The Impact of Prior Knowledge on Searching in Software Documentation

Klaas Andries de Graaf
VU University Amsterdam
ka.de.graaf@vu.nl

Peng Liang
Wuhan University
liangp@whu.edu.cn

Antony Tang
Swinburne University of Technology
atang@ict.swin.edu.au

Hans van Vliet
VU University Amsterdam
hans@cs.vu.nl

ABSTRACT

Software documents are used to capture and communicate knowledge in software projects. It is important that this knowledge can be retrieved efficiently and effectively, to prevent wasted time and errors that negatively affect the quality of software. In this paper we investigate how software professionals search for knowledge in documentation. We studied the search behaviour of professionals in industry. Prior knowledge helps professionals to search software documents efficiently and effectively. However, it can also misguide professionals to an incomplete search.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process; D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*Documentation*

Keywords

Software Documentation; Information Retrieval; Search Strategies; Prior Knowledge; Heuristics; Cognitive Bias

1. INTRODUCTION

In software industry, it is a common practice to capture information about a software system, its design, and architecture in file-based documents, e.g., in text documents and diagram files. It is important that software professionals can quickly and correctly answer questions from these documents. Otherwise valuable time could be wasted, costly errors could be made, and software may not be built according to specification, which increases the cost of software projects and decreases the quality of software [13].

The organisation of documents by directories, titles, and sections typically does not support all of the questions asked by software professionals [4, 13]. Spelling errors, abbreviations, and synonyms make keyword searching ineffective and professionals may not know the right keywords to find answers [9]. Exhaustive exploration of all document content is time-consuming and impractical in a large document set.

These issues introduce search uncertainty and make it hard for professionals to find complete and correct answers within reasonable time. Professionals waste time searching for answers in unstructured documentation [13]. The obstacles to finding the right information can be so great that it discourages professionals from trying to search at all [9, 10].

In this paper we investigate how software professionals search for knowledge in software documentation. We studied how 26 software professionals in industry retrieved knowledge from documentation to answer architecture-related questions. The software professionals were asked to think aloud while answering questions about software and architectural elements such as subsystems, components, behaviour, requirements, and decisions. We measured how much time was spent on finding answers to the questions and whether answers were complete and correct.

We found that the search behaviour of software professionals is heavily influenced by their prior knowledge about the documentation and the software specified in this documentation. Prior knowledge is used to guide predictions about, e.g., the location of knowledge, which keywords can be used to find knowledge, and whether the knowledge found is correct and complete. Professionals use their prior knowledge as a short-cut to find answers to their questions, i.e. they use a heuristic (or 'experience-based') approach [15, 18] to searching.

Use of prior knowledge helped some of the participants in the study to quickly find the location of correct answers, even when the document organisation did not support the questions asked. The participants preferred to use their prior knowledge instead of exhaustively exploring documentation content.

We however observed that availability and confirmation bias can occur when using prior knowledge, which results in wasted time and incomplete answers. Availability bias and confirmation bias are cognitive biases that cause errors in judgement. Participants made inaccurate predictions about whether documents contained answers and whether searching for certain keywords would lead to answers. Moreover, several participants only looked for confirmation of answers that they said to know from their prior knowledge.

DocEng'14, September 16–19, 2014, Fort Collins, CO, USA.

http://dx.doi.org/10.1145/2644866.2644878.

Table 1: Demographics of Participants in Study

Number of participants	Primary role of participants	Average years in role at Océ	Average years in role	Average years working at Océ
6	Domain architect	3.60	4.77	9.92
5	Software engineer	6.47	6.81	7.47
5	Software project manager	3.83	5	14
4	Product or system test engineer	9.75	11.75	11.625
4	Workflow architect	7.25	7.25	18.75
1	Configuration manager	3	10	3
1	Software designer	1	1	1

In this paper we first describe how prior knowledge is used by professionals to search knowledge in software documentation. We then evaluate the use of prior knowledge in terms of search efficiency and effectiveness and report cognitive biases that lower this efficiency and effectiveness. These findings provide guidance for software practitioners to make optimal use of their prior knowledge when searching knowledge in software documentation.

We make the following contributions:

1. Report how professionals use prior knowledge to search in software documents.
2. Identify cognitive biases that may occur when using prior knowledge to search in software documents.
3. Report how prior knowledge and cognitive bias affect the efficiency and effectiveness of searching.

Section 2 details on the study design and identification of the search strategies and cognitive process of participants. Section 3 reports and evaluates how prior knowledge is used when applying the search strategies and how cognitive biases may occur. Lessons learnt for document users and writers are described in Section 4 and Section 5 discusses threats to validity. In Section 6 we discuss related work and Section 7 reports our conclusions.

2. DESIGN AND ANALYSIS OF SEARCH BEHAVIOUR STUDY

2.1 Study Design

We conducted a study to investigate how software professionals search for knowledge in software documents. The study was conducted in a software project at the R&D department of Océ technologies in the Netherlands. Océ applies an agile development methodology to encourage creativity and productivity.

Participants are all software professionals at Océ R&D who are involved in the software development process. Table 1 gives the demographics of the participants.

The architecture documents used are:

- Two Software Architecture Documents (SAD) of 3 and 9 pages, respectively. SADs detail the design of functionality, behaviour, and components. One SAD gives an overview of what knowledge can be found in the other SAD.
- Four Software Behaviour Documents (SBD), ranging in size from 8 to 18 pages. SBDs describe the behaviour of software together with all requirements and settings for that behaviour. One SBD describes all possible settings for behaviour.
- One System Reference Document (Sysref), of 19 pages. The Sysref document details the high level system design and decomposition, in terms of subsystems, components, and interfaces, and gives design decisions and rationale on system design.
- One Design Document containing three UML diagrams that detail on the design of subsystems, components, and interfaces.

