Computer Supported Cooperative Work, vol. 13, no. 1 (2004), pp. 35-61
Preprint

Small-Scale Classification Schemes: A Field Study of Requirements Engineering

Morten Hertzum

Centre for Human-Machine Interaction, Risø National Laboratory, Denmark

Abstract. Small-scale classification schemes are used extensively in the coordination of cooperative work. This study investigates the creation and use of a classification scheme for handling the system requirements during the redevelopment of a nation-wide information system. This requirements classification inherited a lot of its structure from the existing system and rendered requirements that transcended the framework laid out by the existing system almost invisible. As a result, the requirements classification became a defining element of the requirements-engineering process, though its main effects remained largely implicit. The requirements classification contributed to constraining the requirements-engineering process by supporting the software engineers in maintaining some level of control over the process. This way, the requirements classification provided the software engineers with an important means of discretely balancing the contractual aspect of requirements engineering against facilitating the users in an open-ended search for their system requirements. The requirements classification is analysed in terms of the complementary concepts of boundary objects and coordination mechanisms. While coordination mechanisms focus on how classification schemes enable cooperation among people pursuing a common goal, boundary objects embrace the implicit consequences of classification schemes in situations involving conflicting goals. Moreover, the requirements specification focused on functional requirements and provided little information about why these requirements were considered relevant. This stands in contrast to the discussions at the project meetings where the software engineers made frequent use of both abstract goal descriptions and concrete examples to make sense of the requirements. This difference between the written requirements specification and the oral discussions at the meetings may help explain software engineers' general preference for people, rather than documents, as their information sources.

Keywords: Classification schemes, Conceptual design, Cooperative work, Coordination, Requirements engineering, Requirements specification, Small-scale classification.

1 Introduction

Classifications are so commonplace that we often fail to realise their numbers and the myriad ways in which they influence our lives (Bowker and Star, 1999). Classifications are employed to achieve structure, a key constituent in how people manage their world. "There is merit to the claim that much problem solving effort is directed at structuring problems, and only a fraction of it at solving problems once they are structured" (Simon, 1973). For most real-world problems there is, however, not one right structure but a multitude of possible structures, which appeal to different groups of actors because the different structures emphasise different aspects of the problem. Often, different ways of structuring a problem point toward different solutions. In these situations, classifications have politics. A number of studies have deconstructed influential large-scale classification schemes to document these politics; for example, the Dewey Decimal Classification (see Olson, 1998) and the International Classification of Diseases (see Bowker, 1998; Bowker and Star, 1991). Other studies provide more general discussions of the implicit agendas promoted within the explicit motivation of using classification schemes to achieve standardisation (e.g., Albrechtsen and Jacob, 1998; Bowker and Star, 1999).

This study looks at small-scale classification schemes. On the one hand, classification "in the small" is like classification in general, only it affects fewer people and spans shorter periods of time. On the other hand, small-scale classification schemes may be different because fewer resources will be available for the creation

and evolution of the classifications, the use situations will normally be more narrowly defined, and the distance between creation and use of the classification scheme will be smaller in terms of both people and time. Small-scale classification schemes are created and used by local actors to – at least locally and temporarily – establish, structure, and share the meaning of selected aspects of their world for purposes of debate, resolution, access, overview, standardisation, and so forth. Since it may prove prohibitively difficult or resource-demanding to conceive of a really good classification scheme, many activities are performed – and often successfully so – on the basis of partial and inconsistent classification schemes. Small-scale classification schemes are an important topic in CSCW (Computer Supported Cooperative Work) research because they are a widespread example of the artefacts people produce to articulate their work, organise their information, and coordinate their cooperative activities (Schmidt and Bannon, 1992; Simone and Sarini, 2001).

The classification scheme analysed in this study was developed during the conceptual-design stage of a software-engineering project and concerned the classification of the system requirements. In line with the work on Viewpoints (Finkelstein et al., 1992; Nuseibeh et al., 1994), the classification scheme was seen as a tool for integrating and managing multiple perspectives on the new system. The up-front purposes of the classification scheme were to support communication between developers and users during the requirements-engineering process and to provide the developers with a conveniently structured requirements-specification document for use during the continued design process. The requirements-engineering process was the paramount project activity during the first three months of the project but stayed a major concern for a considerably longer period. The project concerned the complete redevelopment of an information system which has been used by municipal authorities for almost two decades. While redevelopment is conceptually different from development from scratch it should be noted that most systems are introduced to support, extend, and/or replace tasks that are already carried out and, hence, most projects consist of redeveloping existing tools and ways of working (Carroll et al., 1991). This study is based on an ongoing field study but focuses on data obtained during the first twelve months of the project.

Studies of design have repeatedly found that fluctuating and conflicting requirements are among the major problems in the design of large software systems (Boehm, 1991; Curtis et al., 1988; Keil et al., 1998). Other major issues are the thin spread of domain knowledge – which emphasises the key importance of project staffing – and communication bottlenecks or breakdowns (Curtis et al., 1988; Walz et al., 1993). In their study of the conceptual-design activities in a software project, Potts and Catledge (1996) find: (1) a painfully slow convergence on a common system vision, (2) repeated returns to certain issues on which the engineers failed to reach closure, and (3) a failure to ground the system architecture as an abstraction in considerations of concrete details and user needs. In addition, the engineers carried most contributions to the architecture around in their heads. It was in struggling with issues similar to these, the software engineers studied in this paper created a scheme for classifying the system requirements. To understand the role of this classification scheme it is necessary to study the interactions between the design process and the specific design artefact that embodied the classification scheme, namely the requirements specification. More specifically the purpose of this study is threefold:

- To make it apparent that classification, in terms of the requirements classification and several other small-scale classifications, is a constituent element of requirements engineering.
- To show how the requirements classification served other purposes besides organising system requirements and, for example, implicitly helped the software engineers reconcile two important but conflicting aspects of their interaction with the users.
- To compare the written requirements specification and the oral discussions at the project meetings in terms of contents and how they support the software engineers in acquiring information.

Two influential CSCW approaches to analysing classification schemes are coordination mechanisms (Schmidt and Simone, 1996) and boundary objects (Bowker and Star, 1999), both presented in the next section. Then, Section 3 describes how the field study was conducted, and Section 4 introduces the studied project. In Section 5 it is analysed how the classification scheme was created, and in Section 6 how it was used. Section 7 provides a comparison of the classification scheme and the oral discussions at the project meetings.

2 Coordination mechanism, boundary object, or simply document structure?

Concretely, the classification scheme studied in this paper manifests itself as the table of contents of the requirements-specification document. Research on *document structure* has focused on identifying (1) the document structure that best suits electronic documents as opposed to paper documents (e.g., Dillon, 1994; Kircz, 1998) and (2) the document components readers use in structuring and mobilising information in

documents, especially scientific journal articles (e.g., Bishop, 1999; Dillon, 1991). While this research investigates how the structure of documents affects and affords reading, information seeking, and certain other uses of documents, it has not addressed how the structure of work-related documents enters into articulating and coordinating cooperative work. These issues have instead been addressed in the research that approaches classification schemes from the concepts of coordination mechanisms and boundary objects.

The concept of *coordination mechanisms* (Schmidt and Simone, 1996) is a generalised description of how cooperating actors use artefacts such as timetables, standard operating procedures, and classification schemes for coordination purposes. Coordination mechanisms are used in cooperative work settings where the complexity of the field of work makes it necessary to devote work specifically to restraining, scheduling, aligning, and meshing the various activities that enter into the overall effort. Coordination mechanisms are indispensable in most industrial work settings because many actors are involved, projects span extended periods of time, work is distributed, tasks are performed concurrently, and solutions to problems require input from multiple disciplinary specialities. Coordination mechanisms consist of an *artefact*, that is a permanent symbolic construct, and a *protocol*, which stipulates the intended ways of using the artefact. The protocol is not a causal scheme but rather a normative construct, which is based on a 'precomputation' of conditions and interdependencies. Further, the protocol may play varying roles ranging from weak roles in which the artefact is largely made available to the actors and used at their discretion to strong roles in which the protocol gives detailed prescriptions and the artefact is part of a highly regulated practice. The weak and strong roles of the protocol may be exemplified by a map and a script, respectively (Schmidt, 1999). Whereas artefacts can be introduced top-down, their use as coordination mechanisms can only be constituted locally through the involved actors' adoption of work practices that accord with the protocol. The protocol need not be formally defined in procedures; it can also exist as conventions that are defined implicitly by the local work practice.

