A revised version of this is published as: Holmström, J. and Sawyer, S. (2011) "Exploring Information Systems Developers' Black-Boxing of the Emergent Character of Requirements." *European Journal of Information Systems*, to appear.

REQUIREMENTS ENGINEERING BLINDERS: EXPLORING INFORMATION SYSTEMS DEVELOPERS' BLACK-BOXING OF THE EMERGENT CHARACTER OF REQUIREMENTS

Abstract:

In this paper we focus empirical and conceptual attention on the social construction of information systems (IS) requirements, and illustrate that IS developers too often choose to ignore, and thus effectively black-box, the complexities of gathering requirements in order to simplify both the difficulties of their work and their relations with customers. The empirical contribution of this paper is evidence drawn from a study of how IS developers pursue requirements engineering and how they conceive its value. Factors we found to be important in this process include: the changing needs of the organization, the ways in which structured IS methods are enacted via experience and social competency, the formation of project groups, and finally engagement in interpersonal conflict and negotiations. Our conceptual contribution is theorization on the nature of developing requirements as a process of social learning.

Keywords:

Requirements engineering, Information systems development, Qualitative research, Social Construction of Technology, Social Learning

1. Requirements Engineering and Information Systems Development

Contemporary literature shows that certain problems related to developing Information Systems (IS) are no less acute now than in the 1980s. For instance, Avison and Fitzgerald (2003) find that IS development projects continue to fail despite increasing attention to method innovation. Thus, the aim of the research presented here is to better understand how requirements engineering is enacted. More specifically, we seek empirical and conceptual insight into the ways in which information systems developers ignore or "black box" the emergent character of requirements (thereby addressing why it is that practitioners are continuing to pursue paths that the research literature clearly indicates are inherently problematic).

Attempts to understand and guide the efforts needed to gather and manage requirements has coalesced into both an academic and professional field called requirements engineering (RE). The RE remit includes discovering, prioritizing, documenting, representing and maintaining a set of requirements for building a computer-based information system (Loucopoulos and Karakostas 1995; Thayer and Dorfman 1990; Zave 1995; Nusibeh and Esterbrook 2000). Moreover, RE is concerned with analyzing and documenting requirements, by processes including needs analysis, requirements analysis and requirements specification (Sommerville and Sawyer 1997; Hanisch and Corbitt 2007).

In this paper we argue that RE's socially-constructed nature and the inherent conflicts among multiple users' needs, are either incompletely addressed or – worse – intentionally 'black-boxed' by professional IS developers. We note that this black-boxing reduces the potential value of RE, turning it into a rather benign process of documenting rather than the intended pursuit of value. Building on evidence drawn from a field study of requirements gathering practice, we consider theoretical aspects of requirements as a social learning activity, based in constructivist, rather than reductionist, approaches.

Our interest is rooted in Brook's (1987) poignant reminder that there is no silver bullet in developing software. While this is seemingly well-known, there continues to be ongoing quests by members of the IS development community to discover such ideal methods, and requirements engineering is one of the approaches for which there are great (possibly excessive) expectations.

To make our case, the paper is developed in five more sections. In section two we develop our conceptual position, and present related research on IS development, requirements engineering and requirements management. In section three we present the social construction of technology (SCOT) approach as our theoretical lens, and in section four we present the details of our research approach and our results.

2. Requirements Engineering and Information System Development

In this section we first review the RE and IS literature, since together these streams provide the grounding for our investigation of RE practices. We then introduce the two components of RE examined in this study: (1) RE in the context of heterogeneous environments and (2) RE as a series of negotiations. So, we also review here the theoretical and empirical work related to these two components.

2.1 Requirements Engineering and Requirements Management

There is longstanding interest among RE scholars in the social aspects of requirements (e.g. Malcolm, 2001). For example, Wastells (1996) argued that methods are used both because IS development practices are inherently complex and to satisfy developers' needs to appear professional (see also Avgerou and Cornford 1993). The problems associated with the design and uses of IS, particularly the difficulties that arise when client organizations' requirements are regarded as fixed lists of specifications, rather than dynamic, emergent sets of needs with social dimensions, have also long been known (e.g., Paul, 1993).