These documents are stored in 3 directories. A directory SAD contains the overview SAD and one subdirectory with the other SAD. A directory 'SBD' contains SBDs. A directory 'Sysref' contains the Sysref and design document.

The documents specify the software for a series of document printing machines developed at Océ and are a subset of all documentation used in the project. An Océ professional estimated that there are around 50-75 users of these documents. The documents are written in English, and consist of 79 pages, 1,794 paragraphs, 3,183 lines, and 13,962 words. Participants could search the documents using a file explorer (MS Windows Explorer), document editor (MS Word), and UML editing tool (MagicDraw).

We formulated 7 questions about the knowledge in the documents. Criterion for selection of these questions include that the interpretation of the questions is similar between different participants and that their answers can be quantitatively assessed, i.e., the questions should not be open-ended. Part of the questions have been obfuscated for non-disclosure reasons: 'QQ', 'XX', 'YY', and 'ZZ' replace an actual software entity or concept.

1A: *Which settings have an impact on behaviour "History"?*

1B: *Which settings have an impact on behaviour "Alert Light"?*

2: *Which requirements for behaviour "XX" should be satisfied (realized) by component "Settings Editor"?*

3A: *Which decisions have been made about component "Settings Editor"?*

3B: *Which decisions have been made on the configuration of behaviour "YY", "ZZ", "History", and "XX"?*

4A: *Which subsystem is interface "QQ" part of?*

4B: *Which other interfaces are offered by this subsystem?*

13 of the 26 participants answered questions 1A, 1B, and 2 and the other 13 participants answered questions 3A, 3B,

Table 2: Encoding of search actions from video recordings

Search action	Description and criteria for identification
Exploring directories	
Open Dir	Participant opens directory.
Inspect Dir	Participant has contents of directory on screen for 3 seconds or more.
Open Doc	Participant opens document.
Dir keyword search	Participant searches for keyword in the documents in a directory.
Inspect Dir search result	Participant inspects the list of documents found by using a keyword search in directory.
Exploring documents	
Scan section	Participant has content of document section or diagram on screen for 3-5 seconds.
Detailed scan	Participant has content of document section or diagram on screen for more than 5 seconds.
Scroll to section	Participant scrolls to a specific section and does not inspect intermediate sections.
Scroll to see section title	Participant scrolls to see the title of section currently being read.
View TOC	Participant looks at Table of Contents for 3 seconds or more.
Click TOC	Participant clicks on an entry in Table of Contents to navigate to section.
Keyword Search	Participant searches for keyword in document.
Inspect context of search result	Participant looks at keyword search result and surrounding text for more than 3 seconds.

4A, and 4B. Answering these questions was part of an experiment reported in [4]. An ontology-based documentation approach was used by participants to answer the remaining questions. For example, the participants that answered questions 1A, 1B, and 2 using file-based documentation would subsequently answer questions 3A, 3B, 4A, and 4B using ontology-based documentation. This use of ontology-based documentation is outside the scope of this paper.

In total 91 answers to the 7 questions were given by 26 participants when using file-based documentation. The researcher conducting the study read the 7 questions aloud to the participants. We asked all participants to search until they were satisfied with the time spent on an answer and its perceived correctness and completeness. Participants were instructed that this satisfaction should reflect their normal way of working.

We measured efficiency by recording how much *time* participants spent on accomplishing each task, namely, searching and providing an answer to a question. Effectiveness was measured by recording the *recall* of participants, i.e. the completeness of their answers, and *precision*, i.e. the correctness of their answers. A complete answer (resulting in perfect recall) to questions 1A, 1B, 2, and 4B inluded multiple knowledge elements, e.g., two settings, three requirements, or four interfaces.

The 'ground truth' for evaluating recall and precision was verified in a pilot with two Océ professionals who did not participate in the study. They were asked whether an answer for a given question was complete and correct. We use "completeness" to refer to recall and "correctness" to refer to precision in the rest of the paper for a better understanding.

The two professionals that participated in the pilot also proposed improvements to the question set. They evaluated whether each question was representative of the questions that software professionals at Océ normally ask and whether each question was relevant to software professionals in different roles. The questions were also evaluated on their representativeness and relevancy by five participants in a questionnaire after the experiment reported in [4]. They evaluated all questions as relevant and representative for their jobs except for question 3A, which one participant evaluated as irrelevant and not representative.

The researcher conducting the study kept track of what participants indicated to be answers to a question. When a participant stopped searching, said s/he found an answer, or said s/he was satisfied, the researcher verified with the participant whether this was the final answer to the question.

We captured the search actions of participants by video recording their monitor screen. We used the think aloud method [19] and asked participants to think aloud when searching and recorded their voice in the video recordings.

2.2 Identification of Search Strategies and Prior Knowledge

We identified around 2,500 search actions in over 11 hours of video recordings. Table 2 details the different types of search actions that we identified and encoded from the videos. We collected the search actions used to find 90 of the 91 answers given by the participants. The video record of one participants answering one question was corrupted beyond repair and is thus excluded from our analysis.

Not all participants were talkative, so the think aloud recordings for some questions were more detailed than others. Also, some phrases and parts of sentences said in video recordings of 22 of the 90 answers could not be heard clearly due to low sound recording volume and low volume of participants' voices. We however could often still infer what was said from the context of the search. One researcher spent 8 weeks to encode the search actions and transcribe think-aloud recordings from the videos.

From the identified search actions and think aloud verbalization we constructed Problem Behaviour Graphs (PBG) [11]. The construction of PBGs is a form of protocol analysis [6] which can be used to identify how people use their intelligence to solve problems in complex real-world environments [2]. In [2], Chen and Dhar used PBGs to model and investigate the cognitive process of people engaged in online document-based information retrieval. In our case the problem space consists of finding answers to questions using the document organisation, content, and the search functions of the documentation tools.

A PBG starts with the initial state of knowledge one has about a problem. In our case the initial state of knowledge was the question asked and the existing prior knowledge of participants about the documentation, its content, and the software system and project it specifies.