Of the studies that led to the formulation of the concept of coordination mechanisms, Carstensen and Sørensen (1996) provide an empirical study of a product classification scheme. Generally, the protocols associated with classification schemes prescribe that the classifications are used whenever objects are classified but apart from that the role of classification schemes is rather to provide a map of what is available. The classificatory principles at the various levels of a classification scheme will to some extent prescribe the decisions involved in traversing the scheme and selecting an item but compared to other coordination mechanisms (see Schmidt and Simone, 1996, for examples) the protocol of classification schemes seems to play a weak role. The concept of coordination mechanisms provides a basis for understanding how classification schemes support people in working toward a common goal. In requirements engineering and other situations involving people with different, possibly conflicting interests the concept of coordination mechanisms should be complemented with work on the more indirect – or even disabling – roles and consequences of classification schemes. In their work on classification and boundary objects, Bowker and Star (1991, 1999) have focused on rendering these indirect roles visible to reveal the nature and consequences of classification.

The concept of *boundary objects* (Bowker and Star, 1999; Star and Griesemer, 1989) arises out of work on how artefacts such as maps, forms, and classification systems may mean different things to different people and, at the same time, be the mechanisms that enable these people to work together. That is, boundary objects are a way of capturing how collaboration is accomplished in a world where the baseline is one of diversity, rather than one of agreed-upon goals and unambiguous categories. Boundary objects are those objects that are both plastic enough to be adaptable to different viewpoints and robust enough to maintain identity across them (Star, 1989; Star and Griesemer, 1989). Boundary objects are, by definition, intended to mediate between different groups of actors, but the concept does not imply that these actors are necessarily collaborating in the sense of working toward a common goal. Further, boundary objects are not expressly linked to the presence of artefacts and may, thus, be merely mental constructs.

Classifications are a pervasive class of boundary objects and several longstanding, wide-scale classification schemes have been analysed from the point of view of boundary objects. Bowker and Star (1991, 1999) analyse the evolution and consequences of the International Classification of Diseases (ICD), which has been used internationally since the 1890s to classify causes of death and disease. To accommodate the concerns of different actors some level of ambiguity has been accepted regarding the classification of a number of diseases, but it has at the same time been a major concern to make the ICD as robust as possible. One way of making room for uncertainty while maintaining a fair level of robustness is to distribute the residual categories. 'Other' categories appear throughout the ICD except at the top level. Since uncertainty is inevitable it is crucial to allow for it and at the same time confine it to the more detailed levels of the classification. This combination of plasticity and robustness is exactly what makes the ICD a boundary object. However, Bowker and Star (1991) also show that the plasticity has its limits in that the ICD readily describes some kinds of diseases whereas other kinds (e.g., chronic as opposed to acute diseases) are rendered more or less invisible. This attention to possibilities that are made invisible by the way in which situations are described is highly important to

requirements engineering, which is about recognising desirable changes in the way in which things are currently perceived and done.

3 Method

The data on which this study is based were collected over a period of twelve months. To follow the course of a project for such an extended period of time it was necessary to restrict the data collection to selected points in the process. The selection of these data points was governed by two considerations: The data points should themselves be important project activities, and they should include some kind of account of the period since the previous data point. This led to choosing observation of project meetings as the primary means of data collection. Altogether the data collection comprises: (1) attendance at the two-day start-up seminar, (2) observation of the project status meetings, which took place once every two weeks, and some additional meetings, (3) interviews with eleven of the core project participants, and (4) inspection of various project documents. The meetings and interviews were recorded on tape and subsequently transcribed.

The purposes of the *project status meetings*, which gathered the entire project group, were to provide a forum for sharing information about the status of the project, maintaining awareness of the entire project, coordinating activities, discussing problems and progress, making decisions, and reviewing major project documents. During the meetings, the author of this paper was seated at the meeting table with the other people present. From their point of view, the author has been invisible in that he was not to be spoken to and have himself remained silent. During the breaks, the author talked informally with people. A total of 24 meetings (52 hours) have been observed in this way. In addition to the tape recording of the meetings, the author made notes about particularly interesting topics and copied the contents of the whiteboard when it was used.

Documents handed out at the meetings were also handed out to the author. Remaining project documents were available to the author upon request. The documents include, among other things, the project plan specifying subprojects and milestones, the information plan scheduling marketing and training, the project diary with brief notes about important events and decisions, the user's manual for the existing version of the system, excerpts of the legislation regulating the application domain, and the final as well as several preliminary versions of the requirements specification, the business model, and the scenarios. The documents, which comprise about a thousand pages, provide evidence of the evolution and intermediate outcomes of the project.

The *interviews* provided an opportunity to talk to people about their individual experiences and concerns, and to dig deeper into issues and discussions that were merely hinted at during the meetings. The interviews, which lasted 1-1½ hour each, were loosely structured by a set of guiding questions prepared for the individual interview. At the time of the interviews the author knew the project quite well and the interviews progressed as conversations focused on the parts of the project in which the interviewee was involved. Frequently, an interviewee's statements could be confirmed by or contrasted with information from the meetings, the documentation, or the other interviews and this prompted the interviewee to clarify misunderstandings, provide further details, and elaborate on disagreements. In addition, the interviewees were asked about their views on what was critical to successful completion of the project. This part of the interviews was based on a walkthrough of a 13-item list of top software-project risks produced by merging the top-10 lists of Boehm (1991) and Keil et al. (1998), see Appendix.

4 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 municipal authorities in the handling of cases concerning child support and alimony (CSA). The CSA project, estimated to last three years, is to completely redevelop the company's existing CSA system, which has been in nation-wide operation for almost two decades. During the first year of the project it was staffed with a project manager, eleven designers/developers, two service consultants, a methods-and-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. This multidisciplinary group is comprised of people with an average of more than ten years of professional experience. When referred to as a group, the members of the CSA project will be termed CSA engineers, irrespective of their different professional backgrounds.

To accomplish their task, the CSA engineers have to interact with external stakeholders such as user representatives and the governmental bodies responsible for the administration of the legislation regarding child support and alimony. Naturally, they also have to interact with management, marketing, technical services, the quality function, and other stakeholders internal to the company. 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 by other project groups in the company. This reduces the amount of code to be developed by the CSA engineers but increases the amount of coordination with people who are external to the CSA project and, thus, have other deadlines and responsibilities to take into consideration. Hence, the integration and reuse that motivate the adoption of component-based design necessitate close cooperation with a number of people outside the CSA project to negotiate, settle, and follow up on component definitions and the progress of component development (Hertzum, 2000). In addition to component-based design, the project involves a move to a technological platform with which the CSA engineers are less familiar – the Web.

The CSA engineers created a classification scheme in order to manage the system requirements and gain an overview of the work with which they had been tasked. In the requirements specification, the 221 requirements were maintained as individual entries and organised by means of the classification scheme, which had three hierarchical levels. At the top level, two of the eight categories were copies of headings in the company's template file for requirements specifications but several of the top-level categories were not used very much (see Table 1). The bulk of the requirements were contained in the top-level category on "User Requirements". Table 2 shows this part of the classification scheme along with the number of requirements in the different subcategories. At the second level, the user requirements were classified by the primary stakeholders – credit users, debit users, and management – and a few additional categories, notably "Common Requirements" which are requirements common to credit and debit users. The third-level categories map out the main functionalities involved in handling CSA cases, with the addition of a few categories concerning facilities required by secondary stakeholders.

