents aid management decisions by classification of responsibility (Katzenelson, 1971).

Several papers proposed metrics to quantitatively *measure documentation benefit*. Among all the papers in the repository, we found two main metrics that have been used to measure benefits

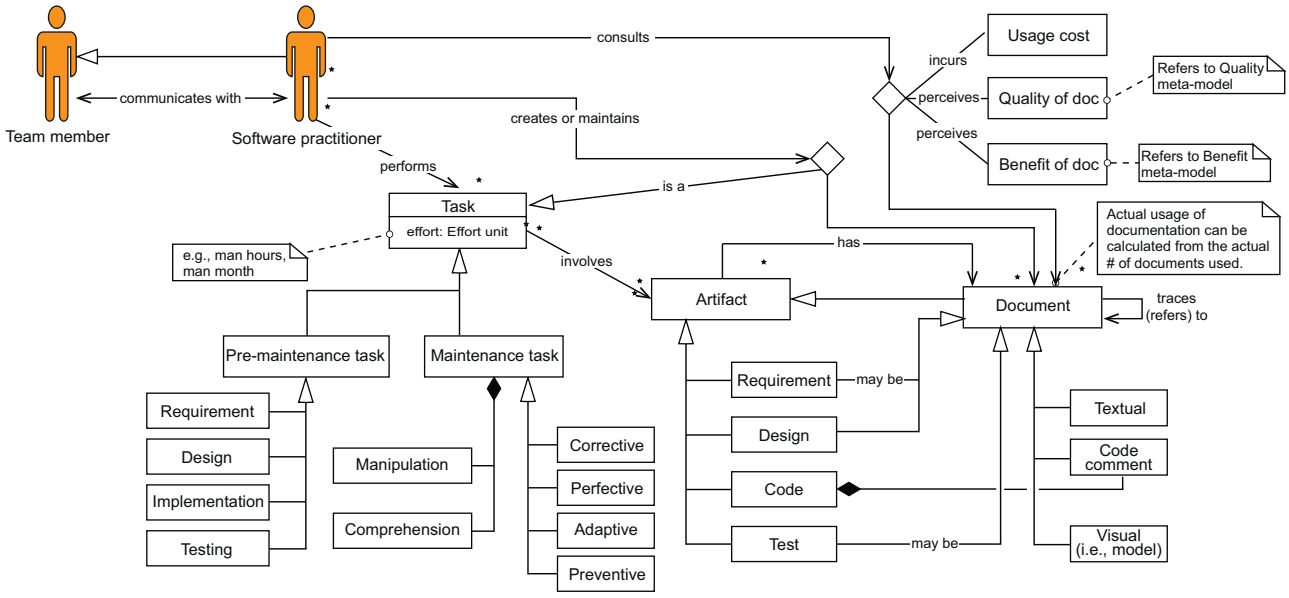


Fig. 2. A meta-model for documentation development, usage-and cost process.

quantitatively: *reduction in effort* and *perceived importance*. *Reduction in effort* refers to that software practitioner would benefit in a way that would save his/her effort (e.g., less time spent on a particular maintenance task) with the aid of documentation. *Perceived importance* is used by several papers (de Souza et al., 2006; Stettina and Heijstek, 2011; Wayne and Carolyn, 2007), describing *software practitioners* perception of the importance of documentation. The latter metric is often measured using questionnaire-based surveys.

5.2.2. Documentation quality

To help us classify quality-related attributes, we constructed a unified model which incorporates all quality attributes (as shown in Fig. 4). A typical software practitioner would usually utilize a type of documentation management system or infrastructure to access (retrieve) software documents. Examples of documentation management systems include online systems (e.g., Wiki), or conventional desktop word processing tools (e.g., Microsoft Word) (Barker, 1990). Each document entity has various attributes: title, author(s), abstract, keywords, format, structure and contents. Note that document structure is believed to have impacts on document quality based on the common sense that, given other attributes the same, a well-structured document is probably easier to read than non-well-structured ones.

We created an abstract class named "Quality" that refers to high-level encapsulation of quality for any entity in the context. *Documentation Quality*, as a type of *Quality*, denotes the intrinsic quality of a document. In our model, software document *Format* and *Structure* are assumed to have impact on *Documentation Quality*.

Document content is an integral part of a document object and has its own quality attributes. We modeled the quality of content using *Content Quality*, which has several attributes as its subclass: *accessibility, accuracy, author-related, completeness, consistency, correctness, information organization, format, readability, similarity, spelling and grammar, traceability, trustworthiness, up-to-date-ness. Accuracy* is a sub-class of *correctness*. This is based on the

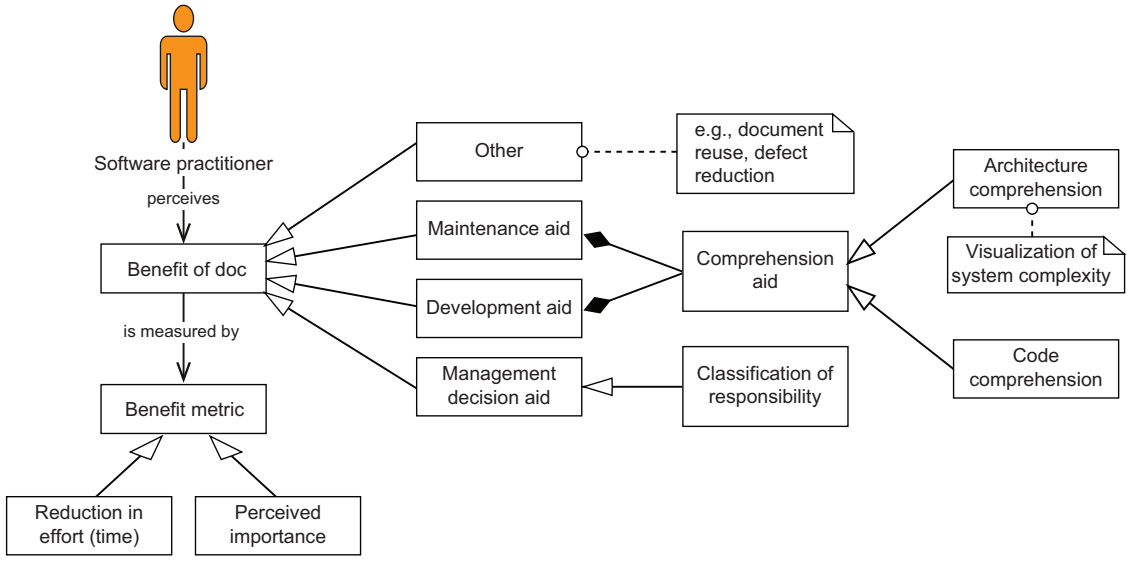


Fig. 3. A meta-model for documentation benefit.

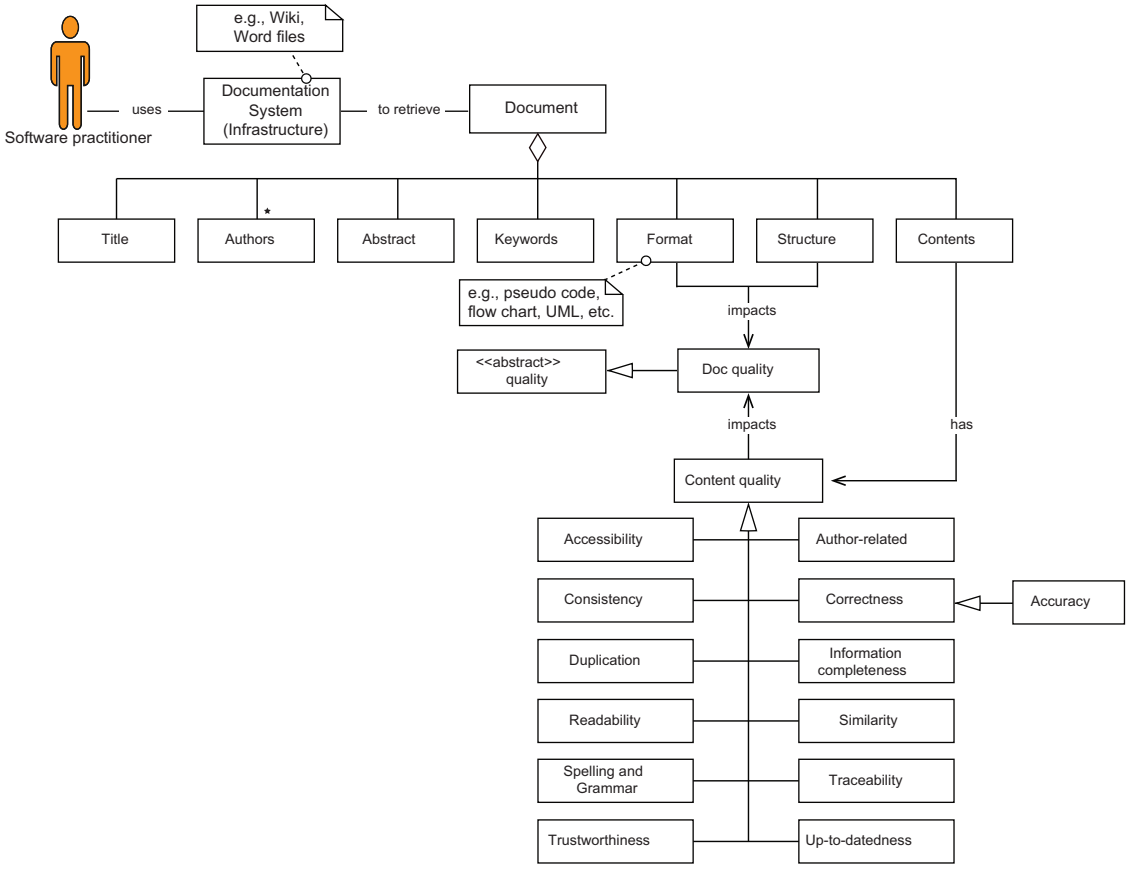


Fig. 4. A meta-model for documentation quality.

fact that *correctness* is the premise of *accuracy*. In other words, an accurate document must be correct, but a correct document can be very general or described in a high-level manner and not necessarily accurate, e.g., a general sentence such as "The system consists of many components" instead of accurate description such as "System XSD consists of three modules and five sub-systems". We included the *author-related* attributes in the model because, in practice, the authoring process of documents is an important factor which could impact documents quality (Kylmkoski, 2003).

5.3. Final systematic map

Table 1 shows the final systematic map that we developed following the process described in Section 5.2. In the table, for each RQ (column 1), the corresponding attributes (column 2), the attribute's potential value set (column 3) and whether it is a multiple choice or a single-choice attribute (column 4), are shown. The last column indicates whether, for each attribute, multiple selections can be applied. For example, for RQ 2 (research fact type), the corresponding value in the last column is 'S' (Single), indicating that one study can only have one research facet type. It is either a solution proposal, a validation research, or any other option listed in the scheme. In contrast, for RQ 1 (contribution type) for example, the corresponding value in the last column is 'M' (Multiple), which indicates that one study can contribute more than one type of options (e.g., method, tool, etc.). Sections 5.3.1–5.3.10 present the details about each attribute scheme listed in Table 1.

5.3.1. Type of paper: contribution facet

The first set of categories in our scheme is related to the contribution facet of the study. The term "contribution facet" is taken from Petersen et al. (2008). The term describes the types of contributions such as being a method/technique, tool, model, metric, process, survey or empirical results. We also added another type: survey or empirical results, since we found that many papers contribute such results. If a study could not be categorized into any above-mentioned types, it would be placed under "Other".

5.3.2. Type of paper: research facet

The second set of categories in our scheme deal with the nature of the research reported in each paper. Similar to "contribution facet", the term "research facet" is defined by Petersen et al. (2008) to classify the research into several categories, including validation, evaluation, etc. The aim of this mapping is to provide insights into the level of empirical foundation used in this domain. The "research type" categories include (Petersen et al., 2008):

- Solution proposal: a paper in this category proposes a solution to a problem. The solution can be either novel or a significant extension of an existing technique. The potential benefits and the applicability of the solution are shown only by a small example or a good line of argumentation.
- Validation research: a paper in this category provides preliminary empirical evidence for the proposed techniques or tools. These papers either proposed a novel technique/approach and its limited application in a certain context to demonstrate its effectiveness, or conducted a survey or interview among a certain number of participants to answer a particular research question. More formal experimental methods (e.g., hypothesis testing, control experiment) or results are further needed to build relevant theories.
- Evaluation research: these papers go further than "Validation research" by using strict and formal experimental methods in

Table 1
Final systematic map developed and used in our study.

