*New Review of Information Behaviour Research: Studies of Information Seeking in Context*, vol. 1 (2000), pp. 135-149.
Preprint version

# People as Carriers of Experience and Sources of Commitment: Information Seeking in a Software Design Project

Morten Hertzum

Centre for Human-Machine Interaction
Risø National Laboratory, Denmark
morten.hertzum@risoe.dk

## ABSTRACT

Engineers in co-operative work settings must engage in a variety of information-seeking activities to accomplish their tasks, and they often turn to other people for information. This study investigates the role of people as information sources during software design, the early stages of software engineering. The purpose is to analyse how information seeking is woven into co-operative work and to inform the design of systems for assisting engineers with finding informed colleagues as well as informing documents. Based on a field study of a software design project, it is found that the involved software engineers are often looking for (1) practical experience as opposed to hard facts and (2) commitment as opposed to information. Few organisational mechanisms are in place to manage the flow of experience among projects. This leaves people as the primary carriers of experience, and they are the sole sources of commitment. Further, the engineers need a synthesis of the new system and use situation and are only analysing the present as a way of getting at the future. The ability to transcend current practice and envision the future is specific to people – though they find it difficult.

## INTRODUCTION

The information-seeking behaviour of engineers has been studied extensively for more than three decades (see, e.g., King, *et al.*, 1994). Though many issues have yet to be resolved, it is generally agreed that engineers primarily rely on oral communication with information sources internal to their company, e.g. face-to-face communication with work group colleagues. A good few prototype systems attempt to support people in finding other people with specific competencies. These include Referral Web (Kautz, *et al.*, 1997) which helps find research experts based on co-author relationships, Yenta (Foner, 1999) which matches people based on textual analysis of personal profiles, and Answer Garden (Ackerman, 1998) which routes users to recorded information, if available, and otherwise to knowledgeable people. However, we still lack a solid understanding of what it is that makes people such good information sources (Hertzum & Pejtersen, 2000; McDonald & Ackerman, 1998).

This study investigates the role of people as information sources within a group of software engineers tasked with designing a completely new version of a system for administrating child support and alimony payments. Thus, this study looks at information seeking in a software design context. Software design designates the early, formative stages of software engineering and is crucially dependent on the involved engineers' abilities to acquire, assess, and integrate information from a web of sources. At a high level of abstraction, the information-seeking activities of software engineers are concerned with three domains of discourse: the users' present work, the system/design context, and the new use situation. Within this framework, it is analysed how the engineers in the studied software design project acquire the information they need and – through that analysis – how their frequent preference for people as information sources is intended to utilise qualities specific to people. In this sense, the purpose of this study is to discover and describe people's affordances with respect to being used as information sources. A more in-depth understanding of what it is that makes people good information sources will both tell us something about the basics of co-operative work and give us a better basis for devising systems that support information seeking in various work contexts.

## DOMAINS OF DISCOURSE IN SOFTWARE DESIGN

At a high level of abstraction, the information needs of software designers are captured in the task-artefact cycle (Figure 1), in which designers respond to user requirements by building artefacts, which in turn present or deny possibilities to the users (Carroll, *et al.*, 1991). It is crucial to understand the cyclic and nontrivial nature of this process. The users' understanding of their tasks is determined by the tools they currently use and, at the same time, their understanding of their tools is determined by the tasks they are using the tools for. Likewise, software designers' understanding of the technological options is determined by their knowledge of tasks that need to be performed and, at the same time, their understanding of the users' tasks is determined by the possibilities and restrictions of the tools they currently know of. Thus, people's familiarity with certain tools and certain tasks shape their understanding of what their tasks are and what technology has to offer, and this understanding, in turn, constitutes a perspective that points to certain technological options and makes people blind toward others (Naur, 1965). This makes it inherently difficult for people to transcend their current way of perceiving things and envision how tasks, users, and technology should interact in constituting the future use situation.

