CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

How software documentation helps communication in development teams: A case study on architecture and design documents

Bachelor of Science Thesis in Software Engineering and Management

Omid Manai

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Supervisor: Michel R. V. Chaudron
Examiner: Richard Berntsson Svensson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

How software documentation helps communication in development teams: A case study on architecture and design documents

Omid Manai
Computer Science and Engineering Department
University of Gothenburg
Gothenburg, Sweden
gusmanom@student.gu.se

***Abstract*—[Context] Communication between developers and within development teams takes place through various communication channels. Software documentation acts as a communication channel among software professionals. But, to what extent documentation can help communication in development teams? [Objective] This paper evaluates how software documentation and specifically architecture and design documents help communication among/within development teams. [Method] We performed a multiple-case study at the IT departments of two Swedish manufacturing companies. The research method followed a qualitative approach consisting of a survey with 24 participants, two semi-structured interviews, and two work diaries. [Results] By performing the case study, the following results were derived: (1) software documentation complements communication rather than replacing it; (2) documentation usage frequency depends on its up-to-dateness and accuracy; (3) the main reasons for using documentation are assistance in development and maintenance phases, knowledge transfer and architecture comprehension, although incomplete/outdated documentation is the main concern; (4) architecture and design documents complement communication channels, and their usage is affected by company policies and education/employment background; (5) knowledge evaporation results in excessive time and cost consumption however, software documentation is a possible remedy to that. [Conclusions] It is concluded that design documents and documentation in general mainly complement communication in development teams and avoid knowledge evaporation, however, their usage depends on their accuracy, company policies and employees' background.**

***Index Terms*—documentation, communication, architecture and design documents, software development, knowledge evaporation**

I. Introduction

Software development is no longer the task of a mere individual to develop standalone software, and it has shifted to a collaborative activity, highly dependent on shared understanding [1], [2]. Software developers seek to communicate in order to collaborate and contribute to various development efforts. Documentation is one of the means of conveying information within projects, and therefore, it is a form of communication among various project members [3], [4], [5]. In this research, the software documentation that we mainly consider are product and process documentation, including system documentation. The documents can be any type of artefact related to a software system which holds information relevant to that, e.g., requirements, process, architecture and design, testing, code comments, etc. Although relevant research have been conducted regarding the importance and inseparability of documentation from software development processes, there is a lack of research in how design documents and documentation in general help with communication in development teams.

Effective and productive communication is a substantial need in team-work software development efforts; however, to avoid incompleteness and vagueness, communication requires a basis and the means which serve its needs. Forward [4] has argued that by comprehending our natural limitations for effective communication, documentation can serve and fulfill the purpose of productive communication. Storey et al. [5] have found that software developers have various communication channels in order to share externalized knowledge (or explicit knowledge, the knowledge that has been recorded in a certain form, e.g., code, model, documentation [6]) and tacit knowledge (the knowledge that resides in people's head and cannot be accessed, e.g., experience of working with various tools, specific details regarding systems [6]) including software documentation. However, the extent to which documentation and specifically design documents can help communication in development teams and their activities is unknown. For instance, is documentation capable of being the primary communication channel replacing verbal communication in a software development activity?

In this paper, the author is exploring the feasibility and effect of having documentation and specifically design documents as a means of communication. The results of a multiple-case study, including the data triangulation of three data collection methods are presented:

- A qualitative survey on developers contributing to collaborative software development activities. The survey investigates the views on software documentation as a communication channel and its effect on knowledge evaporation.
- Semi-structured interviews with selected participants from two software development organizations who have

participated in the survey as well. The interviews expand on the survey by studying what developers expect of documentation as a communication medium, their reasons for using documentation, and documentation's role in knowledge disappearance.
- Work diaries of the same developers from the two software development organizations who have participated in the survey. Work diaries mainly focus on the usage frequency of software documentation and design documents. They also seek to find the primary reasons for which the developers seek software documentation.

The case study intends to further research the relation between documentation and communication, and explore the effects of documentation on communication among/within software development teams. Recent studies have focused on various types of software documentation, and their views on them among software professionals. However, there have not been any research on how documentation specifically helps or affects communication. Moreover, knowledge evaporation and disappearance were not studied in the context of documentation. Furthermore, this study confirms and extends existing results on the reasons in which developers seek to use documentation, and any findings related to software documentation as a communication channel. The study results will benefit software professionals including software developers, maintainers, architects, and middle management to have a better understanding of software documentation as a communication channel, and the extent to which documentation can serve as a channel. Moreover, software decision-makers can benefit from the results in order to design tools and features which will improve software documentation.

The rest of the paper is structured as follows: section II reviews the existing literature on the topic. Section III introduces the research objectives, research questions, and the research methodology. Section IV presents the findings of the case study, along with discussions and interpretations of the results. Section V discusses threats to validity. And finally, section VI draws conclusions from the paper and explains possible future research.

II. Review of the literature

Various research have been conducted on the usage and benefits of software documentation. Ding et al. [6] state that software documentation is an indivisible part of the software development process in which it acts as a communication medium for stakeholders and especially among individuals and teams that are geographically distributed. They further state that software documentation is capable of transforming tacit knowledge into explicit knowledge, and minimizing the risk of knowledge evaporation. Similarly, Glinz and Fricker [1] believe that due to the complexity of software, implicit shared understanding is not sufficient in software development processes, and access to explicit documentation is crucial. Moreover, they state that software systems evolve and requirements and design decisions that are in the form of implicit shared knowledge can get lost; therefore they suggest that such information are better to be recorded as explicit documentation. However, they further elaborate that merely relying on explicit shared understanding in the form of documentation is economically incorrect as it wastes considerable amounts of cost and time.

Accordingly, Storey et al. [5] have found that along with three main communication channels for software developers that are code hosting sites, face-to-face interactions, and Q&A sites, developers have also mentioned software documentation as a communication channel for learning, finding answers, and being up-to-date. Moreover, Garousi et al. [7] conducted an industrial case study and investigated the usage and usefulness of technical software documentation. They found that software developers use documentation in order to comprehend and understand a system both during and after the development cycle. They state that quality technical documentation (e.g., software architecture and design documents), 1) allow communication among team members, and 2) reduce the risk of domain-specific knowledge disappearance. Moreover, they explain that software engineers communicate their required information either through face-to-face interactions with their colleagues and/or by referring to the documentation.

On the other hand, Curtis et al. [3] designed a field study of 17 unstructured interviews with personnel from various large system development projects. They investigated the problems of designing such systems at individual, team, project, and company levels. At the individual level, they found that although documentation is a form of communication, documentation is a weak communication medium, and incomplete documentation is problematic. Moreover, they state that written documentation does not entirely serve the communication needs of teams. Although their study is focused on large software systems, some of their findings are challenged in our study. However, we affirm some of the mentioned software documentation issues as well.

Fernández-Sáez et al. [8] conducted a single-case study by interviewing 31 software maintainers regarding the use and the purpose of design documents and specifically UML diagrams. They found that more than half of the interviewees perceive UML diagrams as a communication tool, i.e., allowing communication within or among teams, transferring knowledge to newcomers, and other stakeholders. Interviewees have also expressed that design documents and specifically UML diagrams help software professionals to complement their verbal or textual communication as it follows a standard notation known by the software industry. In regards to software documentation relevance, Forward and Lethbridge [9] describe that software documentation is an essential communication tool which should follow a purpose and transfer knowledge to its intended audience whether the knowledge is up-to-date or not, just like any other means of communication. Building on Forward and Lethbridge's study, Lethbridge et al. [10] found that software documentation can be the only communication channel for developers when the main developers who designed the software system are not available. Moreover, they found that abstract documentation mainly in

TABLE I
INTERVIEWEES BACKGROUND

Participant	Experience	Context	Educational field	Educational level	Company role
[Int1], [Diarist1]	High	Front-end DevOps	Computer science	Master's degree	DevOps engineer
[Int2], [Diarist2]	Very high	-	Computer science	Master's degree	Delivery lead Developer

the form of architecture and design documents do not have to be updated to impart knowledge in contrast to testing and quality documents that require high accuracy.