While these concerns have been raised by some in the scholarly community, they appear to have had less impact on RE in practice. In particular, evidence suggests that developers tend to "black box" or avoid addressing issues associated with the emergent character of requirements. Fifteen years ago Loucopoulos and Karkostas (1995: 19-20) noted that most contemporary software development methods "focus on the deliverables of the process rather than on the process itself." More recent empirical reports suggest that the negotiation process among developers and stakeholders remains a crucial and under-studied activity in developing requirements (e.g., Rooksby, Sommerville, and Pidd, 2006; Charkraborty, Sarkar and Sarkar, 2010). This is particularly interesting since current theories of RE do not take sufficient account of local context issues, and too often ignore the difficulties that socially marginalized people face in working in contemporary organizations (e.g., Ackerman, 2001; Nusibeh and Esterbrook, 2000). The functional rationalism that underpins the one-solution-fits-all paradigm has recently come under significant pressure from systems and engineering theorists who argue that RE is a complex and heterogeneous process (Bergman et al, 2001).

Within the functional-rationalist framework, at least three assumptions are not made explicit (or, worse, disappear into the background). First, requirement collection is based on the assumption that an organization remains unchanged sufficiently long for a complete list of required specifications to be gathered and a system developed. A second assumption is that users are expected to both understand and be able to communicate their present and future needs in clear and ordered ways (Truex et al., 1999). A third assumption is that the main actors within an organization have congruent ideas about the organization's aims and objectives (Bergman et al. 2002a).

There is clear evidence that these assumptions have tenuous validity. For example, Truex et al. (1999) describe how IS developed within the framework of these assumptions are likely to fail to meet commissioning organizations' needs. Contemporary organizations must be flexible and adjust rapidly to dynamic situations, hence their IS must be similarly flexible. The contemporary organization's reality is not static (and may never have been) so it is at best unhelpful to model the specification process on a static environment. It must be possible to transform information systems that are developed for modern organizations continually and quickly. Thus, system requirements should be analyzed and reassessed constantly (Truex et al., 1999).

One reason for the steady rate of failure in IS development projects may be historical: most of the traditional development methods approach requirements management in an incomplete manner (e.g., Truex et. al, 1999; Bergman et al., 2001). The traditional premise – what Stewart

and Williams (2005) call the "design fallacy" – is that requirements are conceptual things that 'exist' in an organization, which the skilled IS developer can identify and meet. Truex, Stewart, Bergman and their respective colleagues all argue that requirements cannot be readily identified in an organization. Instead, requirements are incompletely sketched in the minds of various important organizational stakeholders. Such a view shifts the emphasis from a "gathering" or "harvesting" metaphor towards negotiating or learning. In this second view, the premise is that stakeholders possess understanding of the problems and needs for a future IS that is partially developed, possibly conflicting, and probably dispersed in various forms among many stakeholders.

The purposes of gathering requirements are to understand and depict the customer's needs and thus establish the IS's required properties. Requirements are often formulated according to particular specifications, which may include detailed specifications regarding the IS's functions, appearance, and performance. These specifications serve as the guide for subsequent development activities. In this model, requirements become the foundation upon which the project is based, hence the gathering and management of requirements are important, if not central, aspects of IS development. A central premise in software engineering, information systems development, and the design of software-intensive systems is that when requirements are collected in a way that faithfully represents the needs of the stakeholders, the likelihood of success is improved. Conversely, poorly developed requirements – ones that do not match user or stakeholder needs – are likely to derail the project and make the resulting information system significantly less valuable.