The up-front purpose of the classification scheme was to support the communication between the CSA engineers and the user representatives – a group of prominent users of the existing system. Another significant, concrete motivation was to distribute the requirements onto major units of system functionality and thereby help ensure that the full set of requirements relevant to each unit of functionality was considered during the subsequent design of the units. Thus, a primary purpose of the classification scheme was to enable the CSA engineers to divide their task into subtasks (which often corresponded to components) without forgetting requirements relevant to individual subtasks.

The CSA engineers acknowledged that the requirements contained in the classification scheme might be biased toward revisions of and extensions to the existing CSA system. For that reason the requirements specification explicitly stated that unless it was in conflict with stated requirements the new CSA system should be able to perform the functions provided by the existing system, even if these functions were not stated explicitly in the requirements specification.

Insert Table 1 about here

Insert Table 2 about here

5 Creating the classification scheme

The creation of the classification scheme started at the very first project meeting and was completed, apart from minor edits, about a month later. Initially, the CSA engineers collected the readily available input for the requirements specification. These inputs consisted of the records from the call centre, which users contact when they need help from the service consultants, and the documentation of seven interviews with users of the existing CSA system. These inputs were supplemented with contributions from the CSA engineers, especially the service consultants, based on the domain knowledge they had acquired during their longstanding involvement in the development, maintenance, and operation of the existing CSA system. Based on this initial corpus of requirements the CSA engineers engaged in devising the classification scheme.

5.1 CLASSIFICATORY PRINCIPLES

The requirements to be satisfied by the CSA system were interrelated in various ways and presented the CSA engineers with multiple candidates for classificatory principles. While the interrelations among the requirements clearly formed a network, the CSA engineers appeared to consider a hierarchical requirements-classification scheme self-evident. The issue was never brought up for discussion, and a hierarchy was tacitly selected as the most appropriate structure on which to base the classification scheme. The immediate reason for

this was probably that the CSA engineers did not perceive the classification of the requirements as an autonomous task but as part of the production of the requirements-specification document. Such documents normally have a structure – reflected in the table of contents – that divides the linear sequence of pages into a hierarchically structured sequence of text chunks. By choosing a hierarchical classification scheme, the CSA engineers restricted themselves to structuring the requirements according to one set of concerns. Alternatively, the requirements could have been classified along several independent, overlapping, or even conflicting dimensions and made available as a hypertext. Such a multi-faceted classification of the requirements would extend the ways in which the requirements specification could be used but it would also require that more work and attention was devoted to the task of designing the classification scheme. The CSA engineers coped with the absence of means for representing more than one set of relations among the requirements by adopting a pragmatic approach where extra relations were, for example, represented by allowing categories to overlap.

The discussion about how to structure the requirements classification was opened with these statements about the classificatory principles to be used at the different levels of the hierarchy (the names of the CSA engineers are not their real names):

Anne: *How do we do it? Do we agree that we split [the requirements] into credit and debit, or what do you say, Dorothy?*

John: *Don't you think that is the easiest […] And afterwards we could split by…*

Anne: *By main functions.*

This suggestion was not accepted immediately but eventually the distinction between credit and debit users became a key element of the classification of the "User Requirements", and the main functions made up a large part of the categories at the third level of the classification (see Table 2). It was, however, debated whether it was appropriate to introduce a distinction between credit and debit users at one of the top levels of the classification. This distinction was very prominent in the existing system and some of the CSA engineers wanted to leave it more open whether the new system should adopt a similar distinction. Though the distinction reflected the division of labour in most municipalities, CSA work was organised differently in some municipalities. Furthermore, the CSA engineers wanted to arrange the categories in a sequence that corresponded to the stages in the life cycle of a CSA case. As an illustration of this, an initial attempt to create the classification scheme was discarded by the project manager because it was judged to appear unnatural and confusing to a reader who started at the beginning of the requirements specification and proceeded onward in a sequential fashion. Eventually, it turned out that CSA cases have separate, though intricately related, credit and debit life cycles so this distinction remained a primary characteristic of the requirements classification.

The top level of the classification scheme (see Table 1) was added late in the process to make room for a number of requirements that did not fit the initial focus on what the users required from the system. Some of these top-level categories concerned the negotiation process through which CSA engineers and user representatives reached agreement on the action to be taken in response to certain requirements: "Unsettled Requirements" (Category 4) and "Rejected Requirements" (Category 5). Thus, in addition to structuring the specification of the new system the requirements classification also contains explicit traces of the process that brought this specification about.

The CSA engineers' ability to adapt hierarchical classification to their concrete needs was especially apparent in relation to the distinction between credit users (Category 2.2) and debit users (Category 2.3). A large number of requirements were common to these two user groups but rather than entering these requirements into both categories a third category was created specifically for the common requirements (Category 2.4). This illustrates that, on close inspection, hierarchical classifications often disintegrate because the multiple concerns and interrelations relevant to the represented real-world phenomena cannot be reduced to a strict hierarchy (Higgins and Safayeni, 1984). A primary reason for the "Common Requirements" category was to facilitate updates since a number of the requirements were likely to be modified during the requirements-engineering process:

Chris: *We have to be careful not to write something under credit or debit, that is actually the same. Then one user group will say "Couldn't we adjust this a little" and then we suddenly have two [requirements] that were once the same but no longer are.*

Anne: *Yes, they are going into the common one [i.e., the "Common Requirements" category].*

The "Common Requirements" category provided a way to adapt a hierarchy to the CSA engineers' needs without losing the perceived benefits of hierarchies. Overall, the CSA engineers were worried that dividing the requirements onto a number of categories could lead to oversights, and this led them to adopt a "one point of access" strategy in devising the requirements classification. The CSA engineers strove for a classification in which requirements they would need to consider collectively were in the same category. Consequently, they generally considered clear-cut borders between categories more important than strong coherence internal to the categories. This led the CSA engineers to deliberately not create a large number of categories, which would be difficult to distinguish from each other. Instead, the requirements classification was designed to filter away the requirements clearly irrelevant to a category but left it to the reader to determine the relevance of the remaining requirements each time the category was used.

When the CSA engineers turned from the initial effort at creating the requirements classification to refining it by starting to classify actual requirements they were, at first, reluctant to assign requirements to just one category defined by just a few words of description. The categories had not yet developed an established meaning and, presumably, appeared too unrefined to reflect the richness of the requirements. Gradually, the meaning of the categories was elaborated and agreed upon – through additional discussions and through the use of the classification scheme for classifying the accumulated requirements. This process revealed that categories needed advocates; otherwise they were eventually removed from the requirements classification. Similarly, requirements with strong advocates among the CSA engineers tended to end up in central categories whereas requirements that received weaker support from the CSA engineers were assigned to more peripheral categories. Along similar lines, Eodice et al. (1999) divided the requirements in the project they studied into those with and those without an advocate. They report the strong finding that whereas virtually all the requirements with an advocate were eventually implemented not a single one of the requirements without an advocate were implemented. Curtis et al. (1988) allude to a related phenomenon when they note how visionary designers tend to lead development efforts because they are able to articulate the design clearly.

Finally, the classification of the requirements also had a direct impact on who would be responsible for satisfying the requirements. The functionality of the CSA system would be distributed onto a number of components. Some of these components were to be developed by the CSA engineers, the rest by other project groups in the company. Like many of the categories in the requirements classification, these project groups were typically defined in terms of the user group they were directed toward or the functionality they provided to users. Thus, the adoption of component-based design meant that assigning a requirement to one category could lay down that the CSA engineers had to deal with it themselves while assigning it to another category could be equivalent to handing it off to another project group. That is, the requirements classification conflated the specification of *what* the system should be capable of with that of *who* should implement it. This may have affected the classification of some requirements because the CSA engineers made the classification single-handed – though other project groups could subsequently contest the classifications. The division of labour among the project groups was an additional reason for maintaining a clear-cut distinction between credit and debit users at a high level in the requirements classification. This distinction made it easy for the CSA engineers to separate out the requirements pertaining to debit users. These requirements were to be handled by the project group responsible for the evolution of the company's debit system, and the CSA engineers' only involvement would be in designing the technical interfaces between the components handling the debit users' requirements and the rest of the CSA system.