The initial knowledge state in a PBG changes to other knowledge states as the problem-solving process progresses. Problem-solving progressed when participants executed search

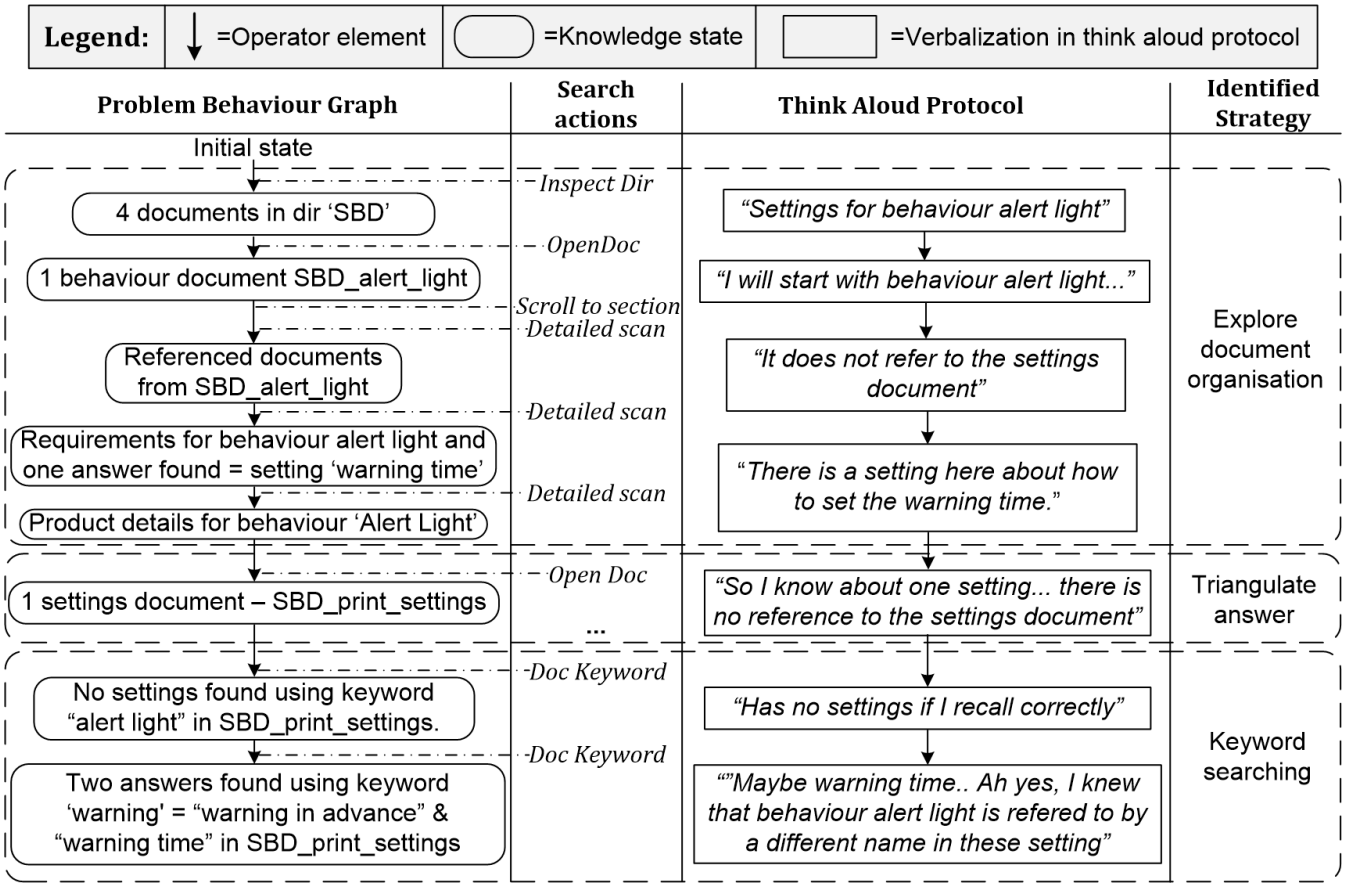

Figure 1: Problem Behaviour Graph of participant answering question 1B (*Which settings have an impact on behaviour "Alert Light"?*).

actions in order to obtain new knowledge about the search problem. When a solution is found the problem-solving process ends. In our case the problem-solving ended when an answer to a question was given by a participant.

Figure 1 shows one of the constructed PBGs in which a participant had to find settings for behaviour '*Alert Light*' in order to answer question 1B. The time sequence of search actions is from top to bottom in this PBG. The knowledge states are represented by boxes with rounded corners, that each contain the additional knowledge acquired by the participant when searching for knowledge.

We identified four search strategies using PBGs which are detailed below with a concrete example. We could identify to which search strategy each of the 2,500 encoded search actions belonged. In most cases multiple search strategies were used when answering a question.

In the PBG example shown in Figure 1 the first search strategy used by the participant is to **explore the document organisation** in directory *SBD*. The documentation was organized by means of directories, documents, and sections. Part of the information in the contents of document sections was organised by lay-out and text notations, e.g., in the phrase "*REQ_1: users can save login credentials in Comp_3: UI*" which makes a requirement explicit. Information was organized in diagrams by means of UML notations that, e.g., denoted interfaces and interactions.

Figure 1 shows that using this strategy the participant finds document *SBD_Alert_Light* by inspecting the content of directory *SBD*. The title of *SBD_Alert_Light* relates to behaviour '*Alert Light*' in question 1B, which indicates that this document contains information relevant for answering question 1B. The participant then opens *SBD_Alert_Light* and checks if it contains a reference to a dedicated settings document. In the next two search actions the participants scans for requirements and product information related to behaviour '*Alert Light*' and one setting is found.