Moreover, the traditional view of IS development as a technical or engineering activity is increasingly seen as incomplete (e.g., Sawyer, 2001a; Madsen, Kautz and Vidgen, 2006; Howcroft and Light, 2010). Truex et al. (1999) regard requirements management as being, to a large extent, a process of compromise to address prioritized changes that occur within an organization and the resources at one's disposal. In order to develop a system that is suitable for modern organizations, one should welcome a healthy degree of conflict between the users regarding their system. If these conflicts can be brought to the surface then changes to improve the system can be more rationally discussed and identified (Truex et al., 1999). The role of conflict management within system development has been examined to a certain extent, for example within the Scandinavian school of IS development (Ehn, 1988, 1993). However, it has been rarely applied in practical system development, and Bergman et al. (2002a) argue that there is a need to further refine (and implement) its use in this context.

2.2 Requirements in and from a heterogeneous environment

Moving away from static and consensus-driven views of an organization's goals and needs – moving beyond the "design fallacy" – makes requirements gathering an even more complex activity. In noting this, the issue is not one of moving from a well-defined space to a less-well defined space. Rather, the issue is to make clear that requirement gathering is done in a complex social milieu in which current models of this effort do not adequately represent the situation.

According to the dynamic model of organizing, identifying, understanding and representing an organization's problems, its needs should be seen as evolutionary, ongoing and constantly

changing (e.g., Hansen, Berente and Lyytinen, 2008; Holmström et al., 2010; Truex et. al., 1999). Hence, developers of an IS should be concerned not only with the system itself, but also with the larger socio-economic context in which the IS will be used (Bergman et al. 2002b). That is, requirements gatherers should be social actors, aware of the situated and institutionally-framed actions in which they and their colleagues participate (e.g., Lamb and Kling, 2003; Rowlands, 2008; Madsen, Kautz and Vidgen, 2006).

The premise of the socially-constructed view of requirements begins with the observation that in any reasonably-sized IS development project – any that involves more than a few users from more than one department – important actors will have different opinions about which requirements are most salient. It will also be unusual for all the members of an organization to actively participate in the definition of system requirements. Instead, a smaller number of members will be asked to, or assigned to, represent the organization's interests and influence the project. These delegates, whom we call the *participating* actors, will be those who, acting with the requirements engineers and other IS staff, formulate the requirements. The assumption is that participating actors are either agents in service of the important actors, or are themselves the important actors.

The relationship between importance and participation is fundamental to characterization of RE as social learning. It is through participation that requirements are developed and systems are constructed. The participating actors thus play defining roles in the requirements management process – whether or not they are the most knowledgeable or the most suitable representatives of the organization and its needs (Bergman et al. 2002b).

In larger IS projects that involve multiple departments, the participating actors are likely (if not certain) to have different opinions on how goals should be met and the direction in which the organization should proceed. Those who make the decisions during the course of the project clearly play a defining role when everyone is not in agreement. In an ideal situation, the optimal approach would be to collect all the relevant information about the situation and then let the main actors come together in order to make decisions. In practice it is difficult to gather the disparate, relevant, information related to the development of an IS, much less organize it in ways that readily allow decision-makers to make informed responses (e.g., Hansen, Berente and Lyytinen, 2009).

The conflicts that arise among participating actors are partly due to disagreements about which requirements should be prioritized and (hence) who should make decisions. Organizations rarely have the full complement of resources needed to take into account the needs of all the involved actors. This leads to situations where selections are continually being made in order to decide whose needs will be met. The omnipresent risk of being unable to complete an IS within the allotted time and budget often drives organizational decision-makers to decide if more resources will be allotted or requirements will be neglected. Negotiations are often necessary to (re-) determine whose interests will be fulfilled. Moreover, this can develop into a cycle of continuous evaluation and re-evaluation of resources and needs, leaving any decisions open to future discussion and renegotiation (Bergman et al. 2002a).

