In P. Romero, J. Good, E. Acosta Chaparro & S. Bryant (Eds). Proc. PPIG 18 Pages 168 - 182

Empirically Refining a Model of Programmers' Information-Seeking Behavior during Software Maintenance

Jim Buckley[1], Michael P. O'Brien[1], Norah Power[1]

[1] Department of Computer Science,
University of Limerick, Limerick, Ireland

{Jim.Buckley, MichaelP.OBrien, Norah.Power}@ul.ie

Abstract. Several authors have proposed information seeking as an appropriate perspective for studying software maintenance activities. However, there is little research in the literature describing holistic information-seeking models in this context. Additionally, in the one instance where an information-seeking model has been proposed, the empirical evidence presented in support of that model is extremely limited. This paper presents a small quasi-experiment that serves to further evaluate and refine this preliminary information-seeking model. Talk-aloud data, generated by two professional programmers, engaged in real software maintenance activities, was captured and then coded. This evaluation largely validated the model but also suggested several important refinements. The study, its results and its impact on the information-seeking model are discussed in this paper.

2 Introduction

Information seeking has been defined as the searching, recognition, retrieval and application of meaningful content [17]. Several researchers have argued that information seeking is a core element of software maintenance [8] [29], [31], [32]. Sim [32], for example, refers to maintenance programmers as task-oriented information seekers, focusing specifically on getting the answers they need to complete a task using a variety of information sources.

Seaman [29] and Singer [31] used questionnaire and interview-based empirical studies to further probe the information sources used by professional programmers during maintenance, and the factors that affected the perceived quality of these sources. They found that programmers relied predominantly on source code, a finding in agreement with that of [33]. However, other valued sources of information that these programmers identified were customers, users, the original development team, other system maintainers, 'Lessons-learnt' reports and execution traces (to recreate software bugs).

Bradac et al [4], and Liu et al. [22] have carried out some related research in the area of information 'Blocking'. Blocking arises when progress on a software engineering activity is halted because the engineers cannot get access to the information

sources they require, when they require it. Both of these studies show that blocking can consume a large proportion of time, in one case up to 60% of the total time required during a software engineering activity [22].

While this work identifies information-seeking as a core software maintenance concern, our literature review suggests that limited research has been carried out to develop a holistic Information-Seeking Model (ISM) for software maintenance activities. In the absence of such a model, there exists no encompassing framework to provide guidance for a more complete research program in this area.

To address this concern, O'Brien and Buckley [25] proposed an ISM for programmers involved in software maintenance. They provided empirical support for their model in the form of talk-aloud data produced by 2 professional programmers involved in real-world software maintenance tasks. However, this support can only be considered provisional as, only two maintenance sessions were assessed and only selected quotations were taken from the talk-aloud transcripts of the programmers.

This paper reports on a quasi-experiment, carried out to further evaluate and refine this ISM. Thus it embodies Basili's assertion that knowledge should be evolved through 'modelling, experimenting, learning and remodelling' [2]. The study reported on here captured talk-aloud data generated by 2 professional programmers, as they maintained a large-scale, proprietary software system in vivo [2]. However, in contrast to O'Brien and Buckley's initial study, all the talk-aloud data generated during this study is classified and reported on. Consequently, the results presented here more fully illustrate the degree to which the ISM is reflected in talk-aloud data.

This paper starts by describing the ISM proposed by O'Brien and Buckley [25]. Section 3 moves on to characterize the work scenarios within which the empirical work was performed and discusses the data-collection protocol employed. Section 4 details the data analysis performed and section 5 describes the alignment of this analysed data with the ISM discussed in Section 2. Section 6 then moves on to discuss refinements suggested to the ISM by the results with section 7 detailing some threats to the validity of our empirical study.

2 An ISM For Programmers

ISMs have been proposed for several domains, including science, psychiatry and industrial engineering [3], [7], [9], [16], [17], [18], [19], [21], [23], [38], [39]. These models break the information-seeking process into a set of phases and stages, through which the information seeker must progress in order to address a perceived need [20]. While some of these models focus on specific stages of the information-seeking process [9], [10], and others generalize over stages expanded on by others [18], [19], [38], [39], 2 core phases and 5 constituent stages can be identified [25]. The first phase is the Problem Oriented phase. This is when the information seeker becomes aware of the problem and is concerned with refining his or her understanding of that problem. It consists of 2 stages:

1. Awareness of Problem: Typically, people seek information in order to solve some perceived problem. This stage refers to when the information seeker first becomes aware of and forms an initial understanding of the problem;

2. Focus Formulation: The information seeker will attempt to define and understand the problem more fully, thus formulating specific queries to be addressed in subsequent stages.