Exploring document organization is a search strategy that was used when searching for 88% of the answers in the study. Participants that spent time on exploring this document organisation would often quickly gather relevant clues about which locations contained answers to questions. In [4], which is partly based on the data described in this paper, we found that exploration of document organisation was often time-effective for answering the questions. During a subsequent analysis of the study in [4] and its replication we found evidence that the use of document organisation that relates to a question correlates with the time-effectiveness of answering that question.

The organisation of documents does not always fully relate to the questions that document users have to answer. For example, none of the directory and document titles used in the study revealed where the decisions for question 3A and

Table 3: Overview of prior knowledge, evaluation, and cognitive biases identified in study

Prior knowledge	Gain if correct	Loss if incorrect	Cognitive bias and possible underlying reasons
Answer is in location X	large	small	
Answer is not in location X	small	large	Availability bias: difficult to recall examples of answers found in unfamiliar location X.
Keyword X leads to answer	large	large	Availability bias: keywords that are often used for searching are familiar and more easily remembered.
Answer can be triangulated	large	small	
Answer is already known	small	large	Confirmation bias: focus on confirming known answer.

3B could be found. Only the section titles inside two documents were explicitly related to these decisions and the documents had to be opened to discover this. Participants could exhaustively explore the document organisation to discover documents and sections that related to a question, however, this took a lot of time for several participants.

Alternatively, participants would directly **search the expected locations of answers** by predicting in which location (directories, documents, and sections) they would most likely find answers to a question. Participants that used this alternative strategy directly navigated to certain locations at the start of a question, without exploring the available document organisation beforehand[1]. Searching the expected location of answers is a search strategy that was used when searching for 29% of the answers.

The third and fifth sentences in the think aloud protocol in Figure 1 show that the participant thinks aloud about another document. After finding an answer in document *SBD_Alert_Light* the participant decides to open the other document *SBD_ Print_Settings* so s/he can verify the correctness and completeness of the answers. We named this strategy '**triangulate answer**', as multiple sources are used to verify and improve the answer. This strategy was used by participants when searching for 11% of the answers.

The participant subsequently uses a strategy of **keyword searching** for the name of behaviour '*Alert Light*'. After an unsuccessful keyword search, the participant recalls from prior knowledge that the settings may not be mentioned by the exact name of behaviour '*Alert Light*', and starts to use a different keyword.

The file explorer, document editor, and UML tool used in the study provide keyword search functions that show their users which document titles, text fragments, and UML elements match a given keyword. We identified **keyword searching** as a strategy that participants used when searching for 62% of the answers.

3. USING PRIOR KNOWLEDGE TO SEARCH UNDER UNCERTAINTY

In the think aloud recordings the participants voiced that they were uncertain about the correctness and completeness of 34 answers, out of the 90 answers (38%) given in the study . 13 of these 34 answers were actually correct and complete. Participants also voiced for 11 of the 34 answers that in everyday practice they would verify the answer with a colleague.

Typical remarks about this uncertainty are: "*It is difficult to know whether you found everything in the documentation. [I am] 70% sure of [my] answer*", "*Because searching was difficult I am not sure if this is [the] correct [answer].*", "*I think there is a 50% change that I have found all answers*", and "*I have reasonable confidence that I have not missed [any parts of the answer]*".

We observed how participants used their prior knowledge to deal with their uncertainty. Prior knowledge was used to predict which documents might contain answers when the document organisation did not relate to the question. Participants were able to recall from prior knowledge what different spelling variations, synonyms, and acronyms existed for technical terms required in the search, and this enabled them to quickly find answers by keyword searching. Participants also used prior knowledge to recognize answers and to predict whether an answer was correct and complete.

Participants talked about how to use their prior knowledge when applying the search strategies identified in Section 2.2. For example, they voiced which documents might be relevant ("*I think only Sysref contains answers*"), which keywords to search for ("*I know that Alert Light is referred to by a different name*", also see Figure 1), and which answers were complete ("*This setting is the answer. I already knew this setting.*"). Participants acquired this prior knowledge by, e.g., having used the documentation, working on the software system, and by attending meetings, presentations, and conversations with other professionals.

In the next subsection we first describe how participants acquire prior knowledge in the study. In subsections 3.2 to 3.6 we report the different ways in which participants used their prior knowledge to search for answers. We also report cognitive biases that may occur during the use of prior knowledge.

We describe the gain when use of prior knowledge leads to complete and correct answers and the loss when it does not lead to answers. The gain is categorized as 'small' or 'large' in terms of time saved (compared to the average time spent on a question) and whether the use of prior knowledge helped participants to find complete and correct answers. The loss is similarly categorized as 'small' or 'large'. If a large gain means that participants found many answers in little time, then a comparatively large loss is that many answers were missed and much time was wasted.

We have summarized the findings in Table 3. The first column denotes what prior knowledge a searcher may have and the last column denotes what cognitive bias may occur

[1] We observed from the search actions and think aloud statements that participants required 3 seconds or more to recognize and explore the document organisation when it was shown on their screen. After 3 seconds or more the participants acted upon the information by exploring the document organisation and they talked about this information, e.g. "*I see a settings document in this directory*". The participants did not actively explore the available document organisation if it was shown on their screen for less than 3 seconds. Instead the participants directly navigated to directories, documents, and sections in which they expected to find answers.

when using this prior knowledge. Column '*Gain if correct*' denotes whether a small or large gain was observed when the prior knowledge was correct and led to answers. Column '*Loss if incorrect*' similarly denotes the loss when the prior knowledge was incorrect and did not lead to answers

3.1 Acquiring Prior Knowledge

Several participants voiced that they learn about how knowledge is organised in the documentation when searching. For example, one participant voiced: "*From the previous question I have gained knowledge about behaviour History*" and "*I already have seen that this knowledge is described in SBDs and not in SADs. So I already have an approach that works for [searching] requirements*". A participant explicitly voiced that this learning process was intentional: "*I would need to build up a kind of model in this environment, in documentation, to find an approach for searching. I need to open a few documents in order to come to that approach.*".