2.3 Negotiations and Conflict Resolution

Gathering, representing and prioritizing requirements involves engaging in negotiations. Consequently, addressing conflicts among participating actors is a common (if not inevitable), part of RE activity. The reality of this conflict is commonly noted (e.g., Hansen, Berente and Lyytinen, 2009), and some empirical information related to this issue has been published in both the software engineering and IS development literature (Guinan, 1988; Urquart, 1999; Boehm, 1981; Robinson and Volkov, 1998; Robey, 1994; Sawyer, 2001b). Bergman et al. (2002a) note that this literature highlights the importance of recognizing and addressing conflicts, but also reminds us that bringing forth and resolving conflict is not the only important factor; it is also important to examine the underlying structures to understand the source of these conflicts. Given this, many of today's IS development methods seem to focus too little on reducing conflicts or increasing the participating actors' understanding of one another's perceptions of IS plans.

In IS development projects a central goal is for participating actors to have an understanding of the system's requirements. The process of requirement specifications can help meet this goal, and the resulting requirements can act as a type of contract between IS developers and the organization commissioning the system. These requirements (and resulting specifications) are of great importance in that the developers can gain a common understanding of the main features and capabilities that the system is supposed to incorporate.

Too often, however, this understanding is lacking – even if a set of specifications has been formally articulated. This leads to a situation where the espoused requirements may have been met, but the system does not achieve its intended goals. A second issue associated with incomplete (or poorly developed) requirements is that participating actors may interpret the specifications in different ways; individual actors may interpret them as favorably as possible for themselves and ignore all conflicting interpretations (Bergman et al., 2002a; Zappavigna and Patrick, 2010). An effective negotiation process may arise through the main actors confronting each other and discussing their individual points of view in order to come to a mutual understanding about the requirements that are most necessary. If possible, these actors may then successfully compromise as long as one interest is not met by excessively damaging interests that are important to other actors. Certain conflicts and divisions are unlikely to be resolved, however, without the intervention of someone who has sufficient power to decide how resources will be allocated by the commissioning organization.

Taken together this line of reasoning suggests that understanding organizations as complex, dynamic and conflict-ridden must be accompanied with an appreciation that requirements are emergent, rather than static. Building on this, Howcroft and Light (2010) advance a convincing argument that this must be seen as a social construction. With all this, there seems to be a disjuncture between leading research on requirements engineering and much RE practice. The former is built on the assumptions that contemporary organizations are characterized by conflict rather than harmony, and that requirements are emergent rather than static and objective (Ghezzi & Nuseibeh, 1998; Rooksby, Sommerville and Pidd, 2006). In the latter, in contrast, the understanding of organizational action is largely in line with the view that organizational life is harmonious, and requirements are seen as being objective rather than emergent.

3. Social Construction of Technology

We build here on the social construction of technology (SCOT) approach as outlined originally by Pinch and Bijker (1984). According to Bijker and colleagues (Bijker et al., 1987; Bijker, 1995; Bijker & Law, 1992), technological artifacts are open to sociological analysis that considers not only their design and use, but also their technical content. The SCOT approach upholds the principle of symmetry between social and technological elements, avoiding any reference to material characteristics of a technology in its analyses. Technological change is explained by reference to social practices, particularly processes of interpretation, negotiation, and closure by different actors and social groups. Technology is understood as a social construction that can only be explained by focusing on the social processes that have constructed it. Given the socially-constructed nature of Information Systems' requirements, SCOT provides an analytical framework for pursuing a deeper understanding of their construction (e.g, Howcroft and Light, 2010).

The RE effort reflects features common to constructivist studies of technology. Most importantly, social constructivism includes a conception of technological development as a contingent process. Accordingly, technological change cannot be analyzed as following a fixed, unidirectional path, and cannot be explained by reference to some inner technological logic. Rather, technological change is seen as being best explained by reference to a number of technological controversies, disagreements, and difficulties that involve different actors. These actors or groups engage in strategies to win concessions from their opponents and shape technology according to their own aims. Hence, the social constructivist approach places society (and social action) as the driver of technological change and regards artifacts as being developed according to social interests. This leads to the concept of "interpretive flexibility", i.e. that the value, purpose and meaning of an object, like technology, is contingent on interpretation.