5.2 MULTIPLE, INTER-RELATED CLASSIFICATIONS

In creating the classification scheme, a significant part of the CSA engineers' task was to piece together a number of pre-existing classifications and distinctions in a way appropriate to the redesign of the CSA system. Several of these classifications reach into the CSA legislation and the ways in which people organise themselves after a divorce and, thus, bring intricate aspects of the application domain into the requirements classification. Other of the pre-existing classifications and distinctions concern the administration of the CSA project and bring aspects related to the software-engineering process into the requirements classification. Through the incorporation of these classifications and distinctions, the requirements classification is itself incorporated in a classificatory system of much larger scope. Bowker and Star (1999) talk about *boundary infrastructures* consisting of numerous, inter-linked boundary objects, classifications, and standards. This means that though the requirements classification is itself small-scale it must be seen and read with reference to at least seven other classifications:

- The standard classification of requirements into functional and non-functional, with the addition of data requirements. These three classes of requirements can be found in textbooks on software engineering (e.g.,

Sommerville, 1996) and they appear as suggested headings in the company's template file for requirements specifications. "Data Requirements" appear as a top-level heading in the requirements classification, which is otherwise strongly biased toward functional requirements.

- A classification of the primary users of the CSA system into debit users, credit users, and management. This classification reflects the division of labour in most, but not all municipalities (some small municipalities do not have separate debit and credit users, and some larger municipalities divide work along other dimensions). The three primary user groups constitute the major classificatory principle at the second level of the requirements classification, to the extent of neglecting other stakeholders (see Section 7).
- The existing system constitutes one view of the use situation and thereby emphasises some aspects of CSA work while others are made virtually invisible. This way the facilities of the existing system provide a classification of the tasks that enter into CSA work. About half of the third-level categories in the requirements classification correspond to main functions of the existing CSA system (see Table 2, especially Categories 2.2 and 2.3).
- A supplementary classification of the requirements into three priority levels: *mandatory* (i.e. requirements that must be satisfied), *desirable* (i.e. requirements the CSA engineers will attempt to satisfy), and *luxury* (i.e. requirements that will only be satisfied if it can be done without delaying the project). Each requirement is labelled with one of these priority levels and this provides the CSA engineers' response to the user representatives in terms of the importance assigned to taking action to satisfy the requirements.
- A classification of citizens who – following a divorce – are involved in a CSA case. This classification, which shows up in the wording of numerous requirements, divides citizens into payers, payees, and children or spouses. It may, however, be a nontrivial matter to determine paternity, and over time the concept of spouse has expanded in scope from the sphere of married couples, over unmarried couples living together, to homosexuals in registered partnerships.
- The legislation regarding child support and alimony is, in effect, a classification of circumstances and events with associated rights and obligations for the involved citizens. As a simple example, payers are divided into three groups based on their income, and this affects the size of their CSA payments. These circumstances and events give rise to, among other things, a variety of notifications (Category 2.4.7), which trigger changes in how cases are handled.
- A division of the work in the development organisation onto a number of longstanding projects. Several of these projects are responsible for facilities that are considered for inclusion in the new CSA system. Different decisions about how to classify a requirement may, as already mentioned, have an effect on who will be responsible for satisfying it.

This illustrates how the requirements classification brought a number of complexly interwoven categories and conceptions together in a single structure. Thereby the creation of the classification scheme involved a number of decisions about the relative importance of such categories and conceptions, as well as workarounds to reconcile issues such as the wish for representing more than one set of concerns in a hierarchical classification. Through its inclusions, exclusions, and balancing of concerns the requirements classification created one framework for the CSA project out of a vast number of possible ones.

6 Using the classification scheme

The initial purpose of the classification scheme was to facilitate communication with the user representatives during the requirements-engineering process. CSA engineers met with the user representatives to walk through the accumulated requirements and bring forth additional requirements. As a result of these meetings some adjustments were made to the classification scheme but the user representatives were primarily concerned with the actual requirements. The classification scheme was mostly attended to when requirements were inappropriately classified or categories were difficult to make sense of. Yet, the classification scheme was used repeatedly, but inattentively during the meetings in that it supported the CSA engineers and user representatives in considering related requirements concurrently and in maintaining an overview.

After the requirements specification was completed it assumed its double role of, on the one hand, contract between users and development organisation and, on the other hand, checklist for the CSA engineers during the development and evaluation of subsequent design artefacts. In these roles, the requirements specification and its classification scheme had a primarily indirect effect on the design process. For example, the business model and the scenarios – two of the design artefacts developed after the requirements specification – were not generated from the requirements specification. Rather, they were developed on the basis of the CSA engineers' knowledge of the domain and the users' tasks, supplemented by discussions with the user representatives and

some reading of the legislation. The classification scheme was used most visibly when the requirements specification was brought in at selected points in the process, for example to validate that design artefacts such as the scenarios met the full range of requirements. However, the classification scheme also affected the design process in another, more fundamental way as a constituent part of the assumptions about the scope of the project.

6.1 CONTROLLING THE SCOPE OF THE PROJECT

The requirements specification for the new CSA system inherited a lot of its structure from the existing CSA system. One reason for this might be that it is 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 (Carroll et al., 1991; Naur, 1965). In envisioning a new system it is important to break away from the present understanding of how the users' tasks and the technological options define the use situation, but it is equally important to identify the aspects of the users' work that should be preserved. Software engineers have to strike this balance between tradition and transcendence (Ehn, 1989) throughout the systems-development process.

In the CSA project, the tradition was not carried by the users only but also by the CSA engineers through their longstanding involvement in the development and operation of the existing system. Further, the accumulated records of the users' critiques of the existing system were an important source of input for the creation of the requirements classification. This introduced a potentially undue bias toward tradition. The CSA engineers were aware of this risk but explicitly argued that it was more important that the classification depicted the world in a way readily recognisable to the user representatives. This significantly increased the ease with which draft versions of the requirements specification could be communicated to the user representatives. By building the requirements classification largely around the accumulated critique of the existing system, it also became easier to classify these requirements. While these two consequences of a bias toward tradition were convenient, they also illustrate how the requirements classification indirectly constrained the requirements-engineering process to requirements that could be conceived of within the framework of the existing system. This is apparently at odds with the activities undertaken to facilitate the user representatives in an open-ended search for the optimal balance between tradition and transcendence (e.g., a vision workshop conducted as part of one of the meetings). The CSA engineers were, however, faced with two contradictory concerns. On the one hand, the CSA engineers needed to conduct the requirements-engineering process in a way that honoured expectations of adequate user involvement. On the other hand, they needed to maintain some level of control over the direction, scope, and outcome of the requirements-engineering process, which concluded in a specification of what the customers had requested and the developers agreed to deliver – a contract. The requirements classification played a discrete but important role in the CSA engineers' handling of these two concerns in that it enabled the CSA engineers to act in accord with expectations of adequate user involvement while at the same time constraining the process. On several occasions, the CSA engineers explicitly asked the user representatives for new ideas and visions regarding the system but, at the same time, the meetings with the user representatives evolved around a walkthrough of the classified requirements, one category at a time. Under these circumstances, the user representatives had few ideas for new facilities that would enhance the system. During an interview, one of the CSA engineers commented on this in the following way:

> *John: I have asked myself whether it is because it [i.e., the user representatives' task in general and the requirements specification in particular] is too big or too burdensome, whether they cannot relate to it and transform it to their own day-to-day world, or whether they simply agree. […] I have said to several [of the user representatives] "Are you aware that it is now you have to come forward if you have something?" […] They have not had a clear idea about where we're going and what the system is going to look like. They have focused – and that's understandable – on the inconveniences they experience in the existing system.*

The tension between open-ended user involvement and the contractual aspect of requirements specification was rooted deeply in the CSA engineers' perception of their work, and they considered disregard of this tension tantamount to being unprofessional. This was, for example, a problem in their relations with a usability specialist who considered it her role to systematically "adopt the users' perspective". Following Garfinkel (1967), the requirements classification was a means of handling the CSA engineers' normal, natural troubles; that is, the sort of troubles that are inherent in complex issues, aggravated by conflicting interests, and coped

with through local work practices. To the CSA engineers handling conflicting interests is normal, natural practice to the extent that they probably remained largely unaware of how effective the requirements classification was as a means of controlling the scope of their project.