RQ	Attribute	Value set	(M)ultiple/(S)ingle
1	Type of paper – Contribution facet	{Method/technique, tool, model, metric, process, survey or empirical results, other}	M
2	Type of paper – Research facet	{Solution proposal, validation research, evaluation research, experience papers, philosophical papers, opinion papers, other}	S
3	Development life-cycle model	{Waterfall, iterative, agile, not mentioned explicitly, other}	S
4	Target artifact of documentation	{Requirement, design, code, test, process, quality, generic, other}	M
5	Documentation format	{Formatted text, models, code comments, specific documentation tool, generic, other}	M
6	Attributes of objects under study: – Number of systems – System types – LOC of systems – Number of organizations – Number of human subjects	Number of systems: Integer System types ∈, {Open-source, commercial, government, academic experimental} LOC: Integer Number of organizations: Integer Number of human subjects: Integer	S
7	Focus	{Documentation cost, documentation usage/benefit, documentation quality}	M
8	Documentation cost attributes	{Development cost, maintenance cost, usage cost, document size, other}	M
9	Documentation usage and benefit attributes	{Development aid, management decision aid, maintenance aid, architecture/design comprehension, code comprehension, perceived importance, reduction in effort (time), actual usage, other}	M
10	Documentation quality attributes	{Accessibility, accuracy, author-related, completeness, consistency, correctness, information organization/structure, format, readability, similarity, spelling and grammar, traceability, trustworthiness, up-to-date-ness, other}	M
11	Industry involvement	{Academic, industry, government, joint}	S

evaluating novel techniques or tools in practice. Hence, these papers provide more convincing empirical evidence and are helpful to build theories. Comparing the definition of Validation and Evaluation research, we may notice that they can be categorized as empirical research, yet the main difference lies in empirical methods used. If one empirical papers employs such strict and formal methods as hypothesis testing or control experiment, then we categorized it as Evaluation research. In other cases where less rigorous methods are used in the study, e.g., a questionnaire survey or a qualitative interview, we categorize such papers as Validation research.

- Philosophical papers: These papers sketch a new way of looking at existing things by structuring the area in form of a taxonomy or conceptual framework.
- Opinion papers: These papers express the personal opinion of the author(s) around whether a certain technique is good or bad, or how things should been done. They do not significantly rely on related work or research methodologies.
- Experience papers: Experience papers explain how something has been done in practice, based on the personal experience of the author(s).
- Field study: A study that aims to gather data about software engineering in real scientific software environments. Examples data collection methods used include questionnaires, interviews, focus groups, and participant observation.
- Other: A catch-all category in the event that the work reported in a paper does not fit into any of the above research types.

5.3.3. Development life-cycle model

The next attribute in our map deals with the types of development life-cycle model applied. The justification for this categorization is that in different development life-cycle models there might be different requirements on documentation quality or different cost-benefit analysis. Extracting such data is helpful to establish the relationship on the variable pair *<Development life-cycle models, cost/benefit/quality of documentation>*.

5.3.4. Target artifact of documentation

This attribute specifies the target artifact for which documentation is made, e.g., a requirement document. If a study did not specify what type of artifact it is discussing, it was classified under "Generic".

5.3.5. Documentation format

This attribute is intended to investigate what type of format documentation is presented. By 'format', it refers to the specific form that documentation is presented. Documentation can be presented using static, manually written text, models (including graphical modeling language such as UML, etc.) and code comments. It can also be automatic, such as electronic documents presented with the aid of specific tools support (e.g., Javadoc, etc.). If one paper does not mention explicitly the documentation format, it falls into the category "Generic".

5.3.6. Objects under study

Most primary papers had objects under study which were either software systems or human subjects. These objects varied depending on the research type of the paper. Often for solution proposals or some experimental papers, authors introduced or applied their approach in a novel software tool for the purpose of illustration or validation. For the other empirical papers that conducted a questionnaire survey or interview, the objects were human participants (usually software practitioners). Characterizing the primary papers by the objects under study in each paper enabled us to assess their level of evidence and experience, for the purpose of generalization validity. For example, a given paper which quantitatively and qualitatively discusses the challenges of software documentation with experience based on 18 organizations (Mira, 2005) will provide higher degree of evidence than another paper which reports similar challenges based on two organizations only (Das et al., 2007).

For each study, we collected, if any, the information including: number of systems under study, whether the system is open-source, commercial, government-related, or for academic experimental purposes. Also, lines of codes (LOC), number of

Table 2
Documentation cost attributes.

Cost attribute	Description
Development cost	Time or effort spent on creating, developing and producing a document
Maintenance cost	Time or effort spent on modifying or updating a document
Usage cost	Time or effort spent on using (mainly reading) documentation
Document size (length)	Metrics used to measure the size of documents, e.g., number of words
Other	Any attributes that do not fit in the above-mentioned category of cost attributes

participating organizations, and number of human subjects (whenever provided) were collected.

5.3.7. Focus of study

Since the purpose of this study is to investigate the cost, benefit or quality of documentation, it is necessary to know the focus of each study on any aspects of above mentioned types. Note that this question is multiple-selection, which indicates that one study can focus on one or more aspect of these types.

5.3.8. Documentation cost attributes

The process for extracting attributes related to documentation cost was iterative. Two rounds of reviews among the authors were conducted. After the refinement process, we derived the attribute scheme presented in Table 2. Note that our definition of 'cost attribute' is a general concept which includes the strictly cost aspects (e.g., time spent, etc.) and also the cost drivers (e.g., document size) that have impacts on the actual cost measurements. Note that the reason of including document size in our cost scheme is based on the belief that document size is one of the factors impacting documentation costs. Generally, the larger a document is, the more maintenance or producing cost is associated with that document.

5.3.9. Documentation benefit attributes and metrics

Similar to extracting documentation cost attributes (Section 5.3.8), the process of extracting attributes related to documentation benefit was iterative. Two rounds of reviews among the authors were conducted to ensure the accuracy of the extracted attributes. After the refinement process, we derived the attribute scheme presented in Table 3.

5.3.10. Documentation quality attributes

For constructing this scheme, the authors applied an iterative process. At the first round, we extracted the data related to documentation quality from each included paper. In order to minimize the personal bias, our data were extracted from the explicit textual description in the papers. We ensured not to introduce any self-invented attributes other than those proposed by the authors of the primary papers.

Extracted data were checked by undergoing at least two rounds of reviews by other authors than the extractor. Then we applied attribute aggregation or clustering. During the aggregation step, all data were put together and synonyms were merged. This is to avoid the situation that there are more than one attributes in the final scheme refer to one highly similar aspect, e.g., *Accuracy* and *Preciseness*.

However, there are still some pairs of attributes that have certain semantic overlaps. Through the discussion of the author, it is decided that we kept all those groups of attributes that have only minor overlaps. For example, *Consistency* may also mean content organization consistency, format consistency, etc. which overlaps with the attributes *Information Organization* and *Format*. Because the three attributes emphasize different quality aspects, it is not appropriate to merge any two of them, thus three of them were kept in the final scheme.

Note that if one quality-related attribute appeared only once in existing literature, we classified it into the category of 'Other'. The results of this step were a list of quality attributes related to software documentation. In Table 4, we present the final attribute scheme and provide the description of the attributes and how and why such attribute impacts documentation quality.

Table 3
Documentation benefit attributes.

Benefit attribute	Description
Development aid	Attributes related to how documentation aids the pre-maintenance tasks.
Management decision aid	Attributes related to how managers benefit from documentation during decision-making process, such as using documents to classify responsibilities among developers.
Maintenance aid	Attributes related to how documentation aids the maintenance tasks.
Architecture comprehension	Attributes related to how documentation aids software practitioners understand system architecture or design rationale.
Code comprehension	Attributes related to how documentation aids software practitioners understand code-level details of the system.
Perceived importance	Measures describing to what extent software practitioners perceive documentation is important.
Reduction in effort	Measures describing to what extent the effort is reduced or saved with the use of documentation.
Actual usage	Metrics used to measure the actual usage of documentation, such as number of visit per document, documentation consultation frequency, etc.
Other	Any attributes that do not fit in the above-mentioned category of benefit attributes.

5.4. Data extraction

To extract data, the papers in our pool were reviewed with the focus of each RQ and the required information was extracted. To increase the preciseness of our classification scheme, we utilized the "observer triangulation" method (Runeson et al., 2012) in data extraction (mapping) phases.

For RQ 8–10 (quality, benefit and cost attributes), each attribute was assigned a two-point scale value (1–2) to annotate the degree of empirical evidence. More specifically, if the attribute under question (e.g., benefit of documentation in development aid) was discussed in a paper together with quantitative empirical evidence (e.g., survey data, controlled experiment, etc.), then the attribute was assigned a degree of two. In contrast, if that attribute was only mentioned or discussed in a qualitative manner without any quantitative validation, evidence or evaluation, then such attribute was assigned the degree of one.

6. Results of systematic mapping (RQ 1–11)

This section presents results related to RQ 1–11.

6.1. Mapping the papers by contribution facet (RQ 1)

Fig. 5 shows the distribution of the type of papers by contribution facet, for all the 69 papers included in our study. Based on their contributions, some papers could be classified under more than one facet. For example, Lehner's study (Lehner, 1993) makes three contributions: (1) a measurement method, (2) a tool, and (3) a metric to measure document comprehensibility and readability.

Table 4
Documentation quality attributes.

Quality attributes	Description
Accessibility	*Accessibility* measures describe the extent to which the content of documentation or document itself can be accessed or retrieved by the software practitioners. Synonyms include 'availability', 'information hiding' and 'easiness to find'. The attribute impacts how practitioners actually use the documentation. In our repository, quite a few papers discuss how this attribute impacts documentation quality, both quantitatively (Forward, 2002; Dautovic et al., 2011; Wayne and Carolyn, 2007) and qualitatively (Sametinger, 1991; Arthur and Stevens, 1989; Correia, 2008; Das et al., 2007; Ericsson et al., 2011; Parnas, 2011; Parnas and Vilkomir, 2007; Sametinger, 1994; Sommerville, 2001; Wingkvist et al., 2010a, 2011).
Accuracy	*Accuracy* measures describe the accuracy or preciseness of documentation content. Synonyms include 'preciseness'. The preciseness of documentation content is generally believed to have impacts on how easy it is for the exact information to be conveyed to the practitioners. If a document is written in a way that the phrasing is vague or the descriptions are too abstract without presenting concrete, exact examples, then it may create barriers for practitioners to retrieve the information and thus impacts the documentation quality (Parnas, 2011; Parnas et al., 1994; Parnas and Vilkomir, 2007).
Author-related	This attribute refers to those attributes related to document authors, including traces of who created the documents, author collaboration, etc. In practice, the authoring process is important for guarantee document quality (Kylmkoski, 2003).
Completeness	*Completeness* measures describe how complete document contents are in terms of supporting development/maintenance tasks. Software documentation is expected to contain all the information needed for the systems or modules described, so that when practitioners read documentation, they can retrieve the information needed for their tasks. If any necessary piece of information is missing, the documentation is perceived not being able to serve its purpose and not being useful in the scenario of need (Mira, 2005; de Souza et al., 2005).
Consistency	*Consistency* measures describe the extent to which documentation, including information presented in documents, document format, etc. are consistent and have no conflict with each other. Synonyms include 'uniformity' and 'integrity'. If the documentation contents are presented inconsistently with conflicting elements, it may confuse practitioners and results in unnecessary mental efforts to resolve those artifacts during the usage of such documentation (Forward, 2002; Mira, 2005).
Correctness	*Correctness* measures describe whether the information provided in the documentation is correct or is in conflict with factual information. If the document presents incorrect information, it is likely to mislead practitioners and creates unnecessary barriers for them to finish the tasks. This attribute is included based on common sense.
Information organization	This attribute describe the extent to which information is organized in documents. If the documentation is organized in a way that is clear and in a structure that is natural to practitioners to understand, such documentation is like to be perceived as in high quality.
Format	This attribute refers to quality of documents' format, including writing style, description perspective, use of diagram or examples, spatial arrangement, etc. This attribute is included because practitioners may prefer certain types of writing styles which are easier for them to understand and use. For example, the decision of choosing to use graphical elements in the documentation is empirically investigated to have impacts on the programming understanding (Tilley and Huang, 2003).
Readability	*Readability* measures describe how easy documents can be read. Synonyms include 'clarity'. This is a subjective quality attributes that is up to the practitioners to decide. Several papers in our repository provide empirical evidence related to this quality attribute (Schreck et al., 2007; Lehner, 1993; de Souza et al., 2005).
Similarity	*Similarity* measures the similarity level in different documents and whether information is duplicated. Some papers use the following notions instead: 'uniqueness' and 'duplication'. Content duplication results in redundancy in the documentation content and leads to unnecessary mental efforts to read and process them.
Spelling and grammar	This attribute refers to those attributes related to the grammatical aspects of documents. If a technical document is presented with a large number of spelling and grammatical errors, it will impact how practitioners read that document.
Traceability	*Traceability* measures describe the extent to which the document modification is able to be tracked; relevant information includes when/where/why the modification is performed and who performed. This attribute deals with the evolution of software documentation which requires special attention in technical documentation. This is because documentation needs to be kept up-to-date together with the software systems or code. The traceability attribute ensures that during the evolution, all the changes to the documentation should be justified and verifiable.
Trustworthiness	*Trustworthiness* measures describe the extent to which software practitioners perceive the documents are trustworthy and reliable. Similar to *Readability*, such attribute is subjective and up to the practitioners to evaluate.
Up-to-date-ness	*Up-to-date-ness* measures describe the extent to which the documents are kept updated during the evolution of software systems. Similar to the description of the attribute *Traceability*, technical documentation is expected to evolve together with software systems. In ideal case, each version of new software release is accompanied with a corresponding version of technical documents. Documentation contents that describe the past release of software systems may provide incorrect information, or miss new information, regarding the new system and thus mislead practitioners.
Other	Several other attributes related to documentation quality were mentioned in several papers, including abstractness (Parnas and Vilkomir, 2007), perceived goodness (Wayne and Carolyn, 2007), etc.