Interpretation is, in turn, developed within one, or several, interpretive frameworks as various groups of actors come to grips with (interpret) the object under consideration. Thus, the properties of objects are not inherent in the objects themselves, but are products of social construction. For example, Woolgar's (1991) constructivist view holds that the capacity of technologies remains essentially indeterminate both before their sale (during their conception, design, and development) and later, while they are in use.

Borrowing from Bijker et al. (1987), Orlikowski and Gash (1994) developed ideas about the vital roles technological frames play in this process. Bijker et al. (1987) refer to technological frames as the ways in which relevant social groups attribute meanings to an artifact. A technological frame is the repository of knowledge, cultural values, goals, practices and exemplary artifacts shared by a social group, which structures their understanding of objects and processes in technical innovation, and their subsequent actions. In analyzing a particular process of technological innovation, the analyst can choose to include not only the technological frame(s) of social groups that have been influential in determining the outcome of the process, but also those, and changes therein, of groups whose voices have not been heard.

In this socially-constructed view, requirements are not things that exist, or have been clearly articulated, ready to gather, in an organization. Rather, requirements are generated through negotiation and conflict resolution among the different actors involved in the project. Such a view suggests that seeing requirements gathering as a negotiation will increase the chances that

requirements will be developed that meet the commissioning organization's needs and (hence) that the IS development project will lead to a useful system. This, in turn, requires the IS developer to be skilled in engaging in and managing such negotiation processes; a conclusion that the academic literature in information systems and software engineering has been consistently articulating since the 1980s (Guinan, 1988; Urquhart, 1999; Mathiassen & Nielsen, 2000; Bergman et al., 2002a).

The socially-constructed view of requirements builds on the literature of science and technology studies (STS), which regard a technology as the result of social actions, something that arises from debate, discourse and discussion. Social-constructionist research on technology began with studies by the Tavistock Institute in the 1950s (e.g., Berger and Luckman, 1966; Mumford, 2003). This approach has been advanced as a means to understand the development of software (Howcroft and Light 2010). Strangely, this robust literature has been rarely cited in the RE literature, despite the strong advocacy for just such a representation by RE scholars (e.g., Rooksby, Sommerville and Pidd, 2006). There is, however, a stream of literature in method enactment in IS development (e,g, Rowlands, 2008; Madsen, Kautz and Vidgen, 2006). Our work can be seen as contributing more evidence to support the arguments made in the method enactment literature that IS approaches are social constructions. Since our work focuses on RE, and not IS development more generally, we also aim to contribute more specific conceptual insights.

The social-constructionist approach to understanding the development of technologies focuses on the social actions relative to technology decisions and is grounded in detailed empirical studies of the work of scientists and engineers. The scholarly traditions of STS build from the conceptual premise of historians and sociologists that technologies represent an outcome of a political process framed by the interests, resources, negotiations and discussions among central actors. These arrangements result in the social construction of the means and forms of a technological artifact (e.g., Grint & Woolgar, 1997). Such a framing is ideally suited for better understanding the socially constructed nature of an information system's requirements as it focuses attention on: (1) the importance of negotiation among stakeholders (users and other organizational stakeholders like decision-makers) and developers; and (2) differences in social power (between the relevant social groups).

4. Research Effort

In this section we build on our field research and secondary sources to explore the ways in which IS developers engage in RE and how they conceive its value. We develop this as an exploratory study, focusing on developing an understanding of how RE is enacted, which we then apply to reflect and theorize (Weick, 1995) about the nature of RE and associated processes. In this way we extend current conceptualization of RE, a process that Vaughan (1992) calls theory elaboration. In addition to better explicating the basis, premise and elements of RE, we also infer implications for professional practices in RE and IS development.