The second phase arises when the information seeker moves to address the problem, and this is called the Solution Oriented phase. It consists of 3 stages:

3. Information Collection: This is the stage where the information seeker identifies, browses and extracts information from various representations, in order to solve the problem and address the queries identified in the first 2 stages. Ellis [9] proposes that this stage consists of 3 sub-stages:
    - Identify Source and Chain: This sub-stage refers to the obtaining of information sources. Here the information seeker identifies sources of information, obtains these sources of information and uses these sources of information to 'chain' (identify) other possible sources of information.
    - Browse and Differentiate: Here the information seeker studies the individual sources of information and identifies seemingly relevant knowledge for extraction.
    - Extract: Here the information seeker extracts information from the information sources.
4. Examine Results: When relevant information has been successfully extracted, the information seeker will assess it in terms of its usefulness towards the initial problem;
5. Problem Solution: The information-seeking process is complete when the seeker's information requirements are sated - that is when the problem has been solved.

These phases, stages, and sub-stages, are the basis for the ISM proposed for software maintainers by O'Brien and Buckley (see Figure 1). This model allows the information seeker to reflect and retreat to any preceding stage in the model in accordance with Wilson's observations on the non-monotonic nature of information-seeking [38], [39], while also allowing progression through the stages linearly.

3 The Empirical Study

These studies were undertaken in the Management Information Systems Department of a National Health Authority. In addition to its other activities, this department maintains a 'Health in the Community' Management Information System (MIS) that was initiated over 20 years ago. The system was approximately 1.4 million Lines Of Code (LOC) in size and was considered by management in the Health Authority to be 'fairly stable'. It was written almost entirely in MUMPs, and runs on VAX Alphas.

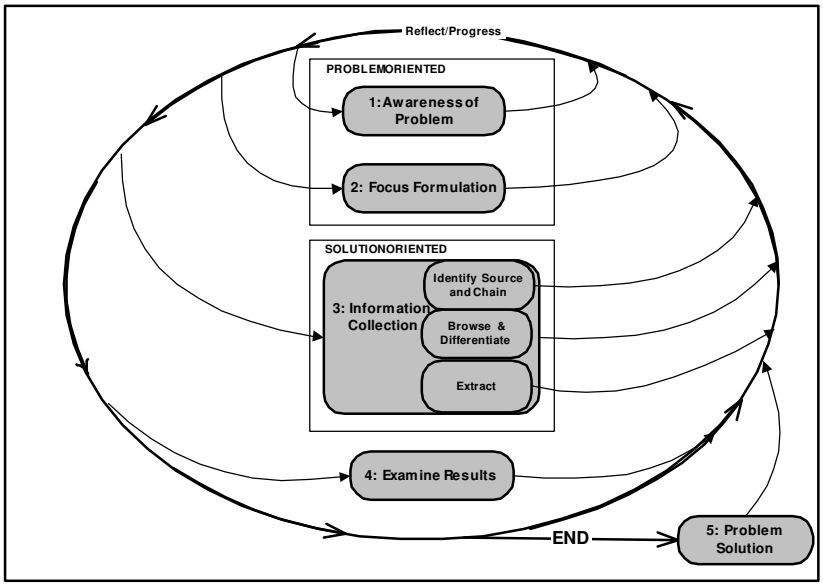

Fig. 1. Preliminary Model of Programmers' Information-Seeking Behavior [25]

As part of their on-going maintenance of the system, managers assign maintenance tasks to programmers periodically. This paper describes the information seeking behavior of two of these programmers whilst performing the maintenance tasks assigned to them by their managers. Hence both tasks reported on here were highly representative of their normal maintenance activities.

3.1 Participants

The first participant was a professional programmer (P1) who had been employed by the Health Authority for the preceding seven years. For 3.5 of those years he was a payroll officer, becoming a programmer-analyst 3.5 years ago (his current position). He rated his knowledge of the programming language used (MUMPs) as 3.5/5 (where 1 is novice and 5 is expert). He stated that, while he had worked on the larger, 'Health in the Community' system, he was unfamiliar with the sub-system that he would be working on for this maintenance task.

This programmer was trying to extract data from one of the system's tables, and generate a flat file, that could be used to 'synchronize' the MIS system with that of an external agency. Specifically, his task was to create a batch program that would detect daily changes on an audit file, and copy the records that had been altered out to a flat file format. He started the task 2 days before the study took place and he estimated that the task would take him 3-4 days in total.

The second participant was again a professional programmer (P2), employed by the Health Authority. He had worked as a programmer-analyst in the Authority for the last 4 years and self-rated his ability in MUMPs as 4/5. He stated that he was working with the relevant system in an 'on-going' basis over the last 4 years. He esti-

mated that his maintenance task would take him 4-5 days (including testing) and stated that he was on his third day.

The second participant's session shared many common features with the first. The system was the same and the task was highly related. Here the programmer's task was to ensure that the necessary audit files, to prompt extraction of clients' records to the flat file format, were generated when changes were made to clients' details.

