Modelling the Information-Seeking Behaviour of Programmers – An Empirical Approach

Michael P. O’Brien & Jim Buckley

Department of Computer Science & Information Systems
University of Limerick
{michaelp.obrien, jim.buckley}@ul.ie

Abstract

Information seeking has been recognised as a core subtask in software maintenance. This paper reviews, merges, and adapts existing information seeking models for different domains to propose a non-linear information-seeking model for programmers involved in software maintenance. Talk-aloud data from two empirical studies of industrial programmers carrying out real software maintenance tasks is presented, to illustrate and validate (in part) this model.

1. Introduction

Programmers find themselves daily in situations where they must make a decision, answer a question, locate a fact, comprehend something, or solve a problem. Studies by Singer et al. [25] clearly show that programmers spend a considerable amount of their time seeking information, in other words, at the heart of the majority of their tasks, is an information requirement(s). In all their studies, Singer et al. found that searching is done far more often than any other activity.

Sim [27] refers to programmers as task-oriented knowledge seekers, focusing specifically on getting the answers they need to complete a task using a variety of information sources. Over the past few decades, research has been carried out to examine the information seeking and retrieval methods of selected professions such as librarians, social scientists, physicians and engineers [18, 5, 12, 13, 3]. However, attempts to discern the information needs of computer programmers have only been undertaken recently, with very a very limited number of studies carried out. These studies have mainly relied on questionnaires and interviews to probe the information sources and needs of software engineers [24, 26].

This paper discusses information seeking theory and presents several currently accepted models that attempt to conceptualise the information seeking process in various different domains. It identifies the main information sources available/used by programmers and expands the literature by proposing an information seeking process for programmers. Talk-aloud data from preliminary empirical studies of two professional programmers in-situ, maintaining large commercial software systems, is presented to illustrate this model.

1.1. Information Seeking

Problem solving is one of the main reasons for seeking information. Essentially, problem solving can be defined as “thinking that is directed toward the solving of a specific problem that involves both the formation of responses and the selection among possible responses” [28].

The term information seeking often serves as an umbrella overarching a set of related concepts and issues. In the simplest terms, information seeking involves the search, retrieval, recognition, and application of meaningful content [13]. This search may be based on specific strategies or serendipity, the resulting information may be embraced or rejected, and the entire experience may be carried through to a logical conclusion or aborted in midstream. Indeed, there may be a million other potential results.

Information seeking has been viewed as a cognitive exercise, as a social and cultural exchange, as discrete strategies applied when confronting uncertainty, and as a basic condition of humanity in which all individuals exist [13]. In fact, information behavior may be a more encompassing term, rather than information seeking, to best describe the multi-faceted theory [21]. As seen in Figure 1, problem solving is a superset of information seeking.
Figure 1 – Information Seeking in Context

The information seeking process itself may lead to either a success or a failure. If successful, information is located, which will be used. This may result in the satisfaction or non-satisfaction of the original perceived need. Satisfaction occurs when the located information has been analyzed and placed in the context of, for example, the original bug; in other words, satisfies the original need. Non-satisfaction occurs when the information does not satisfy the original need. With non-satisfaction, information seeking may be iterated until satisfaction occurs. A failure to find information may result in the process of information seeking being continued. Krikelas [14] stated that: information seeking begins when someone perceives that the current state of knowledge is less than that needed to deal with some issue (or problem). The process ends when that perception no longer exists. That is, the information seeking process ends when the perceived need has been satisfied.

Pirolli and Card [23] use the term information foraging to characterize the process of information seeking, by suggesting that people when possible, will modify the cognitive strategies they use in this process to maximize their rate of gaining valuable information at lowest cost. A cognitive strategy will be superior to another if it yields more useful information per unit cost [23] and these strategies evolve over time to improve returns on foraging. They suggest that humans seeking information adopt different strategies, sometimes with close parallels to those of animal foragers. They claim that providing people with MORE information is not the problem; rather, the problem is one of maximizing the allocation of human attention to information that will be useful [23].