4.1 Research Approach

This study builds from an interpretive epistemology (Klein and Myers 1999; Walsham 1993). Interpretive research "attempt(s) to understand phenomena through assessing the meanings that people assign to them" (Orlikowski and Baroudi 1991: 5). This ambition is central to our study,

which relies on qualitative interviews with developers and applies qualitative analysis to interpret the RE practices. Interpretive researchers often use an underlying theory for both framing and analyzing research data (Holmström, 2005; Walsham 1993), and in this case we draw upon SCOT to guide the data collection, analysis and reporting. In an interpretive case study, the researchers become integral to the research method and are obliged to acknowledge their influence over it (Golden-Biddle and Locke 1993; Klein and Myers 1999; Mason 1996). In our case, the first author conducted the empirical study, and the second author collaborated in interpreting the findings and writing the manuscript. Since it was anticipated that developers' experiences might differ, depending upon the numbers and types of projects they had engaged in, participants were deliberately selected to ensure variation with respect to age and experience.

The aim of this research is to better understand how requirements engineering practices – the gathering and representation of stakeholder's needs for an information system – are enacted. To meet this aim we interviewed working professionals for whom requirements gathering is a major part of their work. For pragmatic purposes, snow-ball sampling was used to select interviewees from the local business community. Given the focus on requirements gathering, we judged that the exploratory work would not be compromised by the sampling approach.

4.2 Data Collection and Analysis

In this section we focus first on the selection and collection of data, then on analysis of the data.

4.2.1. Data Selection and Collection

We conducted 26 interviews with IS developers from five IS developing companies: Martinsson, Sogeti, Tieto Enator, Umeå datakonsulter, and WM-Data, all of which are relatively small IT consultancy firms (though many are parts of larger organizations) located in Umeå, Sweden. All of the interviewed consultants had between two and 16 years of professional experience of requirements engineering and IS development.

In the 26 interviews the data were gathered, analyzed and discussed with the participants within an interpretivist paradigm. This interpretivist approach, with its goal of revealing the participant's views of reality, allowed the underlying reasons for actions in requirements engineering practice to be elicited. As noted above, our 26 interviewers, and their five companies, were not randomly selected, nor can we argue that they reflect a representative sample across all important characteristics. For these (and perhaps other more idiosyncratic) reasons, allegations of bias are commonly raised about interpretive research. Bias can refer to the way in which a particular point of view may affect the way one observes and interprets a specific situation, or it may refer to a systematic error in the research process. In particular, opportunities for the researcher to influence findings occur in any case study research where relationships among researchers and participants can be frequent and close. In addition, participants may not report fully to the researcher if they perceive that the information given may display them in a negative light.

Aware of these concerns, we agreed with the companies involved in the study at the outset that the consultants would be referred to by pseudonyms, and thus not attached to a specific activity and/or argument. Further, all participants were volunteers who were aware that we had made arrangements to support their confidentiality. These steps provided assurance to the interviewees

that they could be open and frank about their actual work practices. Thus, we refer to the consultants and their organizations by pseudonyms. The consultants are referred to as A-Z in order to maintain their anonymity while also making it possible to trace their voices in the results section.

The interviews, which were audio recorded and transcribed, were between 50 and 85 minutes long and conducted by the first author. With respect to use of existing theoretical constructs to guide theory-building research, we worked within the explicit conceptual framework provided by SCOT. Such a framework becomes a "researcher's first cut at making some explicit theoretical statements" (Miles & Huberman, 1994, p. 91). In the context of our study, the use of SCOT as an orienting framework helped us make sense of occurrences and ensured that important issues were not overlooked. We return to this in section 6.