3.2 Protocol

In an initial meeting with the programmers in the Management Information System's Department, the authors requested volunteers to participate in empirical studies of 'software maintenance'. The programmers were told that the aim of this study was to test an academic model of software-maintenance behavior for validity, but they were not given any details as to the nature of this model. Two programmers expressed their willingness to participate in such a study and their managers agreed.

Subsequently, an appointment was made to visit the Management Information System Department, after the 2 volunteer programmers had been assigned maintenance tasks. On arrival, the experimenter asked the programmers to fill in a short questionnaire on their previous experience, the system they were maintaining and the maintenance task they were about to perform. Then the programmers were given a small voice recorder and were shown how it functioned. They were asked to talk-aloud, stating everything that came into their mind as they progressed through their maintenance task. These guidelines are in line with the guidelines suggested by Ericsson and Simon [12] for capturing valid talk-aloud data: concurrent data capture, capturing mental state (rather than attempting to capture mental processes) and telling the participant to report on *everything* that enters their mind. Ericcson and Simon also suggest prompting the participants when they fall silent but, we chose the less invasive protocol of leaving the participant in their working environment without an observer.

When the programmers were ready, the voice recorder was set to 'Record', and placed on their person. The experimenter left the office, the programmer started their task and their talk-aloud data was captured, for later transcription. Roughly 2 hours later, the experimenter returned and, when the programmer next stopped or paused, the recording was halted. Finally the programmers were asked to provide a written summary of the maintenance task they had just performed on a post-study questionnaire. The programmers were each given €100 each for their participation.

4 Results and Data Analysis

Later, the talk-aloud data from both studies was transcribed, generating 43 pages of transcript. The first author carried out a detailed analysis of this data, naming and categorizing each statement or utterance made by the programmers. In grounded theory analysis [35], this procedure is called 'open coding' and is carried out without the aid of a coding manual, the coder effectively creating the categories from scratch. Accordingly, this coder immersed himself in the transcript data, seeking to gain as many insights as possible into the information-seeking behavior of the programmers,

and began to create categories based on the contents of portions of the transcripts being examined. This analysis was performed iteratively, initially on several small sections of the transcripts. Over time, a number of stable categories began to emerge with respect to these sections. Those categories were then applied to other sections of the transcripts and refined by means of merging and renaming categories. Finally, as the set of categories proved increasingly resistant to change, even when applied to new sections of the talk-aloud transcript, a final set of 10 categories was established.

In grounded theory terminology, these categories would be termed: 'saturated'. Saturation is said to occur when a particular concept or code has so much supporting data associated with it that no significant changes to it can reasonably be expected, and additional data can no longer contribute to discovering anything new about it [35]. This determination requires judgment by the researcher, who is said to be creating a theory from the data. The 10 categories discovered thus are as follows:

- Task Statement: An utterance where the programmers stated the global task or sub-tasks they needed to perform. (P2) …'so I need to find out what its actually doing and where its actually doing this particular update…'
- Mechanical-Facilitation: Utterances showing that the programmer is working with their electronic environment to facilitate their information searching during maintenance. These utterances do not suggest actual searching or maintaining, but rather pre-cursors to searching and maintenance. Examples include going into a file as in the following quotation: (P2) '…just going to load up <program name>'
- Mechanical-Search: Utterances stating that the programmer is performing a search. These search-statements can identify the TYPE of representation the programmer is searching (documents, code, the running system, other programmers), the LARGE-grained item-instance within each type that the programmer is looking for (the program name, the document name, the programmer) or the occurrence of some SMALL-grained items within a large-grained item instance. (P2)'…this particular <data file> has to be updated on the system… so search for that…' (LARGE Mechanical Search)
- Found: A statement showing that something has been found, or that something (which was expected) has not been found. (P2) '…to search for that, ah… I've found that particular section of code now…'
- Doing: Utterances stating that a maintenance change will be done, is being done, has been done or will not be done. (P2) '.am going to insert a trigger in here, to update the <audit file>…'
- Explanation: An utterance showing the programmer's knowledge of the system. (P2) 'when a client is given a new card number… for the first time… the card number is kept on the <main client index>…'
- Disruptive-Shifts-in-Focus: These are utterances that reflected the frequent interruptions suffered by programmers during their work: (P2) 'the last section of code seems to deal with… em… (telephone)… Hello, <programmer> here how are you doing… she's looking at it now is she?…'
- Bucket: These are utterances that do not neatly fit into any of the above categories. (P1). '…Its recording now its started…OK…'

Table 1 shows the number of utterances made by each programmer in each category, during their maintenance sessions.