Spelling out the information needs inherent in the task-artefact cycle in more detail, Kensing & Munk-Madsen (1993) describe three domains of discourse in the software design process: (1) *Users' present work*. The designers need concrete experiences with the users' present work, and designers and users need relevant structures on the work, which can provide a common language for communication about desired changes and consequences of proposed designs. (2) *Technological options*. The designers must have and maintain an overview of the possibilities and constraints of various technologies, and to enable the users to play an active role the designers must provide them with concrete experiences with diverse technological options. (3) *New system*. Both designers and users need abstract descriptions of design proposals to assign priorities and make decisions, but to better understand and more

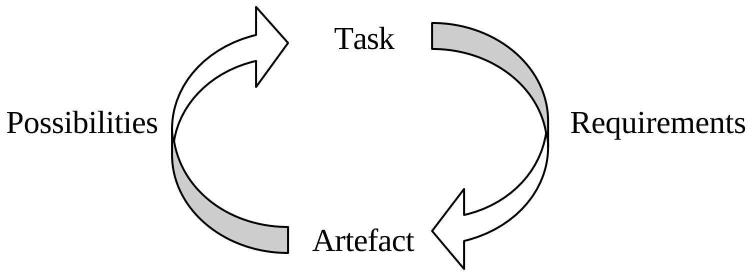

Figure 1. The task-artefact cycle

thoroughly assess proposed designs users as well as designers also need concrete experiences with more or less sophisticated prototypes of the new system.

When a new system is developed from scratch, the designers have to spend some time familiarising themselves with the users' work. When designers are tasked with developing a new version of an existing system, the situation is often markedly different. If the designers have themselves been involved in developing the present system, they already have at least an elementary understanding of the users' work. With respect to the technological options, designers are usually well informed but an important exception arises when company policy or project needs cause a change of technological platform, for example a move to web-based applications. In this situation, designers lack knowledge of common conventions and they lack the practical experience necessary to tell what is well supported by the new platform from what is difficult to achieve. The new system is an independent domain of discourse because it involves a fundamental breaking away from the present understanding of how the users' task and the technological options define the use situation. The new system is unknown at the outset but will gradually come into being as the design process progresses.

In a sense, the users' present work and the technological options are only of interest because system designers have no direct way of getting information about the new system and use situation. This is interesting from an information-seeking point of view because it points out the massive indirectness of the information-seeking process in software design. Software designers seek information about the users' present work, as opposed to their future work, and the technological options, as opposed to the future system, for one reason only: because they cannot get the information they really need. This is an instance of what Star & Ruhleder (1994), following Bateson, call a double bind. Double binds are defined in relation to Bateson's division of messages onto three communicative levels, which extend messages about physical reality with increasingly more context information. A double bind occurs when a message is given at more than one level simultaneously, or an answer is simultaneously demanded at a higher level and negated on a lower level. For example:

> *The parent may say, for instance, "go get your coat, it's cold outside" while closing the door to the cloakroom and standing in front of it. Attempts to point to the contradiction are met with denial, "of course I want you to get your coat; didn't I say so?"* (Star & Ruhleder, 1994)

When software designers are asked to design a new system they are, at the same time, prevented from getting crucial information about what properties this new system should have. What stands in front of the cloakroom is not a parent but how people's familiarity with their present tasks and tools blocks their ability to envision radically new solutions. To get around this double bind, software designers' information-seeking activities assume the characteristics of a process of analysis (of the present) and synthesis (of the future).

This study makes use of an adapted version of the three domains of discourse (see Figure 2). The adaptations have been made to fit the domains to the distinguishing characteristics of the software design project studied in this paper. First, the studied project concerns a system where the users' work practices are strongly regulated by environmental factors, e.g. legislation, and this influence has been made explicit. Second, the technological options are considered part of a domain regarding the system/design context. This domain also includes the present system, which is to be replaced by a new version, and the options and constraints relating to the design project. Third, the domain involving the new system has been split into a part regarding the new system and a part regarding new work practices. This is done to emphasise that what is being designed is just as much new work practices as it is a new computer system.