We have used SCOT to understand the ways in which IS requirements are socially constructed and why IS developers too often choose to ignore, and thus effectively black-box, the complexities of gathering requirements in order to simplify both the difficulties of their work and their relations with customers. In so doing we have approached RE practice as a contingent process. In such a view, technological change is seen as being best explained by reference to a number of technological controversies, disagreements, and difficulties that involve different actors. Overall, our approach relates to the principle of interpretive research (Klein & Myers, 1999), about abstractions and generalizations of data through the use of theories. We structured the data collection and analysis following the key tenents in SCOT as applied to RE practices: identifying key encounters related to the RE process? What actors were involved and what were their interests? What negotiations took place? What were the effects of these negotiations? What role did requirements play in these negotions?

However, while early identification of possible constructs allows them to be explicitly used in the interview context (Eisenhardt, 1989), it is equally important to recognize that the identification of constructs is tentative in theory-building research. We found this to be true as new factors were found during data collection that needed to be added to the analysis. An important issue to resolve for reaching closure is when to stop conducting interviews. Ideally, researchers should stop adding cases when theoretical saturation is reached (Eisenhardt, 1989). Theoretical saturation is the point at which incremental learning becomes minimal because the researchers are observing phenomena seen before (Glaser & Strauss, 1967). In practice, we decided to stop conducting interviews when new interviews did not add to what we already knew.

In addition to the interviews, we collected more than 200 documents relevant to the present study, including organizational charts, annual reports, special reports and/or administrative documents (including documents relating to requirements). These documents serve to situate both the interviewees and their responses.

4.2.2 Analysis

To do the analysis, the interviews were separated question by question and categorized in order to break them down into smaller segments (Miles and Huberman, 1994). As part of this process, we also drew on the collected documents. We built on and theorized around the concepts we

drew from the literature review. The basic principle of SCOT used to frame the analysis is the *interpretive flexibility* of a potential requirement. This interpretive flexibility is at the root of how technological choices are socially negotiated. Thus, we sought evidence of interpretation and negotiation among members of the two relevant social groups: participating actors who are members of commissioning organizations and IS developers.

Interpretation and negotiation lead to conflicts among relevant social groups and these conflicts in turn are illustrative of choice points. These conflicts are resolved and the choices are resolved through the process of stabilization and closure (when consensus has been reached). Closure can be achieved in many ways. Some of the most common are rhetorical (where the issue is finessed through negotiation) and redefinition (in which the focus is shifted from the conflict problem to an aspect for which there is agreement). However, the focus of this paper is not on the processes whereby closure is achieved, but on the presence of negotiation around requirements. In this respect, SCOT is used here to frame the identification of conflicts/choice points among the two relevant social groups.

To do this, we followed the recommended procedures for qualitative research and grounded theory (Eisenhardt 1989; Miles and Huberman 1994; Strauss and Corbin 1990). Specifically, we adopted the "Straussian" approach toward grounded theory, which explicitly permits researchers' exposure to related literature to guide the data analysis process (Strauss and Corbin 1994). We followed an iterative coding process that involved identifying the emerging concepts, examining empirical evidence for support, consolidating similar concepts to create more refined ideas, and collecting more data until theoretical saturation was reached.

Data analysis was based on the three types of coding suggested by Strauss and Corbin (1990): open, axial, and selective. The data analysis process was facilitated by using Atlas.ti software, which was designed for managing complex data and supporting qualitative analysis. We first identified 55 codes, each supported by two or more text segments, during the open coding stage. During this stage, we drew on SCOT for guidance, and focused on identifying the nature of interactions among the two relevant social groups.

During the axial coding stage we consolidated codes that were conceptually similar. Finally, during the selective coding we strove to integrate the identified codes and formulate a storyline that offered a coherent and insightful account of the RE practices. This third stage was again guided in large part by SCOT concepts as we were looking for evidence of social constructions and the social definitions of meanings for what was to be a requirement for the IS. We provide an example of our selective coding effort in Appendix 1.