Table 1. Quantifying the amount of utterances in programmer's talk-aloud data by inspection category

Category	P1	P2	Total
Task Statement	11	4	15
Mechanical Facilitation	42	84	126
Mechanical Search (Type)	23	5	28
Mechanical Search (Large)	35	34	69
Mechanical Search (Small)	76	121	197
Found	58	149	207
Doing	142	78	220
Explanation	59	87	146
Disruptive Shifts in Focus	5	13	18
Bucket	32	8	40

5 Results Interpretation

Relating the grounded categories presented in section 4 and the ISM described in section 3 a number of relationships become apparent:

- Task Statement utterances, where the programmer states the global task or sub-tasks they face, reflect the 'Problem Oriented' phase. That is, they reflect either the programmer's initial awareness of the problem or their focusing on that problem. Using an expanded version of the Task Statement example presented in Section 4.1, we can see both of these sub-stages: (P2) 'it doesn't seem to be updating <the client index>… but it needs to somewhere (AWARENESS)… so I need to find out what its actually doing (FOCUS FORMATION) and where its actually doing this (FOCUS FORMATION)…'
- Mechanical Search utterances referring to searching as they do, relate directly to the 'Information Collection' stage in the model, where the information seeker identifies information sources, browses through them and extracts information from them. In terms of the sub-stages associated with the 'Information Collection' stage:
    - 'Mechanical Search TYPE' utterances, where the programmer comments on the type of information source they are going to use in their search, would seem to reflect the 'Identify Source and Chain' sub-stage: (P1) 'I'm going to save out this program… because I need to confer with a programmer (IDENTIFY SOURCE)… to see if I'm going along the right way…'
    - 'Mechanical Search LARGE' utterances, where the programmer is searching for specific large-grained objects like a specified program or document, also seem to reflect the 'Identify Source and Chain'

sub-stage: (P2) '… OK, the next program I think we need to look at… is… <program name> (IDENTIFY SOURCE)'

- 'Mechanical Search SMALL' utterances where the programmer is going through the contents of a large scale object and searching for small grained objects, would seem to reflect the 'Browse and Differentiate' sub-stage of the ISM: (P2) 'Just scrolling down…searching again…and (have found the update line)' (BROWSE AND DIFFERENTIATE).

From the data however, the difference between 'Mechanical Search LARGE and Mechanical Search SMALL was sometimes unclear. Consider when a programmer is browsing through the output of a utility that lists all the programs that access a certain data table (as one did, in order to identify programs relevant to his task). In this instance the programmer is identifying LARGE grained objects (sources) by browsing and differentiating: (P2) 'I'm going to the search utility… and searching for the <data table> (BROWSING)… to make sure that everything is being looked for… this should bring up any of the programs (IDENTIFY SOURCE) we inserted the trigger in…'. In this instance an 'Identify Source' activity is carried out through 'Browsing and Differentiating' the output of a system utility. As both interpretations were equally valid, a decision was made to code all 'MS LARGE' utterances as reflecting the 'Identify Source and Chain' sub-stage.

- Found statements in the transcripts refer to when the programmer finds information. Thus, they relate directly to the 'Extract' stage of the ISM, where the information-seeker takes information from an information source. (P2) '…OK, I found that point in the program (EXTRACT)…'
- A number of 'Doing' utterances made by programmers, where the programmers discuss what they propose to change in the system, seem related to the 'Examine Results' stage in the ISM. More specifically, if the Doing statements are based on information previously found by a programmer, then they can be regarded as a result of the programmer examining the information found. Thus they are reflective of an 'Examine Results' process. Such utterances can be illustrated using an extended version of the example given above: (P2) '…OK, I found that point in the program (EXTRACT)…and I'm now going to insert a trigger underneath it (EXAMINE RESULTS)…'

A refined analysis of the Doing statements was performed, identifying those that were, in part, based on the programmers' previous findings. For P1, 95 out of 142 Doing utterances were found to be finding based and thus reflective of examining the results. For P2, 63 out of 78 Doing utterances were finding based.

Given this correspondence between the observed categories and the stages of the ISM, Table 2 presents the number of utterances observed for each stage in the ISM. It also reports the percentage of total utterances made by each programmer for each stage. The remaining categories: Mechanical-Facilitation, Explanation, Disruptive-Shifts-In-Focus and Bucket, would not initially seem to be related to any category in the existing ISM but this will be commented on further in Section 6.

Table 2. Quantifying the amount of utterances in programmers' talk-aloud data by ISM stage

Category	P1	P1 %	P2	P2 %	Total
Awareness of Problem	3	0.62	1	0.02	4
Focus Formation	8	1.66	3	0.51	11
Identify Source and Chain	58	12.01	39	6.69	97
Browse and Differentiate	76	15.74	121	20.75	198
Extract	58	12.01	149	25.56	207
Examine Results	95	19.67	63	10.81	158
Problem Solution	0	0	0	0	0

6 Evaluation of the ISM

In this section, the results are discussed with respect to the ISM presented in section 2. Firstly section 6.1 discusses the correctness of the ISM in the light of the data obtained here. Section 6.2 then compares the information-seeking behavior of the 2 programmers and attempts to explain these differences based on the programmers' tasks and their past experience.