We could observe that participants most often acquired knowledge about documentation by exploring the document organisation (one of the search strategies). Participants visited or ignored locations based on what they had learned from exploring the document organisation during preceding questions. Participants also used keywords that were successful in earlier searches and used keyword spelling variations they found when exploring document organisation.

3.2 Predicting Which Locations Contain Answers

Several participants voiced in which locations they expected to find answers. For example, four participants voiced in which locations they expected an answer to the first question 1A before they started to search: "*I will first look in SBD*", "*I will look in SBD_history, it has standards settings*", "*Behaviour is in SBDs*", and "*Then I would look in the requirements.*".

After these statements the participants directly navigated to directories, documents, and sections instead of exploring the available document organisation. They acted on their prior knowledge about the documentation. One participant explicitly voiced this: *"From my knowledge I know I should look in SBD_history and SBD_print_settings. I would not expect something in SBD_docbox . . . I however do not claim that this is indeed the case*". Such experience provides a starting point for the search.

The participants intuitively predicted from prior knowledge that certain locations contained relevant information or an answer because they found (similar) information or answers there before. This is an availability heuristic, described by Tversky and Kahneman in [18], which people use to estimate the probability or frequency of an event by recalling occurrences of similar events from their memory.

Correctly predicting that a location contains an answer resulted in a large gain. Namely, participants that directly navigated to locations found 19 answers to questions in the expected location and spent, on average, 37% less time compared to the average time spent on searching these answers.

Incorrectly predicting that a location contained an answer resulted in a small loss compared to the gain above. Namely, one participant wasted 70 seconds searching for an answer that was not in the expected location. The participant however still spent less time than average to answer this question. After the unsuccessful search in the expected location, the participant used other search strategies (explore document organization and keyword searching) and then found the answer. He used an agile search approach by switching to a different search strategy after the initial search strategy did not work.

3.3 Predicting Which Locations do not Contain Answers

Prior knowledge was also used to predict that an answer could *not* be found in certain locations, namely, in specific directories and documents. Participants ignored locations, i.e., they did not search in locations where it was unlikely to find an answer. This helped to cut down the search space and thereby save time. However, participants also gave incomplete and incorrect answers to questions because they ignored certain locations.

One of the participants said that a document containing answers to questions 1A and 1B was not related to these questions. During question 1B he voiced: "*SBD_print_settings has nothing to do with [behaviour] 'alert light'. This I know.*". Three participants ignored locations with answers, but gave no explicit reason as to why they ignored the locations. Another participant said that he decided to not open the Sysref document because he was not very familiar with it: "*I cannot do much with the Sysref . . . I do not really know the Sysref that well*".

In [18] Tversky and Kahneman describe how estimating the occurrence of an event is affected by the ease with which one can bring instances of this event to mind from personal experience. Events that are familiar to a person are more easily retrieved from memory than less familiar events, and this biases the use of availability heuristics. The participant that explicitly voiced that he was not familiar with a location had difficulty in recalling examples of answers in this location. The participant chose to visit locations he was more familiar with and not the location that he was less familiar with. This suggests that the participant missed answers because of availability bias.

Correctly predicting that a location can be ignored, i.e. ignoring a location that indeed does not contain answers, resulted in a small gain, namely, time was saved by not having to inspect this location.

Incorrectly predicting that a location can be ignored however resulted in a large loss compared to the gain above. Namely, participants did not find complete answers to 7 questions because they ignored the location containing the answer. Moreover, they wasted time searching for answers in other locations that they did not ignore. When these participants did not find answers they did not reconsider and check the locations they ignored. We asked all participants in the study to search until they were satisfied with the time spent and answer found. In this case the participants decided to stop searching without finding an answer within reasonable time.

3.4 Predicting Which Keywords Lead to Answers

The names of certain knowledge elements, e.g. decisions, settings, and subsystems, can be recorded using different spelling variations and acronyms. For example, the component in questions 2 and 3A has three spelling variations in the documents; '*settings editor*', '*settingseditor*', and '*setting editor*', and one acronym; '*SE*'. Keyword searching for only

one of these spelling variations does not return all locations that mention this component.

A participant voiced this problem quite clearly after keyword searching for requirements realized by component settings editor (question 2): "*I am not sure if [my answer] is complete. There could be requirements that do not contain the name 'settings editor'. Or [the name] is recorded differently*". Another participant voiced concerns about how to spell the interface fo question 4A: "*IJ-I. I wonder if there are different ways of writing it*". One participant emphasized the importance of prior knowledge in this situation: "*So context, about how we call certain things within Océ, is really needed to search fast*".

Moreover, certain keywords only led to part of the answers, because these keywords were not recorded in all descriptions of these answers. This was often the case for keywords that indicated a type of knowledge. For example, only part of the decisions could be found using keyword 'decision' because several descriptions of decisions did not contain the actual word 'decision'. People used prior knowledge to predict the 'coverage' or 'frequency' of keywords.

We observed that 8 of the 26 participants used part of a name in their keyword search, which allows multiple spelling variations to be covered in one keyword. For example, they used keyword '*editor*' to search for component '*settings editor*' in question 2 and 3A. One participant voiced this use of partial keywords for question 2: *"Maybe I can search for something like 'setting' or 'editor'"*. Participants that used partial keywords however found much knowledge that was irrelevant for their question, and this resulted in lower average efficiency than the use of full names when keyword searching.

The participants had a clear preference for using certain keywords over others. They used keyword searching when trying to find 56 of the 90 answers, but in only 7 cases they searched for all the keywords phrased in the question they tried to answer. Participants voiced that they are familiar using certain keywords: "*I do not know how this is always written in the text so I always search for settings editor concatenated and settings editor with a space in between*" and "*Normally I would have to search for keyword 'decision'*". One participant used keyword '*requirement*' to answer questions 1B and 2 because he had successfully used the keyword during preceding question 1A: "*I already have an approach that works for [searching] requirements*".