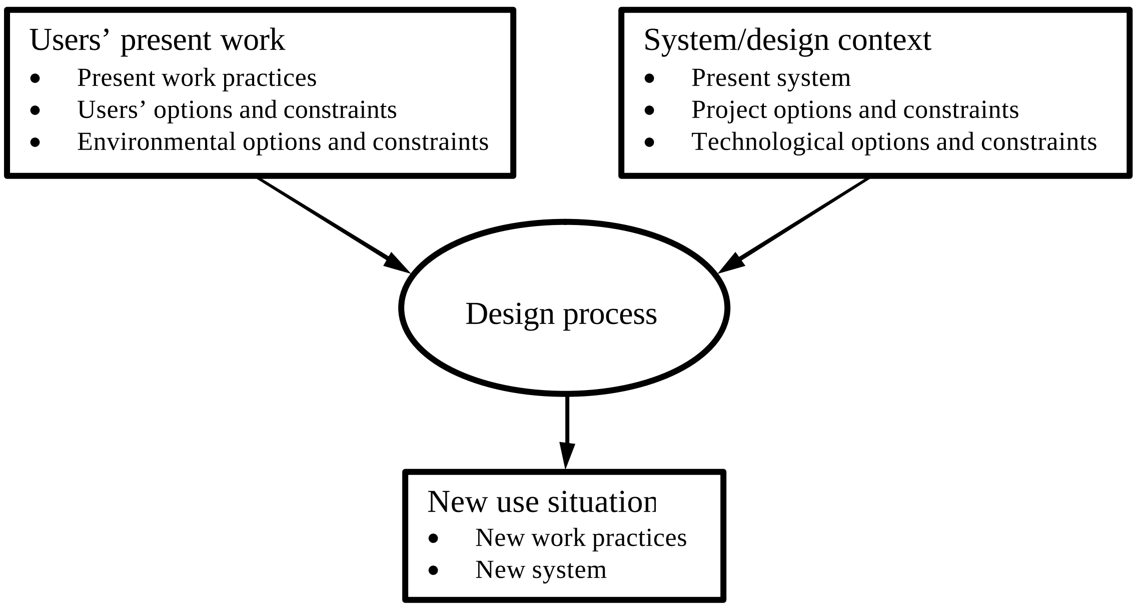
Users' present work
Present work practices
Users' options and constraints
Environmental options and constraints
System/design context
Present system
Project options and constraints
Technological options and constraints
Design process
New use situation
New work practices
New system

Figure 2. Domains of discourse in software design

## THE CSA PROJECT

The company where the field study took place is a large software house, which has developed and marketed a range of systems for use in local government institutions. The studied project concerns a system to support local government authorities in the handling of cases concerning child support and alimony (CSA). The CSA project was initiated in 1999 and will, according to the project plan, last two years. The first eight months of the project, the period analysed in this study, concerned the requirements specification and the business modelling. During this period, the project was staffed with a project manager, eleven designers/developers, two service consultants, a methods & tools consultant, a usability specialist, and a secretary. The project manager and six of the designers/developers worked full time on the CSA project, the remaining ten persons were assigned to the CSA project on a part-time basis. In the following, the members of the CSA project will be termed CSA engineers, irrespective of their different educational backgrounds.

The CSA engineers are to completely redevelop the existing CSA system, which several of them have been heavily involved in developing and maintaining. Whereas the existing CSA system contains substantial amounts of code that duplicate functionality from other systems made by the company, the new CSA system will distribute this functionality onto components that are to be developed outside the CSA project. This philosophy of component-based design means that the CSA engineers have to co-operate closely with a number of people inside the company to negotiate, settle, and follow up on component definitions and how the development of the components progresses. Naturally, the CSA engineers also have to interact with management, sales & marketing, technical services, the quality function, etc. Moreover, they need to communicate with external stakeholders such as user representatives and the governmental bodies responsible for the legislation regarding child support and alimony.

## METHOD

The data collected for this study cover the formative eight-month period from the initiation of the CSA project, through the requirements specification, to the completion of the business modelling. I have followed the project by (1) participating in the two-day start-up seminar, (2) being present at the fortnightly status meetings and some additional meetings, (3) conducting interviews with eleven of the core project participants, and (4) collecting various project documents. This study is based on an analysis of the 16 meetings that have been observed, whereas the other empirical data provide crucial background information.