Fig. 5 indicates that proposing new techniques or improving an existing technique has attracted the most research with 26 papers (38%). 20 papers (29%) contributed survey/empirical results. There were six papers (9%) which could not be categorized into the five contribution facets of our scheme, thus we categorized them under 'Other'. The contributions of the papers in the category of 'Other' include a new experimental methodology (Tilley and Huang, 2003), general guidelines for producing documents (Parnas, 2011; Robert, 1985; Schoeffel, 1976; Sommerville, 2001) and a summary of lessons learned (Tilley, 2009).

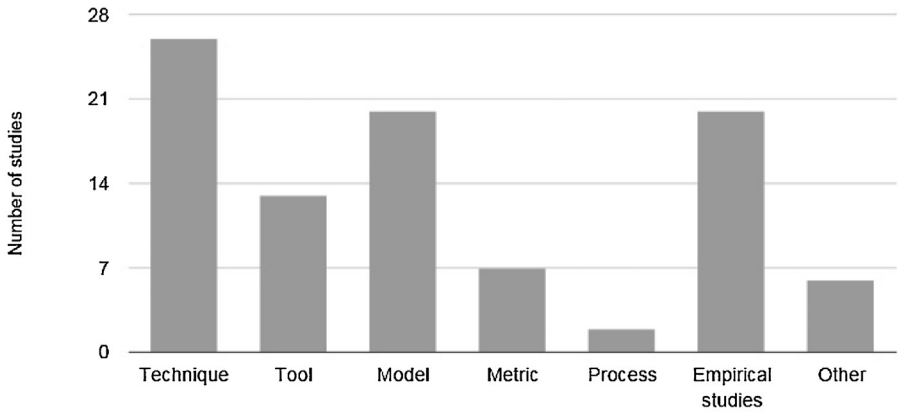
The annual trend of the same data is shown in Fig. 6. In recent years, there is a focus on a mix of different contribution facets. In terms of time, the earliest study (Katzenelson, 1971) in our paper pool was published in 1971. This study is entitled "Documentation and the management of a software project—a case study". It is a case study in a university software project, conducted by Katezenelson. The results show that documentation assisted project participants in making code-related decisions. The study concluded that documentation played an important role in the success of the students' projects.

Readers may notice that Figs. 5 and 6 look slightly different in terms of the sizes of stacks. This is because the contribution facet of a study is multiple-choice while the research facet is single-choice.

6.2. Mapping the papers by research facet (RQ 2)

Based on the classification scheme described in Section 5.3, we classified the papers into five categories. Fig. 7 shows the classification of the 69 selected papers according to the type of research they report. Exact paper references have also been provided under the figure. Note that each paper was categorized in a single category.

Nearly one third of all papers (21 out of 69) relate to presentations of solution proposals without further validation or evaluation. Moreover, a large ratio of papers is validation research (27 papers).



Technique: [9, 13, 14, 16, 17, 19, 25, 27, 28, 32-34, 37, 39, 40, 43, 45, 47, 57, 61, 63, 66-69]
Tool: [2, 3, 7, 9, 11, 15, 23, 25, 27, 32, 33, 57, 60]
Model: [3, 10, 12, 14, 15, 18, 20, 21, 24, 26, 28, 35, 41, 42, 45-47, 64, 65, 67]
Metric: [3, 5, 7, 12, 26, 32, 40]
Process: [34, 37]
Survey/Empirical results: [1, 4-6, 8, 21, 22, 29, 30, 36, 42, 48, 50, 52, 54-56, 58, 62, 66]
Other: [38, 44, 49, 51, 53, 59]

Fig. 5. Frequency of contribution types mentioned in paper.

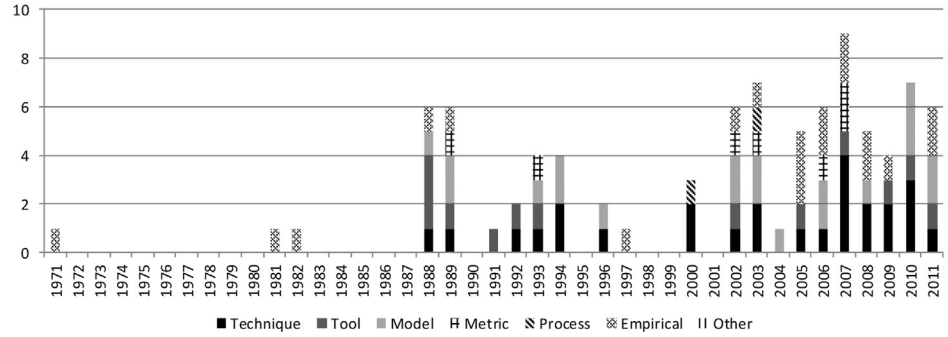
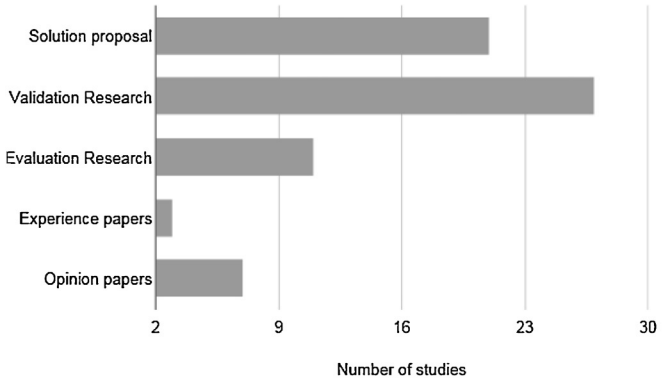


Fig. 6. Trend of contribution facets mentioned in papers over time.



Solution proposal: [2, 11-13, 15, 17, 20, 25, 26, 34, 35, 37, 39, 45, 47, 57, 60, 61, 63, 64, 69]
Validation Research: [6-9, 14, 16, 18, 22, 24, 28-33, 36, 43, 51, 52, 54, 56, 58, 62, 65-68]
Evaluation Research: [1, 3-5, 21, 23, 27, 42, 48, 50, 55]
Experience papers: [40, 41, 59]
Opinion papers: [10, 19, 38, 44, 46, 49, 53]

Fig. 7. Frequency of research types in papers.

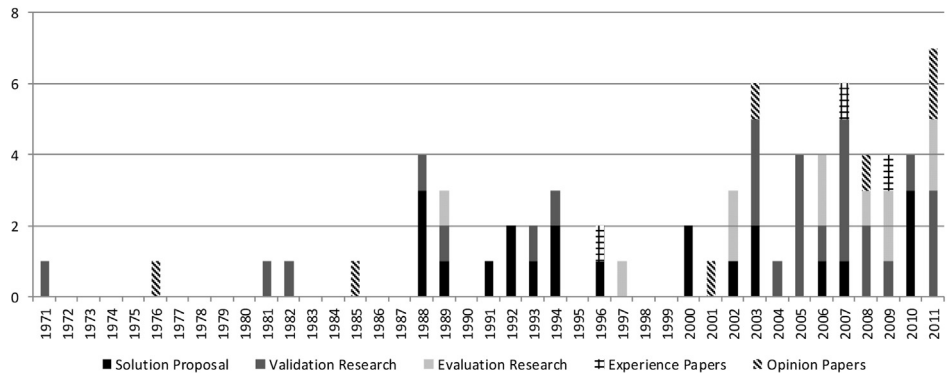


Fig. 8. Trend of research facet used in papers over time.

In contrast, 11 papers (16%) are evaluation research. The remaining 10 papers are opinion papers (7 papers) or experience papers (3 papers). There is no secondary study or philosophical study papers in our repository.

We assume that papers presenting solution proposals are based in an understanding of the specific conditions and problems related to this area. Such papers contribute novel ideas instead of empirical evidence. However, because the repository size (69 papers) is relatively small, more empirical evidence, especially evidence from rigorous experiments, are needed to help this research area grow to maturity.

Fig. 8 shows the annual trend of the papers per research facet during the past 40 years. From the figure the numbers of papers about software documentation cost, benefit and quality are generally increasing since around year 1988. The reason may be that the number of software engineering related papers has generally increased since that time. Furthermore, in two other recent SM papers authored by our colleagues (Banerjee et al., 2013; Garousi et al., 2013) in the areas of GUI and web application testing, the authors observe the increasing trends of the number of papers as well.

During the late 1990s, there is a short period of decrease in paper numbers (around 1971–1999). Since 2005, there have been at least four papers in this area each year, indicating a constant interest among researcher in this area.

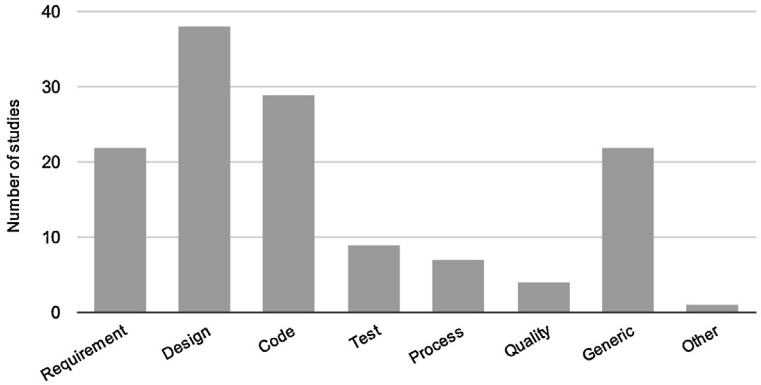
In terms of annual trend of research facets, in recent years (after around 2004), more and more papers are conducting validation or evaluation empirical papers and the share of pure solution proposals are decreasing. This demonstrates the growing research interests to empirically validating idea proposals in this area.

6.3. Types of development life-cycle model (RQ 3)

Most of the papers (61 papers, 88%) do not explicitly mention the type of development life-cycle model used. Agile development is only mentioned in six papers (Forward, 2002; Ambler, 2011; Rubin and Rubin, 2011; Rueping, 2003; Stettina and Heijstek, 2011; Wong and Tilley, 2002) the earliest of which were published in 2002 (Forward, 2002; Wong and Tilley, 2002). Considering that "Agile" is a concept that started to gain popularity since the late 1990s, we could probably infer that the papers published before that period could be considered to have followed traditional waterfall or iterative development processes.

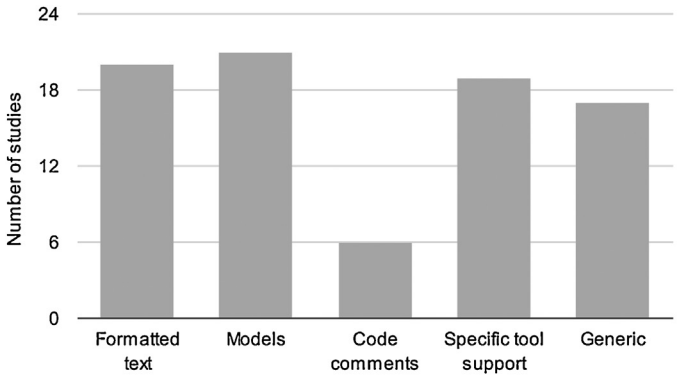
6.4. Target artifact of documentation (RQ 4)

Fig. 9 shows the distribution of documentation artifacts mentioned by the papers. If one paper discusses any of the common types of artifact such as requirement or design documents, code comments, software quality related documents (e.g., process



Requirement: [3-5, 9, 10, 12, 15, 17, 23, 26, 29, 30, 33, 37-39, 45, 48, 51, 53, 54, 62]
Design: [1, 3-6, 9, 10, 12, 13, 15-17, 23, 26-28, 30, 34, 35, 37-41, 45, 47, 48, 50, 51, 53, 54, 57-59, 61-63, 69]
Code: [1-3, 5-10, 12, 15, 21, 25, 26, 30, 35, 37-39, 41, 42, 45, 47, 53, 54, 60-62, 69]
Test: [1, 3, 16, 23, 37, 53, 54, 62, 69]
Process: [10, 15, 26, 37, 48, 53, 62]
Quality (i.e., QA documents): [3, 26, 53, 62]
Generic: [11, 14, 18-20, 22, 24, 31, 32, 36, 43, 44, 46, 49, 52, 55, 56, 64-68]
Other: [26]

Fig. 9. Target artifacts of documentation.



Formatted text: [3, 6, 8, 10, 21, 23, 26, 29, 30, 32, 37, 38, 40, 46, 50, 53, 54, 57, 67, 68]

Model: [3-5, 10, 13, 14, 17, 21, 24, 28, 34, 35, 37, 45, 46, 50, 51, 54, 58, 59, 63]

Code comments: [3, 7, 42, 46, 54, 56]

Specific tool support: [2, 3, 9, 11, 15, 19, 20, 25, 27, 33, 37, 39, 43, 47, 52, 56, 58, 60, 61]

Generic: [1, 12, 16, 18, 22, 31, 36, 41, 44, 48, 49, 55, 62, 64-66, 69]

Fig. 10. Frequency of documentation formats.

record, Q/A documents, etc.), that paper is classified under the corresponding type(s). Note that one paper could discuss more than one type of artifact. Similar to Section 6.2, if the paper does not mention explicitly what type of documentation artifact, or its type of documentation could not be categorized into any type of above-mentioned types, it is placed under the 'Generic' category.

We can observe that design, code and requirement documentation types, in descending order, are three dominating types. The histogram also shows that 22 papers do not provide explicit type of documentation. In contrast, process related types (test, process or quality documents) are less frequently discussed among the papers.