Several participants in our study voiced that they used certain keywords because they were familiar with using these. They recalled from experience that these keywords could be used to find answers, however, these keywords did not always lead to answers in this study. This selection of keywords based on familiarity suggests an availability bias.

Accurately searching for keywords, i.e. using keywords that lead to descriptions of answers, resulted in a large gain. Namely participants found 34 answers to questions by keyword searching even though they spent 17.7% more time than the average for these questions.

Keyword searching allowed participants to find answers without having to spend a lot of time on exhaustively exploring document content. Several participant voiced this as their motivation: "*I find a lot of text here so I will switch to keyword searching*", "*It is a relatively huge document so what I will do is a keyword search*", and "*I have no other option than to use keyword searching*".

Inaccurately searching for keywords, i.e., using keywords that do not lead to descriptions of answers, resulted in a large loss compared to the gain above. Participants could not find 22 answers to questions via keyword searching and spent 8% more time on these questions than the average time required for these questions. These participants also used other search strategies, however, they gave up searching without finding a complete answer to 15 of these 22 questions. They gave up partially because relatively much time was spent on keyword searching without finding an answer.

3.5 Predicting Whether Answers can be Found in Multiple Locations (Triangulation)

After finding an answer in some location, several participants tried to verify whether the answer was complete and correct by searching in other locations. This strategy is called *triangulation*, which we describe in Section 2.2. This strategy is applied when, after a participant finds an answer in one location, s/he navigates to other locations to verify and improve the answer.

Participants predicted from prior knowledge whether they could find the same - possibly more complete - answer in another location. For example, one participant first found an answer to question 4B by searching in document Sysref and then voiced "*there should be a diagram here somewhere*". The participant then visited the UML document to find a more complete answer.

The answers for questions 1A, 1B, 4A, and 4B were described in multiple locations. For example, participants looking for settings whilst answering question 1B often could not find all the settings in *SBD_Alert_Light*. Settings were not very explicit in this SBD and the section titles did not clearly show where an answer could be found. Participants who also looked for an answer in *SBD_Print_Settings* would however find the settings more explicitly recorded.

Accurately triangulating answers, i.e. an answer is improved by searching for the same answer in another location, resulted in a large gain. Participants improved the completeness and correctness of 3 answers by triangulating the answers, even though they spent 95% more time on average (one participant spent 420% of the average time finding 1 of the 3 answers). 16 out of the 90 answers (18%) in this study could have been more complete and correct if the participants had triangulated their answers.

Inaccurately triangulating answers, i.e. searching for the answers in another location but not finding relevant information to improve the answers, resulted in a small loss compared to the gain above. Six participants triangulated one of their answers in another location but did not improve the answer. They spent on average 23.6% more time than other participants that answered the same questions.

3.6 Estimating Whether an Answer is Complete and Correct

Most participants had extensive experience using the documents and building the software specified in it. Several of these participants had a good idea what would be the likely answers to the questions. When participants said that they knew an answer from prior knowledge, we instructed them to nevertheless answer the question using the documentation.

Participants used their prior knowledge about possible answers to recognize answers while searching and to estimate the completeness and correctness of the answers they

found. This often worked well, however, in some cases the prior knowledge about possible answers was incomplete. We found that 5 out of the 26 participants made a false assumption because of this.

These 5 participants falsely assumed that an answer they knew from prior knowledge was complete and correct, whilst in reality their answer was incomplete. Two of these five participants gave an answer that was both incomplete and incorrect.

For example, one participant voiced: *"I think my answer is right and I am thus satisfied. I however already knew this was the answer"*. Another participant who had to find two settings for question 1B voiced: *"this is indeed the only setting"*. All these participants had the missing parts of their answer on screen for some time but ignored this.

The search behaviour of these 5 participants was affected by confirmation bias [12]. The participants searched for answers that they knew from prior knowledge and confirmed that these answers were recorded in documentation, i.e., they confirmed their prior beliefs. In the process they ignored other answers and information that was inconsistent with these beliefs.

Accurately estimating whether an answer is complete and correct based on prior knowledge resulted in a small gain. 10 participants gave a correct and complete answer that they claimed to already know from prior knowledge. These participants spent, on average, 3% less time to find this answer than other participants.

Inaccurately estimating that an answer is complete and correct based on prior knowledge or prior belief, resulted in a large loss compared to the gain above. Five answers to questions that were assumed to be correct and complete by 5 participants were in fact incomplete and incorrect. The participants answered the five questions in 74% of the average time that other participants spent on these questions. This was because they searched briefly or they stopped searching immediately after finding an answer known from prior knowledge.

4. LESSONS LEARNT

4.1 Lessons for Documentation Users

Prior knowledge can be used to quickly find correct answers to questions when the document organisation does not support the questions. We found that the use of certain prior knowledge yields larger gains than other prior knowledge. Moreover, there is a difference in losses when incorrect predictions are made based on prior knowledge.

Table 3 shows that it was rewarding for participants to visit the expected location of an answer and to triangulate an answer in multiple locations known from prior knowledge. If this prior knowledge proves to be incorrect the loss is small, and it is therefore relatively safe to use. Using prior knowledge to predict which keywords lead to an answer often yields a large gain, however, incorrect predictions may result in a large loss and this prior knowledge should thus be used with caution. Ignoring certain locations or estimating that an answer is complete and correct from prior knowledge yields a small gain and such cognitive biases increase the chance of a large loss.

Being more aware of cognitive biases can prevent the aforementioned losses. It may be hard to remember examples of answers being recorded in a certain location because one is not familiar with this location, however, this does not imply that the location indeed contains no answers. Certain keywords are easily remembered because they are often used and familiar. Using these keywords in searching may however be counter-productive. Prior knowledge about the answer to a question may be incorrect, incomplete, or outdated.