The main purpose of the meetings was to provide a forum for sharing information about the status of the project, maintaining awareness of the entire project, co-ordinating activities, discussing problems and progress, making decisions, and reviewing major project documents. During the meetings, I was seated at the meeting table with the other people present. From their point of view, I have been invisible in that I was not to be spoken to and have myself remained silent. During the breaks, I have talked informally with people. All the meetings have been recorded on tape and transcribed.

The data analysis involved two passes. First, nine transcripts were examined sentence by sentence and all references to information sources were marked up and annotated. This bottom-up analysis, combined with findings from the literature (see Allen, 1977; King, *et al.*, 1994; and above), provided the input for creating a coding scheme. Second, all 16 transcripts were examined to identify the incidents involving information sources and categorise them according to the coding scheme. The incidents comprise the situations where the project members make each other aware of available sources, discuss the quality of sources, decide on the sources to approach in a variety of situations, follow up on whether appropriate sources have been involved in settling complicated issues, etc. In the following, these incidents will be referred to as the information-seeking incidents. To give a feel for the data, four sample incidents are reproduced below:

1. *We have to be good at asking other people than [two of the user representatives]. Don't get me wrong: They have that area only and they are very competent and careful. But they are not always representative of an average user in an average municipality.*
2. *I have just received an internal report on workflow management. It's the review version, and I have only had time to leaf through it. I don't know whether it has reached any of you yet? I'll have a look at it and either distribute it or make a resume.*
3. *The person we have to go through when we need something done in relation to [one of the software components] is [a person from another project]. He is controlling the available resources, and he knows their calendars and who can be assigned to the task.*
4. *At the meeting yesterday with the user representatives we had a workshop where they had to be visionary. They should think of a normal use situation and ask themselves: What happens frequently and has to run smoothly? Afterwards they were to express their wild wishes regarding how such situations should be handled in a brave new world.*

The incidents were coded with respect to the three domains of discourse depicted in Figure 2. For each incident it was determined whether the source was considered/consulted in its capacity of being able to provide information about one or more of the three domains, which were (the numbers in parentheses give the coding of the four sample incidents): users' present work (1, 4), system/design context (2, 3), and new use situation (4). In addition, the incidents were coded with respect to four binary distinctions:

- *People* or *documents*. People sources include individuals, project groups, and organisations (for example, 1, 3, 4). Document sources include electronic and paper documents as well as information systems (for example, 2).

- *Internal* or *external*. Internal sources are sources internal to the company – but external to the project group (for example, 2, 3). External sources are sources external to the company (for example, 1, 4).
- *Information* or *commitment*. In the case of information, the source delivers insights, opinions, pieces of information, etc. (for example, 1, 2, 4). In the case of commitment, the source takes on a task, i.e. agrees to perform a specified piece of work (for example, 3).
- *Pull* or *push*. Pull refers to information and commitments actively acquired by the CSA engineers (for example, 1, 3, 4). Push refers to the situations where people external to the project volunteer information or require that the CSA engineers take on a commitment (for example, 2). This distinction is not always applicable as some of the information sources are primarily involved to output information from the project.

## DISCUSSION

The 16 meetings include 362 incidents where the CSA engineers' attention was devoted to information sources. On average, the meetings contained one such information-seeking incident every 5.7 minutes. Figure 3 shows that 268 (74%) of the incidents concern information sources knowledgeable about only one of the domains of discourse, whereas 94 (26%) concern sources capable of providing information about two or all three domains. The CSA engineers have needed information on all three domains of discourse but more than half of the information sources were considered/consulted in their capacity of being able to provide information about the system/design context. A total of 263 (73%) incidents have to do with the system/design context, 133 (37%) with the users' present work, and 80 (22%) with the new use situation.

The four binary distinctions are all strongly dominated by one of the two possible values (Table 1). Whereas the people-documents distinction and the push-pull distinction appear to be rather independent of the domains of discourse, the two other binary distinctions vary with the domain. Commitment, as opposed to information, is more frequent in the system/design context than in the two other domains. The balance between internal and external sources is roughly equal for the users' present work, strongly skewed toward internal sources for the system/design context, and somewhere in between for the new use situation.