6.1 Evaluating the ISM

In this section the results from the study are used, to comment on various attributes of the ISM. These attributes include:

- Bloating: Does the data suggest there are unnecessary phases and stages in the model?
- Completeness: Does the data suggest any extra stages or transitions between stages, for inclusion within the model?
- Fit: Does the data suggest the replacement of any phases or stages within the model?
- Iteration: Does the data re-enforce the assertion that the model should be non-linear?

6.1.1 Bloating in the ISM

No 'Problem Solution' and few 'Problem Oriented' utterances were made during the maintenance sessions. However, on reflection, this is hardly surprising given that both sessions were small segments of bigger maintenance tasks. So, for example, neither programmer finished their task and thus, neither programmer generated utterances that could be placed in the 'Problem Solution' stage. Likewise, both programmers were 2-3 days into their maintenance task and focusing on the primary task seemed to be unnecessary at that stage. In fact, only one utterance, made by P1 at the start of his session, referred to the global task: (P1) 'The job that I am required to do is write a program that will extract the data from the <client index>… this extracts on the basis of <an audit table>... In other words if you corrected something on the system that the

job that I will be doing will extract <updates> done'. Most of the Problem Oriented utterances that we found reflected the sub-tasks that the programmers were trying to achieve instead: (P2) '…and just now we are going to have a quick look at… to see if we need to insert the trigger on the <named sub-system>…'

6.1.2 Completeness of the ISM

The preliminary ISM does not entirely encompass the software maintenance process. Indeed, this is to be expected: software maintenance, by definition, involves changing the software system. At first glance, such changes would seem to be beyond the scope of the ISM.

Previously we incorporated many of the Doing utterances in the model by stating that they reflected the examination of results. However, it could also be argued that such changes should be part of the ISM in their own right. In this reading, programmers interpret the results of their information collection and change the system Thus they create a new representation for subsequent study, feeding back into the information-seeking cycle. The execution of updated code, for example, gives a behavioral representation of the system that was employed by the programmers: (P2) ' …what I need to test is that the program takes that it has been updated… that they all will put people on the <audit file>… so the best way to test this is to go in and change people's details for the programs that were updated…' Alternatively, the updated code itself could be the basis for a review by another programmer, allowing directional information to be obtained: (P1) '… to see if I'm going along this the right way, so I'm just going to print out the program and just ask <other programmer>'

While this type of behavior was not prevalent in the maintenance sessions, it did exist and it is likely that, as the programmers moved towards completion of their tasks, the behavior would increase. These observations argue for the inclusion of an 'Information Prompted Action' stage in the ISM that feeds back into further information-seeking activities.

The data from this study also suggests that the model should accommodate interruptions. On average, there were 9 disruptive interruptions for each programmer during their 2-hour sessions. In one programmer's session alone, there were 13 such episodes, over the 2 hours. These ranged in duration from a couple of seconds ('I do (want)…coffee, yeah') to the more disruptive interrupts of lunch and requests for assistance on other systems: (P1) 'Chinese takeaway… that would be really good……', (P2) 'date of birth, sex, title…<programmer> just wants to ask me a question, so hold on…'

Disruptive interruptions have been seldom studied in this area. Most of the software comprehension studies performed to date have been tightly controlled experiments where disruptions were not allowed [36], [24], [6]. In contrast, this study suggests that disruptions, and indeed frequent disruptions, are part of every-day life for programmers involved in maintaining software systems. These disruptions were sometimes long (one lasted for over 10 minutes in our study) and often made the programmer spend time re-focusing on their current state-of-play, when they returned to their maintenance task. These findings, related to the 'blocking' work referred to earlier [4], [22] suggest that this would be a worthwhile area of focus for further research.

6.1.3 Fitting the Data to the ISM

As discussed in Section 5, it can be difficult to discriminate between 'Identify Source and Chain' episodes and 'Browse and Differentiate' episodes in the data: (P1) 'the first (table) I see is the <client index>… this holds a volume of data, the <ID number> … the surnames, forenames, date of birth (BROWSE)……ah the <health support table> (EXTRACT) is another <table> that I need to look at… (DIFFERENTIATE or CHAIN SOURCE) and this holds…'

Ellis's sub-stages seem to be appropriate for industrial engineers and scientists (Ellis's original domain), who use journals and white papers. Here the 'Sourcing and Chaining' stage can be explicitly related to identifying journal articles and cross-referencing them. 'Browsing and Differentiating' can be related to reviewing the abstracts of selected articles, looking at content pages and selecting pieces to focus in on.

However, Marchionini's [23] model would seem to be more relevant for programmer involved in software maintenance. Marchionini's model expressly deals with electronic environments, the predominant working environment of maintenance programmers. In this model, the 'Solution Oriented' stages involve identifying a search system, formulating a query for that search system, executing the search and examining the results. There are many instances indicative of this in the study, where the programmers explicitly followed many, or all, of these steps. For example: (P1) '…just confirming that this one such program has (does not exist) <name of program> (FORMULATING QUERY)… and its saying to me, I'm loading it up (EXECUTING SEARCH)… and it came back to me no such program (EXTRACT). So what I'm going to do … <is create that program>' (EXAMINING RESULTS). Here the programmer implicitly identifies a 'directory-type' utility as a search system, formulates his 'program-name' search, executes the search and examines his results. Another example can be seen in section 6.1.5 (starting ' I know there is a download…')

Another advantage of using this model is that it allows incorporation of many of the Mechanical Facilitation utterances made by programmers, (as these often indicate that programmers are trying to 'identify a search system').