Existing prior knowledge can be evaluated and updated, by thoroughly exploring document organisation and content, i.e., conducting empirical investigation and seeking disconfirmatory evidence [15]. This prevents inaccurate predictions and cognitive biases later on. An additional benefit of exploring the document organisation and content is that it can remove search uncertainty and the need for using prior knowledge. A searcher may find document organisation that is fitting for the question asked, and this organisation often leads to complete and correct answers.

4.2 Lessons for Documentation Writers

We observed several causes for search uncertainty:

- Document organisation does not relate to a question because document writers do not plan the document organisation to answer all questions that could arise.
- Documents that are not searched might contain answers.
- The same knowledge might be referred to by multiple spelling variations and acronyms.
- The type of knowledge might not be consistently recorded.
- Complementary knowledge might be described in multiple locations.

Consequently, a searcher might only become certain that an answer is correct and complete when all text in the available documentation is read. We observed that none of the participants exhaustively inspected all available document contents. Participants either quickly found answers using document organisation that was fitting for the questions, or they used their prior knowledge to predict which locations and keywords were relevant when searching.

Searchers make predictions from their prior knowledge to deal with search uncertainty. These predictions can however be inaccurate and are prone to cognitive biases. This results in inefficient and ineffective use of documentation, and in turn lowers the incentive to spend resources on producing good documentation, creating a vicious cycle [13]. Moreover, incomplete and incorrect answers are used to build software and may result in costly errors.

Addressing causes for search uncertainty removes the need for searchers to use prior knowledge and lowers the chance that cognitive biases occur. Creating a document organisation that fully relates to commonly asked questions removes much search uncertainty. Introducing spelling conventions and consistently recording what type of knowledge is described makes keyword searching more efficient and less error-prone. Recording the same type of knowledge in one location prevents scattered descriptions of the same knowledge that can become inconsistent over time, and in turn removes the need for searchers to triangulate answers.

These solutions are difficult to realize when writing and maintaining documentation in a linear file-based format with

multiple authors. As an alternative, ontology-based documentation can address above issues, whilst it also provides benefits from the use of explicit semantics and non-linear organization of knowledge. Ontology-based documentation encapsulates relationships between pieces of information, which allows users to traverse and search knowledge more effectively [4]. Participants in our study were also asked to retrieve knowledge from ontology-based documentation, as part of a larger experiment reported in [4]. We found that the use of ontology-based documentation was significantly more efficient and effective for answering the questions as compared to the use of file-based documentation [4].

5. THREATS TO VALIDITY

The participants also used ontology-based documentation to answer questions as part of the larger research reported in [4]. We did not find evidence that the use of the ontology-based documentation approach had an influence on how participants used the reported search strategies and their prior knowledge in file-based documentation.

To verify the consistency of the used encoding scheme, described in Section 2.2, two researchers independently applied the scheme to identify search actions from the videos. After applying the scheme they checked if they came up with the same set of search actions.

We constructed PBGs using the format depicted in [2] and Figure 1. We observed that participants acquired new knowledge with each executed search action. For example, one participant voiced: "*I have already seen that this knowledge is described in SBDs and not in SADs.*". As such the participants did not return to a previous state of knowledge, which is proposed as a PBG modelling construct by Newell and Simon in [11] but not applied in this paper.

The use of SADs, SBDs, Sysref, and design documents, and the search functions of the tools used for searching these documents, can be considered generic documentation practice in industry. This suggest that the identified search strategies are also generalizable to other software projects.

The evaluation in terms of efficiency and effectiveness in this paper is specific to the questions and Océ documentation used in this study. The use of prior knowledge and cognitive biases that may occur are however largely independent of the question asked and documentation searched. The use of prior knowledge and impact of cognitive biases also apply when searching in software documentation in other domains and is therefore generalizable.

6. RELATED WORK

How knowledge is used in software documentation is systematically reviewed in [5], and in this section we only discuss related work on using prior knowledge to search software documentation.

Chen and Dhar describe in [2] how prior knowledge is used during online document-based information retrieval and how prior knowledge affects the selection of search strategies. In [14] Shute and Smith identify the use of prior knowledge as 'subject-dependent expertise' for searching bibliographic databases. For example, the 'known-item-instantiation' strategy described in [2] uses subject-dependent expertise. Shute and Smith describe how participants talk about using their intuition and "gut feeling", which suggests that they use their prior knowledge.

In [3] de Boer and van Vliet describe how software professionals in the same team have similar mental models of documentation. They have a shared understanding of the contents of these documents. Moreover, the development process affects the level of shared understanding within a team. Such mental models of documentation are part of the prior knowledge that we discuss in this paper.

In [9] LaToza *et al.* describe how developers rely on implicit code knowledge and spend much effort to maintain a mental model of code. Developers recall details, e.g., about the architecture and design of their code, as part of their prior knowledge, i.e., they know code details by heart. Moreover, they often do not consult documentation to check if their mental model is consistent with documentation.

In [17] Tang describes how software design is affected by cognitive biases. Designers use prior beliefs and intuition to make judgements, and cognitive biases may occur because of this. In [1] Calikli *et al.* investigate how confirmation biases during software testing can be prevented.

In [15] Stacy and Macmillian describe how software professionals develop and use their mental models during software engineering activities and how this is influenced by cognitive biases. For example, code features that are easily remembered by software professionals may be judged to occur more frequently than other features due to availability bias. They suggest that keywords that are very long, occur in recently read documents, or occur in code recently worked on may be more easily remembered than other keywords, and this may cause availability bias. Similarly, we observed that software professional tend to search keywords that are familiar and easily remembered from prior knowledge, and that this may result in availability bias.

In [8] Korkala and Maurer identify communication waste, e.g., outdated and scattered information, in a software project. The software documentation in our study is used to communicate knowledge. As such, the losses during the use of prior knowledge, described in sections 3.2 through 3.6, can be regarded as communication waste. The identified causes for search uncertainty can in turn be regarded as causes for communication waste.