### Putting experience to work

Figure 3 indicates that the CSA engineers devote much attention to augmenting their knowledge of the system/design context and feel comparatively less in need of acquiring information about the users' work. Several of the designers/developers among the CSA engineers know about the users' work from their longstanding involvement in the development and maintenance of the existing CSA system, and the two service consultants in the group have spent years helping users out when they had trouble using the system. Conversely, the system/design context involves the use of many new tools and techniques, such as the introduction of component-based design, a change of development tool, and the transition to a Web-based user interface.

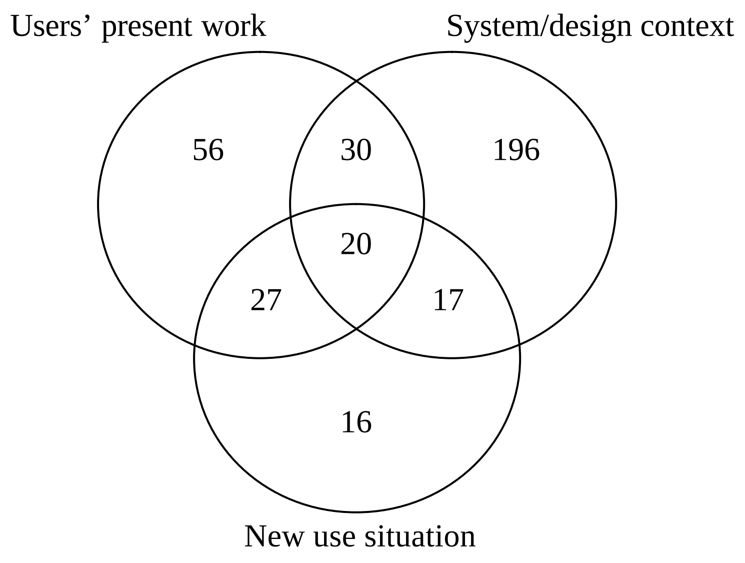

Figure 3. Distribution of the incidents on the three domains of discourse

Both document sources and people sources are available in all three domains of discourse but 77% of the information-seeking incidents involve people sources (Table 1). A major reason for this is that the CSA engineers tend to prefer information sources that are informed by formal expertise or, preferably, practical experience to relying on their own inexpert interpretation of a document (see Hertzum, 2000). Another, closely related reason is that document sources leave something out. Whereas the handling of CSA cases is prescribed in detail in written legislation, there is a large gap between the terse texts and the richness of real-world cases. To close this gap, the users of the CSA system have to interpret the legislation with respect to the concrete cases they are confronted with. Over time, this leads to a practice that is based on the legislation but not inherent in it (see Leith, 1986, for a substantiation of this argument). Consequently, the CSA engineers have to talk to users to get information about the users' present work, they cannot simply read the legislation. In a

| | Users' present work | System/design context | New use situation | Total |
|---|---|---|---|---|
| 1. | | | | |
| Information | 79.0 (87%) | 147.5 (65%) | 37.5 (84%) | 264 (73%) |
| Commitment | 12.2 (13%) | 78.7 (35%) | 7.2 (16%) | 98 (27%) |
| 2. | | | | |
| Internal | 38.2 (42%) | 197.2 (87%) | 29.7 (66%) | 265 (73%) |
| External | 53.0 (58%) | 29.0 (13%) | 15.0 (34%) | 97 (27%) |
| 3. | | | | |
| People | 67.2 (74%) | 180.7 (80%) | 31.2 (70%) | 279 (77%) |
| Documents | 24.0 (26%) | 45.5 (20%) | 13.5 (30%) | 83 (23%) |
| 4. | | | | |
| Pull | 77.7 (85%) | 187.2 (83%) | 30.2 (68%) | 295 (81%) |
| Push | 9.0 (10%) | 27.0 (12%) | 4.0 (9%) | 40 (11%) |
| Other | 4.5 (5%) | 12.0 (5%) | 10.5 (24%) | 27 (7%) |
| Total | 91.2 | 226.2 | 44.7 | 362 |