Relevant to the related works, our case study delves into the relationship between software documentation and its effect on communication among and within software development teams.

III. RESEARCH METHODOLOGY

A. Research questions

The authors seek to answer the following research questions:

- **RQ1)** *What are the views on software documentation as a communication channel?*
- **RQ2)** *How often do developers refer to software documents and specifically architecture and design documents?*
- **RQ3)** *Why do/do not software developers seek to use software documentation?*
- **RQ4)** *In what way do/do not design documents help software developers to communicate within/among development teams?*
- **RQ5)** *How does knowledge evaporation affect software development teams, and what role can software documentation play in that?*

B. Multiple-Case Study

In what follows, we mainly follow the descriptions and methodologies proposed by Runeson and Höst [11], and Yin [12] regarding case study research. A multiple-case study research method has been chosen for this research as this method allows studying a phenomenon in its real-life context [11]. Case study research seeks to have an in-depth focus on a specific number of cases to better understand their dynamics [12], [13]. Yin [12] states that suitable research questions for the case study method are either explanatory questions or descriptive questions (e.g., how/why, or what), very much the same as the research questions in this study. Our multiple-case study follows exploratory empirical research by seeking and exploring insights regarding a specific phenomenon and deriving new ideas and theories from that [11]. The multiple-case study relies on multiple sources of evidence and compared to a single-case study, it results in more convincing findings [12].

The authors have selected two software development companies as the two cases of the multiple-case study. The first company (Company A) was the IT department of a Swedish multinational manufacturing company. The second company (Company B) was the IT department of a manufacturing company based in Gothenburg. The main participants of the study were software developers working in various teams within the organizations. Although developers were the main subjects of the study, architects, testers, maintainers, and middle management were also encouraged to participate in the study, in order to have a richer picture of the phenomenon.

In each case, we had one data point as a participant of our research, and the participants' backgrounds are presented in table I.

Furthermore, we chose to do data triangulation to increase confidence in the research data and give a better picture of the phenomenon from different viewpoints [14]. For the data triangulation, three different data sources are chosen, interviews, work diaries, and a qualitative survey in the form of a questionnaire. Regarding the order of data collection, first, the survey was conducted during a time span of 24 days from April 23, 2019, to May 17, 2019. On May 1, 2019, the authors started the interview process by doing the first interview on the same day. In the meanwhile, on May 2, 2019, the work diary templates were sent to the participants. The case study approach is represented in figure 1.

C. Data collection

Interviews – From the six sources of evidence that Yin [12] suggests, the authors have chosen to run semi-structured interviews with the subjects. Semi-structured questions seemed more appropriate with qualitative research in which they would allow the authors to further explore the opinions and perceptions of the subjects on critical issues and in-depth clarifications [15].

In the context of the case study, we performed two interviews, one with the participant from company A ([Int2]), and the other with the participant from company B ([Int1]). Both interviewees have had experience in software development in their entire work experience, and they were currently occupied with software development activities within their companies. All interviewees received open-questions with no predefined choices in order to allow them to better explain their ideas and express themselves.

The first two questions represent demographic questions, and the specific ones which have *'pre'* notions were accompanied by an explanation of the topic and some examples allowing the interviewee to have a better understanding of the question. Moreover, prior to the interviewees receiving the interview questions, they received a short introduction to the interview objectives, their anonymity, how the interview data was going to be handled, and a description of the technical terms including various communication channels, different software documentation types, etc. Both interviews

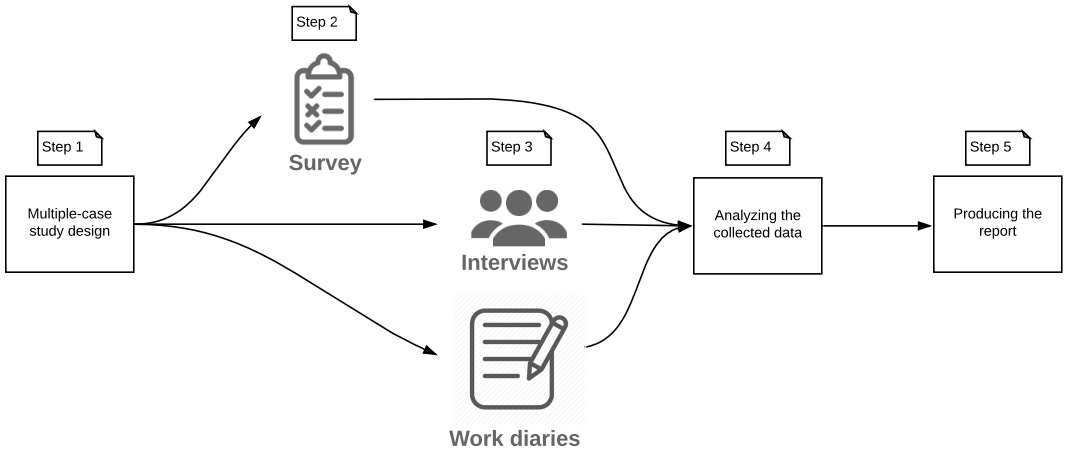

Fig. 1. Case study approach

took from 30 minutes to 45 minutes in order to cover all the structured questions and their follow-up questions. The structured interview questions can be found in appendix A, although each question had one or more follow-ups in the interview.

Survey – Survey method gathers information from a group of specific people who represent the sample of a larger population [16], [17]. The survey objective was to collect information on the use of software documentation, how it can act as a communication channel, and its effect on knowledge evaporation. A questionnaire was constructed and used as the survey instrument. The questionnaire consisted of six partially-structured questions and four open questions. The questionnaire can be found in appendix B, and it is also accessible online through this link [18]. The survey population was mainly software developers working in development teams, but software architects, testers, maintainers, and middle IT management were also encouraged to participate in the survey as they are part of software development teams, and their use of software documentation could be evaluated. The survey followed the convenience sampling method, and hence people were asked to participate by accessing the shareable link of the survey in LinkedIn, alternatively the authors contacted specific people and asked for their participation.

At the beginning the survey was limited to the respondents from the two case companies, but it was opened to other companies when the authors found that, 1) the company name is not being recorded in the responses and therefore, the responses can not be traced mainly to the two case companies, 2) the sample group will be extremely limited. The survey was opened to various IT organizations in Gothenburg and Stockholm. As the data was not entirely related to the case companies anymore, the authors removed the survey data from the case study and did not analyze it with the interview and work diary data. However, the survey data is just included in the results as the findings were still showing the opinions of software professionals regarding their usage frequency of documentation and the reasons for that. This also introduced a threat to the construct validity of the study which has been discussed in section V.

As shown in figure 2, the sample group was mainly developers from various IT organizations in Gothenburg and Stockholm, and the sample size reached a total of 24 individuals.

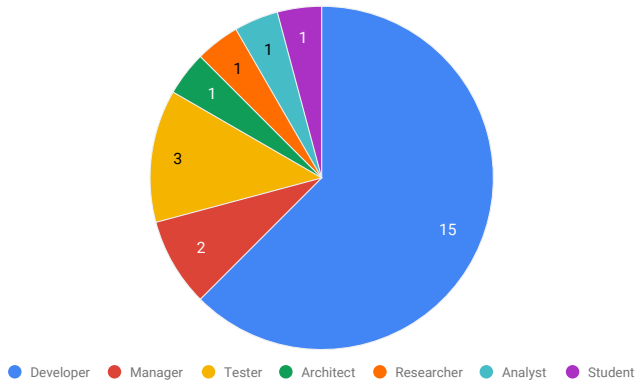

Fig. 2. Survey sample group

Work diary – Additionally, work diaries require the participants to record their flow of thoughts and actions regarding a specific behavior [19]. In this method, diarists were the two participants who have participated in the survey and interview methods as well. The work diary template can be found in appendix C, and it is also available online through this link [20]. To fill the template, diarists received an email having the template attached, and including a description of the diary objectives, how the data is handled, their anonymity, and an explanation of the technical terms including knowledge evaporation, different software documentation types, etc. Diarists had to fill in the work diary templates within a normal working day (8 hours).