2. Information Seeking Models

Kingrey [13] states that information seeking is a very difficult discipline to fully conceptualize for the purpose of explanation. However, there has been considerable interest in recent years in producing conceptual models or frameworks for information seeking and retrieval research [11]. These models present a section, or a full sequence of activities, which, essentially, lead to obtaining relevant information [21]. Most of them describe the process of information seeking itself, usually as a set of stages (e.g. [7, 15, 16, 20]). In fact, Large et al [17], claim that in order to carry out a search for any kind of information, it is necessary to progress through a series of stages/steps, typically called the search process. This is the case regardless of whether the information seeker is a novice or has experience, or is a subject expert or beginner, and so on [17].

This section of the paper examines key models proposed to characterize the information-seeking process and subsequently attempts to integrate elements in these and psychology of programming literature, into a behavioural model of the information search and retrieval process of programmers.

2.1. Ellis et al’s Model

Ellis et al [6, 7, 8] propose and elaborate a general model of information seeking behaviour based on studies of the information seeking patterns of social scientists, physicists, chemists, engineers, and research scientists in an industrial firm. Essentially, this model describes six categories of information seeking activities, namely, starting, chaining, browsing, differentiating, monitoring, and extracting.

Starting: This stage of the information seeking process is concerned with the identification of sources of interest that could serve as starting points for the search. (Initial examination of these sources may point to additional sources of information, e.g. a relevant journal).

Chaining: The process of chaining can be carried out in either a backward or forward direction. Backward chaining, which is normal practice among researchers and scientists, takes place when pointers or references from initial sources to previous work on which that document relies, are followed. Forward chaining, less commonly used, takes place when the information seeker identifies and follows up on other sources that refer to an initial source or document (as is possible with on-line facilities, such as ‘Cite seer’ [http://www.citeseer.com])

Browsing: This stage takes place only when the information seeker has located relevant sources and documents. They ‘browse’ through this information in a semi-directed fashion, simplifying the process by
looking through abstracts, summaries, contents pages, and so on.

**Differentiating:** This stage is concerned with the filtering and selection of information relevant to the task at hand, by noticing differences between the nature and quality of the information available.

**Monitoring:** Here the information seeker remains ‘current’ by monitoring the task area at hand, i.e. regularly checking ‘core’ sources.

**Extracting:** This activity is concerned with working through an information source methodically, in order to identify material of interest.

While this model is presented as a very linear or staged approach to information seeking, it is likely, however, that is probably does not ‘strictly’ follow these stages in order. Instead, the information seeking process can be inter-leaved to identify new sources as new information is found.

### 2.2. Kuhlthau’s Model

In a much more specific context but of wider scope, Kuhlthau [15, 16], devised a model of how students search for information as part of a research process. Kuhlthau [15, 16] uses the term *information search process* to describe the initiation, selection, exploration, formulation, collection and presentation, of information. Essentially, this model focuses on the ‘search process’ itself, rather than how to use and evaluate the information found and again, presents a staged (linear) approach to information seeking.

**Task Initiation:** This is the stage when a user’s focus is on determining their context for information seeking and recognizing that information is needed to complete the task or solve a problem.

**Topic Selection:** During selection, the task is to ‘select’ the general topic to be pursued.

**Pre-focus Exploration:** This stage involves gathering information, which is general to the topic, rather than information that is specific or especially pertinent.

**Focus Formulation:** From the information gathered in the exploration stage, the user now forms a focused perspective on the topic. A clear focus enables the seeker to move to the next stage.

**Information Collection:** Here, information specifically related to the defined focused topic is gathered and knowledge of the general subject area is increased.

**Presentation:** The task at this point is to complete the search and to use the findings.

Kuhlthau [16] proposes that this staged model of information seeking is valid across diverse user groups as well as appropriate for describing the search process longitudinally. It is a significant model, in that it presents the user as an ‘active participant’ in the information search process. The user’s knowledge grows as s/he interacts with the information. Throughout the process, the individual engages in cognitive strategies such as brainstorming, contemplating, predicting, identifying, and defining, among others. However, in many scenarios, the context for information seeking, is not set by the individual as presented here, e.g. a software bug, etc., and this must be considered when placing this model in a software maintenance context.