Table 1. The distribution of the four binary distinctions across the three domains of discourse. Since the domains overlap, incidents relating to two domains have been counted as contributing one half to each domain and incidents relating to all three domains as contributing one third to each domain. All percentages give the distribution of the two values of the binary distinction within the domain.

similar way, the new tools to be used by the CSA engineers are comprehensively documented but while this documentation can be scrutinised it is entirely up to the reader to ask the questions that are pertinent in relation to the development of the CSA system. Passively receiving a lot of information by reading documents has repeatedly been found to result in a reduced depth of understanding compared to active assimilation of the information through experiencing things for oneself (Hertzum, 1999). Consequently, asking colleagues who have experience with the tools from projects similar to the CSA project is likely to produce more reliable information – and be less work – than reading the lengthy documentation.

The percentage of external sources is 58% for the users' present work and 13% for the system/design context (Table 1). This is, of course, an indication that the users are the primary source of information about their work, whereas the primary sources of information about technological options and project constraints are the CSA engineers' colleagues in the organisation. Part of the explanation for the magnitude of the difference is, however, a difference in the CSA engineers' attitude toward involving internal and external sources. Internal sources are involved whenever there is a potential need. The project manager is very conscious that the CSA project utilises the resources made available to it by the organisation, and that appropriately experienced colleagues are drawn on often and early to ensure efficient execution of project activities. Internal experts and consultants are involved to an extent where some of the CSA engineers at times feel overwhelmed by the amount of decisions that are only made after consulting company experts:

> *Have you drawn on [an internal consultant]?*
>
> *Yes, but it is difficult when you're sitting for five minutes in doubt with yourself about how to do something. It's more about trying something out for yourself [than about consulting an expert].*

External sources are involved more sparingly. Some of the CSA engineers are, at times, concerned that they spend too little time probing the users about their needs and wishes. It seems that external sources are not involved until there is a recognised need, as opposed to a potential need or some uneasiness with regard to basing decisions on the currently available information:

> *I would much prefer that we delimit it so that we do not start by saying to the users: "Let's start all over again from Adam and Eve." We have been out talking with them. […] We should not start by going out, we should start by collecting the information we have, and then record the questions that pop up along the way and let them form the basis for going out to do some interviews. […] In order to focus the process a little.*

The reason for this is partly that internal sources tend to be conceived as consultants, who can be asked at no risk, whereas external sources are stakeholders, who may have their own interests. Asking the users what they want involves a risk that it will subsequently be necessary to argue with them about why some of these wishes will not be included in the system. This is a conflict between a contractual relationship (the users as customers) and an information-seeking relationship (the users as sources of information about their present work and wishes for the future system). Everybody in the CSA project are well aware of this conflict but it remains a cause of profound disagreements, for example between project management and the CSA engineers responsible for ensuring the users' say in the process.

## Managing the flow of experience

The CSA engineers have to actively acquire most of the information they need in that little of it is pushed into the project by customers or sources set up by the company. As much as 81% of the information-seeking incidents are initiated by the CSA engineers and only 11% of the incidents by people external to the project (Table 1). A record is kept of the questions and problems customers report when they call the service department, and this record was fed into the CSA project when they started working on the requirements specification. The CSA project has received some additional information from task forces formally set up to clarify certain issues and broadcast the resulting company decision to all projects. Furthermore, some information has been volunteered informally by colleagues. For the vast majority of their information needs, it has however been up to the CSA engineers to recognise their information needs as well as to find the needed information.

The number of information-seeking incidents involving internal sources indicates that lots of valuable experience is available within the company, but no system or set of procedures is in place to ensure that the lessons learned in one project are propagated to future projects. In the absence of a corporate approach to the management and utilisation of these experiences, individual employees become the vital carriers of experience. This is a key factor in explaining why 77% of the information-seeking incidents involve people sources. The CSA engineers' general knowledge about recent and ongoing projects tells them that colleagues with a range of concrete experiences are around. At the same time, the CSA engineers know that their colleagues' memories are likely to be the only record of these experiences because few experiences are recorded in shared databases or broadcasted to development projects. This means that the primary means of managing the flow of experience in the company is to carefully staff projects with people who carry the experiences from pertinent previous projects.