By filling the work diary templates, the participants were obliged to record the efforts they made to reach software

documentation by providing 1) the time they have accessed the document, 2) the amount of time they have spent on the document, 3) the type of the software document, and 4) a short description of the reason they have reached the documentation. Moreover, diaries also included questions that were supposed to be filled at the end of the day.

D. Data Analysis

The last phase in the research methodology was analyzing the collected data. As three different sets of data were collected, each dataset was analyzed individually, and the analyses of the three sources were used for interpreting the results and drawing conclusions. Analyses of the interview data and the work diaries [21] were carried out in parallel with the data collection processes of the methods. All three types of collected data were analyzed by performing a thematic analysis, and by following Braun and Clarke's [22] guide for doing a thematic analysis.

The thematic data analysis consists of the following six steps:

1) **Familiarizing with the data**: the interview and work diary data were transcribed into text, and the survey data was gathered into spreadsheets. The data from both sources were read several times, and the initial ideas noted.
2) **Generating codes**: interesting and salient parts of data were highlighted and coded, and the data relevant to specific codes were collated.
3) **Looking for themes**: the codes were focused and grouped into themes.
4) **Reviewing themes**: the themes and codes were compared, and a thematic map of the analysis produced. Also, themes were defined and named clearly in this step.
5) **Defining and naming themes**: the potential themes were refined, and names and concise definitions for each theme were provided.
6) **Producing the report**: examples of the datasets were extracted, and the analysis was used to relate back to the research questions and to finalize the report.

The data analyses of the work diaries and interviews were integrated at the fourth step of the thematic analysis, and their results and interpretations are included in section IV. However, the findings based on the survey data analysis are reported separately from the other two methods in section IV. Also, several charts have been included relating to the results of the semi-structured questions in the survey. The thematic map of the identified themes and codes is shown in figure 4. Although, some of the identified codes have not been used in the case study as they were mainly related to the survey data.

It is worth noting that the authors have used "NVivo" (a computer-assisted qualitative data analysis package) to perform the analysis phase.

IV. Results and Discussion

This section presents the results of the case study, along with the interpretation of the findings and their relation to the published work. The results come from all three data collection methods; survey, interviews, and work diaries. By following Braun and Clarke's [22] guide for doing a theoretical (deductive) thematic analysis, the collected data has been transcribed, coded and, the identified themes are explained excessively in their subsections relevant to the research questions.

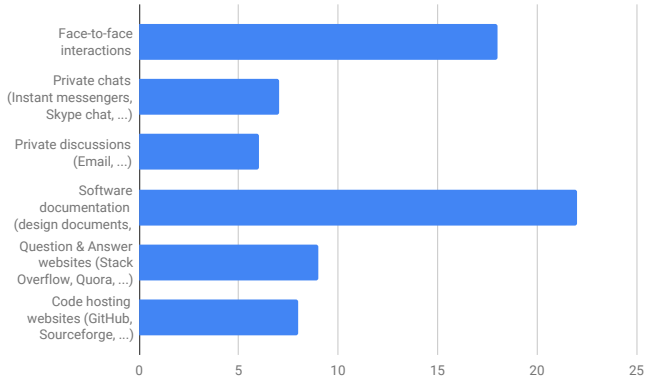

Fig. 3. Which communication channel(-s) do you find more helpful to understand a system?

A. Documentation as a communication channel (RQ1)

The first research inquiry was devoted to the views on software documentation as a communication channel. The second question in the survey specifically asked the participants to reflect their views on software documentation as a communication channel for transferring information among people and projects. The findings show that all of the respondents think that software documentation can act as a communication channel to transfer information among people and projects. Furthermore, the third question in the survey asked the participants about their preferred communication channel(-s), in terms of being helpful for understanding a system. As shown in figure 3, the majority of respondents (91.7%) have chosen software documentation in the first place. The second is face-to-face interactions with 75%, and the rest are Q&A websites 37.5%, code hosting sites 33.3%, private chats (instant messaging, etc.) 29.2%, and private discussions (emails, etc.) 25%.

The thematic analysis on the work diary and interview data shows that the participants from both cases believed that software documentation helps communication among/within development teams in both development and maintenance phases. It allows the team members to understand the overall functionality and architecture of a system: "*software documentation is the most important a maintenance and development team can have for the system they work with. Software documentation is an important channel or an important way to understand the functionality, work with the system, and*

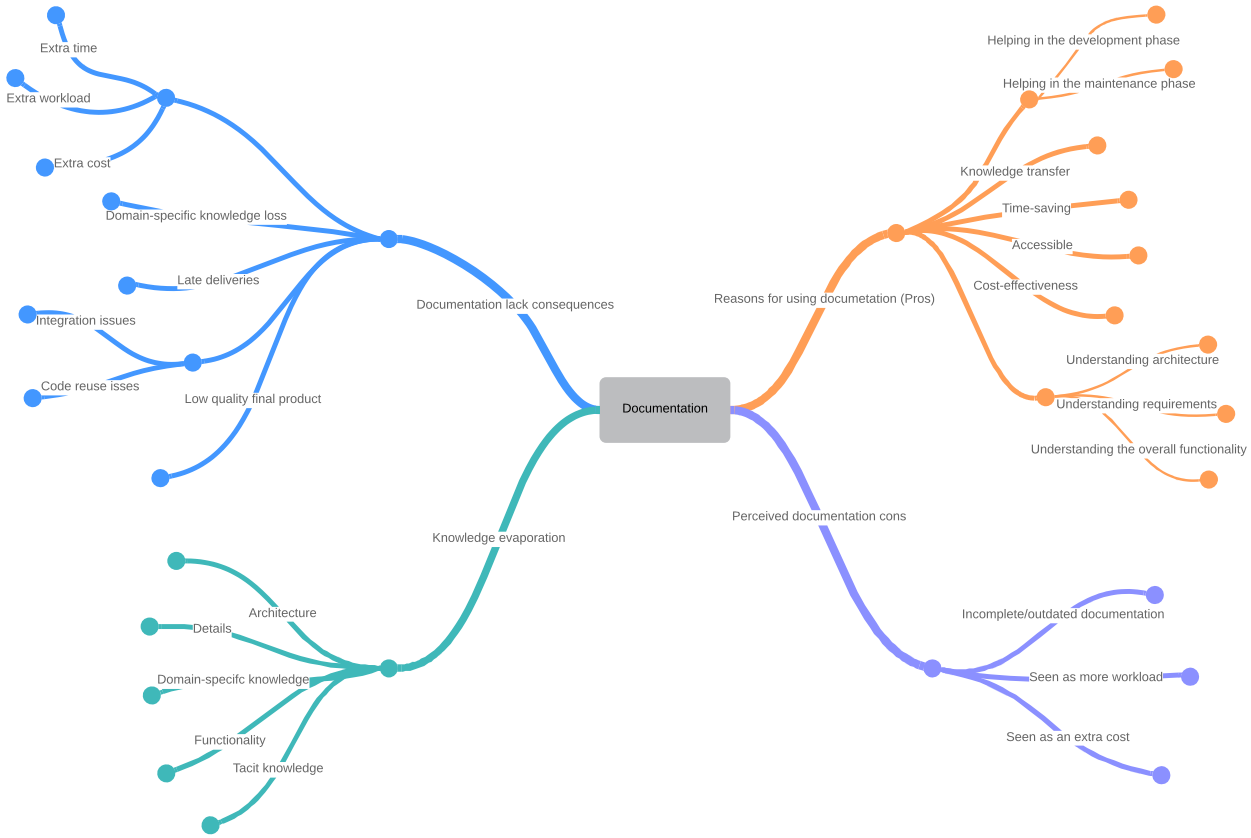

Fig. 4. Thematic Map

further develop it" [Int2]. The interviewee from company B states that documentation acts as a communication channel at certain points: "*Software documentation can act as a communication channel at points. It can act as a communication channel when people want to refer to something which they have discussed before or they want to review it later*"[Int1]. Company A interviewee believes that software documentation should 'ideally' replace face-to-face interactions, verbal communication, and other communication channels when the purpose of communication is for understanding the overall functionality and architecture of a system: "*To act as the primary communication medium and reduce the amount of verbal communication and face-to-face meetings*" [Diarist2].