In Marchionini's model, 'examining the results' is equivalent to the 'extract' stage in Ellis's model. We propose keeping Ellis's 'Extract' terminology, so that the stage: 'Examine Results' can be used to represent the interpretation of extracted information. The ease of modeling programmer behavior with this hybrid model argues for the adoption of these sub-stages in the 'Solution Oriented' stage.

6.1.4 The Iterative Nature of the ISM

One of the innovative aspects of the ISM proposed by O'Brien and Buckley is that information seeking is recognized to be non-monotonic. That is, it allows information seekers to retreat back to earlier stages in the model if required [25]. The extension of existing ISMs in this way is supported by the talk-aloud data captured here. While many of the utterances reflected a sequential process through the stages, there were a number of occasions when the programmer skipped a stage or where they retreated back to an earlier stage: (P1) '…so <Program> is a routine that inputs the details and it kills a format file, creating a file called <program name.txt>(EXTRACT)… and that's exactly what we need to do… (EXAMINE RESULTS) but different information

we require (FOCUS FORMATION)… so I'm looking at this program (BROWSE AND DIFFERENCIATE)…'

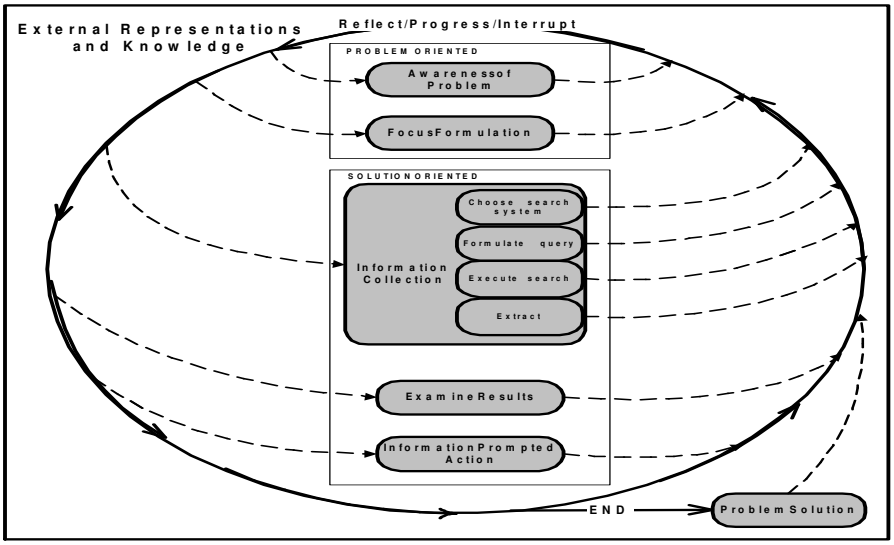

Fig. 2. Refined Model of Programmers' Information-Seeking Behavior

6.1.5 Knowledge-Based Information Seeking

The model, as it stands, doesn't explicitly allow for programmers to obtain information from recall, a prevalent feature of all software comprehension models [5], [13], [37], [26]. There were many episodes in the talk-aloud data where the programmers demonstrated knowledge (146 'Explanation' utterances were found) that allowed them localize potential changes in the system. 2 examples from the talk-aloud data are: (P1) 'I know there is a download…there is… a…system programming on the same principles as what we are trying to achieve…so I…do a search (using a system utility that allows programmers to see each file and their function) within the system and… at the moment I have come up with the name of a program <program name>'. (P1) 'I'm going to ask <programmer> do I need that .. (I know) he is good at that'. In accordance with this insight we argue that the ISM should explicitly represent internal as well as external representations as a basis for the information seeking process. These and the changes suggested above are summarized in our revised version of the ISM contained in Figure 2.

7 Threats to Validity

There are a number of threats to validity in this study, the primary one being the limited number of maintenance sessions used to obtain the results. However, given that these 2 sessions resulted in 43 pages of transcribed talk-aloud data, it would have been infeasible to perform a larger study and still undertake the detailed analysis of data appropriate at this stage of the models' evolution. It is envisaged that, as the model

grows in maturity, this pilot-type study will give rise to more controlled, larger studies with a tighter coding technique.

The second threat to validity was due to the utilization of the talk-aloud technique. Even though talk-aloud data provides the 'richest source of a subject's mental state' [27] and in this case was gathered in line with best practice for capturing valid data [12], we did not prompt the participants when they fell silent, as suggested by Ericsson and Simon. While this lessened the invasiveness of the protocol, it is possible that we missed out on certain types of episodes that could have impacted on the model. Also, despite our best efforts, the participants were continually made aware of their participation in the study by the presence of the voice recorder.