### 2.3. Marchionini’s Model of Information Seeking in Electronic Environments

Marchionini [20] proposes a model of the information seeking process, tailored purposely, to electronic environments. In this model, the information seeking process consists of eight sub-processes, which develop in parallel (unlike the previous models): (1) recognition and acceptance of the problem itself, (2) defining and understanding the problem, (3) choosing a search system, (4) formulating a query, (5) executing the search, (6) examining the results, (7) extracting the information, and (8) Reflecting / Iterating / Stop.

Because these sub-processes are controlled by the information seeker, they most often take heuristic or opportunistic paths according to skills and level of experience. These paths can depend on ongoing judgements about the costs and benefits of the progress being made along with judgments on the information retrieved.

**Recognizing and accepting an information problem:** This stage reflects when the user becomes ‘aware’ of a problem.

**Defining and understanding the problem:** This is a crucial step in the information seeking process. As long as the user is seeking information, this step remains active. Understanding the problem is dependent on knowledge of the task domain and may also be influenced by the setting.

**Choosing a search system:** is somewhat guided by the information seeker’s knowledge of the domain. Experts in a particular domain are familiar with the primary search systems specific to that domain (e.g., research students use books and journals, programmers use their favorite IDE and the source code, etc). Marchionini [20] claims that information seekers prefer colleagues or human sources to formal sources of information but
that in practice however, information seekers consult several search systems as they move toward solutions to their problems.

**Formulating a query:** This stage of the process involves matching understanding of the task/problem with the search system selected. Experts in a particular domain will know the terms that directly relate to query formulation. Marchionini [19] argued that electronic systems significantly help in this process, as they provide a much broader range of ways to articulate queries.

**Execute the search:** This is essentially, the “looking-up” stage, driven mainly by the information seeker’s mental model of the search system.

**Examining the results:** When a query is executed, a result is obtained from the search system. Here, the information-seeker examines results and judges them in terms of the quantity, type, relevance, and format of the findings, with relevance being paramount.

**Extract the information:** This stage bears the same name as Ellis’ “extracting” activity, discussed earlier, but the two processes are different. Essentially, this extraction stage involves the information-seeker applying skills such as reading, scanning, listening, classifying, copying, and storing relevant information. Ellis on the other hand, defines this “extraction” as “systematically working through a particular information source(s) to identify material of interest”.

**Reflection/Iteration/Finish:** This process can be repeated until the information-seeker’s information needs are sufficient to solve the problem.

Some or all of the above sub-processes will additionally require redefinition and reimplementation if the results of the search are deemed unsatisfactory. Marchionini [20] further elaborates this model by suggesting that, like learning and problem solving, information seeking demands general cognition, knowledge, and skill, and is influenced by personal attitudes and preferences. Belkin et al [1, 2] focus on information seekers’ anomalous states of knowledge. They claim that information seekers are concerned with a problem, but the problem itself and the information needed to solve the problem are not clearly understood. Information seekers must go through a process of clarification to articulate a search request, with the obvious implication that search systems should support iterative and interactive dialogues with users.

### 2.4 Wilson et al’s Model

A model, often employed in research concerned with information use and users, is that proposed by Wilson [30, 31, 32]. The aim of this model is to provide a means for conceptualising the stages of problem resolution within which information seeking is embedded and motivated. In other words it shows the user in action, progressing from problem identification through information seeking, to the stage of problem resolution. Wilson [32] suggests that each stage in this model can subsume the other models presented in this section. The model splits problem solving up into four consecutive process stages and the possibility of new information contributing to an increase in uncertainty and returning the problem solver (information seeker) to a previous stage, is represented by feedback arrows (see Figure 2).

**Problem identification:** Essentially, this is where the information seeker identifies the problem and asks a question, such as, “what sort of problem is this”?

**Problem definition:** Now that the problem has been identified, the individual begins to define the problem and get a tighter definition as to the nature of the problem. (A typical question they may ask at this stage is “what is the nature of my problem”?)