However, the second interviewee continues that: "*That [having documentation as the primary channel] is the ideal case and that is what we are striving for but, the challenges such as the constant need for updating documentation in new releases, new functionality, new customer input, and the changing world around us in general*" [Int2] diminishes the ideal case for using software documentation. Moreover, the interviewee believed that a communication channel is a medium that transfers information, and in most cases it is difficult to have software documentation as the primary communication channel replacing the verbal communication as it would better serve if accompanied by other communication channels in order to enhance and complement the information: "*For me a communication channel is just a facility that transfers some type of information. So, I would rather not call documentation a channel. I interpret it for example if you send the documentation over email then you have sent it over the mail channel*" [Int2].

The views on software documentation in our two cases seem to be aligned with the recent works of Forward [4], Fernández-Sáez et al. [8], and Zhi et al. [23]. Forward [4] and Fernández-Sáez et al. [8] support the concept of documentation as a communication channel and, stress that documentation is written for the purpose of communication among software engineers and maintainers. However, Zhi et al. [23] see documentation as an 'aid' for communication among developers and maintainers.

> Based on our empirical findings, software documentation helps communication within/among development teams, but it mainly complements other communication channels rather than replacing them or acting as the primary channel for communication.

B. Documentation usage frequency (RQ2)

The second research inquiry is devoted to the software documentation and specifically architecture and design documents usage frequency by developers and development team members.

The fourth question in the survey is asking for the frequency of referring to any type of software documentation in a normal working day. As shown in figure 5, less than half of the respondents (43.4%) refer to software documentation 1 or 2 times a day. 21.7% of respondents do not refer to software

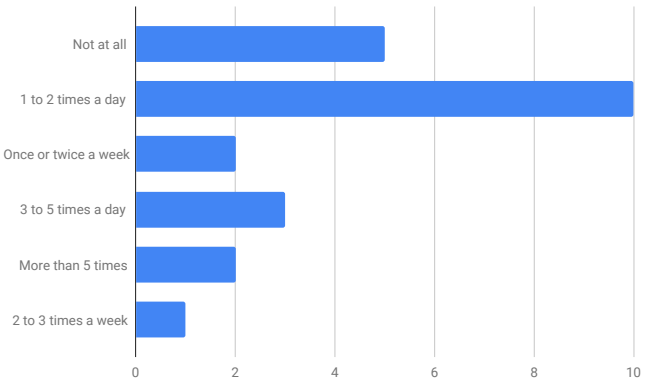

Fig. 5. How often do you refer to software documentation in a normal working day?

documentation at all. The rest are as follows: 8.7% more than 5 times a day, 13% 3 to 5 times a day, 4.3% 2 to 3 times a week, and 8.7% once or twice a week. The fifth question in the survey focuses merely on the frequency of referring to architecture and design documents in a normal working day. As shown in figure 6, 37.5% have stated that they do not refer to design documents at all. 29.2% have mentioned that they do access 1 or 2 times a day and the rest are as follows: 8.3% 3 to 5 times a day, 8.3% once or twice a week, 8.3% sometimes, 4.2% once in a sprint, and 4.2% at most once a month.

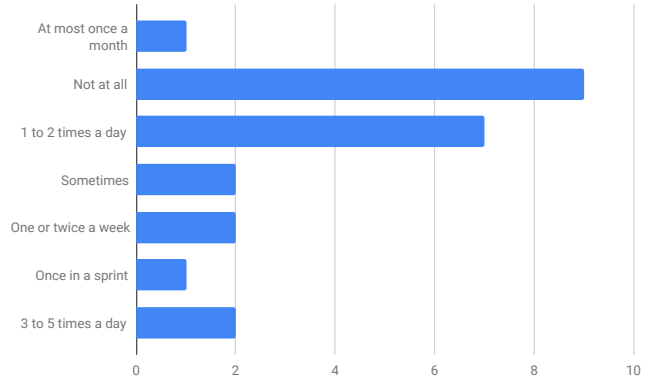

Fig. 6. How often do you refer to architecture and design documents in a normal working day?

Based on the interview data, the company B interviewee ([Int1]) stated that the software documentation regarding some of their projects in the company are not up-to-date, and this lack of valid information has led the developers to refer to documentation only 'from time to time': "*Well, we hope that we have a very updated information somewhere that we can refer to them, but sometimes with some applications that we are using, the documents are not very updated and don't have valid information. Then we have to choose another way to find those information [..] in general, we use documentation from time to time but of course we have a daily meeting or scrum*

meetings, and then we discuss about what we are going to do during the day" [sic] [Int1].

He mentioned that company B is going under a major transformation from old tools and technologies to new ones, and also migrating some of their IT projects from Gothenburg to development sites in India. The interviewee mentioned that: "*everyone in the company have felt the need for correct and complete documentation mainly in this period, but there is not much that we can do about it now*"[Int1]. Deployment guides, installation documents, health check documents, and even meeting notes are some of the documents that they use 'several times during a week'. Regarding architecture and design documents, he mentioned that they use and update these documents every three to four weeks, at the end of each sprint, when the new functionalities are added to the system: "*We use them (design diagrams) after the sprints when it is done and we have added the new features to the application [..] for us each sprint takes 3 to 4 weeks*"[Int1].

On the other hand, company A's development teams use architecture and design documents, and documentation in general on a daily basis, but their exact frequency of usage depends on the situation, and the document: "*It depends on the issue and what we are looking into. So, a software architecture document is more on a very high level. It usually describes our system is integrated in xyz based on these technologies, etc, etc. But, for example one delivery specific document that we have is BMD, Busines Mapping Document, and that describes exactly what goes in each integration. So, that is more a detail if you have an integration problem, we need to go to that detail. But, I would say on daily basis" [sic]* [Int2]. Moreover, he stressed that company A follows strict guidelines on the creation and availability of documentation: "*We have a strict process with designated gates on when a project can hand over and go to maintenance mode. If a project lacks documentation, it will not pass those gates*"[Int2].

Additionally, based on the work diary data, we found that the frequency of using documentation in company B is primarily based on the information that the teams residing outside of Sweden are seeking. For instance, they have recorded a 1.5-hour usage of a software architecture document for transferring knowledge to the remote team. The diarist ([Diarist1]) mentioned that they have such usage of architecture documents as these documents are more 'reliable'. Also, these sessions take place when a remote team requests to have a deeper knowledge of the system, and recently, the sessions have been happening more. He continued that they are currently using documents such as health check documents and release notes a few times a week as well.

In contrast, company A has shown a more rigorous usage of documentation in their work diaries ([Diarist2]). They have recorded the usage of various types of documentation including technical documents (e.g., JavaDoc, API reference guides, code comments), and business documents(e.g., business mapping document) on a daily basis. The diarist belonging to company A ([Diarist2]) has recorded several instances of using documentation in a single day and has mostly spent less than five minutes on a document. Comparing the time spent on documentation between the two companies, we found that company A's diarist have reported less time being spent on documentation than the diarist from company B. Company B diarist ([Diarist1]) reported ten minutes to one and a half hour for using various types of documentation.

The findings are in line with previous works of Garousi et al. [7], and Lethbridge et al. [10] that the frequency of using and consulting documentation has a direct relationship with its accuracy. Company B had problems with the accuracy and up-to-dateness of its documentation; therefore developers and team members use documentation from time to time. On the other hand, as company A had strict processes to create and update documentation, their frequency of usage was on a daily basis. Furthermore, Lethbridge et al. [10] state that software engineers consider abstract documents such as design documents to be more accurate. Regarding company B, we have found that although some of the documentation were not up-to-date, their usage of design documents was significant.

> Based on our empirical findings, the frequency of software documentation usage depends on the up-to-dateness and accuracy of the documents. Concurrently, developers consider architecture and design documents to be more accurate.