The situations where information is pushed to the development projects include some where a pilot project is used to gain experience with a new tool or technique before it is decided whether to adopt it on a company basis. It is however difficult to learn from trying out a new tool or technique in one project since first-time use is usually troublesome. Furthermore, the bad experiences acquired during the first project stick to the tool or technique for some time even if it is decided to adopt it. When the next engineers start to use the tool, they will also experience the troubles involved in learning something new. Turning to their colleagues for help, they are however more likely to find a colleague with frustrations similar to their own than one who will defend the tool and be able to explain how to get around the problem. This creates a prolonged transition period where the engineers who have (slightly) more experience with the new tool confirm the frustrations of the other engineers, who may start to doubt that it was a wise decision to adopt the tool. In the midst of the transition process, the engineers are caught in a double bind when they consult their more experienced colleagues for help. On the one hand, the tool has been adopted based on these colleagues' recommendations and, on the other hand, the same colleagues express reservations about it, which resonance with the engineers' own frustrations. It should be emphasised that as the engineers gain experience with the tool, they also develop a more balanced view of its qualities.

The CSA project provides an illustrative example involving two of the designers/developers (D1 and D2) and the methods & tools consultant (C). The excerpt is from a discussion of the difficulties the CSA engineers experience with their new development tool:

*D1: What about the people in [another project], which has been using it [the new development tool]? What do they think about it?*

C: *They hated and detested it when they were working with it. That's pretty normal in a pilot project.*

D1: *That's why the company decided to adopt…*

C: *You get that reaction from most people who try something new and experience a lot of problems. Those, who have started later, have not run into the same problems, and they have been able to see the positive sides of the tool.*

D2: *That beats me.*

C: *– Even the advantages.*

## In search of commitment

The success of the CSA project is dependent on a number of activities performed outside the project. For that reason, the CSA engineers are often looking just as much for commitment to future actions as they are looking for information. A total of 27% of the information-seeking incidents concern commitment, and most of them fall within the system/design context (Table 1). Many of the commitments have to do with clarifying the functionality and interface of the software components that will be developed outside the CSA project. This is a delicate and time-consuming process since the people who are being tasked with developing the components are fully occupied with other tasks and thus have to reschedule part of their activities. Consequently, negotiations are needed to convince other project managers that the needs of the CSA project are the more urgent, that satisfying these needs is not prohibitively time consuming, and that good people should be selected for the task. Such negotiations are made more frequently in the system/design context where the other party tends to be internal, and the balancing of the conflicting needs of several projects against each other is a central management issue.

To the CSA engineers, information seeking is normally a matter of finding the right person to contact and the distinction between information and commitment is easily blurred. A request for information may turn into a commitment if the source is unable to provide the information directly but offers to find out. Likewise, requesting that someone take on the task of investigating a certain issue may turn into a request for information if the source, unexpectedly, can provide the requested information directly. The CSA engineers are very aware of the situations where they ask for their colleagues' commitment to time-consuming activities because that usually involves some negotiation. But these situations apart, the CSA engineers' overriding concern is that they need input from other people to accomplish their task, and it is clearly a secondary issue whether this input ends up being information or commitment. The capacity to make commitments sets people apart from documents and makes people more versatile as information sources. The definite demand for this capacity signifies that it is important to the CSA engineers.

In a co-operative context such as software design, information seeking is an integral element of people's everyday activities, not a separate activity. A central reason for this is that the information needed by the CSA engineers does not exist in advance; it has to be created. The CSA engineers need information about the new use situation, but can only get at it indirectly by analysing the users' present work and the technological options. As a result, their information-seeking activities are divided up into numerous, often brief incidents, which are subsumed in other activities. Attempts to support engineers' information-seeking activities, for example with information technology, must therefore approach information seeking as an activity that is born out of other more continuous activities and feeds information back into these activities.