**Problem resolution:** At this stage, the individual may use a variety of information sources to resolve the problem (e.g. “how do I go about finding the answer to my problem”?)

**Solution statement:** This is where the information-seeker is satisfied that the problem has been solved (e.g. “That’s that done”!)

According to this model, the process of information seeking may be carried out at each stage of problem solving and may contribute to the resolution of uncertainty and progression to a subsequent stage. Unlike the previous models, feedback loops exist in Wilson’s model, allowing the information seeker to return to a previous stage, depending on their level of knowledge of the task at particular time.

### 3. Information-Seeking Behaviour in Software Maintenance

This section of the paper explores information seeking behaviour in the context of software maintenance. It examines the types of information programmers’ use during software maintenance activities, which have been covered in the literature. However, to date, the information seeking process of programmers during software maintenance has not been specifically addressed and this section proposes a model to conceptualise this process.
3.1. Information Types

Jarvelin & Repo [9, 10], propose three categories of information people look for and it is proposed that these three types can also be applied to software engineers as they seek information relevant to their maintenance task(s), namely:

- **problem information**: information, which describes the structure, properties, and requirements of the problem at hand
- **domain information**: which consists of known facts, concepts, laws, and theories in the domain of the problem
- **problem-solving information**: this type of information describes:
  - how problems should be seen and formulated
  - what problem and domain information should be used
  - how it should be used, in order to solve the current problem.

These three information categories represent three different dimensions and have different roles in addressing a problem. All categories are necessary in problem treatment, but depending on the task, only some of the categories may be available to a software engineer in solving a problem or answering a question relating to the system under study.

So, what are the information types available to industrial software engineers? Currently, programmers rely on several sources of information about a system they are trying to adapt, perfect, or update [24]. Much of the research in program comprehension focuses specifically on the source code as the main source of information for maintenance engineers. Indeed, many studies show that “source code is basically the bible”, and programmers tend to rely heavily on this representation [26, 24, 29]. Research is needed, however, on what software practitioners consider to be other good information sources. Certainly, the source code is one obvious information source; on the other hand, much information can also be attained from other programmers, from the documentation and from system execution (dynamic representations).

Singer [26] carried out an interview study at 10 industrial sites to probe the work practices of software maintenance engineers (two engineers, who worked on the same system were interviewed from each company). Seven of the ten companies claimed that:

- the source code is the primary source of information used by programmers when carrying out enhancements to software systems.
- when these respondents were asked about *documentation* as an information source, they stated that due to the fact that it can often be inconsistent, they only occasionally refer to it. However, the more abstract the documentation was, the more respondents trusted it. The main reason given for the general distrust of documentation is due to the fact that documentation is time consuming to create and maintain and is therefore, often incomplete.
- Studies carried out by Seaman [24] also reflect the belief among programmers that source code is “king”, but found that maintenance engineers also frequently use *execution traces*. Those who regularly used execution traces claimed they were a most accurate and efficient way to understand a problem, especially, when used in conjunction with the source code.
- when the source code is unfamiliar to engineers, they tend to consult other in-house programmers involved with the system. Seaman found supporting evidence for the role of in-house programmers when surveying 45 software practitioners from two different organisations, each with varying degrees of experience in maintenance and interestingly found that human information sources (i.e. people who presently/previous worked on the system, original developers, operators, etc.) were generally “most accurate” and a “crucial” source of information.

This research into the information sources of professional software engineers is predominantly survey/interview based, and other real-time evidence must be attained.

3.2. An Information-Seeking Model

This proposed model of information seeking behaviour of industrial software maintenance engineers, suggests that the programmer perceives a need in the context of their environment. That is, in a given context (for example, finding a bug or carrying out an enhancement to the system), the programmer will perceive an information need (e.g. locate the bug). The perceived need will lead them to search for information, making demands upon a variety of information sources. As shown in the survey discussed earlier, these information sources include the source code, the executing system, relevant documentation, and other resources, such as colleagues or the original developers.
This theory of information seeking recognizes that programmers seek information from a variety of sources to satisfy their information needs [24]. These needs are often initially vague (possibly due to their level of familiarity with the domain/system) and evolve during the search process. Essentially, this theory illustrates a different emphasis but is built on the models and theories of information seeking discussed earlier. However, stage one of both Wilson’s model and Kuhlthau’s model, where a problem is ‘identified’, is somewhat irrelevant in the case of software maintenance, as the overall problem/task is often defined in advance by someone other than the programmer.