6.2 FOCUS ON IMMEDIATE USE

Most of the requirements embodied in the existing CSA system never made it explicitly into the requirements specification and, consequently, the entire backbone of the classification was largely implicit. To make sense of the requirements specification it is essential that the reader understands the implications of the sentence saying that the new CSA system should, by default, be able to perform the functions provided by the existing system, irrespective of whether these functions were explicitly stated in the requirements specification. It would have taken considerable resources to unfold this sentence in writing, and though that was considered highly important early in the requirements-engineering process it was ultimately not considered worthwhile. This extensive use of condensed writing, which left most of the context unsaid, preserved resources during the requirements-engineering process but at the same time reduced the number of people capable of understanding the requirements specification. Most of the CSA engineers and user representatives possessed the necessary knowledge of the existing system but those among the CSA engineers who had not been involved in the development and maintenance of the existing system were unable to get a coherent understanding of their task from reading the requirements specification.

The condensed writing must be elaborated if the requirements specification is to be understandable to people such as municipal managers and future CSA maintenance engineers. The research on, for example, design rationale and reuse generally assumes that such elaboration would pay (e.g., Moran and Carroll, 1996; but see Grudin, 1996, for a counter-argument). However, several of the CSA engineers felt that the time it would take to put the requirements embodied in the existing system into writing could be spent more productively on other activities. To some extent, the competent readers of the requirements specification saw elaboration of the condensed writing as redundant and annoying in that it would introduce scores of what they perceived as self-evident requirements, which nevertheless would consume time and divert attention from more important issues (see also Brown and Duguid, 1996). Since the competent readers encompassed most of the immediate users of the requirements specification, the CSA project could progress with a requirements classification that left many requirements unstated. It could be argued that these unstated requirements were ‘stated’ so clearly in the functionality of the existing system that everybody would be able to read them off. This was however not the case since only some of the features of the existing system were to be preserved, others were minor issues that could be dealt with in manifold ways, and still others were obliterated by the new system. Instead of clarifying which aspects of the existing system that should be carried over to the new system the requirements specification presumed, among other things, a firm understanding of the existing system.

When professionals such as the CSA engineers are busy or simply absorbed in getting their work done they are likely to devise – and be content with – classification schemes that support them in their own sense-making process. They are much less inclined to spend time creating classification schemes that make their work accessible to people outside the group of their immediate collaborators. This implies a focus on immediate use, rather than long-term use and reuse, and is reinforced by the necessity of getting each day’s work done within the limits of the available resources. Consequently, small-scale classification schemes differ from large-scale classification schemes in their balancing of condensed writing and broad understandability. Large-scale classification schemes are meant to be of use also outside the group of their immediate creators and must, therefore, pay much more attention than small-scale classification schemes to elaboration of condensed writing.

7 Comparison of classification scheme and project meetings

The written requirements classification can be analysed further by comparing it with the oral discussions about system requirements. To initiate such a comparison, which will reveal some of the condensed writing, the contents of the requirements classification and the discussions at the project meetings have been mapped onto a number of stakeholder groups and abstraction levels. Multiple groups of people have a stake in the new CSA system. The requirements specification explicitly states that it, directly or indirectly, reflects the requirements of the following stakeholders: (1) *The municipalities*, which are the primary users of the CSA system and sub-specified into credit users, debit users, and management. (2) *Other public institutions*, which form a diverse group of secondary users including the taxation authorities and the national bureau of statistics. (3) *The citizens* who are involved in a CSA case. (4) *The development organisation*, which is sub-specified into management, marketing, service, operations, and the CSA project group itself. It should be noted that the people who have a stake in the system – and thus may have input to the requirements specification – form a heterogeneous group including users as well as non-users. In addition to considering the different stakeholders, the CSA engineers

also have to address their design task at several levels of abstraction. Correspondingly, numerous methods for work analysis distinguish between at least three levels of analysis: goals, tasks, and actions. Whereas the task level is concerned with *what* goes on, the goal level provides the reasons *why* it is going on, and the action level describes *how* it is carried out. As an example, the abstraction hierarchy developed by Rasmussen et al. (1994) distinguishes between five levels, termed goals, priority measures, general functions, work processes, and physical objects. During a design process, designers repeatedly move up and down in the abstraction hierarchy to clarify what the system is to achieve and how that can be implemented as well as to clarify what is implementable and how that constrains what the system can possibly be made to do.

Figure 1 shows how the 221 system requirements map onto the four groups of stakeholders identified in the requirements specification and the five abstraction levels defined by Rasmussen et al. (1994). It is immediately visible that the requirements specification is strongly focused on the general functions required by the municipalities; that is, the primary users' functional requirements. As much as 80% of the requirements concern this one combination of abstraction level and stakeholder group (Figure 2 details how these 177 requirements are distributed onto the three municipal stakeholders). In the requirements specification, each of the 221 requirements appears as an independent entity, which is classified with some requirements and thereby differentiated from the remaining requirements. No distinctions are made between requirements at different levels of abstraction and there are, consequently, no links indicating that a certain priority measure is intended to balance certain goals and is implemented through a certain subset of general functions. It is left to the reader of the requirements specification to hold and recall information about goals and to infer how they give rise to the general functions. This incompleteness was recognised by the CSA engineers who, on several occasions, discussed the urgent need to clarify and reach closure on what they called the *worldview* or *mindset* that formed the basis for the design of the new CSA system. They considered to lay out this worldview in the requirements specification as a coherent, high-level, lead-in description of what the new system was to achieve. However, this description was never made. Such absence or severe incompleteness of information about the intentions underlying designs seems to be a recurrent feature of design documentation (Button and Sharrock, 1996; Hertzum and Pejtersen, 2000; Parnas and Clements, 1986).

Insert Figure 1 about here

Insert Figure 2 about here

In contrast to the contents of the requirements specification, the CSA engineers spent a considerable portion of their meetings discussing the goals, constraints, and priority measures that entered into making sense of the requirements. Figure 3 covers the two meetings during which the main structure of the requirements classification was created and shows how the discussions at these meetings, which lasted a total of nine hours, map onto the abstraction levels and stakeholder groups. This figure excludes the 26% of the CSA engineers' utterances that were considerations about how to go about their work. After removing these process considerations the remaining utterances were grouped into chunks, each dealing with a single stakeholder group at a single level of abstraction. Whenever a chunk treated several topics, for example several different requirements, it was subdivided into chunks dealing with a single topic each.