## CONCLUSION

This study has investigated the role of people as information sources during software design, the early stages of software engineering. The purpose has been to analyse how information seeking is woven into co-operative work and to inform the design of systems for assisting engineers with finding informed colleagues as well as informing documents. Based on a field study of a software design project, it is found that:

- Engineers are involved in a construction process and for that reason they need a *synthesis of the new use situation* and are only *analysing the present* as a way of getting at the future. The ability to transcend current practice and envision the future is specific to people – though they find it difficult.
- Engineers are involved in an applied process and for that reason they are often looking just as much for *experiences* with certain tools or work tasks as they are looking for *facts*. Such experiences are seldom available in writing.
- Engineers are involved in a co-operative process and for that reason they are often looking just as much for *commitment* to future actions as they are looking for *information*. The capacity to make commitments is specific to people.

In the studied organisation, few organisational mechanisms are in place to manage the flow of experience among projects, and this leaves people as the primary carriers of experience. Even when the engineers are in possession of the authoritative documents, for example the legislation the system is to administer, they often feel in need of checking their interpretation of a document with that of a more qualified person. The rationale for this is that practical experience is normally needed to read a document competently. Commitment, which is mostly sought from people internal to the company, underlines that information seeking cannot be understood in isolation from its context of co-operative work.

## ACKNOWLEDGEMENTS

This work has been supported by a grant from the Danish National Research Foundation. I wish to thank the members of the CSA project group who have put up with my presence in spite of their busy schedule.

## REFERENCES

Ackerman, M.S. (1998). Augmenting organizational memory: A field study of Answer Garden. *ACM Transactions on Information Systems*, 16(3), 203-224.

Allen, T.J. (1977). *Managing the flow of technology: Technology transfer and the dissemination of technological information within the R&D organization*. Cambridge, MA: MIT Press.

Carroll, J.M., Kellogg, W.A., & Rosson, M.B. (1991). The task-artifact cycle. In J.M. Carroll (Ed.), *Designing Interaction: Psychology at the Human-Computer Interface* (pp. 74-102). Cambridge: Cambridge University Press.

Foner, L.N. (1999). *Political artifacts and personal privacy: The Yenta multi-agent distributed matchmaking system*. Cambridge, MA: MIT [PhD Thesis]. Available from: http://foner.www.media.mit.edu/people/foner/PhD-Thesis/Dissertation (consulted May 10, 1999).

Hertzum, M. (1999). Six roles of documents in professionals' work. In S. Bødker, M. Kyng, & K. Schmidt (Eds.), *ECSCW'99: Proceedings of the Sixth European conference on Computer Supported Cooperative Work* (pp. 41-60). Dordrecht: Kluwer.

Hertzum, M. (2000). The fundamental importance of trust in engineers' assessment and choice of information sources. Submitted for publication.

Hertzum, M., & Pejtersen, A.M. (2000). The information-seeking practices of engineers: Searching for documents as well as for people. *Information Processing & Management*, 36(5), 761-778.

Kautz, H., Selman, B., & Shah, M. (1997). Referral Web: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3), 63-65.

Kensing, F., & Munk-Madsen, A. (1993). PD: Structure in the toolbox. *Communications of the ACM*, 36(6), 78-85.

King, D.W., Casto, J., & Jones, H. (1994). *Communication by engineers: A literature review of engineers' information needs, seeking processes, and use*. Washington, DC: Council on Library Resources.

Leith, P. (1986). Fundamental errors in legal logic programming. *The Computer Journal*, 29(6), 545-552.

McDonald, D.W., & Ackerman, M.S. (1998). Just talk to me: A field study of expertise location. In *Proceedings of the ACM CSCW'98 Conference on Computer Supported Cooperative Work* (pp. 315-324). New York: ACM Press.

Naur, P. (1965). The place of programming in a world of problems, tools, and people. In *Proceedings of the IFIP Congress 65* (pp. 195-199). [Reprinted in P. Naur (1992), *Computing: A human activity* (pp. 1-9). New York: ACM Press].

Star, S.L., & Ruhleder, K. (1994). Steps towards an ecology of infrastructure: Complex problems in design and access for large-scale collaborative systems. In R. Futura & C. Neuwirth (Eds.), *Proceedings of the ACM CSCW'94 Conference on Computer Supported Cooperative Work* (pp. 253-264). New York: ACM Press.