C. Documentation as an asset vs a liability (RQ3)

The third research inquiry is devoted to the reasons that developers and development team members do/do not seek to use software documentation. Concerning this research inquiry, the sixth question in the survey asks the participants for reason(-s) they would use software documentation. By coding the answers of the open-ended question, the authors found that more than half of the participants (62.5%) use software documentation to acquire more knowledge regarding the overall functionality of a system. In the second place, 16.6% of the respondents use software documents to know more about the design and architecture of a system. The rest are as follows: 16.6% seek to have better understanding about the requirements of a system, 12.5% believe that documentation helps in the development phase, 8.3% seek to know more about the test cases and the reported bugs, 4.1% believe that documentation helps in the maintenance phase, and 4.1% have the intention of avoiding knowledge evaporation.

By analyzing the interviews and work diaries, the following reasons and documentation advantages have been identified in both cases (ordered from the highest amount of relevant codes found during the thematic analysis to the lowest):

1) **Helping in the development phase**: the most common reason that the participants have mentioned is that software documentation helps in the development phase in different forms. An interviewee mentioned that it helps developers and team members to recall the forgotten domain-specific knowledge: "*Let's say, the development team is trying to add new features and some people are working with some services or components, and in the*

future they leave that team for some time then there is a lack of knowledge transfer here, so you will lose that information. You have to keep it [..] to be able to continue development"[Int1]. Another diarist mentioned that documentation related to programming languages, frameworks, and libraries help them in implementation: "*To understand how to use a Java method; Usage, type of return value, type of parameter values, type of exception that the method may throw*"[Diarist2]

2) **Transferring knowledge**: sharing and transferring knowledge is one of the most common reasons mentioned by the participants for using documentation. Documentation transfers the tacit knowledge of an individual by making it into explicit knowledge and providing information to the people who are seeking it: "*The goal for documentation is for it to be picked up by a new person and can be understood. [..] In my opinion, if we have the correct documentation then I can just give it to you and leave the building*" [Int2].
3) **Understanding the architecture**: the next most common reason for using documentation is to have a better understanding of a system's architecture and design. Design documents provide an abstract, higher level view of the components and relationships within a system: "*We need to have the great picture of the application so for example we should know the integration between various subparts of the system. Then we search based on those components that we are looking for and then as each component has a specific team behind it, we can ask that team to help us regarding some incident or problem that we might have*" *[sic]* [Int1].
4) **Helping in the maintenance phase**: the next most common reason for using documentation is how documentation helps maintainers acquire more knowledge about the specific system they are maintaining. There are cases that maintainers of a software system are not the ones who implemented the system in the first place. In these situations, documentation plays a key role in a successful maintenance effort: "*Every new development that has been added to the application should be explained to the support and maintenance team via documentation that they should know what are the expected incidents or challenging problems you may face in the future*" [Int1], "*Beside executable code that is running in production, and in different pre-production environments, software documentation is the most important a maintenance and development team can have for the system they work with*" [Int2].
5) **Time-saving**: another reason mentioned by the participants for using documentation is the fact that documentation saves time by minimizing the need for consulting people by various means of communication such as face-to-face meetings, and seeking information from them: "*The greatest thing about the documents is that you can refer to them all the time. And, you are not anymore wasting other people's time, spending so much time for having a conversation, or let's say asking for a meeting regarding some information that you can always find in the documents*" [Int1].
6) **Accessible**: one of the other reasons for using documentation is the accessibility of documents that allows anyone who holds the right to access documents refer to them. Moreover, most organizations have centralized online repositories for storing documentation today, and this removes the geographical limits of accessing and using documentation: "*We should have documents that are accessible for everyone, so anyone can easily find it and review it quickly. [..] the best way I would say is having a centralized domain of information somewhere that can be accessible for everyone*" *[sic]* [Int1]. "*The presence of documentation helps to track all the aspects of an application however it is highly important to make documents accessible as much as possible and also readable so users can easily refer to the required documents and find the information that they are searching for*" [Diarist1].

There are other reasons as well that were rarely mentioned by the participants, including: understanding the overall functionality, cost-effectiveness, and understanding the requirements.

The reasons for using documentation were diverse. Although participants saw documentation as an asset and were mostly supportive of using it, they mentioned some disadvantages for documentation as well. The disadvantages were related to their previous jobs (in the case of company A) or the struggles within their current companies (in the case of company B). The following cons were identified within the interview and work diary data, and are sorted from the highest amount of relevant codes to the lowest:

1) **Incomplete/outdated documentation**: although incomplete or outdated documentation is not truly a problem with documentation itself, the extent to which documents have not been updated or left incomplete have brought a disadvantage for accessing valid information in some organizations. Among our cases, company B seemed to have this problem: "*We hope that we have a very updated information somewhere that we can refer to them but sometimes with some applications that we are using the documents are not very updated and don't have valid information then we have to choose another way to find those information. [..] still the document, the template, or the tool that we are using they are highly limited [..] so, still face-to-face interaction is the first and most usable way to communicate*" *[sic]* [Int1].
   Regarding company A, the company strives to have updated documents: "*The tricky thing in having documents related to a software is to keep them updated [..] because when you have a software you do a first release, you have documents to that, then it comes to the customer and wants to change maybe a component, add a component or remove a component, or add an*

integration, then you need to trace back and see that all changes are reflected back into all the documents. When you miss to do that then the documents are out of date" [Int2]. He continues: "*That is what we are striving for. But, the challenges are new releases, updated functionality, new customer input, a changing world around us. Both from what the customer wants but also the technology that we need to deliver the service on. So, that is also important to measure, we are not just being hit by the customer and getting the support in production, we are being hit by a machine part that is getting old that we need to update all the time in order to have the right level of support.*" [Int2].

2) **Seen as more workload**: writing and updating documentation requires time and effort from the ones who hold the knowledge that needs to be documented, and everyone else who are responsible for keeping track of the information to be up-to-date. Therefore, some organizations might see this effort as an extra workload. Among our cases, we have found that company B developers and development team members follow this mindset: "*Since we do not have people that have this only task (writing documentation) then it is something that other people need to do. It is gonna be more workload for them. So, they are quite busy with other stuff so, they might skip that process but in the future they will realize that yeah, still they need to have that information somewhere. [..] it is not followed that well due to the time consumption and the amount that people need to spend time on writing documentation. And of course, I would say it is not a very interesting task, you know, people prefer to speak about it rather than write and keep that information somewhere.*" *[sic]* [Int1]
3) **Seen as an extra cost**: considering documentation as an extra cost relates to an organization's policy or management decision, and it is not very much a decision made by teams and their members. Moreover, it is dependant on the type of documentation and the specific needs of that organization which has come to have such a view on documentation. Seeing document as a cost was not present in any of the two organizations in our case study, but the 2nd interviewee explained a previous assignment with his/her former company which had such a notion on documentation: "*I was working in a consultancy which I will not mention them. I was on an assignment for a big manufacturing company here in our region, and I got to know that company, and the customer had agreed upon not having any test cases because of the costs. [..] that lead to the delivery company receiving the worst grades for the delivery because the customer was expecting a qualitative delivery each time but there were no test cases [..] I can understand, it is either the customer or the delivery team providing it to the customer that did not understand the criticality of these documents, they rather took a short-term cost-saving and did not saw what this will bring further down the road*" [Int2].

The findings on the advantages and reasons for using documentation are aligned with the results of previous studies, especially the works of Aghajani et al. [24], Curtis et al. [3], Garousi et al. [7], and Forward and Lethbridge [9]. Based on the findings, 'helping in the development phase' is the first reason for using documentation, this is in line with Garousi et al. [7] that found documentation to be more effective in the development phase than maintenance. Also, knowledge transfer was seen as an ability of documentation to impart knowledge [9].

Regarding the disadvantages, Curtis et al. [3] have found as well that incompleteness and tardiness of documentation make it a weak medium for communication. We have seen that company B was struggling with incomplete/outdated documentation, and this might be the reason that they were having issues with knowledge transfer to their remote teams, and the best solution they have come up with was face-to-face interactions. Additionally, as we saw in our cases, Garousi et al. [7] found that the larger a document gets, the more it adds to the workload, and the lower the cost-effectiveness will be. Also, Curtis et al. [3] mention that as a software project gets larger, the time spent on writing documentation shifts toward verbal communication due to the heavier workload.