6.5. Documentation format (RQ 5)

Fig. 10 shows the distribution of documentation formats. The distribution seems to be a fine mix, except the code comments type. Three types of format, including formatted text (20 papers), models (21 papers), and specific documentation tools (19 papers) had nearly the equal share of papers. 17 papers (25%) are classified as "Generic". Interestingly, code comments as a documentation format received a very small share (6 papers, 9%).

When comparing Fig. 10 with Fig. 9, we can see that while a large number of papers (29 papers) are discussing documentation artifacts related to code, format of code documentation is not usually in the form of code comments (in only 8 papers), and perhaps other documentation formats are used for code (external formats such as word documents).

6.6. Objects under study (RQ 6)

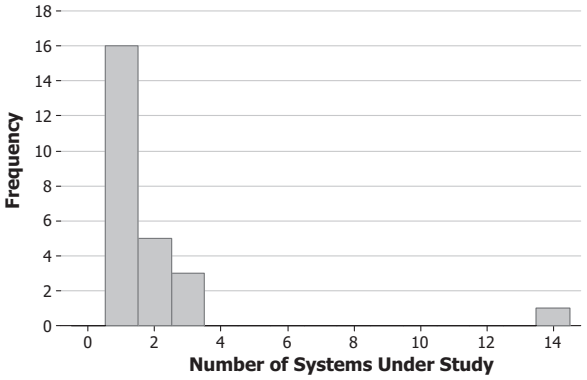
6.6.1. Software systems discussed

Where a given primary study provided information about the software system(s) under study, we extracted the following three pieces of information: (1) number of systems discussed in each paper, (2) system type (academic, commercial, open-source, governmental), and (3) size in LOC. The results of each attribute are presented next.

6.6.1.1. Number of systems under study. Fig. 11 shows the number of systems. In total, there were 25 papers in the repository that discussed systems under study. Among them, 16 papers have only one system for analysis. In contrast, the number of papers which have two or more systems under study dropped sharply. Five papers (7%) studied two systems under study and three papers had three. One extraordinary study (Cook and Visconti, 1994) analyzes 14 systems.

6.6.1.2. Types of systems. Fig. 12 shows the distribution of type of systems under study (SUS). Note that the categories of system type are differentiated by who sponsors the development of the system. For example, if a system is developed for academic experimental purposes, then it may have different requirements on documentation from those sponsored to fulfill commercial needs. From the figure, we can see that most systems (12 out of 25) are for academic experimental purposes. Eight papers involve commercial systems while three papers mentioned documentation in the context of open-source systems. Only two papers discussed systems related to governmental applications.

6.6.1.3. Size in LOC. In our two other recent SM papers (i.e., (Banerjee et al., 2013; Garousi et al., 2013)), we plotted LOC



One system under study: [1-4, 6, 7, 14, 15, 17, 23, 27, 28, 30, 42, 43, 51]

Two systems under study: [5, 24, 35, 41, 67]

Three systems under study: [8, 21, 68]

10+ systems under study: [18]

Fig. 11. Frequency of number of systems under study.

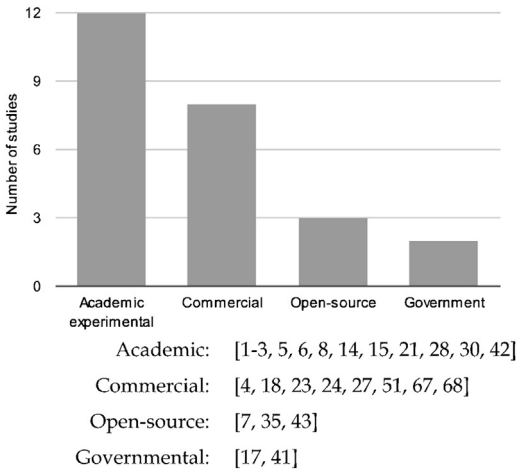


Fig. 12. Type of systems under study.

sizes with the year of paper. Similarly, we intended to investigate whether there is a rising LOC trend from older to new papers. In our repository, only eight papers (8 out of 69, 12%) report the LOC information of their systems under study. Note that most papers do not specifically provide the exact LOC number in their paper. Based on the limited statistics of those eight papers, there is a slight increase in LOC sizes with increase in years. We calculated the Pearson correlation coefficient between the paired variable (year, average LOC size) and the value is 0.38. This indicates that the systems LOC size does not increase strictly over time, but there is a weak correlation. Nevertheless, we spotted an extraordinary case in one study conducted in 2007 (Schreck et al., 2007), which analyzed a system with over 1,600,000 LOC. If more papers had reported the LOC size of their SUSs, the analysis would have been more statistically representative. This calls for the attention of the research community that researchers may need a standard template for reporting and provide precise parameters about their papers so as to enable comparison and statistics gathering.

6.6.2. Survey participants

As discussed in Section 6.1 (RQ 2), 20 papers (29%) contributed survey/empirical results. Out of those 20 papers, only 18 of them (26%) report the number of their survey participants. Fig. 13 shows the histogram of number of subjects in those papers.

From the chart, we can see that the number of participants in most survey papers is below 160. Papers with participant numbers below 40 are dominant. Interestingly, one study (Kylmkoski, 2003) has approximately 1000 participants. The histogram shows that surveys with large number of participants are rare, since inviting and involving large population of participants is practically challenging. This is something that we have also experienced in our own surveys in other topics (e.g., testing practices (Garousi and Varma, 2010)).

6.6.3. Participating organizations

In terms of the organizations where survey participants were from, the results vary. Fig. 14 shows the histogram of number of organizations and its occurrence frequency. Exact number of organizations is not provided in those papers. In total, 13 papers (19%) reported that number.

Not surprisingly, most participants in the selected papers were from one or two organizations. There are also a few cross-organization papers. Another interesting study is that of Visconti et al. (Visconti and Cook, 2004) which reported the results of assessing documentation process among "91 projects at 41 different companies over a seven year period".

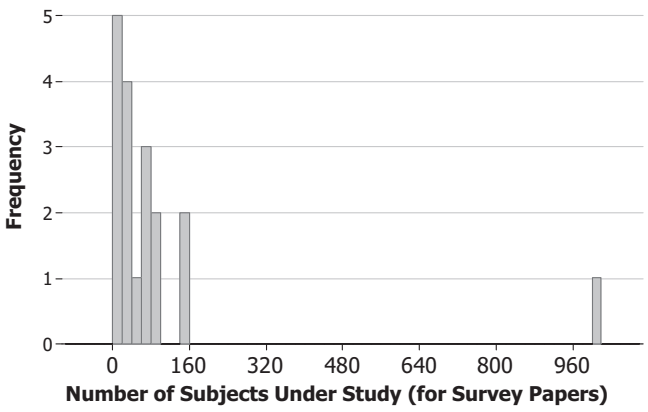


Fig. 13. Frequency of number of subjects under study in survey papers.

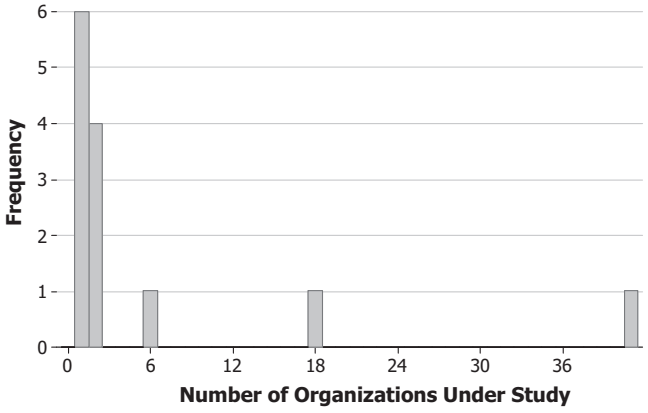


Fig. 14. Frequency of number of organizations.

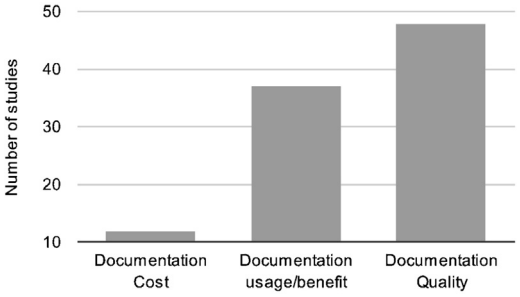


Fig. 15. Focus of papers.

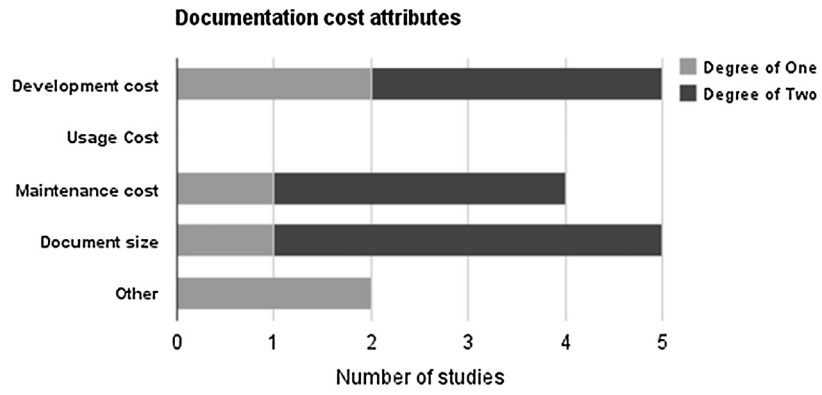
6.7. Focus on documentation cost, benefit and quality (RQ 7)

The histogram in Fig. 15 shows the distribution of papers that discuss each of the three documentation aspects of our interest. From the chart, we can see that 48 papers (71%) discuss documentation quality, followed by 37 papers (54%) on benefit. Surprisingly, and only 12 papers (18%) related to documentation cost. More detailed interpretations of the results will be presented in Sections 6.8–6.10.

6.8. Documentation cost attributes (RQ 8)

Fig. 16 shows the distribution of papers that discuss documentation cost attributes. The data were collected based on the cost attribute scheme described in Section 5.3.8.

Among the papers in the pool, only six papers discuss the development or production cost of software documentation. Four papers



Attribute	Degree of One	Degree of Two
Development cost	[10, 44]	[6, 48, 56]
Maintenance cost	[62]	[4, 5, 23]
Usage cost	N/A	N/A
Document size	[62]	[7, 23, 42, 48]
Other	[38, 44]	-

Fig. 16. Documentation cost attributes.

concern with documentation maintenance costs. The remaining two papers (Parnas, 2011; Robert, 1985) have general discussions on document recording, storing, producing cost, etc. From this statistics there are a very limited number of papers on documentation cost. In other words, documentation's cost aspects seemed to be neglected in the community.

In terms of cost metrics, six papers used document size to measure the cost. Among the two papers that fell into "Other" category, one discusses the "waste of time caused by unstructured documents or incorrect information" (Parnas, 2011).

In terms of degree of evidence (Section 5.4), 12 out of 19 papers report quantitative empirical results. Such a ratio (63%) shows that, although there were a small number of papers on documentation cost, existing papers have provided high levels of empirical evidence by quantitatively measuring documentation cost.

We did not find any paper that addresses the aspect of documentation usage costs. Nevertheless, we modeled this aspect in our meta-model (Fig. 2). The lack of papers regarding this aspect indicates a future research opportunity. We expect to see more empirical evidences that investigate what is the actual cost while using software documentation.

Overall, it seems that, to date, there are only a handful of papers on documentation cost. This is worth the attention of research community. The authors believe that future research in this area should focus on cost measurement and control. Documentation, as a major investment, needs proper tuning so that overall software life-cycle costs are optimized.

6.9. Documentation benefit attributes (RQ 9)

Fig. 17 shows software documentation benefit attributes and their degree of evidence. The data were collected under the benefit attribute scheme described in Section 5.3.9. From the chart, 29 papers (42%) considered documentation as an aid to software maintenance, which is the dominating attribute. However, among these 29 papers, only two papers (Arisholm et al., 2006; Curtis et al., 1989) reach two in degree of evidence. 16 papers (55%) discussed documentation as a development aid. 10 papers are related to documentation as an aid to comprehend system architecture while 14 papers are on low-level code comprehension. The ratio of papers that reached the degree of two in empirical evidence is relatively small (6 out of 24).

Four survey studies (5%) are conducted to survey the perceived importance of documentation among professionals. Only three papers are related to how documentation saved time of developers. Among these three, only (Dzidek et al., 2008) measures the reduction quantitatively. The paper (Dzidek et al., 2008) reports a controlled experiment on the benefit of using UML to save time in the context of maintaining object-oriented software. The metric used in (Dzidek et al., 2008) to measure savings in performing the tasks was time, which aligns with our model in Section 5.2.1.

In terms of actual usage, all three papers (Forward, 2002; de Souza et al., 2005; Wingkvist et al., 2011) reach two in degree of evidence. The metrics used in those papers include actual usage percentage (de Souza et al., 2005), number of visits per document (Wingkvist et al., 2011) and mean consultation number (Forward, 2002).

Overall, the limited number of papers reaching two in degree of evidence indicates that most papers discussed documentation benefit in a qualitative manner. This is probably due to the difficulty of measuring how exactly documentation aids development tasks in actual projects.

6.10. Documentation quality attributes (RQ 10)

Fig. 18 shows software documentation quality attributes and their degree of evidence. Note that the data were collected under the quality attribute scheme described in Section 5.3.10.