Insert Figure 3 about here

Like the requirements specification, the discussions at the meetings were strongly focused on the municipal users. Slightly more attention was devoted to the other stakeholder groups, but the marginal attention devoted to the citizens (the ultimate 'victims' of the system) is an important characteristic of the entire requirements-engineering process. The requirements classification does not have a single category for recording input from citizens or considerations about their needs. With respect to abstraction levels, the 184 chunks resulting from the analysis of the project meetings comprise 71 concerning goals, constraints, and priority measures, 62 concerning general functions, and 51 concerning the two most concrete levels of abstraction. This is a much more even distribution than in the written requirements specification and illustrates what could be a central difference between oral and written communication. At the meetings, the CSA engineers paid a lot of attention to the intentions underlying the requirements. This was not done as a separate lead-in step to the requirements classification but as an activity integrated with clarifying the meaning and contents of the individual requirements and categories. Some issues were clarified by moving from lower to higher levels of abstraction, for example from general functions to goals. These transitions seemed essential to the CSA engineers' building of their understanding of what the new system was to achieve, but the goal-level understanding arising from

these transitions was not recorded in the requirements-specification document. Another set of issues were clarified by moving downward from general functions to work processes-. Most of these transitions represent the introduction of real-world examples in terms of either descriptions of details in typical courses of events or accounts of special cases. Very few of these examples were included in the requirements-specification document. The near absence of such contextual and explanatory information may be an important factor in understanding why software engineers prefer to get most of their information from informed people, rather than documents.

Leveson (2000), Moran and Carroll (1996), Pejtersen and Albrechtsen (2000), and others propose to extend typical design documentation with intent specifications, design rationales, and other goal-oriented descriptions to more fully support the tasks people perform during software engineering and evolution. An argument against such specifications is that the cost of producing them may outweigh their value, especially for the people most involved in and most knowledgeable about the projects. However, the priority measures, which specify how conflicting goals are balanced against each other, point to a set of delicate discussions among the stakeholders involved in the requirements-engineering process. It was part of the CSA engineers' role to facilitate such discussions among the user representatives, and the CSA engineers generally found this difficult. To focus and drive such discussions, it could be cost-effective to produce specifications of important priority measures.

In the CSA project, the near absence of recorded requirements regarding priority measures indicates that conflicts among goals are generally left implicit. The only three priority measures that appear in the requirements specification are (1) that the credit and debit users in the municipalities have priority over other users, (2) that a CSA case consists of two equally important, related but separate sub cases: a credit sub case and a debit sub case, and (3) that the primary user interface of the CSA system is web-based though a mainframe interface must also be provided. However, a total of nineteen priority measures are discussed at the meetings. Compared to this, the requirements specification depicts CSA work as almost conflict-free. One of the priority measures discussed at the meetings concerns the tension between adhering strictly to the legislation and supporting the handling of real-world cases. On the one hand the system must not support the users in doing things they are not allowed to do, on the other hand the system must support the users' actual activities, not merely their formal task. While this priority measure is of key importance, it also illustrates that some priority measures are kept implicit for good reason – the documentation *must at all times* convey the impression that the system supports its users in handling CSA cases as prescribed by the legislation. It would, however, have been unproblematic to state most of the priority measures explicitly and that would, probably, have been seen as a natural part of the CSA engineers' role as facilitators of discussions among the various stakeholder groups. One such example is how to prioritise a straightforward system against a system capable of handling any special case. It was frequently discussed at the meetings how a small fraction of the CSA cases was responsible for the major part of the complexity of the system. Leaving (some of) these cases for manual treatment could substantially simplify the CSA system and thereby the treatment of the majority of the cases. As it is, the cost of including the special cases was not mentioned in the requirements specification and was therefore easily forgotten when yet another special case was considered for inclusion.

The comparison of the written requirements specification and the oral discussions at the meetings illustrates that a crucial aspect of a classification scheme concerns the decisions about where the artefact stops and oral communication takes over. In complex work domains neither artefacts nor oral communication can be obliterated but their relative contributions can vary within wide limits (Hertzum, 1999; Schmidt and Simone, 1996). Due to resource limitations and a focus on immediate use, rather than long-term use, small-scale classification schemes will normally rely on oral communication to provide substantial amounts of the information necessary to make sense of the classification scheme. The most cost-effective balance between documentation and relying on the involved engineers' memory and exchange of information depends on the specifics of individual projects and must be determined locally and temporarily. While an artefact such as the Viewpoints framework (Finkelstein et al., 1992; Nuseibeh et al., 1994) allows for the inclusion of multiple perspectives, it fails to support software engineers in getting an overview of which perspectives to consider for inclusion. The matrix of abstraction levels and stakeholder groups provides a simple but potentially useful tool for working systematically with such considerations. In the CSA project this could, for example, have increased the CSA engineers' awareness of the extent to which they ended up negotiating priority measures amongst themselves, while the user representatives typically stated goals and requirements without committing to their consequences for other goals and requirements.

8 Conclusion

In organising and coordinating their work, software engineers regularly create small-scale classification schemes. These classification schemes may, for example, be objectified as the structure of written documents

such as the requirements specification of the CSA project. This study has analysed how the requirements classification for the redevelopment of a large information system blended with pre-existing classifications and conceptions and became a constituent element of the requirements-engineering process.

The requirements classification of the CSA project was mainly introduced to help CSA engineers and user representatives gain an overview. This appears to be a key role of classification schemes and is akin to coordination mechanisms in which the protocol serves the role of a map. Explicitly, the protocol provided opportunities for – rather than stipulated ways of – working with the requirements, but the main consequences of the requirements classification remained implicit. Implicitly, the requirements classification, which inherited a lot of its structure from the existing CSA system, rendered requirements that transcended the framework laid out by the existing system almost invisible. This way the requirements classification contributed to constraining the requirements-engineering process and thereby supported the CSA engineers in maintaining some level of control over the scope of the project. Thus, to the CSA engineers the requirements classification was an important means of discretely balancing their management of the contractual aspect of requirements engineering against facilitating the users in an open-ended search for their system requirements. The distinction between explicit and implicit consequences of classification schemes is absent from the concept of coordination mechanisms in that it elaborates how artefacts such as classification schemes *enable* cooperative activities. In situations involving people with different, possibly conflicting interests, coordination mechanisms should be complemented with work on the implicit or even disabling consequences of classification schemes.

The implicit consequences of classification have been at the fore of the analyses approaching classification schemes as boundary objects. While this proved a valuable focus, the CSA project also revealed that the requirements classification was, partly, intended to serve as a checklist within the group of CSA engineers; that is, within rather than across boundaries. Here, the CSA engineers used the classification in coordinating their activities at one point in time with those at a later point in time. Such activities are excluded from the concept of boundary objects, which is explicitly confined to mediation between different groups of actors (i.e., *across* boundaries). Further, the ways in which the requirements classification affected the communication between CSA engineers and user representatives reflected that the requirements classification was created and controlled by the CSA engineers, while the user representatives were in a merely reactive position. This is evident in the CSA engineers' use of the requirements classification as a means of discretely controlling the scope of the project but also in several of the other concerns that shaped the classification of the requirements. These concerns illustrate that the requirements classification is not only determined by how the requirements relate to each other in the application domain but also by issues internal to the software-engineering effort, such as facilitating updates of the requirements classification without introducing inconsistencies, ensuring one point of access to all requirements relevant to a piece of functionality, and affecting the allocation of who will be responsible for satisfying which requirements. The close ties between the creators of the classification scheme and one of the actors involved in using it is a distinguishing characteristic of small-scale classification, as opposed to large-scale classification.

The requirements specification of the CSA project is strongly focused on the primary users' functional requirements. The vast majority of these functional requirements are not accompanied by any description of the context in which they occur – neither in terms of goals, constraints, and priority measures nor in terms of real-world examples or other detailed descriptions of typical or exceptional cases. It is left to the readers to supply this information, and unless they can do that they will not be able to make much sense of the requirements. Comparing the written requirements specification with the oral discussions at the meetings, it is evident that even the CSA engineers, who know the domain well, need to devote considerable attention to elaboration of the context in order to work out the meaning of the requirements and the best way to organise them. At the meetings where the requirements classification was created the CSA engineers divided their attention about evenly between goals, constraints, and priority measures (38%), general functions (34%), and work processes and physical objects (28%). The near absence of goals, constraints, and priority measures in the written requirements specification (only 4% of the requirements) provides documentation of the CSA engineers' extensive reliance on condensed writing, which leaves the context implicit. The condensed writing is only understandable to a select group of people and may be a key factor in explaining software engineers' general preference for getting their information from informed people, rather than documents.