Also, many of the models of information seeking behaviour, such as Kuhlthau’s model (section 2.2), have adopted a representation depicting a linear sequence of activities in which the information-seeker progresses from identifying a problem, to refining a solution. In the case of industrial software maintenance, the author argues that these ‘linear’ models run the risk of under-representing the indeterminate nature of the complex tasks and the consequent twists and turns of human exploration and creativity. It is therefore suggested that the information seeking behaviour of software engineers is an iterative process (more inline with the parallel nature of Marchionini’s model – but different) where the programmer reflects on a particular stage, progresses to another, or iterates a stage(s), depending on the quality and/or usefulness, of the information retrieved. Also, in line with Wilson’s model, programmers may reflect at all stages and may go back a step to should they require to redefine or get a more detailed understanding of the task at hand.

Figure 2 attempts to correlate the key models of information-seeking behaviour in terms of their parallel (analogous) stages. For instance, Wilson’s problem identification stage is somewhat analogous to Marchionini’s selection stage. Likewise, stages two and three of Kuhlthau’s model are analogous to, and on the same “level” of, stage two of Wilson’s model and Marchionini’s exploration stage (stage 2). There is also a correlation between stage 1 of Ellis et al’s model (identification of information sources), stage 3 of Wilson’s model, and stages 3 and 4 of Marchionini’s model, in that here the information seeker is concerned with identifying information sources relevant to solving the problem. Stage 3 of Ellis et al’s model, where the information-seeker extracts the retrieved information is analogous to Marchionini’s stages 6 and 7, and Kuhlthau’s stage 5 (see Figure 2).

Table 1 – Grouping Related Information-Seeking Stages

<table>
<thead>
<tr>
<th>STAGES IN PROPOSED MODEL</th>
<th>RELATED STAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wilson</td>
<td>Kuhlthau</td>
</tr>
<tr>
<td>Awareness of Problem</td>
<td>1</td>
</tr>
<tr>
<td>Focus Formulation</td>
<td>2</td>
</tr>
<tr>
<td>Information Collection</td>
<td>3</td>
</tr>
<tr>
<td>Examine Results</td>
<td>5</td>
</tr>
<tr>
<td>Problem Solution</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 2 – Correlating Information-Seeking Theories

This research proposes a five-stage model, which has at its foundation, the stages associated with aforementioned models of information seeking behaviour (see Table 1). This five-stage proposed model of information-seeking behaviour of programmers is presented in Figure 3.
Essentially, this non-linear model proposes five stages of information seeking behaviour in software maintenance. Each stage(s) can be active at any given time and programmers can start the information seeking process at any point and progress to another stage based on their current knowledge. In identifying a problem, the programmer begins with an explicit or implicit information requirement and actively seeks out information (defining/redefining the problem/task at hand). Understanding the problem is dependent on knowledge of the domain and also the programming language at hand [22].

In the focus formulation stage, queries are generated as to the nature/location of the problem, and from there, the programmer may refer to the knowledge and information sources at hand. In line with Carter (reported in [4]) programmers may formulate queries to discover what is happening, to check that they are on the right track, or form an opinion. Again, this stage may be iterated until the programmer’s information needs are satisfied.

Essentially, the Information Collection stage is where the programmer gathers information specifically related to the solution, and here, their knowledge is increased. The information collection stage is subdivided into three main components:

**Identify Source & Chain:** Here the programmer attempts to identify and collect the information sources available to him/her and begins to follow initial pointers or references from these initial sources (backward chaining). Alternatively, the programmer may identify and follow up on other sources that refer to an initial source or document (forward chaining). In solving a bug, for example, a programmer may decide to use the source code, the system documentation and execution traces.