An alternative data-capture technique was to video-tape the programmers' sessions. However, for confidentiality reasons the company was unwilling to allow this. In addition, the presence of a video camera, while possibly enriching the data stream by capturing programmers' expressions, programmers' actions and the source code, would increase the programmer's awareness of their participation [40], thus again reducing the ecological validity of the study.

When participants know they are being studied, there behavior may become altered, an effect known as the Hawthorne effect [1]. Another potential threat to validity in this study was that programmers, aware of their participation, may attempt to please the experimenter during the study [28], an effect referred to as the 'placebo effect' [40]. Typically this behavior is instigated by some (possibly subliminal) gesture or utterance by the experimenter. However, in this study the possibility of this was reduced, as the experimenter left the room for the sessions.

Ideally, we should have studied complete maintenance tasks rather than studying a 2-hour timeslot. Given our current modus operandi, it is entirely possible that certain types of information seeking episodes, specific to starting the task or winding up the task, were missed in our data capture. However, getting access to real maintainers, maintaining real software systems over a long period of time is difficult and our current industrial partners favor shorter exposure time for their programmers.

A final threat to validity was the coding process used in this study. It relies on one person's analysis of the transcripts and thus, reliability cannot be assessed. This coder was also aware of the hypothesized model and this may have impacted on his interpretation of the data. Without safeguarding reliability in the coding, [34] warns: It is the nature of a hypothesis when once a man is conceived it, it assimilates everything to itself as proper nourishment and, from the first moment of your begetting it, it generally grows the stronger by everything you see, hear, read or understand'. In future studies, 2 researchers will code samples from the transcripts, based on a coding manual for the categories of the ISM. This dual coding will occur at the start and end of the analysis. The kappa [15] from these dual-coded samples will allow assessment of reliability and drift in the coding process.

8 Conclusion

This paper proposed a number of changes to the ISM of O'Brien and Buckley [25], based on complete analysis of talk-aloud data generated in a small quasi experiment.

The findings, by in large validate the original ISM. Specifically the findings suggest that there is little bloating in the model and that its iterative nature mirrors actual practice. However, the results do suggest that the model lacks completeness, arguing for the explicit inclusion of an 'Information Prompted Action' stage, interruptions and programmer knowledge. In addition, the data suggests that the information collection stage be modeled in line with Marchionini's ISM to more accurately portray the behavior of programmers involved in software maintenance.

Future work will be directed at collecting more rigorous empirical data to evaluate and refine the model. Initially this will involve the analysis of talk-aloud data that we have already gathered from other industrial collaborators.

Another issue for future work is that of task-granularity, an issue that has implicitly appeared several times in the data presented in this paper. As information was extracted by programmers, other information seeking tasks often emerged (see Section 6.1.4 for an example) resulting in the emergence of new sub-tasks, new super-tasks, or just related tasks. This mirrors Wilson's model of Information Seeking, where advances in knowledge as information-seeking processed, prompt new goals and new iterations through the information-seeking process. It is an area of future research to establish the relationship between these tasks and between these tasks and the ISM.

References

1. Adair G. "The Hawethorne Effect: A Reconsideration of the Methodological Artifact". Journal of Applied Psychology. Vol 69. No 2. 1984. pp 334-345.
2. Basili v.. "The Role of Experimentation in Software Engineering: Past, Present, and Future". Keynote address: International Conference on Software Engineering 1996
3. Bowden, C., Bowden V., "Survey of Information Sources Used by Psychiatrists" Bulletin of the Medical Library Association, Vol. 59, 1971, pp 603-608
4. Bradac MG, Perry DE, Votta LG. "Prototyping a Process Monitoring Experiment", IEEE Transactions on Software Engineering, 1994 Vol 20 no 12. pp 774-784.
5. Brooks R., "Towards a Theory of the Comprehension of Computer Programs". International Journal of Man-Machine Studies, Vol. 18, 1983, pp. 543-554
6. Burkhardt J.M., Detienne F., Wiedenbeck S., "Object Oriented Program Comprehension: Effects of Expertise, Task and Phase". E. S. E. Vol. 7. No. 2. pp 115-156.
7. Chen, C., "How Do Scientists Meet Their Information Needs", Special Libraries, No. 65, 1974, pp 272-28
8. Curtis, Bill, Herb Krasner, and Neil Iscoe. "A field study of the software design process for large systems." Communications of the ACM, Vol. 31, no 11. 1268-1287, November 1988.
9. Ellis, D., Haugan, M., "Modeling the Information Seeking Patterns of Engineers & Research Scientists in an Industrial Environment", J. of Documentation, Vol. 53, No. 4, pp 384-403
10. Ellis, D., "A Behavioural Approach to Information Retrieval Design", J. of Documentation, Vol. 46, No. 3, 1989, pp 318-338
11. Ellis, D., Cox, D., Hall, K., "A Comparison of the Information Seeking Patterns of Researchers in the Physical & Social Sciences", J. of Documentation, Vol. 49, No. 4, pp 356-369
12. Ericsson K.A. and Simon H.A.. "Protocol Analysis, Verbal Reports as Data". MIT Press.
13. Good J., "Programming Paradigms, Information Types and Graphical Representations: Empirical Investigations of Novice Program Comprehension". Ph.D. Thesis, 1999
14. Harrison W.. "N=1 Editorial". E. S. E. Vol 2. no. 1. 1997 pp 7-10.