As the figure shows, the quality attributes of documentation which attract most attention of previous researchers is "completeness" (17 papers, 28%). It might indicate that incomplete documentation has been one of difficulties that most researchers or professionals have attempted to address. Following the attribute of completeness are "consistency" (16 papers) and "accessibility" (14 papers). Besides, there are a certain number of papers on documentation format (13 papers, 19%) and up-to-date-ness (12 papers). In terms of degree of evidence, six papers out of 13 on documentation format reached two. Papers such as (Curtis et al., 1989; Sheppard et al., 1982) have provided strong empirical evidence for

the document format and its influence. There are also five and four empirical papers that have the degree of two on completeness and up-to-date-ness, respectively. In total, 47 out of all 132 degrees shown in Fig. 18 (35%) reached the maturity degree of two. It indicates that, for the documentation quality aspect, researchers have put efforts to empirically evaluate this particular aspect.

6.11. Industry's involvement (RQ 11)

This RQ aims to investigate industry's involvement in studying documentation cost, benefit and quality. Since documentation issues are a constant challenge in industrial projects, it is important to connect the efforts of industry and academia in this subject.

To address this RQ, we first investigated author affiliations to see the level of industry-academia collaborations of the papers in our pool. Fig. 19 shows the distribution of authors of the 69 selected papers by affiliation type. We identified four categories of authors' affiliations: (1) papers whose authors are all academic researchers, (2) papers authored by industry practitioners (3) papers from governmental research groups such as NASA, and (4) collaborative work (papers jointly authored by groups of authors affiliated with two or three different categories). As Fig. 19 shows, papers from academia (57 papers, 83%) are the most frequent. Among the other groups, the number of papers written solely by practitioners (7 papers, 10%) is higher compared to papers reported by governmental research groups (no papers) and joint papers (3 papers, 4%).

There are in total seven papers (Ambler, 2011; Kylmkoski, 2003; MacKinnon and Murphy, 2003; Palazzolo and Utt, 2000; Robert, 1985; Rueping, 2003; Sheppard et al., 1982) authored by practitioners. The aspects concerned in these papers include documentation process in agile development (Ambler, 2011; Rueping, 2003), document quality from the perspective of document authoring process(Kylmkoski, 2003), the usage of UML diagrams (MacKinnon and Murphy, 2003), documentation standard (Robert, 1985), document notations and spatial arrangements (Sheppard et al., 1982), and documentation development process (Palazzolo and Utt, 2000).

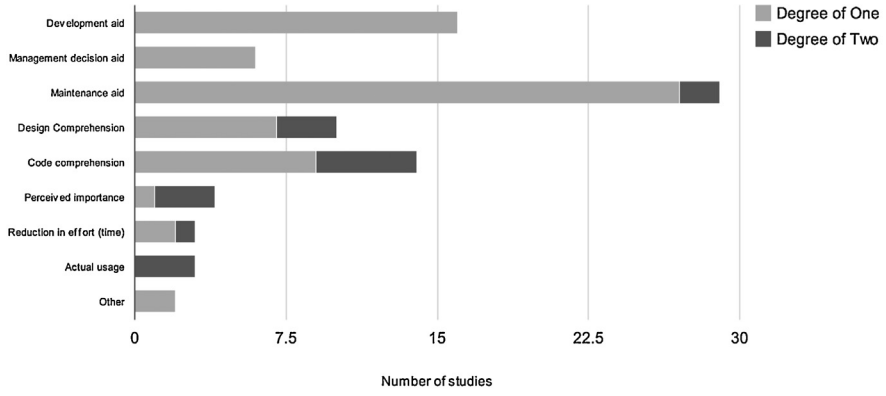
However, compared with the number of papers by academia in this area, the industrial involvement are relatively in short. We hope that industry involvement will continue to increase so as to enrich the literature and help mature this field.

7. Threats to validity

The discussion of threats to validity is important to judge the strengths and limitations of our SM study. For our study, the following issues may introduce threats to validity: selection of search databases, definition of search terms and time frame, researcher bias with regards to exclusion/inclusion, and incorrect data extraction (classification). Using the standard classification scheme of validity threats suggested in Wohlin et al. (2000), we discuss these issues in relation to four types of threats to validity: (1) conclusion validity, (2) construct validity, (3) internal validity, and (4) external validity.

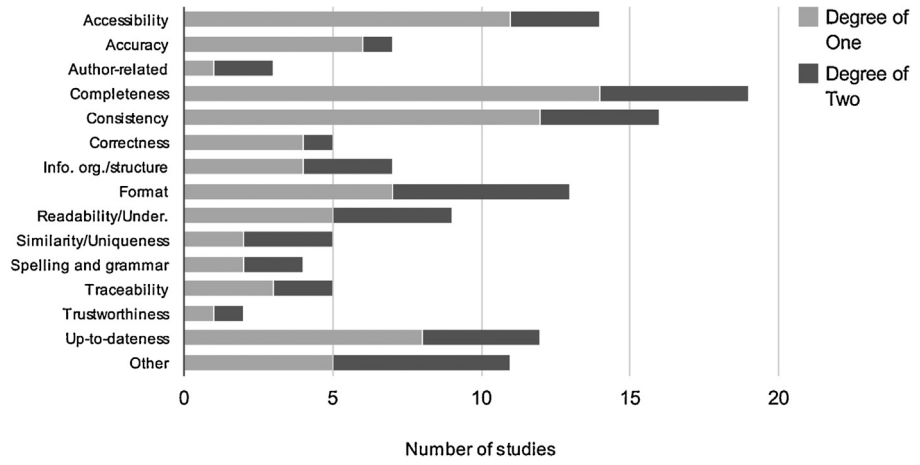
7.1. Conclusion validity

Conclusion validity refers to the degree to which conclusions we reach about the relationships are reasonable. In Section 8, we draw our conclusions about the research landscape in this area, including the academic trend, the topics that have been frequently discussed or thoroughly studied, and the emerging areas that deserve attention. These conclusions are based on our statistical data from the paper repository, i.e., the number of papers focusing on a certain aspects, e.g., research facet, contribution facet, etc. The conclusion validity issue lies in whether there is a relationship between the



Attribute	Degree of One	Degree of Two
Development aid	[3, 6, 19, 21, 27, 28, 30, 38, 40, 45, 50, 56, 58, 60, 62, 69]	-
Management decision aid	[15, 30, 34, 45, 49, 61]	-
Maintenance aid	[1-4, 6, 8, 15, 17, 19, 25, 27, 28, 35, 36, 49, 51, 54-56, 58, 59, 61-63, 66, 68, 69]	[5, 21]
Design Comprehension	[6, 28, 35, 36, 59, 61, 63]	[27, 51, 54]
Code comprehension	[2, 6, 25, 35, 36, 40, 45, 60, 69]	[1, 8, 21, 51, 54]
Perceived importance	[66]	[55, 56, 58]
Reduction in effort (time)	[17, 38]	[4]
Actual usage	-	[3, 54, 68]
Other	[19, 31]	-

Fig. 17. Documentation benefit attributes.



Attribute	Degree of One	Degree of Two
Accessibility	[2, 12, 19, 22, 24, 38, 40, 47, 53, 67, 68]	[3, 23, 66]
Accuracy	[12, 18, 38, 40, 41, 61]	[65]
Author-related	[19]	[3, 23]
Completeness	[2, 11, 12, 15, 18, 28, 30, 37-40, 64, 67, 68]	[7, 14, 23, 36, 54]
Consistency	[2, 6, 11, 12, 18-20, 28, 34, 37, 43, 64]	[3, 9, 36, 65]
Correctness	[10, 14, 39, 67]	[36]
Info. org./structure	[22, 33, 37, 46]	[3, 23, 66]
Format	[22, 26, 37, 46, 66-68]	[3, 8, 21, 23, 51, 52]
Readability	[10, 12, 24, 36, 39]	[7, 29, 32, 54]
Similarity	[13, 24]	[23, 67, 68]
Spelling and grammar	[67, 68]	[3, 23]
Traceability	[11, 12, 27]	[23, 36]
Trustworthiness	[10]	[65]
Up-to-datedness	[13, 18, 19, 27, 41, 61, 64, 67]	[3, 36, 54, 65]
Other	[13, 26, 40, 53, 67]	[12, 23, 56, 57, 65, 66]

Fig. 18. Documentation quality attributes.

number of papers and the actual academic focus and efforts. There are risks that the relationship does not exist.

7.2. Construct validity

Construct validity is related to selecting the right variables to *measure* the phenomenon of interest. The construct validity issue in our study lies in the comprehensiveness of the categorization scheme that we used for the data extraction. One example question that readers may raise is: Are those benefits attributes (Fig. 3) the correct ones and comprehensive enough to measure all document benefits? To mitigate this issue, all the attributes are extracted based on the collected papers and the process of extraction and constructing schemes had been undergone several reviews by all authors. In such a way, we minimized the number of attributes that we might have missed in the papers. Second, we have constructed the formalized models for the three aspects (Section 5.2) in order to cover as many attributes as we can from a theoretical standpoint.

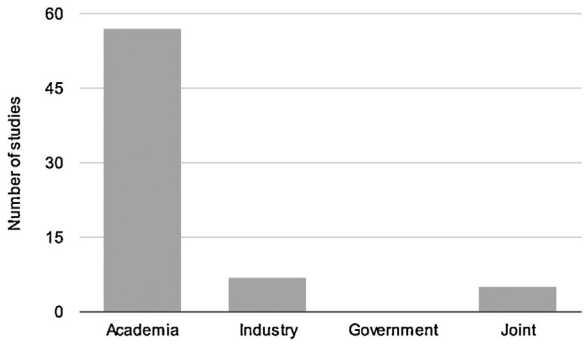


Fig. 19. Histogram of author affiliations.

7.3. Internal validity

The internal validity concerns with how well the causal relationship is warranted. In our study, we attempted to establish relationships between various attributes extracted from the papers and the academic field landscape under study (cost, benefits and quality of software development documentation). The internal validity issues are mainly in the papers' selection process. More specially, the issues include: (1) potentially missing relevant papers, and (2) researchers' bias in papers inclusion/exclusion. We discuss these two issues in the following paragraphs.

The first issue is the completeness of our paper repository. To mitigate this risk, we chose to use the popular academic search engines, including IEEE Xplore, ACM Digital Library, etc. Also, we attempted to use various combinations of the topics of interest and their synonyms related to *software documentation*, or *software documentation* (Section 4.1). These string constructs were extracted from the *IEEE Standard Glossary of Software Engineering Terminology*. Finally, we examined the reference list of the known papers as well (Section 4.3) to find the ones that did not show up in our

initial search efforts. Even if we have done our best in conducting a systematic search, it is possible that we may have missed some relevant papers during our search phase. For example, it is possible that some research projects or studies in this area may not have been reported or they may not have been indexed in the search engines, or have been indexed using a different set of keywords that were not included in our keyword set.

The second issue is in the inclusion/exclusion criteria that rely on the researchers' judgment and experience. Personal bias may be introduced during this process. To mitigate this issue, we applied a few strategies to increase the reliability of our decisions with respect to inclusion/exclusion criteria, as discussed next. First, each of the four researchers examined candidate papers independently and followed the voting process described in Section 4. With regards to data extraction, we split the final pool of papers in three equally large sub-pools and assigned one researcher to each sub-pool plus an additional reviewer. In case of disagreements about inclusion/exclusion criteria or specific classification decisions, a second reviewer was involved. In all such cases, this procedure yielded consensus.

7.4. External validity

External validity is concerned with to what extent we can generalize the results of our SM study. As described in Section 4, our selected literature under study was all written in English language. Papers written in other languages were excluded. One issue lies in that whether the papers included in our repository is able to represent all the relevant works in the area of software documentation cost/benefit/quality. One threat stems from the keywords used, databases we selected and inclusion/exclusion criteria. As discussed in Section 7.1, we followed a comprehensive search procedure to make sure our repository was as inclusive as possible. We believe that relevant literature we selected in our pool contained sufficient information to represent the knowledge reported by previous researchers or professionals, including those non-English authors.

Also, note that the classification schemes that we derived are only applicable in the field of software documentation cost, benefit and quality. Additional related papers and future papers can be categorized using our schemes.