> Based on our empirical findings, developers tend to use documentation as it helps them in development, knowledge transfer, maintenance, understanding architecture, saving time, and its accessibility. However, there are concerns on incomplete/outdated documents and the possibility of the required costly and tiresome efforts.

D. Design documents and communication (RQ4)

The fourth research inquiry is devoted to how architecture and design documents help developers to communicate among/within development teams. Related to this research inquiry, the seventh question in the survey asks the participants to what degree design documents are a remedy for the lack of communication and knowledge disappearance. As shown in figure 7, 37.5% of the participants think that design documents can play that role 'A lot'. 29.2% have mentioned 'Sometimes', and 20.8% have selected 'Always'. The rest are as follows: 8.3% have mentioned 'not at all', and 4.2% 'A little'.

Architecture and design documents are one of the technical software documentation types. This type of documentation is mainly known with various modeling languages and templates such as Universal Modelling Language (UML), which present a unified notation for writing and using design documents. In both interviews and work diaries, we asked the participants to express their opinions regarding how design documents can help communication in development teams, and if design documents can act as the primary and individual communication channel for knowledge transfer without the need of other channels.

In the case of company B, Diarist 1 recorded a one and a half hour usage of architecture documents for transferring knowledge to a remote operation team in India. During an

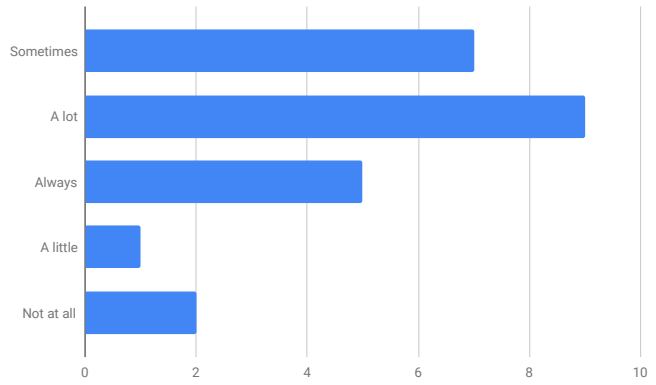

Fig. 7. To what degree do you think that design documentation is a remedy against knowledge disappearing because of people leaving projects?

interview with the same diarist, the interviewee ([Int1]) mentioned that their team uses architecture documents to explain the system to the remote team, but these documents do not cover all the details and they need to have video calls with the remote team to explain the documents in a verbal manner and add the necessary details as well: "*Every session is around 1 or 2 hours and it is quite long, but it is their challenge to store that video and try to review it and understand everything [..] we need to compensate that lack of knowledge with video conference or some video calls but we could have better documentation and just give them the documentation instead of that*" [Int1]. He further adds that the design documents in the company are limited to the point that they cannot be the primary source of information; in other words, the documents need to be complemented with the tacit knowledge of the team members in order to make sense: "*We have to have them (the design documents) open every time and add information or get some information from that document so, still the conversation is the first and most usable way to communicate*" *[sic]* [Int1].

Regarding company A, interviewee 2 explained that they use software documentation, including architecture and design documents on a daily basis. However, he remarked that they are still not able to depend on design documents as the primary channel of communication, but that is their goal: "*The goal for design documents is to be picked up by a new person and can be understood [..] but, always it is easier if you have someone that have been working with this in 20 years and he sits in front of you and he helps you understand the diagram, or describes for you the diagram [..] in the best of worlds, yes I should be able to give my (design) document to someone in India and he/she should be able to take that and do not talk with me. Take it and understand it, because that is the aim of the document*" *[sic]* [Int2].

During the analysis of the interview and work diary data, we found two concepts that affect the usage of design documents:

- **Educational/Employment background**: the interviewees pointed out that the usage of documentation and mainly design documents are in direct relationship with the employees educational and employment background. They elaborated that for the people who have a background in a field other than software engineering and computer science, design documents and documentation are not seen very important in general: "*It depends on what background you come from. So, if you have a software architecture background and you know the processes and you know the importance of documentation, yes, but based on other experience background that may not be seen as so important [..] you need to have the right background, right education to understand the criticality of what you need to have. If you don't understand that, then you are focused on the cost, of course*" *[sic]* [Int2]. Also, as Interviewee 1 mentioned: "*As far as I know designing a UML diagram, it needs a great knowledge, or something that people should know how to interact with it, and how to add the new information, and most people they prefer to just write some notes*" [Int1].
- **Company policies**: another aspect that affects the usage of design documents and documentation in general, is the company policies toward the importance and value of documentation. For example, as mentioned by interviewee 2, company A moves projects from development phase to maintenance when several 'gates' have passed: "*When we take an application from project mode to maintenance mode, there are several gates that needs to be passed within our organization. As the project proceeds with various gates within the project, there are hand-over, pointed out processes or instructions on what type of documentation should be ready from the project hand-over to maintenance along the way. So, there are processes how the documentation should be created, evolved and what type of documentation should be provided [..] if a project lacks documentation, it will not pass those gates*" [Int2]. In contrast, interviewee 1 explained that in company B they have a lack of an authority who would define and check the process of writing and using documentation: "*Each application should have some special roles to lead the project so they should define some policies that everyone would follow [..] designing documents should be accepted by the team and somebody needs to control those policies that everyone is following, regarding what type of task you are doing or developing, somebody needs to keep an eye on that*" *[sic]* [Int1].

The findings are in line with the previous studies of Garousi et al. [7], and Fernández-Sáez et al. [8]. Aligned with Fernández-Sáez et al. [8] study, we found that architecture and design documents complement face-to-face interactions and other communication channels in general, rather than replacing them. However, in the case of company A, development teams have shown interest in setting their goal in having complete and accurate documents to the point that the design documents would replace verbal communication. Moreover, educational and employment background influences the views on design

documents as these document might be seen costly, or not capable of being a communication channel, as reported by [8] as well. As mentioned by Garousi et al. [7], company policies affect the amount of support that the teams receive for using technical documentation, and these policies mainly define the value of such documentation. By looking at the two case companies, we found that in most cases delivering a working product is the first priority of software companies to the extent that producing and updating documentation becomes an unimportant activity, or fades away entirely. Various companies have different values; some might value reducing expenses as much as possible which might target documentation as well. Others might value the software product more than the accompanying documentation.

> Based on our empirical findings, architecture and design documents are seen as a complement to communication channels by enhancing the information being exchanged, rather than entirely replacing it. Educational and employment background, together with company policies, affect the usage and value of design documents in a company.

E. Knowledge evaporation (RQ5)

The fifth research inquiry is devoted to the effects of knowledge evaporation on development teams and the role that software documentation plays in that. Relevant to this research inquiry, the eighth question in the survey asks the participants for any type of information regarding the design that they tend to forget. After coding the answers, we found that 42.8% of the participants tend to forget the details related to the functionality of a system, 28.5% forget the architecture and abstract elements, and 28.5% do not tend to forget any information. The rest have reported that they tend to forget: tacit knowledge (9.5%), and Domain-specific knowledge (4.7%). Additionally, the ninth question asks the participants for their personal experience about the lack of software documentation and its effect on development teams. By coding the answers of the open-ended question, we found that most participants (80.9%) have reported that the lack of software documentation will lead to a consumption of extra time and workload, 33.3% mentioned that it leads to knowledge disappearance, 28.5% think that it results in poor software quality and final delivery. The rest are as follows: 9.5% anticipated a delay in delivery, 9.5% mentioned that software integration and code reuse would be problematic, and 4.7% saw no effect for the lack.

In both interviews and work diaries, the participants were asked to express their opinions about knowledge evaporation and disappearance, and how they affect development teams. In both cases, the participants indicated that software documentation avoids knowledge disappearance and evaporation, but we found that company B is struggling with knowledge evaporation as the first interviewee brought an example of their current project: "*For the current project that I am working on, a couple of weeks ago we wanted to make some adjustments on the database. We tried to add new stuff to that but we realized there are some problems behind it and we did not understand it. Then we should have spent so much time to see what the issue is, because sometimes the errors that we get from the system are not very understandable. Then, after spending some time we realized that this is something that someone else was working on, and we had to contact that person. Sometimes, we might see that the person has already left the project and it becomes scary because now you have a problem with your time planning that you did not expect. So, that becomes a very frustrating situation. We needed to find that person and ask him, what have you done and what can be the issue and the possible solution now [..] the best way is that whatever you do, you store that information somewhere that people can refer to it, and if something went wrong then people can go to the stored information and solve the issue*" *[sic]* [Int1].