15. Hartmann D.P. (1977). "Considerations in the choice of interobserver reliability estimates." Journal of Applied Behaviour Analysis. Vol: 10. pp 103-116.
16. King, D., McDonald, D., Roderer, N., "The Journal System of Scientific and Technical Communication in the United States", King Research, 1978
17. Kingrey, KP, "Concepts of Information Seeking and Their Presence in the Practical Library Literature", Library Philosophy & Practice, Vol. 4, No. 2, 2002
18. Kuhlthau, C., "Developing a Model of the Library Search Process: Investigation of Cognitive and Affective Aspects", Reference Quarterly, Vol. 28, No. 2, 1988, pp 232-242
19. Kuhlthau, C., "Seeking Meaning: A Process Approach to Library and Information Services", New York: Greenwood Publishing, 1993
20. Large, A., Tedd, L., Hartley, R., "Information Seeking in the On-Line Age: Principles & Practice", Bowker-Saur, UK, 1999, ISBN 1-85739-260-4
21. Line, M., "Investigation into Information Requirements of the Social Sciences", Research Report No. 1, Bath University of Technology, 1971
22. Liu WQ, Chen CL, Lakshminarayanan V, Perry DE "A Design for Evidence-based Software Architecture Research", http://www.cs.toronto.edu/
23. Marchionini, G., "Information Seeking in Electronic Environments", Cambridge, England: Cambridge University Press, 1995
24. O'Brien, M. P., Buckley, J., "Inference-based and Expectation-based Processing in Program Comprehension", Proceedings of the 9th IWPC, Toronto, Canada, 2001
25. O'Brien, M. P., Buckley, J., "Modeling the Information-Seeking Behavior of Programmers – An Empirical Approach", Proceedings of the 13th IWPC St. Louis, Missouri. 2005
26. O'Brien M.P., Buckley J. Shaft, T., "Expectation-based, Inference-based, and Bottom-up Software Comprehension", IJSME: Research & Practice, Vol. 16, 2005, pp 427-447
27. Russo, J., Johnson, E., Stephens, D., (1989), "The Validity of Verbal Protocols", Memory & Cognition, Vol. 17.
28. Rosenthal R.. "Experimenter Effects in Behavioral Research". New York, NY, Appelton Century Crofts 1996
29. Seaman, C., "The Information Gathering Strategies of Software Maintainers", ICSM, 2002
30. Singer, J., Lethbridge, T., Vinson, N., Anquetil, N., "An Examination of Software Engineering Work Practices", Proceedings of the 1997 Conference of the Centre for Advanced Studies on Collaborative research, Toronto, Canada, 1996
31. Singer, J., "Work Practices of Software Maintenance Engineers", Proceedings of the ICSM, Washington, Federal District of Columbia, USA, 1998, pp 139-145
32. Sim, S. E., "Supporting Multiple Program Comprehension Strategies During Software Maintenance", Masters Thesis, Department of Computer Science, University of Toronto, 1998
33. Sousa, Maria João Castro, and Helena Mendes Moreira. "A Survey on the Software Maintenance Process." Proceedings of the International Conference on Software Maintenance, Bethesda, MD, November 1998, pp. 265-274.
34. Sterne L., "Tristam Shandy". 1761.
35. Strauss, A, Corbin, J. Basics of Qualitative Research: Grounded Theory Procedures and Techniques. Beverly Hills, CA: Sage Publications, 1990
36. Von Mayrhauser A., Vans A.M., "Program Understanding: Models and Experiments". Advances in Computers. 1995; Vol. 40, No. 4, 1995 pp: 25-46
37. Von Mayrhauser A., Vans A.M., Howe A.E., "Program Understanding Behavior during Enhancement of Large Scale Software". IJSME: Research and Practice. Vol. 9, pp: 299-327
38. Wilson, T. D., "Information Behaviour: An Interdisciplinary Perspective, A Report to the British Library Research & Innovation Centre on a Review of the Literature", 1996
39. Wilson, T. D., "Models of Information Behavior Research", Journal of Documentation, Vol. 55, No. 3, (1999), pp 249-270
40. Xu S. Rajlich V., Dialog-Based Protocol: An Empirical Research Method for Cognitive Activities in Software Engineering