The abstraction level and stakeholder group matrix used in analysing the CSA project offers a simple, high-level tool for working with the scope of requirements specifications. This abstraction-stakeholder matrix emphasises that all stakeholders be considered and that requirements be considered at several levels of abstraction. Furthermore, it points out the specific need for priority measures that balance conflicting goals, such as the interests of different stakeholders, against each other. The abstraction-stakeholder matrix – itself a small-scale classification scheme – captures aspects that appear important in working toward a complete set of well-described system requirements. Further work is, however, needed to determine when it no longer pays to

spend resources creating classification schemes and writing increasingly elaborate specifications and, instead, becomes cost-effective to rely on other means, such as oral communication.

Acknowledgements

This work has been supported by the Danish National Research Foundation through its funding of the Centre for Human-Machine Interaction and by the British Royal Society through its funding of the GRIS project. I wish to thank my colleagues at Risø and the three anonymous reviewers for thorough and constructive comments on earlier versions of the paper. Special thanks are due to the members of the CSA project group who have put up with my presence in spite of their busy schedule.

Appendix: Top-13 list of software-project risks

In discussing factors critical to successful completion of the CSA project, the interviewees were asked to provide their views on the thirteen items on the following list of software-project risks. The list was produced by merging the top-10 lists of Boehm (1991) and Keil et al. (1998). Six of the risks were considered of particular relevance to the CSA project by two or more interviewees. These six risks are marked with an asterisk (*).

1. Lack of top management commitment to the project *
2. Personnel shortfalls *
3. Unrealistic schedules and budgets
4. Developing the wrong functions and properties, i.e. misunderstanding the requirements
5. Developing the wrong user interface
6. Gold-plating
7. Continuing stream of requirements changes
8. Shortfalls in externally furnished components *
9. Shortfalls in externally performed tasks *
10. Real-time performance shortfalls *
11. Straining computer-science capabilities / introduction of new technology *
12. Failure to gain user commitment
13. Failure to manage end-user expectations

References

Albrechtsen, Hanne and Elin J. Jacob (1998): The Dynamics of Classification Systems as Boundary Objects for Cooperation in the Electronic Library. *Library Trends*, vol. 47, no. 2, pp. 293-312.

Bishop, Ann P. (1999): Document Structure and Digital Libraries: How Researchers Mobilize Information in Journal Articles. *Information Processing & Management*, vol. 35, no. 3, pp. 255-279.

Boehm, Barry W. (1991): Software Risk Management: Principles and Practices. *IEEE Software*, vol. 8, no. 1, pp. 32-41.

Bowker, Geoffrey C. (1998): The Kindness of Strangers: Kinds and Politics in Classification Systems. *Library Trends*, vol. 47, no. 2, pp. 255-292.

Bowker, Geoffrey and Susan L. Star (1991): Situations vs. Standards in Long-Term, Wide-Scale Decision-Making: The Case of the International Classification of Diseases. In J. F. Nunamaker and R. H. Sprague (eds.): *Proceedings of the 24th Annual Hawaii International Conference on System Sciences, Vol. IV, Koloa, Hawaii, January 8-11, 1991*. Los Alamitos, CA: IEEE Computer Society Press, pp. 73-81.

Bowker, Geoffrey C. and Susan L. Star (1999): *Sorting Things Out: Classification and Its Consequences*. Cambridge, MA: MIT Press.

Brown, John S. and Paul Duguid (1996): The Social Life of Documents. *First Monday*, vol. 1, no. 1, http://www.firstmonday.dk/issues/issue1/documents/index.html (consulted June 3, 2002).

Button, Graham and Wes Sharrock (1996): Project Work: The Organisation of Collaborative Design and Development in Software Engineering. *Computer Supported Cooperative Work*, vol. 5, no. 4, pp. 369-386.

Carroll, John M., Wendy A. Kellogg and Mary B. Rosson (1991): The Task-Artifact Cycle. In J. M. Carroll (ed.): *Designing Interaction: Psychology at the Human-Computer Interface*. Cambridge: Cambridge University Press, pp. 74-102.

Carstensen, Peter H. and Carsten Sørensen (1996): From the Social to the Systemic. *Computer Supported Cooperative Work*, vol. 5, no. 4, pp. 387-413.

Curtis, Bill, Herb Krasner and Neil Iscoe (1988): A Field Study of the Software Design Process for Large systems. *Communications of the ACM*, vol. 31, no. 11, pp. 1268-1287.

Dillon, Andrew (1991): Readers' Models of Text Structures: The Case of Academic Articles. *International Journal of Man-Machine Studies*, vol. 35, no. 6, pp. 913-925.

Dillon, Andrew (1994): *Designing Usable Electronic Text: Ergonomic Aspects of Human Information Usage*. London: Taylor & Francis.

Ehn, Pelle (1989): *Work-Oriented Design of Computer Artifacts*. Stockholm, Sweden: Arbetslivscentrum.

Eodice, Michael T., Renate Fruchter and Larry J. Leifer (1999): Towards a Theory of Engineering Requirements Definition. In B. Lindemann and V. Meerkamm (eds.): *Proceedings of the ICED 99 International Conference on Engineering Design, Vol. III, Munich, Germany, August 24-26, 1999*. Garching, Germany: Technische Universität München, pp. 1541-1546.

Finkelstein, A., J. Kramer, B. Nuseibeh, L. Finkelstein and M. Goedicke (1992): Viewpoints: A Framework for Integrating Multiple Perspectives in System Development. *International Journal of Software Engineering and Knowledge Engineering*, vol. 2, no. 1, pp. 31-57.

Garfinkel, Harold (1967): "Good" Organizational Reasons for "Bad" Clinic Records. In H. Garfinkel: *Studies in Ethnomethodology*. Englewood Cliffs, NJ: Prentice Hall, pp. 186-207.

Grudin, Jonathan (1996): Evaluating Opportunities for Design Capture. In T. P. Moran and J. M. Carroll (eds.): *Design Rationale: Concepts, Techniques, and Use*. Mahwah, NJ: Lawrence Erlbaum, pp. 453-470.

Hertzum, Morten (1999): Six Roles of Documents in Professionals' Work. In S. Bødker, M. Kyng and K. Schmidt (eds.): *ECSCW'99: Proceedings of the Sixth European Conference on Computer Supported Cooperative Work, Copenhagen, Denmark, September 12-16, 1999*. Dordrecht, The Netherlands: Kluver, pp. 41-60.

Hertzum, Morten (2000): People as Carriers of Experience and Sources of Commitment: Information Seeking in a Software Design Project. *New Review of Information Behaviour Research*, vol. 1, pp. 135-149.

Hertzum, Morten and Annelise M. Pejtersen (2000): The Information-Seeking Practices of Engineers: Searching for Documents as well as for People. *Information Processing & Management*, vol. 36, no. 5, pp. 761-778.

Higgins, Christopher A. and Frank R. Safayeni (1984): A Critical Appraisal of Task Taxonomies as a Tool for Studying Office Activities. *ACM Transactions on Office Information Systems*, vol. 2, no. 4, pp. 331-339.

Keil, Mark, Paul E. Cule, Kalle Lyytinen and Roy C. Schmidt (1998): A Framework for Identifying Software Project Risks. *Communications of the ACM*, vol. 41, no. 11, pp. 76-83.

Kircz, Joost G. (1998): Modularity: The Next Form of Scientific Information Presentation? *Journal of Documentation*, vol. 54, no. 2, pp. 210-235.

Leveson, Nancy G. (2000): Intent Specifications: An Approach to Building Human-Centered Specifications. *IEEE Transactions on Software Engineering*, vol. 26, no. 1, pp. 15-35.