**Browse & Differentiate:** When the programmer has identified relevant/useful sources of information, they begin to browse through the information (such as the source code, documentation, etc.) and prioritize the information and types of information according to their use in solving the particular problem or information requirement.

**Extract:** Here a programmer may systematically work through a ‘particular’ information source and extract relevant information to assist in solving the problem at hand. For example, working through a particular program and identifying a chunk of re-usable code that can be used in part to resolve the current problem.

When the programmer has gathered (collected) all the relevant sources of information, they examine the results of their search in terms of the usefulness of the information retrieved in solving the problem. The information seeking process is complete when the programmer’s information requirements are satisfied (problem solution). This stage is somewhat analogous to Wilson’s ‘solution statement’ stage. The process (or any stage of it) can then be started again to deal with the next information need.

**4. The Empirical Studies**

This section of the paper presents a preliminary evaluation of the proposed model of information seeking, using talk-aloud captured during observational studies of two software maintenance engineers in-situ, performing real maintenance tasks in their respective software companies. Thus, this research explores a new area of searcher behavior that has not been thoroughly examined before, the information behaviour of professional software maintenance engineers. For this reason, a qualitative methodology was selected. Qualitative methodologies are especially well suited for exploring new areas of research, as they allow for the examination of unknown factors and relationships without having to define these in advance. It is the intention of the author to carry out several more such studies in order to gain a more complete and fuller picture of the information seeking behaviour of industrial programmers and thus refine the model.

**4.1. Empirical Study 1**

The initial study observed, in-situ, one professional programmer who has been employed by a software
development company in the Mid-Western region of Ireland, for over 2.5 years. This programmer has 5.5 years experience programming and was asked to self-assess his knowledge of the programming language under study, COBOL. On a 1 – 5 basis (where 1 is novice and 5 is expert) the participant assessed himself as a 5.

As part of his ‘normal’ routine work, the participant was observed for a two-hour period as he altered and tested several programs to ensure that they incorporated new functionality. The programmer was asked to speak aloud as he worked and all verbalisations were captured on an MP3 recording device. The programmer estimated that the overall maintenance task, of which only two hours were observed, may take over one month to complete. During the two hours the programmer was observed, he altered five programs and ensured they worked correctly with a newly created file.

The overall MIS system being worked on consists of tens of thousands of lines of code and currently 31 people are working on various parts of the system in many parts of the world. Essentially, this system is a health insurance system designed for the United States of America. It was initiated over 20 years ago, is entirely written in the COBOL programming language, and runs on an IBM mainframe computer.

During the observation session, the participant used the following facilities to assist him with the task: Xpeditor (a debugging tool), the JCL compiler, the executing system (TSO - IBM Mainframe), browser, and other programmers’ knowledge.

The following talk-aloud data is presented as a means of validation for the proposed information-seeking model, presented in the previous section of this paper.

As the problem was identified in advance of the maintenance activity observed during this session, the programmer began by restating the general problem (awareness of problem). “Okay… I need to set up a new pseudo-production test region and make changes to current programs to incorporate the new region values…”

An example utterance that shows this programmer attempting to gain a tighter focus of the problem (focus formulation) is as follows: “I really need to ensure that it is reading from the region file just created and ensure it is bringing back the correct values… Let’s see… what exactly do I need to do…”

The following utterances show the programmer implicitly identifying the information sources available: “I need to go back into my TSO session”, “Let’s see… In order to find out this, I’m going to have to take a look at the information I have at this stage…”, “…So I’m going to check my e-mail”, “…I’m going to ask Gillian…”

As the programmer browses through and differentiates between the information available to him (browse & differentiate), he states: “…So I’m going to check my e-mail again to find the library, as I can’t remember it…”, “…I’m going back now to the menu screen – the original screen one gets when they log on… and I’m typing in a command in order to browse the menus for information…”, “…I need to browse the source code and screens and different members…”