8. Summary, conclusions and future work

8.1. Addressing RQs

In this study, we addressed eleven research questions (RQ 1 to RQ 11). Below we list the main result(s) of each RQ:

- **RQ 1 – Number of papers by research facet**: What type of research methods are used in the papers?
  In terms of contribution facet, most selected papers (about 38%) proposed new techniques or improved an existing one. A certain proportion (29%) of papers (20 out of 69) contributes empirical evidences.
- **RQ 2 – Number of papers by contribution facet**: What types of contributions are made by the papers?
  Validation research papers are dominating (27 papers or 40%), followed by solution proposals (21 papers, or 31%); evaluation research papers only cover 16% of all papers, followed by opinion papers and experience papers.
- **RQ 3 – Types of development life-cycle model**: In what type of development life-cycle model is the documentation applied?
  Most papers did not mention development cycle explicitly. Agile development was only mentioned in 6 papers.
- **RQ 4 – Type of artifact**: What are the artifact types for which documentation is made?
  Design, code and requirement, with paper number decreasing in order, are three dominating types of documentation; software quality related types (test, process or quality documents) are less frequently discussed among our selected papers.
- **RQ 5 – Documentation formats**: In what format is the documentation presented?
  Three types of format, including formatted text (20 papers), models (21 papers), and tool support (19 papers) had nearly the equal share of papers; documentation mentioned as code comments received a very small share (6 papers).
- **RQ 6 – Objects under study**: What are the attributes of the objects under study?
  Most papers had only one SUS. Most SUS's (12 of 25 papers, 48%) were academic prototypes. Unlike what one would expect, the trend of SUS size does not increase strictly over time. The average number of participants in survey-based papers was 106, the highest one having approximately 1000 participants (Kylmkoski, 2003). Most participants in survey-based papers were from one or two organizations.
- **RQ 7 – Focus of papers**: Among all papers devoted to documentation, what is the percentage of papers that have focused on these aspects of interest: cost, benefit and quality?
  In terms of focus of papers, 50 papers (72%) discuss documentation quality, followed by 37 papers (54%) on benefit, and 12 papers (17%) focusing on documentation cost.
- **RQ 8 – Software documentation cost attributes and metrics**: What attributes and metrics related to the cost of documentation have been studied?
  Only six papers (8%) discuss the development or production cost of software documentation. Four papers (6%) address documentation maintenance costs. In terms of degree of evidence in this category of papers (documentation costs), 12 out of 19 papers reached the maturity degree of two (i.e., full empirical evaluation of the attribute).
- **RQ 9 – Software documentation usage and benefit attributes and metrics:** What attributes and metrics related to the usage/benefit of documentation have been studied?
  In terms of documentation usage, 29 papers (42%) studied documentation as an aid to software maintenance. 10 papers (14%) were related to documentation as an aid to comprehend system architecture during development. 14 papers (20%) were on code-level comprehension.
- **RQ 10 – Software documentation quality attributes and metrics:** What attributes and metrics related to the quality of documentation have been studied?
  The quality attributes of documentation that attracted most attention of previous researchers is "completeness", followed by "consistency" and "accessibility".
- **RQ 11 – Industry's involvement**: What are industry's involvement related to software documentation cost, benefit and quality?
  Papers where all authors are affiliated with universities (83%) are dominating, compared with the number of papers from industry (10%), governmental research centers (0%) or joint collaborations (4%).

8.2. Summary and conclusions

The above-mentioned results of our systematic mapping can help new (e.g., PhD students) or established researchers as well as practitioners to obtain an overview of the research space (approaches, metrics, tools and models) in this field. From RQ 1 to 2 we can see that the majority of existing papers focus on proposing a new approach or techniques (around 40%) and presenting

preliminary validation results on their proposal (29% of all papers contributing empirical evidences). We can infer from such a ratio comparison that a certain number of approaches/techniques proposed have not been reportedly validated in empirical evidences. Also, the lack of papers contributing documentation processes or metrics lets us speculate why these aspects have been neglected in existing literature and what factors lead to such facts. Among the empirical papers, only a small portion of them offer substantial evidences (RQ 6), such as study results reported from large-scale development projects, or from large number of study participants of various organizations. Stronger empirical evidences are still needed to enhance the understanding and to establish profound theories of this field.

On the other hand, our results reveal that in most existing discussion development documents are mostly in certain types (e.g., design, requirement, etc. from RQ 4). Software quality related document types, such as test, process or quality documents, attracted little attention. More research concerning these types may be worthy of consideration. In terms of documentation formats (RQ 5), certain formats, such as text, models, tools, etc. are discussed frequently in collected papers. Questions such as "Are requirement documents mostly textual?", "Do practitioners write design documents using graphical models?" might be raised about the relationship between documentation types and format. This is one interesting topic worthy of future research. Besides, most existing research does not discuss documentation in an explicitly-given context of development cycles (RQ 3). As people's understanding of software development evolves, future researchers or practitioners might need to redefine the role and boundary of documentation according to different SDLC contexts. Specifically, research on documentation in Agile development practice could be an interesting research direction, based on the fact that only six papers have investigated this aspect so far.

Also, the results would help to identify the emerging areas in this field and also the areas that require more attention from the research community. As the results in RQ 7–8 reveal, one important aspect of documentation, the cost, seems to be neglected in existing literature. There are many unanswered questions regarding cost metrics or measurement, such as: "Is document size a good document cost metric?", "What factors lead to the cost?" or "What are the underlying cost-drivers in a system development or maintenance context?" This is a strong contrast to other related research areas which have matured with rich literature, such as software effort estimation. As the future project management requires more accurate cost control over development process, effort consumed in documentation as one significant cost drivers will need to be tuned properly. Therefore, more papers dedicated to understand documentation behaviors and leading toward a documentation cost estimation model will be expected. In terms of documentation benefits (RQ 9), only a limited number of papers provided strong empirical evidences and the majorities only discussed benefits in a qualitative manner. Although there were a few controlled experiments reported on how documentation or UML saved maintainers' time or improved code quality, an aggregation of these empirical evidences and more novel evidences are still needed to achieve a well-established theory so as to achieve better understanding of the process and to generate meaningful benefit measurements. This trend prediction can also be applied to documentation quality (RQ 10). As the results show, the majorities of papers concerning documentation quality attributes have only qualitative discussion. On the one hand, there is a general lack of models to comprehensively and meaningfully incorporate the different aspects of software documentation. To address this issue, we proposed a quality model in Section 5.2.2. On the other hand, current proposed metrics are still far from mature to actually measure document quality. Again, more empirical evidences are worthy of investigation to validate existing models and to investigate how quality-related measurements of software documentation would generate concrete benefits for practitioners.

Recall from Section 1 that the need for this SM was motivated in the context of a multi-year industrial collaborative research and development project, which aims to minimize the cost and amount of documentation across the software development lifecycle for one of our industrial partners. The results of this SM have already started to benefit our research team in that project by providing to us a summary of what has been done in each of the following sub-areas: cost, benefits and quality of technical software documentation. The results of the SM have enabled us to adapt (re-use) some of the existing techniques, thus preventing us from "re-inventing the wheel", and to develop novel methods, models and techniques in this area. Specifically, formalized models of documentation cost, benefit, and quality that we developed during this SM (Section 5.2) have been very beneficial for our team members in a formal analysis of the subject matter.

In terms of industry's involvement in the field, a certain number of papers concerning practical aspects of documentation (e.g., tool building, standard selection, etc.) are reported by our industrial colleagues. We hope that such trend of industry involvement will continue to enrich the literature and help mature this field.

Acknowledgements

This work was supported by the NSERC CRD grant # CRDPJ414157-11, and NSERC ENGAGE grant # EGP-413039. Vahid Garousi was additionally supported by Atilim University and the Scientific and Technological Research Council of Turkey (TÜBİTAK). We would also thank the anonymous reviewers for their insightful comments that helped us improve this article.

Primary Papers

Aguiar, A., David, G., 2005. WikiWiki weaving heterogeneous software artifacts. In: Presented at the Proceedings of the International Symposium on Wikis, San Diego, CA.

Ambler, S., 2011. Agile/Lean Documentation: Strategies for Agile Software Development. Available: http://www.agilemodeling.com/essays/agileDocumentation.htm (Accessed: June 2012).

Arisholm, E., Briand, L.C., Hove, S.E., Labiche, Y., 2006. The impact of UML documentation on software maintenance: an experimental evaluation. IEEE Trans. Softw. Eng. 32, 365–381.

Arthur, J.D., Stevens, K.T., 1989. Assessing the adequacy of documentation through document quality indicators. In: Proceedings of the International Conference on Software Maintenance, Miami, FL, pp. 40–49.

Arthur, J.D., Nance, R.E., Stevens, K.T., 1988. Prospects for Automated Documentation Analysis in Support of Software Quality Assurance. Systems Research Center, Virginia Polytechnic and State University, Blacksburg, VA.

Bachmann, F., Bass, L., Carriere, J., Clements, P.C., Garlan, D., Ivers, J., Nord, R., Little, R., 2000. Software architecture documentation in practice: documenting architectural layers: special report CMU/SEI-2000-SR-004. Software Engineering Institute, Carnegie Mellon University.

Bayer, J., Muthig, D., 2006. A view-based approach for improving software documentation practices. In: Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems, Washington, DC, USA, pp. 269–278.

Blum, B.I., 1988. Documentation for maintenance: a hypertext design. In: Proceedings of the Conference on Software Maintenance, Scottsdale, AZ, pp. 23–31.

Cioch, F.A., Palazzolo, M., Lohrer, S., 1996. A documentation suite for maintenance programmers. In: Proceedings of the International Conference on Software Maintenance, Monterey, CA, pp. 286–295.

Cook, C., Visconti, M., 1994. Documentation is important. CrossTalk 7, 26–30.

Correia, F.F., 2008. Extending and integrating wikis to improve software documentation. In: Presented at the 4th International Symposium on Wikis.

Correia, F.F., Aguiar, A., Ferreira, H.S., Flores, N., 2009. Patterns for consistent software documentation. In: Proceedings of the 16th Conference on Pattern Languages of Programs, Chicago, IL, pp. 12:1–12:7.

Curtis, B., Sheppard, S.B., Elizabeth, K.-B., Bailey, J., Deborah, A.B.-D., 1989. Experimental evaluation of software documentation formats. J. Syst. Softw. 9, 167–207.

Das, S., Lutters, W.G., Seaman, C.B., 2007. Understanding documentation value in software maintenance. In: Proceedings of the Symposium on Computer Human Interaction for the Management of Information Technology, Cambridge, MA.