In [16] Su *et al.* used Information Foraging theory to explain how software professionals search for architectural knowledge in document sections using several foraging styles. Information Foraging theory tries to explain the search behaviour of people in terms of cost and reward when navigating an information topology. The study in [16] however does not evaluate whether the observed search behaviour is time-efficient and does not focus on use of prior knowledge.

In [7] Ko *et al.* report an exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. Their analysis of information seeking behaviour of software developers relates to our work in which software professionals also exhibit information seeking behaviour. However, software maintenance information is searched in source files and on the Internet, whereas in our work software professionals search for knowledge in architecture documents.

7. CONCLUSIONS

Software documents are used to capture and communicate knowledge in software projects. It is important that this

knowledge can be retrieved efficiently and effectively, to prevent wasted time and errors that negatively affect the quality of software. The organisation of software documentation typically does not support all of the questions asked by software professionals. This introduces search uncertainty and makes it hard for software professionals to find complete and correct answers within reasonable time.

We conducted an industry study to investigate how software professionals search for knowledge in software documentation. We found that professionals use their prior knowledge to find answers when the document organisation did not relate to the questions they had to answer. Prior knowledge was used to make predictions about the location of knowledge, which keywords can be used to search relevant knowledge, and whether the knowledge found is correct and complete.

Using prior knowledge is often time-effective, however, inaccurate predictions and cognitive biases can lead to inefficient and ineffective knowledge retrieval. Availability bias may cause searchers to ignore locations and keywords that they are not familiar with, even though these locations and keywords may lead to answers. Using prior knowledge is also prone to confirmation bias when searchers mainly focus on confirming the answers that they already know from their prior knowledge.

Awareness of these cognitive biases may reduce the likelihood that they occur when searching in software documentation. Searchers can evaluate and update their existing prior knowledge by spending time on exploring the document organisation and content, which further reduces the probability that cognitive biases occur. Addressing causes for search uncertainty when writing documentation removes the need for searchers to use prior knowledge and in turn prevents that cognitive biases occur.

Acknowledgements

The authors wish to thank René Laan, Wim Couwenberg, Pieter Verduin, Amar Kalloe, and the other good folks at Océ R&D for their support, interest to participate in this research, and excellent insights. This research has been partially sponsored by the Dutch "Regeling Kenniswerkers", project KWR09164, "Stephenson: Architecture knowledge sharing practices in software product lines for print systems" and by the Natural Science Foundation of China (NSFC) project No. 61170025 "KeSRAD: Knowledge-enabled Software Requirements to Architecture Documentation".

8. REFERENCES

[1] G. Calikli, A. Bener, and B. Arslan. An analysis of the effects of company culture, education and experience on confirmation bias levels of software developers and testers. In *International Conference on Software Engineering (ICSE)*, pages 187–190. IEEE, 2010.

[2] H. Chen and V. Dhar. Cognitive process as a basis for intelligent retrieval systems design. *Information Processing & Management*, 27(5):405 – 432, 1991.

[3] R. C. de Boer and H. van Vliet. Writing and reading software documentation: How the development process may affect understanding. In *ICSE Workshop on Cooperative and Human Aspects on Software Engineering (CHASE)*, pages 40–47. IEEE, 2009.

[4] K. A. de Graaf, A. Tang, P. Liang, and H. van Vliet. Ontology-based software architecture documentation. In *Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA)*, pages 121–130. IEEE, 2012.

[5] W. Ding, P. Liang, A. Tang, and H. van Vliet. Knowledge-based approaches in software documentation: A systematic literature review. *Information and Software Technology*, 56(6):545 – 567, 2014.

[6] K. A. Ericsson and H. A. Simon. *Protocol Analysis: Verbal Reports as Data*. MIT Press, revised edition, 1993.

[7] A. Ko, B. Myers, M. Coblenz, and H. Aung. An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. *IEEE Transactions on Software Engineering*, 32(12):971–987, 2006.

[8] M. Korkala and F. Maurer. Waste identification as the means for improving communication in globally distributed agile software development. *Journal of Systems and Software*, 2014. http://dx.doi.org/10.1016/j.jss.2014.03.080.

[9] T. D. LaToza, G. Venolia, and R. DeLine. Maintaining mental models: A study of developer work habits. In *International Conference on Software Engineering (ICSE)*, pages 492–501. ACM, 2006.

[10] T. C. Lethbridge, J. Singer, and A. Forward. How software engineers use documentation: The state of the practice. *IEEE Software*, 20(6):35–39, 2003.

[11] A. Newell and H. A. Simon. *Human Problem Solving*. Prentice Hall, 1972.

[12] R. S. Nickerson. Confirmation bias: A ubiquitous phenomenon in many guises. *Review of General Psychology*, 2(2):175–220, 1998.

[13] D. L. Parnas. Precise Documentation: The Key to Better Software. In *The Future of Software Engineering*, chapter 8, pages 125–148. Springer, 2011.

[14] S. J. Shute and P. J. Smith. Knowledge-based search tactics. *Information Processing & Management*, 29(1):29 – 45, 1993.

[15] W. Stacy and J. MacMillan. Cognitive bias in software engineering. *Communications of the ACM*, 38(6):57–63, 1995.

[16] M. T. Su, E. Tempero, J. Hosking, and J. Grundy. A study of architectural information foraging in software architecture documents. In *Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA)*, pages 141–150. IEEE, 2012.

[17] A. Tang. Software designers, are you biased? In *Proceedings of the 6th International Workshop on SHAring and Reusing Architectural Knowledge*, (SHARK), pages 1–8. ACM, 2011.

[18] A. Tversky and D. Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131, 1974.

[19] M. W. van Someren, Y. F. Barnard, and J. A. Sandberg. *The Think Aloud Method - A practical guide to modelling cognitive processes*. Academic Press London, 1994.