The following talk-aloud utterances show the programmer extracting relevant information (extract): “There is it… Let’s have a look at this one… So this particular one, I’ll have to make a change to the OOR field…”, “Okay, I’m going to the debugging and code listing for [filename] and put a breakpoint in the first executable statement…”

The programmer examines the results (examine results): “Okay, so now, let’s see [reflect]… Oh, it’s actually a different convention so I’ll have to go back to [iterate] the program again in the library with the program code that we are changing and I’ll have to do a find on [variable name] again in order to see the first occurrence of it…”

Finally, the following utterances show the programmer reaching the problem solution stage (problem solution): “So, it finished correctly… If it didn’t finish correctly, you would be displayed the line of code that was wrong, in the debugging tool… So, again, that would be deemed a successful code change made in that program…”, “So, that change I made in order to initialize rather than add, is deemed a success because it worked…”

4.2. Empirical Study 2

The second preliminary study observed, in-situ, one professional programmer who has been employed in the Information Technology division of a National Health Service for over 5 years. This programmer has many years experience programming and was asked to self-assess his knowledge of the programming language under study, MUMPS. On a 1 – 5 basis (where 1 is novice and 5 is expert) the participant assessed himself as ‘4’.

As part of his ‘normal’ routine work, the participant was observed for a two-hour period as he carried out an enhancement to the existing health board system. Again, the programmer was asked to speak and all utterances were captured on an MP3 recording device. The programmer estimated that the overall
maintenance task, of which only two hours were observed, might take about four or five days to complete. During the two hours the programmer was observed, he carried out an extensive search to see which programs needed to be updated, inserted a ‘trigger’ into each one, and made several changes in the source code, along with fixing any bugs encountered along the way.

The participant has four years experience with this system, which consists of thousands of lines of code. The system is a community-care system, which deals with payments subvention, granting and reviewing medical cards, drug payment scheme, etc. It was initiated over ten years ago and is written mostly in the MUMPS programming language.

During the observation session, the participant used the following facilities to assist him with the task: the source code, the executing system, the data schema, and the brief assistance of one other programmer.

The following talk-aloud data is presented as a means of validation for the proposed information-seeking model of programmers during software maintenance.

This talk-aloud utterance shows the programmer being aware of a problem (awareness of problem): “So what happens when there’s a change of General Practitioner... because one piece of information on the [patient index] is, GP...”

Now the programmer attempts to get a tighter focus on the task at hand (focus formulation): “…there’s about 1, 2, 3, 4 programs that need to be updated... Let’s see...”

Next, the programmer identifies a relevant source of information and works from there (identify source & chain): “I need to search for the actual point in the source code...”

The following utterances show the programmer browsing through an information source (browse & differentiate): “I need to search for the actual point in the source code where it actually does the update on the audit global...”, “Looking down through the source code here just to see where the update is being done... okay scrolling down through it...”, “I'm going to scroll down through the source code... just to see what's happening...”

The extraction process is indicated in the following talk-aloud data (extract): “Okay... based on this so... the next program will have to be MCSAP6 because I've looked at the other one already”, “Okay... it's coming back with information on the list of clients and I'll choose the first client on the list...”

Examine results: “Okay... so I've found the code that updates the [patient index] for a new medical card number...”, “The search has now come back saying that there is no further calls to [patient index]... so global is not being updated in the program...”

Finally, talk-aloud data to indicate the programmer has reached the stage where a problem (sub-problem) has been solved (problem solution): “We can quit this program now as it has successfully been updated...”, “I can see why it's falling over... we've put in a comma too many... which wasn’t required... now, that's that done...”

5. Conclusions & Future Work

Initial observations from these studies suggest that understanding large commercial software systems seems not guided by the clichéd comprehension strategies, said to be employed during maintenance activities [22]. Instead, the process is driven by an information requirement. Essentially, then this paper examined the information requirements of programmers and proposed a non-linear model of the general information behaviour of programmers as they carry out software maintenance activities. This model is based primarily on the theories of information seeking proposed in the literature for different domains. Future work will attempt to further validate and refine this model by carrying out more empirical studies of software maintenance engineers as they debug, expand, and evolve programs.

6. References