The findings show that knowledge evaporation results in its disappearance, and it affects the teams in both development and maintenance phases: "*the best way is to have documentation than to have meetings, let's say we ask information from other people, but then we might forget during time. It's better that we have that information somewhere, so if someday we wanted to understand the problems and solutions, we know that we always have access to that information*" *[sic]* [Int1]. The second diarist also answered a diary question regarding the role of documentation in knowledge evaporation: "*To access the knowledge of former developers who have designed/implemented the system and have left the company*" [Diarist2]. Furthermore, we found that waste of resources (in terms of cost and time) is another consequence of knowledge disappearance and the lack of documentation as company A interviewee explains that: "*If you as a developer have the right documentation with the right content, that's it. Then you are free of the need of anybody else. But, if you don't have the documentation, that is where the tricky things start. Either you need to dig yourself in, or you need to find somebody who was responsible [..] because then you need to have 10 persons maybe to do the documentation job that you should have had, because all the critical parts of the system that are not documented, they should be documented as soon as possible*" [Int2].

Our findings on knowledge evaporation effects on development teams are aligned with the previous research [8], [6], and [7]. As Fernández-Sáez et al. [8] have found as well, software documentation and mainly design documents prevent knowledge evaporation, which affects the teams mainly in the maintenance phase. The tacit knowledge of an individual is a valuable but transient source of information due to the fact that the information might fade away over time, or get lost as people leave projects and assignments. Therefore, the tacit knowledge needs to be transformed into explicit knowledge, and be recorded and archived in the form of documentation to avoid knowledge evaporation and disappearance [6], [7]. Based on both cases, the lack of documentation results in knowledge disappearance and evaporation, and the lost knowl-

edge leads to a consumption of extra time and workload for the development teams.

> Based on our empirical findings, knowledge evaporation and disappearance results in a consumption of extra time and workload for the development teams, but by creating documentation and storing the knowledge, the information will be saved from the risk of getting lost.

V. Validity threats

There are specific validity characteristics in research that represent the quality and validity of it; e.g., construct validity, internal validity, and external validity [12]. A number of threats to validity were identified in the study.

Construct validity is the correct choice of operational measures in a study and their alignment with the research questions and objectives [12]. In this study, multiple sources of evidence were used to increase the construct validity. Moreover, we performed three collection methods on both cases of the case study in order to increase the confidence in research data. However, it is worth mentioning that the survey method did not have any questions regarding the company of the respondents. Therefore, we could not identify the respondents that were from the two companies under study. For this reason, we opted out the survey findings from our case study data sources, and only presented the results in section IV as they were still valuable findings covering the opinions of a sample group within the software industry.

Internal validity ensures that the factors under study in the research were not affected by any another factor outside the scope of the study [12]. During data collection, the authors asked the participants to cooperate in the survey, work diaries, and the interviews. As the same group of participants were approached in various methods, their awareness of the usage and usefulness of software documentation might be exaggerated and altered in comparison to their everyday use of documentation. In order to mitigate this threat, we scheduled a seven-day break between each data collection method. Moreover, an interviewee's feelings toward his company and the state of relationship with his team members might risk the internal validity of the research as well. To lower the risk, we approached the participants with various types of questions, some of them asking the same concept but from different viewpoints, in order to circumvent personal feelings and focusing on professional aspects.

External validity concerns if the findings of the study are generalizable afar from the current study [12]. In our research, we have one data point in each company under study; this might limit the generalisability of our results. However, by running a multiple-case study with various data collection methods, we are having a more in-depth look at the phenomenon through two cases. Also, our results will encourage other researchers to examine more companies and cases by applying our proposed case study approach in order to seek the ability of replicating the results in this context.

Reliability is the extent to which the same results and conclusions of a research can be reached if it is to be operated by other researchers [12]. By using a computer-assisted data analysis software (NVivo), we made a database of the interview recordings, transcriptions, work diary templates, survey responses, and their relevant codes and themes. This presents a transparent view of our case study approach and a chain of evidence from all the data collection methods in the study. By inspecting this data, the study can be replicated by other researchers.

VI. Conclusions and future work

In this paper, we studied the relation between software documentation and communication among/within development teams, and how software documentation and specifically architecture and design documents can be of help to communication. We carried out a multiple-case study on the IT departments of two Swedish manufacturing companies. Our proposed approach consisted of a qualitative evaluation of cases by having a data triangulation of three data sources: two semi-structured interviews, a survey with twenty-four participants, and two work diaries.

In relation to our case study, we found that 1) software documentation complements communication channels rather than replacing them, 2) documentation usage frequency depends on its up-to-dateness and accuracy, 3) the main reasons for using documentation are helping in development and maintenance phases, knowledge transfer and architecture comprehension, although incomplete/outdated documents and assumptions of extra cost and time consumption are the main concerns, 4) architecture and design documents round out communication channels, and their usage is effected by company policies and educational/employment background, 5) knowledge evaporation results in excessive time and cost consumption, however software documentation is a remedy to that.

This study approach can be applied to other software development organizations as researchers will be able to compare the existing results with the ones expected from the context of their cases. Other Possible future works can be toward the following directions:

- What can be improved in various types of documentation in order to act as the primary communication channel replacing verbal communication?
- What are the elements affecting the accuracy and up-to-dateness of software documentation?
- What incentives will encourage software professionals to create and update software documentation? And why?
- To what extent the modelling language or template of an architecture and design document affects its usage and usefulness?
- How software processes can help mitigating the knowledge evaporation effects on software development teams?

Moreover, there is a contradiction between the findings of this study and how agile manifesto emphasizes on working software over comprehensive documentation. Future qualita-

tive research is required to evaluate the findings of this study in correlation with agile methodology.

REFERENCES

[1] M. Glinz and S. Fricker, *On shared understanding in software engineering*. Gesellschaft für Informatik eV, 2013.

[2] S. Ghobadi, "What drives knowledge sharing in software development teams: A literature review and classification framework," *Information & Management*, vol. 52, no. 1, pp. 82–97, 2015.

[3] B. Curtis, H. Krasner, and N. Iscoe, "A field study of the software design process for large systems," *Communications of the ACM*, vol. 31, no. 11, pp. 1268–1288, 1988.

[4] A. Forward, *Software documentation: Building and maintaining artefacts of communication.* University of Ottawa (Canada), 2002.

[5] M.-A. Storey, A. Zagalsky, F. Figueira Filho, L. Singer, and D. M. German, "How social and communication channels shape and challenge a participatory culture in software development," *IEEE Transactions on Software Engineering*, vol. 43, no. 2, pp. 185–204, 2017.

[6] W. Ding, P. Liang, A. Tang, and H. Van Vliet, "Knowledge-based approaches in software documentation: A systematic literature review," *Information and Software Technology*, vol. 56, no. 6, pp. 545–567, 2014.

[7] G. Garousi, V. Garousi-Yusifoğlu, G. Ruhe, J. Zhi, M. Moussavi, and B. Smith, "Usage and usefulness of technical software documentation: An industrial case study," *Information and Software Technology*, vol. 57, pp. 664–682, 2015.

[8] A. M. Fernández-Sáez, M. R. V. Chaudron, and M. Genero, "An industrial case study on the use of uml in software maintenance and its perceived benefits and hurdles," *Empirical Software Engineering*, vol. 23, no. 6, pp. 3281–3345, 2018.

[9] A. Forward and T. C. Lethbridge, "The relevance of software documentation, tools and technologies: a survey," in *Proceedings of the 2002 ACM symposium on Document engineering*. ACM, 2002, pp. 26–33.

[10] T. C. Lethbridge, J. Singer, and A. Forward, "How software engineers use documentation: The state of the practice," *IEEE software*, vol. 20, no. 6, pp. 35–39, 2003.

[11] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, p. 131, 2009.