Moran, Thomas P. and John M. Carroll (eds.) (1996): *Design Rationale: Concepts, Techniques, and Use*. Mahwah, NJ: Lawrence Erlbaum.

Naur, Peter (1965): The Place of Programming in a World of Problems, Tools, and People. In W. Kalenich (ed.): *Proceedings of the IFIP Congress 65*. Washington, DC: Spartan Books, pp. 195-199. [Reprinted 1992 in P. Naur: *Computing: A Human Activity*. New York: ACM Press, pp. 1-9.]

Nuseibeh, Bashar, Jeff Kramer and Anthony Finkelstein (1994): A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification. *IEEE Transactions on Software Engineering*, vol. 20, no. 10, pp. 760-773.

Olson, Hope A. (1998): Mapping Beyond Dewey's Boundaries: Constructing Classificatory Space for Marginalized Knowledge Domains. *Library Trends*, vol. 47, no. 2, pp. 233-254.

Parnas, David L. and Paul C. Clements (1986): A Rational Design Process: How and Why to Fake It. *IEEE Transactions on Software Engineering*, vol. 12, no. 2, pp. 251-257.

Pejtersen, Annelise M. and Hanne Albrechtsen (2000): Ecological Work Based Classification Schemes. *Advances in Knowledge Organization*, vol. 7, pp. 97-110.

Potts, Colin and Lara Catledge (1996): Collaborative Conceptual Design: A Large Software Project Case Study. *Computer Supported Cooperative Work*, vol. 5, no. 4, pp. 415-445.

Rasmussen, Jens, Annelise M. Pejtersen and L. P. Goodstein (1994): *Cognitive Systems Engineering*. New York: Wiley.

Schmidt, Kjeld (1999): Of Maps and Scripts: The Status of Formal Constructs in Cooperative Work. *Information and Software Technology*, vol. 41, no. 6, pp. 319-329.

Schmidt, Kjeld and Liam Bannon (1992): Taking CSCW Seriously: Supporting Articulation Work. *Computer Supported Cooperative Work*, vol. 1, no. 1&2, pp. 7-40.

Schmidt, Kjeld and Carla Simone (1996): Coordination Mechanisms: Towards a Conceptual Foundation of CSCW System Design. *Computer Supported Cooperative Work*, vol. 5, no. 2&3, pp. 155-200.

Simon, Herbert A. (1973): The Structure of Ill Structured Problems. *Artificial Intelligence*, vol. 4, no. 3&4, pp. 181-201.

Simone, Carla and Marcello Sarini (2001): Adaptability of Classification Schemes in Cooperation: What Does It Mean? In W. Prinz, M. Jarke, Y. Rogers, K. Schmidt and V. Wulf (eds.): *ECSCW 2001: Proceedings of the Seventh European Conference on Computer Supported Cooperative Work, Bonn, Germany, September 16-20, 2001*. Dordrecht, The Netherlands: Kluwer, pp 19-38.

Sommerville, Ian (1996): *Software Engineering. Fifth Edition*. Reading, MA: Addison-Wesley.

Star, Susan L. (1989): The Structure of Ill-Structured Solutions: Boundary Objects and Heterogeneous Distributed Problem Solving. In L. Gasser and M. N. Huhns (eds.): *Distributed Artificial Intelligence, Vol. II*. San Mateo, CA: Morgan Kaufmann, pp. 37-54.

Star, Susan L. and James R. Griesemer (1989): Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science*, vol. 19, no. 3, pp. 387-420.

Walz, Diane B., Joyce J. Elam and Bill Curtis (1993): Inside a Software Design Team: Knowledge Acquisition, Sharing, and Integration. *Communications of the ACM*, vol. 36, no. 10, pp. 63-77.

Table 1. The eight categories at the top level of the requirements classification.

	Category	No. of requirements
1	General Requirements	8
2	User Requirements	183
3	Data Requirements	8
4	Unsettled Requirements	3
5	Rejected Requirements	2
6	Documentation	4
7	Technical Requirements	11
8	Test and Release	2
	Total	221

Table 2. The subcategories of the top-level category “User Requirements”.

		Category	No. of requirements
2.1		**CSA Document/Case**	**4**
2.2		**Payee (Calculation and Outgoing Payment)**	**61**
	2.2.1	Create New Case	16
	2.2.2	Edit Case	7
	2.2.3	Calculation	26
	2.2.4	Outgoing Payment	8
	2.2.5	Outstanding Debt	4
2.3		**Payer (Debtor)**	**45**
	2.3.1	Create New Case (Payer)	8
	2.3.2	Charging / Assessment	12
	2.3.3	Division of Incoming Payment	13
	2.3.4	Supplementary Child Benefit	8
	2.3.5	Administrative Reduction of Debts/Arrears	4
2.4		**Common Requirements**	**47**
	2.4.1	Change of Address	9
	2.4.2	International Cases	14
	2.4.3	Cease/Deletion/Suspension/Termination of Case	7
	2.4.4	Repayment	1
	2.4.5	Clearing Arrangement	1
	2.4.6	Archiving / Change Tracking	2
	2.4.7	Notification / Free-Text Notes	6
	2.4.8	P-Info	3
	2.4.9	F&N	4
2.5		**Management and Planning**	**14**
	2.5.1	Statistics / Management Information	4
	2.5.2	Auditing	2
	2.5.3	Statistics Denmark	1
	2.5.4	Tax Information	6
	2.5.5	Ledger Entries	1
2.6		**Output**	**12**
		Total	183

Note. Though labelled with the payee and the payer, Categories 2.2 and 2.3 contain requirements pertaining to the municipalities’ credit users and debit users, respectively. P-Info (Category 2.4.8) and F&N (Category 2.4.9) are systems that provide general information about citizens. Statistics Denmark (Category 2.5.3) is the national bureau of statistics.

Stakeholder group / Abstraction level	Development organisation	Municipalities	Other public institutions	Citizens	Total
Goals and constraints	4 (2%)	2 (1%)	1 (0%)		7 (3%)
Priority measures	1 (0%)	2 (1%)			3 (1%)
General functions	9 (4%)	177 (80%)	10 (5%)		196 (89%)
Work processes	6 (3%)	7 (3%)			13 (6%)
Physical objects	2 (1%)				2 (1%)
Total	22 (10%)	188 (85%)	11 (5%)	0 (0%)	221 (100%)

Figure 1. The written CSA requirements, classified by abstraction levels and stakeholder groups.

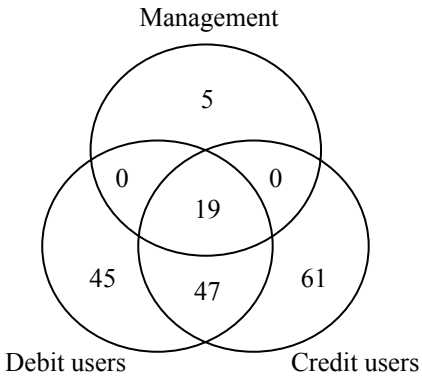

Figure 2. The 177 requirements relating to the general functions required by the municipalities. The breakdown of the requirements shows a clear focus on debit and credit users, at the expense of management.

Stakeholder group / Abstraction level	Development organisation	Municipalities	Other public institutions	Citizens	Total
Goals and constraints	14 (8%)	29 (16%)	7 (4%)	2 (1%)	52 (28%)
Priority measures	3 (2%)	16 (9%)			19 (10%)
General functions	1 (1%)	58 (32%)		3 (2%)	62 (34%)
Work processes	6 (3%)	24 (13%)	5 (3%)	5 (3%)	40 (22%)
Physical objects	4 (2%)	3 (2%)	3 (2%)	1 (1%)	11 (6%)
Total	28 (15%)	130 (71%)	15 (8%)	11 (6%)	184 (100%)

Figure 3. The oral discussions at the requirements meetings, classified by abstraction levels and stakeholder groups.