Dautovic, A., Plösch, R., Saft, M., 2011. Automated quality defect detection in software development documents. In: 5th International Workshop on Software Quality and Maintainability, Oldenburg, Germany.
de Boer, R.C., van Vliet, H., 2009. Writing and reading software documentation: how the development process may affect understanding. In: Presented at the Proceedings of the Workshop on Cooperative and Human Aspects on Software Engineering.
de Souza, S.C.B., Anquetil, N., de Oliveira, K.M., 2005. A study of the documentation essential to software maintenance. In: Proceedings of the 23rd Annual International Conference on Design of Communication, Coventry, United Kingdom, pp. 68–75.
de Souza, S., Anquetil, N., de Oliveira, K., 2006. Which documentation for software maintenance? J. Braz. Comput. Soc. 12, 31–44.
Dzidek, W.J., Arisholm, E., Briand, L.C., 2008. A realistic empirical evaluation of the costs and benefits of uml in software maintenance. IEEE Trans. Softw. Eng. 34, 407–432.
Ericsson, M., Wingkvist, A., Löwe, W., 2011. A software infrastructure for information quality assessment. In: Proceedings of the 16th International Conference on Information Quality, Adelaide, Australia.
Fletton, N.T., Munro, M., 1988. Redocumenting software systems using hypertext technology. In: Proceedings of the Conference on Software Maintenance, Phoenix, Arizona, pp. 54–59.
Forward, A., 2002. Software Documentation – Building and Maintaining Artefacts of Communication. University of Ottawa, Master in Computer Science, Ottawa-Carleton Institute for Computer Science.
Huang, S., Tilley, S., 2003. Towards a documentation maturity model. In: Proceedings of the 21st Annual International Conference on Documentation, San Francisco, CA, USA, pp. 93–99.
Jansen, A., Bosch, J., Avgeriou, P., 2008. Documenting after the fact: recovering architectural design decisions. J. Syst. Softw. 81, 536–557.
Jansen, A., Avgeriou, P., van der Ven, J.S., 2009. Enriching software architecture documentation. J. Syst. Softw. 82, 1232–1248.
Kanter, H., Muscarello, T., Ralston, C., 2008. Measuring the readability of software requirement specifications: an empirical study. J. Inform. Syst. Control 1, 1–6.
Katzenelson, J., 1971. Documentation and the management of a software project—a case study. J. Softw.: Pract. Exp. 1, 147–157.
Kylmkoski, R., 2003. Efficient authoring of software documentation using RaPiD7. In: Proceedings of the 25th International Conference on Software Engineering, Portland, OR.
Lehner, F., 1993. Quality control in software documentation based on measurement of text comprehension and text comprehensibility. Inform. Process. Manag. 25, 551–568.
Lethbridge, T.C., Singer, J., Forward, A., 2003. How software engineers use documentation: the state of the practice. J. IEEE Softw. 20, 35–39.
Maarek, Y.S., Benyfj, D.M., 1989. The use of lexical affinities in requirements extraction. In: Proceedings of Fifth IEEE International Workshop on Software Specification and Design, Pittsburgh, PA, United States, pp. 196–202.
MacKinnon, N., Murphy, S., 2003. Designing UML diagrams for technical documentation. In: Proceedings of the 21st Annual International Conference on Documentation, San Francisco, CA, USA, pp. 105–112.
Meng, W.J., Rilling, J., Zhang, Y., Witte, R., Mudur, S., Charland, P., 2006. A context-driven software comprehension process model. In: 2nd International IEEE Workshop on Software Evolvability, Philadelphia, PA, pp. 50–57.
Mira, K.-M., 2005. A survey of documentation practice within corrective maintenance. Empir. Softw. Eng. 10, 31–55.
Palazzolo, M., Utt, M.H., 2000. A unified process for software and documentation development. In: Presented at the Proceedings of IEEE Professional Communication Society International Professional Communication Conference and Proceedings of the 18th Annual ACM International Conference on Computer Documentation: Technology and Teamwork, Cambridge, MA.
Parnas, D.L., 2011. Precise documentation: the key to better software. In: The Future of Software Engineering. Springer-Verlag, Berlin, Heidelberg, pp. 125–148.
Parnas, D.L., Vilkomir, S.A., 2007. Precise documentation of critical software. In: Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium, HASE '07.
Parnas, D.L., Madey, J., Llewski, M., 1994. Precise documentation of well-structured programs. IEEE Trans. Softw. Eng. 20, 948–976.
Pemberton, L., Gorman, L., Hartley, A., Power, R., 1996. Computer support for producing software documentation: some possible futures. In: The New Writing Environment: Writers at Work in a World of Technology. Springer, pp. 59–71.
Pendharkar, P.C., Rodger, J.A., 2002. An empirical study of factors impacting the size of object-oriented component code documentation. In: Proceedings of the 20th Annual International Conference on Computer Documentation, Toronto, Ontario, Canada, pp. 152–156.
Poshyvanyk, D., Marcus, A., 2007. Using traceability links to assess and maintain the quality of software documentation. In: Proceedings of the 4th ACM International Workshop on Traceability in Emerging Forms of Software Engineering, Lexington, KY, USA, pp. 27–30.
Robert, P., 1985. Selecting software documentation standards. IEEE Softw., 90–91.
Rubin, E., Rubin, H., 2011. Supporting agile software development through active documentation. Acad. J. Requir. Eng. 16, 117–132.
Rueping, A., 2003. Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects, 1st ed. John Wiley & Sons Inc., New York, NY, USA.
Sametinger, J., 1991. DOgMA: A Tool for the Documentation & Maintenance of Software Systems. Doctoral Dissertation. Johannes Kepler University of Linz, Austria, 9783853698686.
Sametinger, J., 1994. Object-oriented documentation. ACM SIGDOC Asterisk J. Comput. Doc. 18, 3–14.
Sanchez-Rosado, I., Rodrguez-Soria, P., Martn-Herrera, B., JosCuadrado-Gallego, J., Martínez-Herráiz, J., González, A., 2009. Assessing the documentation development effort in software projects. In: International Conferences on Software Process and Product Measurement.
Schoeffel, W.L., 1976. An Air Force Guide to Software Documentation Requirements: Defense Technical Information Center. NTIS AO-A027 051.
Schoonewille, H.H., Heijstek, W., Chaudron, M., Kühne, T., 2011. A cognitive perspective on developer comprehension of software design documentation. In: Proceedings of the 29th ACM International Conference on Design of Communication, Pisa, Italy, pp. 211–218.
Schreck, D., Dallmeier, V., Zimmermann, T., 2007. How documentation evolves over time. In: Proceedings of the 9th International Workshop on Principles of Software Evolution: In Conjunction with the 6th ESEC/FSE Joint Meeting, Dubrovnik, Croatia, pp. 4–10.
Sheppard, S.B., Bailey, J.W., Kruesi, E., 1981. The effects of the symbology and spatial arrangement of software documentation in a modification task. In: Proceedings of the 5th International Conference on Software Engineering, San Diego, CA, United States, pp. 207–214.
Sheppard, S.B., Kruesi, E., Bailey, W.J., 1982. An empirical evaluation of software documentation formats. In: Proceedings of the Conference on Human factors in Computing Systems, Gaithersburg, MD, United States, pp. 121–124.
Soloway, E., Lampert, R., Letovsky, S., Littman, D., Pinto, J., 1988. Designing documentation to compensate for delocalized plans. J. Commun. ACM 31, 1259–1267.
Sommerville, I., 2001. Software Documentation. In: Revised Version of Chapter 30 From his Book Software Engineering, 4th ed. Pearson Education Ltd.
Stettina, C.J., Heijstek, W., 2011. Necessary and neglected? An empirical study of internal documentation in agile software development teams. In: Proceedings of the 29th ACM International Conference on Design of Communication, Pisa, Italy, pp. 159–166.
Su, M.T., 2010. Capturing exploration to improve software architecture documentation. In: Proceedings of the 4th European Conference on Software Architecture: Companion Volume, Copenhagen, Denmark, pp. 17–22.
Tang, A., Babar, M.A., Gorton, I., Han, J., 2005. A survey of the use and documentation of architecture design rationale. In: Presented at the Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture.
Tilley, S., 2009. Documenting software systems with views VI: lessons learned from 15 years of research & practice. In: Proceedings of the 27th ACM International Conference on Design of Communication, Bloomington, IN, USA, pp. 239–244.
Tilley, S., Hausi, M., 1991. INFO: a simple document annotation facility. In: Proceedings of the 9th Annual International Conference on Systems Documentation, Chicago, IL, United States, pp. 30–36.
Tilley, S., Huang, S., 2003. A qualitative assessment of the efficacy of UML diagrams as a form of graphical documentation in aiding program understanding. In: Presented at the Proceedings of the 21st Annual International Conference on Documentation, San Francisco, CA, USA.
Tilley, S.R., Müller, H.A., Orgun, M.A., 1992. Documenting software systems with views. In: Proceedings of the 10th Annual International Conference on Systems Documentation, Ottawa, Ontario, Canada, pp. 211–219.
Trese, T., Tilley, S., 2007. Documenting software systems with views V: towards visual documentation of design patterns as an aid to program understanding. In: Proceedings of the 25th Annual ACM International Conference on Design of Communication, El Paso, TX, USA, pp. 103–112.
Tryggeseth, E., 1997. Report from an experiment: impact of documentation on maintenance. J. Empir. Softw. Eng. 2, 201–207.
Visconti, M., Cook, C., 1993. Software system documentation process maturity model. In: Proceedings of the ACM Conference on Computer Science, Indianapolis, IN, United States, pp. 352–357.
Visconti, M., Cook, C., 2004. Assessing the state of software documentation practices. Product focused software process improvement, vol. 3009. Springer, Berlin/Heidelberg, pp. 485–496.
Wayne, G.L., Carolyn, B.S., 2007. Revealing actual documentation usage in software maintenance through war stories. Inform. Softw. Technol. 49, 576–587.
Wingkvist, A., Ericsson, M., Lincke, R., Lowe, W., 2010a. A metrics-based approach to technical documentation quality. In: Proceedings of the 7th International Conference on the Quality of Information and Communications Technology., pp. 476–481.
Wingkvist, A., Ericsson, M., Löwe, W., 2011. A visualization-based approach to present and assess technical documentation quality. Electron. J. Inform. Syst. Eval. 14, 150–159.
Wong, K., Tilley, S.R., 2002. Connecting technical communicators with technical developers. In: Proceedings of the 20th Annual International Conference on Computer Documentation, Toronto, Ontario, Canada, pp. 258–262.

Unavailable Papers

Bayer, J., 2004. View-based Software Documentation: Fraunhofer. IRB Verlag, Stuttgart.
Comer, E.R., 1983. Rigorous software engineering standards through progressive documentation. In: Second Software Engineering Standards Application Workshop, IEEE.

Prause, C., Kuck, J., Apelt, S., Oppermann, R., Cremers, A.B., 2007. Interconnecting documentation-harnessing the different powers of current documentation tools in software development. In: Ninth ICEIS, pp. 63–68.
Robles, G., Barahona, J.M.G., Prieto, J.L., 2006. Assessing and evaluating documentation in Libre software projects. In: Workshop on Evaluation Frameworks for Open Source Software (EFOSS 2006), Como, Italy.

Excluded Papers

Alan, S.N., Roger, M.S., 1983. Playback: a method for evaluating the usability of software and its documentation. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Boston, MA, United States, pp. 78–82.
Alberto, S., Martina, C., Barbara, R., Giancarlo, S., 2005. Managing uncertainty in requirements: a survey in documentation-driven and agile companies. In: Proceedings of the 11th IEEE International Software Metrics Symposium, Washington, DC, USA, pp. 17–27.
Ali, S., Briand, L.C., Hemmati, H., Panesar-Walawege, R.K., 2010. A systematic review of the application and empirical investigation of search-based test-case generation. IEEE Trans. Softw. Eng. 36, 742–762.
Ambler, S.W., 2002. Agile Modeling: Effective Practices for EXtreme Programming and the Unified Process. John Wiley and Sons Inc., New York.
Ashley, W., 2004. The documentation of quality engineering: applying use cases to drive change in software engineering models. In: Proceedings of the 22nd Annual International Conference on Design of Communication: The Engineering of Quality Documentation, Memphis, TN, USA, pp. 4–13.
Azmi, A., Ibrahim, S., 2011. Test management traceability model to support software testing documentation. In: Proceedings of the International Conference on Digital Information and Communication Technology and its Application, Dijon, France, pp. 21–32.
Barker, T.T., 1990. Software documentation: from instruction to integration. IEEE Trans. Prof. Commun. 33.
Bauer, B.J., Parnas, D.L., 1995. Applying mathematical software documentation: an experience report. In: Proceedings of the 10th Annual Conference on Computer Assurance, Gaithersburg, MD, pp. 273–284.
Beck, K., Beedle, M., Bennekum, A.v., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D., 2012. Manifesto for Agile Software Development, http://www.agilemanifesto.org (Last accessed: 29.06.12).
Benson, H.S., Tom, G., 1988. Letting software engineers do software engineering or freeing software engineers from the shackles of documentation. In: Proceedings of the 6th Annual International Conference on Systems Documentation, Ann Arbor, MI, United States, pp. 81–92.
Berglund, E., 2000. Writing for adaptable documentation. In: Proceedings of IEEE Professional Communication Society International Professional Communication Conference and Proceedings of the 18th Annual ACM International Conference on Computer Documentation: Technology & Teamwork, Cambridge, MA, pp. 497–508.
Berglund, E., Priestley, M., 2001. Open-source documentation: in search of user-driven, just-in-time writing. In: Proceedings of the 19th Annual International Conference on Computer Documentation, Sante Fe, NM, USA, pp. 132–141.
Bethke, F.J., Dean, W.M., Kaiser, P.H., Ort, E., Pessin, F.H., 1991. Improving the usability of programming papers. IBM Syst. J. 20, 306–320.
Bill, C., Herb, K., Neil, I., 1988. A field study of the software design process for large systems. CACM 31, 1268–1287.
Briand, L.C., 2003. Software documentation: how much is enough? In: Seventh European Conference on Software Maintenance and Reengineering, Benevento, Italy, pp. 13–15.
Clark, P.G., Lobsitz, R.M., Shields, J.D., 1989. Documenting the evolution of an information system. In: Proceedings of the IEEE 1989 National Aerospace, Electronics Conference, Dayton, OH, pp. 1819–1826.
Clear, T., 2003. Documentation and Agile Methods: Striking a Balance, vol. 35. ACM Special Interest Group on Computer Science Education Bulletin, pp. 12–13.
Clements, P., Garlan, D., Little, R., Nord, R., Stafford, J., 2003. Documenting software architectures: views and beyond. In: Proceedings of the 25th International Conference on Software Engineering, Portland, OR, pp. 740–741.
Courtois, P.J., Parnas, D.L., 1993. Documentation for safety critical software. In: Proceedings of the 15th International Conference on Software Engineering, Baltimore, MD, United States, pp. 315–323.
de Rosis, F., de Carolis, B., Pizzutilo, S., 1999. Software Documentation with Animated Agents (Unpublished).
Delanghe, S., 2000. Using learning styles in software documentation. IEEE Trans. Prof. Commun. 43, 201–205.
Forward, A., Lethbridge, T.C., 2014. Software Engineering Documentation Priorities: An Industrial Study [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.112.7359&rep=rep1&type=pdf
French, J.C., Knight, J.C., Powell, A.L., 1997. Applying hypertext structures to software documentation. Inform. Process. Manag. 33, 219–231.
Galdo, E.M.D., Williges, R.C., Williges, B.H., Wixon, D.R., 1986. An evaluation of critical incidents for software documentation design. In: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, Anaheim, CA, pp. 19–23.
Gerth, T., Schachtschabel, R., Schönefeld, R., Unpublished.
Guillemette, R.A., 1986. Application software documentation: a reader measure, vol. 17. Special Interest Group of the ACM on Management Information Systems Database, pp. 40.
Haneef, N.J., 1998. Software documentation and readability: a proposed process improvement. ACM SIGSOFT Softw. Eng. Notes 23, 75–77.
Hargis, G., 2000. Readability and computer documentation. ACM J. Comput. Doc. 24, 122–131.
Hargis, G., Carey, M., Hernandez, A.K., Hughes, P., Longo, D., Rouiller, S., Wilde, E., 2004. Developing Quality Technical Information: A Handbook for Writers and Editors, 2nd ed. Prentice Hall, PTR Upper Saddle River, NJ, USA.
Horowitz, E., Williamson, R., 1985. SODOS: a software documentation support environment – its use. In: Proceedings of the 8th International Conference on Software Engineering, London, England, pp. 8–14.
Hyland-Wood, D., Carrington, D., Kaplan, S., 2008. Towards a software maintenance methodology using Semantic Web techniques and paradigmatic documentation modelling. Softw. IET 2, 337–347.
Jazzar, A., Scacchi, W., 1995. Understanding the requirements for information system documentation: an empirical investigation. In: Proceedings of Conference on Organizational Computing Systems, Milpitas, CA, United States, pp. 268–279.
Johnson, W.L., Rey, M.D., 1994. Dynamic (re)generation of software documentation. In: Proceedings of the 4th Systems Reengineering Technology Workshop, Monterey, CA, USA, pp. 57–66.
Kalus, D., Torsten, K., 1996. A pragmatic approach to software documentation. Technical report, Technische Universitat Berlin, Germany.
Landes, D., Schneider, K., Houdek, F., 1999. Organizational learning and experience documentation in industrial software projects. Int. J. Hum. Comput. Stud. 51, 643–661.
Landis, L.D., Hyland, P.M., Gilbert, A.L., Fine, A.J., 1988. Documentation in a software maintenance environment. In: Proceedings of the Conference on Software Maintenance, Scottsdale, AZ, pp. 66–73.
Liu, S., 2001. Generating Test Cases from Software Documentation. MSc. Thesis, Master of Engineering, Department of Electrical and Computer Engineering, McMaster University.
Lobato, L.L., O'Leary, P., de Almeida, E.S., de Lemos Meira, S.R., 2010. The importance of documentation, design and reuse in risk management for SPL. In: Proceedings of the 28th ACM International Conference on Design of Communication, Carlos, Paulo, Brazil, pp. 143–150.
Luqi, X., Liang, L., Zhang, V., Berzins, 2003. Software documentation-driven manufacturing: viaduct between software engineering and virtual engineering. In: Proceedings of the 27th Annual International Conference on Computer Software and Applications, Dallas, TX, pp. 472–477.
Lutsky, P., Logananthara, R., Palm, G., Ali, M., 2000. Information Extraction for Validation of Software Documentation Intelligent Problem Solving, Methodologies and Approaches, vol. 1821. Springer, Berlin/Heidelberg, pp. 29–60.
MacKinnon, N., Murphy, S., 2004. Designing UML diagrams for technical documentation: continuing the collaborative approach to publishing class diagrams. In: Proceedings of the 22nd Annual International Conference on Design of Communication: The Engineering of Quality Documentation, Memphis, TN, USA, pp. 120–127.
Magyar, M., 2000. Automating software documentation: a case study. In: Proceedings of the IEEE Professional Communication Society International Professional Communication Conference and Proceedings of the 18th Annual ACM International Conference on Computer documentation: Technology & Teamwork, Cambridge, MA, pp. 549–558.
Marcus, A., Maletic, J.I., 2005. Recovery of traceability links between software documentation and source code. Int. J. Softw. Eng. Knowl. Eng. 15, 811–836.
Marovac, N., 1994. Guidelines for embedded software documentation. ACM SIGSOFT Softw. Eng. Notes 19, 22–28.
Marovac, N., 1998. Embedded documentation for semi-automatic program construction and software reuse. ACM SIGSOFT Softw. Eng. Notes 23, 70–74.
Medina, E.A., 1984. Some aspects of software documentation. In: Proceedings of the 3rd Annual International Conference on Systems Documentation, Mexico City, Mexico, pp. 57–59.
Metzger, A., Heymans, P., Pohl, K., Schobbens, P.-Y., Saval, G., 2007. Disambiguating the documentation of variability in software product lines: a separation of concerns, formalization and automated analysis. In: Proceedings of the 15th IEEE International Conference on Requirements Engineering, Delhi, India, pp. 243–253.
Moallem, A., 2003. Usability of Software Online Documentation: A User Study. Available: http://sjsulug.engr.sjsu.edu/amoallem/papers/834_HCI_Usability_AMoallem.pdf
Odenthal, G., Quibeldey-Cirkel, K., 1997. Using patterns for design and documentation. In: Proceedings of the European Conference of Object Oriented Programming, Berlin, pp. 511–529.
Parnas, D.L., 2005. A family of mathematical methods for professional software documentation. In: Proceedings of the 5th international Conference on Integrated Formal Methods, Eindhoven, The Netherlands, pp. 1–4.
Parnas, D.L., Smith, D., Pearce, T., 1988. Making formal software documentation more practical: a progress report. Technical Report, Dept. of Computing & Information Science, Queens University, Canada.
Peters, D.K., Parnas, D.L., 1998. Using test oracles generated from program documentation. IEEE Trans. Softw. Eng. 24, 161–173.
Phoha, V., 1997. A standard for software documentation. IEEE Comput. 30, 97–98.
Powell, A.L., French, J.C., Knight, J.C., 1996. A systematic approach to creating and maintaining software documentation. In: Proceedings of the ACM symposium on Applied Computing, Philadelphia, PA, United States, pp. 201–208.
Price, J., 1984. How to Write a Computer Manual: A Handbook of Software Documentation. Benjamin/Cummings Pub. Co.
Price, J., Korman, H., 1993. How to Communicate Technical Information: A Handbook of Software and Hardware Documentation. Benjamin/Cummings Pub. Co.