[12] R. K. Yin, *Case study research and applications: Design and methods*. Sage publications, 2017.

[13] K. M. Eisenhardt, "Building theories from case study research," *Academy of management review*, vol. 14, no. 4, pp. 532–550, 1989.

[14] V. A. Thurmond, "The point of triangulation," *Journal of nursing scholarship*, vol. 33, no. 3, pp. 253–258, 2001.

[15] K. Louise Barriball and A. While, "Collecting data using a semi-structured interview: a discussion paper," *Journal of advanced nursing*, vol. 19, no. 2, pp. 328–335, 1994.

[16] S. L. Pfleeger and B. A. Kitchenham, "Principles of survey research: part 1: turning lemons into lemonade," *ACM SIGSOFT Software Engineering Notes*, vol. 26, no. 6, pp. 16–18, 2001.

[17] J. Linåker, S. M. Sulaman, R. Maiani de Mello, and M. Höst, "Guidelines for conducting surveys in software engineering," 2015.

[18] O. Manai. How software documentation helps communication in development teams: A survey. [Online]. Available: https://forms.gle/5rzhwzAEPkrg2QCw8

[19] T. C. Lethbridge, S. E. Sim, and J. Singer, "Studying software engineers: Data collection techniques for software field studies," *Empirical software engineering*, vol. 10, no. 3, pp. 311–341, 2005.

[20] O. Manai. How software documentation helps communication in development teams: Work diary template. [Online]. Available: https://drive.google.com/file/d/1gSZF8wXE1D11Brbd2o3ng8BTD94zKGPW/view?usp=sharing

[21] A. Alaszewski, *Using diaries for social research*. Sage, 2006.

[22] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[23] J. Zhi, V. Garousi-Yusifoğlu, B. Sun, G. Garousi, S. Shahnewaz, and G. Ruhe, "Cost, benefits and quality of software development documentation: A systematic mapping," *Journal of Systems and Software*, vol. 99, pp. 175–198, 2015.

[24] E. Aghajani, C. Nagy, O. L. Vega-Márquez, M. Linares-Vásquez, L. Moreno, G. Bavota, and M. Lanza, "Software documentation issues unveiled," in *In Proceedings of ICSE 2019 (41st ACM/IEEE International Conference on Software Engineering)*.

APPENDIX A
INTERVIEW QUESTIONS

TABLE II
INTERVIEW QUESTIONS

#	RQ	Interview Question
Q1	D	What is your role within the company?
Q2	D	How many years of experience do you have with software development?
Q3	RQ1	(pre) What is the most preferred communication channel(-s) personally for you to seek information in development/maintenance phases? Software documentation, emails, face-to-face communication, Q&A websites, etc.
Q4	RQ1, RQ3	Why do you think software documentation can act as a communication channel?
Q5	RQ1	(pre) Which form of communication channels can yield a faster and a more reliable response in your experience?
Q6	RQ2	(pre) How often do you refer to various types of software documentation?
Q7	RQ2	How often do you refer to architecture and design documents?
Q8	RQ3, RQ4	For what purposes do you mainly refer to design documents?
Q9	RQ4	Do you think design documents are capable of being the primary form of communication in specific situations?
Q10	RQ3	As a developer, do you think software documentation can reduce the time spent on finding answers/solutions compared to other means of communication?
Q11	RQ5	In your personal experience, how does the lack of software documentation affect development teams?
Q12	RQ5	(pre) Does the lack of documentation result in knowledge evaporation?

APPENDIX B
SURVEY (QUESTIONNAIRE) QUESTIONS

University of Gothenburg - CSE department -Thesis research: How software documentation helps communication in development teams

Software developers seek to communicate and engage with each other in order to collaborate and contribute to various development practices. Documentation is one of the means of conveying information within projects and therefore it is a form of communication among various project members. The case study intends to research into the correlation between software documentation and communication, and seeking the effects of documentation on communication among/within software development teams.

The research is designed for DIT565 Software Engineering and Management Bachelor thesis project at the Computer and Software Engineering department of the University of Gothenburg.

Note: All the data gathered in this survey are anonymized and no email addresses, names, or any other forms of identification are recorded.

*Required

1. **What is your role within your organization?** *
   Mark only one oval.
   - ◯ Developer
   - ◯ Architect
   - ◯ Tester
   - ◯ Manager
   - ◯ Other: _____

2. **Do you think software documentation can act as a communication channel to transfer information among people and projects?** *
   Mark only one oval.
   - ◯ Yes
   - ◯ No

3. **Which communication channel(-s) do you find more helpful to understand a system?** *
   Tick all that apply.
   - ☐ Face-to-face interactions
   - ☐ Private chats (Instant messengers, Skype chat, ...)
   - ☐ Private discussions (Email, ...)
   - ☐ Software documentation (design documents, requirement specifications, test documents, ...)
   - ☐ Question & Answer websites (Stack Overflow, Quora, ...)
   - ☐ Code hosting websites (GitHub, Sourceforge, ...)
   - ☐ Other: _____

Fig. 8. Questionnaire page 1

4. **How often do you refer to software documentation in a normal working day?** *
Mark only one oval.

- ◯ Not at all
- ◯ 1 to 2 times a day
- ◯ 3 to 5 times a day
- ◯ More than 5 times
- ◯ Other: _____

5. **How often do you refer to architecture and design documents in a normal working day?** *
Mark only one oval.

- ◯ Not at all
- ◯ 1 to 2 times a day
- ◯ 3 to 5 times a day
- ◯ More than 5 times
- ◯ Other: _____

6. **For what reasons do you lookup in software documentation?** *

_____

_____

_____

_____

_____

7. **To what degree do you think that design documentation is a remedy against knowledge disappearing because of people leaving projects?** *
Mark only one oval.

- ◯ Not at all
- ◯ A little
- ◯ Sometimes
- ◯ A lot
- ◯ Always

8. **Is there any type of the information about the design that you tend to forget?** *

_____

_____

_____

_____

_____

Fig. 9. Questionnaire page 2

9. **In your personal experience, how does the lack of software documentation can affect the software development teams?** *

10. **If you have any comments, please add them here.**

Fig. 10. Questionnaire page 3

APPENDIX C
WORK DIARY TEMPLATE

CHALMERS | UNIVERSITY OF GOTHENBURG

Department of Computer Science and Engineering
Software Engineering & Management Program

Work Diary Template

Date: YYYY-MM-DD

Diarist No. or Initials: For privacy purposes, the first letter of the first name and the last letter of the family name works just fine.

Page No.:

1. How is it common for you to use/refer to software documentation? (Check the box which best describes your usage. From lowest to the highest amount of usage, '1' represents **No at all**, and '5' represents **Most of the times**.) (RQ2)

   1☐ 2☐ 3☐ 4☐ 5☐

2. Please record each effort you make to reach software documentation. If the following table is not enough for all your efforts, please record the rest in another diary template and increment the page number. (RQ2, RQ4)

Time of reference[1]	Time spent on documentation	Type of the software document[2]	Reason for using documentation[3]

(1) The approximate time you referred to software documents. In 24-hr format.
(2) Software documentation can be architecture and design documents (AD), Requirements specifications (R), Source code comments (SC), Test Documents (T), and Process descriptions (P). You can add any extra type of documents as well.
(3) A short sentence on the reason you have used documentation. For example, the type of the document was a design document, and your reason for using it was "for checking the higher-level structure/design of a specific class."

Fig. 11. Work diary template page 1

CHALMERS | UNIVERSITY OF GOTHENBURG
Department of Computer Science and Engineering
Software Engineering & Management Program

Please answer the following questions at the end of the day:

3. Which one of the following cases best describes your view on the use software documentation? You can add any extra cases in the last part as well. (RQ1, RQ3)
   - To access the knowledge of former developers who have designed/implemented the system and have left the company ☐
   - To use in communication with your current colleagues ☐
   - To act as the primary communication medium and reduce the amount of verbal communication and face-to-face meetings ☐
   - To allow knowledge transfer among various teams within an organization ☐
   - (You add other cases here)

4. Do you think software documentation can decrease knowledge evaporation in software development teams? (RQ5)

Fig. 12. Work diary template page 2