Rob, P., 2003. Optimizing your documentation with the help of technical support. In: Proceedings of the 21st Annual international Conference on Documentation, San Francisco, CA, USA, pp. 6–11.
Ruth, A., 1997. Evaluating Generalized Tabular Expressions In Software Documentation. Master of Engineering, Faculty of Engineering, McMaster University, Hamilton.
Sabou, M., 2004. Extracting ontologies from software documentation: a semi-automatic method and its evaluation. In: Proceedings of the Workshop on Ontology Learning and Population (ECAI-OLP), Valencia, Spain.
Tang, A., Liang, P., van Vliet, H., 2011. Software architecture documentation: the road ahead. In: 9th Working IEEE/IFIP Conference on Software Architecture, Boulder, CO, pp. 252–255.
Taulavuori, A., Niemela, E., Kallio, P., 2004. Component documentation – a key issue in software product lines. Inform. Softw. Technol. 46, 535–546.
Tausworthe, R.C., 1976. Standard Classification of Software Documentation. Jet Propulsion Laboratory, Pasadena, CA.
Thomas, B., Tilley, S., 2001. Documentation for software engineers: what is needed to aid system understanding? In: Proceedings of the 19th Annual International Conference on Computer Documentation, Sante Fe, NM, USA, pp. 235–236.
Tilley, S.R., 1993. Documenting-in-the-large vs. documenting-in-the-small. In: Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research: Distributed Computing, vol. 2, Toronto, Ontario, Canada, pp. 1083–1090.
Van der Meij, H., 2000. The role and design of screen images in software documentation. J. Comput. Assist. Learn. 16, 294–306.
Visconti, M., Cook, C.R., 2002. An overview of industrial software documentation practice. In: Proceedings of the 12th International Conference of the Chilean Computer Science Society, Valparaiso, Chile, pp. 179–186.
Walters, N.J., Beck, C.E., 1992. A discourse analysis of software documentation: implications for the profession. IEEE Trans. Prof. Commun. 35, 156–167.
Wang, X., Lai, G., Liu, C., 2009. Recovering relationships between documentation and source code based on the characteristics of software engineering. Electron. Notes Theor. Comput. Sci. 243, 121–137.
Wingkvist, A., Löwe, W., Ericsson, M., Lincke, R., 2010b. Analysis and visualization of information quality of technical documentation. In: Proceedings of the 4th European Conference on Information Management and Evaluation, Reading, UK, pp. 388–396.
Wise, M.R., 1993. Using graphics in software documentation. J. Soc. Tech. Commun. 40, 677–681.

Other References

Banerjee, I., Nguyen, B.N., Garousi, V., Memon, A.M., 2013. Graphical user interface (GUI) testing: systematic mapping and repository. Inform. Softw. Technol. 55 (10), 1679–1694.
Budgen, D., Turner, M., Brereton, P., Kitchenham, B., 2008. Using mapping studies in software engineering. Evidence-Based Softw. Eng. 8, 195–204.
Elberzhager, F., Münch, J., Nha, V.T.N., 2012. A systematic mapping study on the combination of static and dynamic quality assurance techniques. Inform. Softw. Technol. 54, 1–15.
Garousi, V., 2012. Classification and trend analysis of UML books (1997–2009). J. Softw. Syst. Model. (SoSyM) 11 (2), 273–285.
Garousi, V., Varma, T., 2010. A replicated survey of software testing practices in the Canadian Province of Alberta: what has changed from 2004 to 2009? J. Syst. Softw. 83 (November), 2251–2262.
Garousi, V., Mesbah, A., Betin-Can, A., Mirshokraie, S., 2013. A systematic mapping study of web application testing. Inform. Softw. Technol. 55, 1374–1396.
Jia, Y., Harman, M., 2011. An analysis and survey of the development of mutation testing. IEEE Trans. Softw. Eng. 37, 649–678.
Jia, Y., Harman, M., 2012. Mutation testing repository. http://www.dcs.kcl.ac.uk/pg/jiayue/repository (Last accessed: April 2012).
Kitchenham, B., Charters, S., 2007. Guidelines for performing systematic literature reviews in software engineering. Evidence-Based Softw. Eng., Technical report, EBSE Technical Report EBSE-2007-01.
Kitchenham, B., Brereton, O.P., Budgen, D., Turner, M., Bailey, J., Linkman, S., 2009. Systematic literature reviews in software engineering – a systematic literature review. Inform. Softw. Technol. 51, 7–15.
Kitchenham, B., Brereton, P., Budgen, D., 2010. The educational value of mapping studies of software engineering literature. In: ICSE 2010 Education Theme, pp. 1–7.
Kitchenham, B.A., Budgen, D., Brereton, P., 2011. Using mapping papers as the basis for further research – a participant–observer case study. Inform. Softw. Technol. 53, 638–651.
Lientz, B.P., Swanson, E.B., Tompkins, G.E., 1978. Characteristics of application software maintenance. Commun. ACM, 466–471.
Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M., 2008. Systematic mapping papers in software engineering. In: Presented at the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE).
Pfleeger, S.L., Atlee, J.M., 2010. Software Engineering: Theory and Practice, 4th ed. Pearson.
Runeson, P., Host, M., Rainer, A., Regnell, B., 2012. Case Study Research in Software Engineering: Guidelines and Examples. John Wiley & Sons.
Software Engineering Evidence Map, 2012. http://www.dur.ac.uk/ebse/evidence.php (Last Accessed: August 2012).
UML meta-model Infrastructure specification, 2012. http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF (Last accessed: August 2012).
Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2000. Experimentation in software engineering: an introduction. Kluwer Academic Publishers.
Zhang, Y., 2012. Repository of papers on search based software engineering. http://crestweb.cs.ucl.ac.uk/resources/sbse_repository (Last accessed: April 2012).
Zhang, H., Babar, M.A., Tell, P., 2011. Identifying relevant papers in software engineering. Inform. Softw. Technol. 53.
Zhi, J., Garousi, V., Sun, B., Garousi, G., Shahnewaz, S., Ruhe, G., 2012. Cost, Benefits, Usage and Quality of Technical Software Documentation System Mapping Repository (Online), http://www.softqual.ucalgary.ca/projects/SM/doc_cost_benefit_usage_quality (Last accessed: 29.06.12).

Junji Zhi is a PhD student in Computer Systems and Networks Research Group, Department of Computer Science, University of Toronto, Canada. He completed his M. Sc. degree at University of Calgary from 2011 to 2013. Junji's research interests include cloud computing, software testing and software documentation. For more information, please visit: http://www.cs.toronto.edu/~zhij/.

Vahid Garousi is an Associate Professor of Software Engineering in Atilim University in Ankara, Turkey. He served as an Associate Professor of Software Engineering in the University of Calgary, Canada from 2006 until 2014. Vahid completed his PhD in Software Engineering in Carleton University, Canada, in 2006, and his M. Sc. in the University of Waterloo, Canada, in 2003. He is a practicing software engineering consultant and coach, and has provided consultancy and corporate training services in Canada and Turkey in the areas of software testing and quality assurance, model-driven development, and software maintenance. During his career, Vahid has been active in initiating a number of major R&D software testing and software engineering projects in Canada and Turkey. During year 2013, he was a Visiting Associate Professor in the Middle East Technical University (METU) in Ankara, Turkey. He has been involved as an organizing or program committee member in many software engineering conference, such as ICST, ICSP, CSEE&T, and MoDELS. He is a member of the IEEE and the IEEE Computer Society, and is also a licensed professional engineer (PEng) in the Canadian province of Alberta. He has been selected a Distinguished Visitor (speaker) for the IEEE Computer Society's Distinguished Visitors Program (DVP) for the period of 2012–2015. Among his awards is the Alberta Ingenuity New Faculty Award in June 2007. For more information, visit: www.atilim.edu.tr/~vahid.garousi.

Bo Sun is a Data Management Consultant at iSolutions Inc. He completed his M. Sc. Degree at University of Calgary from 2010 to 2012.

Golara Garousi is a software engineer at geoLOGIC Systems in Calgary, Canada. She received her M. Sc. in software engineering from the University of Calgary, Canada in 2012. Her research interests are software development documentation, software engineering decision support and software quality assurance.

Shawn Shahnewaz is Master student at the Department of Electrical and Computer Engineering, University of Calgary since 2012.

Guenther Ruhe holds an Industrial Research Chair in Software Engineering at University of Calgary. From 1996 until 2001, he was the deputy director of the Fraunhofer Institute for Experimental Software Engineering (Fh IESE). Ruhe received an iCORE research award for the period 2001 to 2007. Since 2007, he had served as an Associate Editor of the Journal of Information and Software Technology, published by Elsevier. His main research interests are in the areas of Product Release Planning, Software Project Management, Empirical Software Engineering as well as Search-based Software Engineering. He is a Senior member of IEEE and a member of the ACM. Dr. Ruhe is the Founder and CEO of Expert Decisions Inc., a University of Calgary spin-off company created in 2003.