Following an initial coding effort, additional data collection and coding efforts were made until theoretical saturation was reached. To verify the plausibility of identified concepts, we further reviewed the dataset for corroboratory evidence and used data from different sources and methods for results triangulation to ensure the validity of our findings (Miles and Huberman 1994). The results provide insights explaining the processes associated with RE practices, in particular with how RE – the gathering and representation of stakeholder's needs for an information system – is enacted.

4.3 Findings

We report four findings from our study of how IS developers enact RE and conceive its value. We find that RE reflects: (1) the changing needs of the organization, (2) the way in which structured IS methods are enacted via experience and social competency, (3) the formation of project groups, and (4) the resolution of conflicts and negotiations.

4.3.1 The Changing Needs of the Organization

Respondents note that participants in contemporary IS development projects generally presume that development will lead to a complete and finished IS. They further note that the need for adjustment is driven by an implicit conceptualization of organizational work as static: the IS is designed to match what was specified, not support what is now needed. For example, developer "D" notes:

It worked for a while [after the system went into operation] but then after a while they change your activities, of course, "we will do this too". We then have to go in and adjust the system. […]. We are always doing this [changes]. Always and never-ending. (D)

This is not a revelatory finding, just further evidence that professionals in practice seem to adopt an (empirically and conceptually) unsupportable worldview of requirements (and the organizations that develop them) as temporally immutable. Some developers assign their static worldview of an IS to those using the IS:

I believe that they [the customers] think that the system will cost a certain amount and will look like this and this. And on this date we say goodbye to the consultants and, hopefully, will never see them again. (A)

Other developers note that users do not have a fixed-point view of requirements or of the organization:

I believe that today most are aware that you have to budget 20-25% of the development's costs for the maintenance of the system. (G)

Developers further noted that requirements are constantly being added or changed during the IS development. Respondents viewed changing requirements as commonplace, obvious, but problematic:

Today the goals and demands are over there (points in one direction). Later, when we have finished the project, they are over there (points in another direction). So you have to constantly keep your eye on the moving goal. (R)

Developers note there are many reasons why requirements change. It may be that the developers did not properly understand their customer's work or working environment. It may be that the IS customer believed that certain things would be included, but did not clearly express these demands from the beginning. It may be that requirements were not properly prioritized, or, worse, that they were forgotten:

He [a customer] was the type that said, "Yes, but you should have known that". He assumed that we knew things which we could not have known and then you could have done something that he had asked for and sent it to him. Then he said "Yes, but this is wrong, of course. This should also be included." "Yes, but we did not know that." "Yes, but you should have known." Then it became a whole lot of back and forth and redoing and adding to. (P)

Developers called attention to the importance of customers being good at prioritizing their needs:

It is assumed that those who create the system are those who need to know what is needed from it, but it is important that the customer also knows what they want. Then they understand that they must present as much information as possible which could have an effect on the system. This is partly our task, and partly the task of the customer. (T)

The focus on establishing requirements seems to provide little help for either developers or the developer's stakeholders to grapple with the well-known problems of dealing with changing requirements.

4.3.2 Structured IS Methods are Enacted via Experience and Social Competency

Respondents reported that they used some form of the prescribed structured method to gather requirements. While the particulars of the structured method vary from firm to firm, each method has a phase or activity specifically focused on requirements gathering, and the respondents were aware of these methods. Respondents further noted that they do not follow these methods as rules, but more as guides. That is, methods are not followed, they are enacted (as other authors have shown, e.g. Stolterman, 1992; Mathiassen and Stage, 1992). Method enactment is an area of active scholarly inquiry (e.g., Rowlands, 2008). We note here only that our work is both supportive of that research and that our findings reflect the premises of incompleteness and resulting social negotiation on which method enactment literature rests.

Beyond the rules imbued in structured methods, developers noted that experience, working knowledge of the field, and social competency were all deployed in enactment of the methods:

You never walk a straight line throughout an entire project, there are tons of potholes. Methods can, of course, help you avoid falling into these holes. […] The other competencies are used when the unforeseen happens. (M)

Through working and spending time with personnel, and by examining the routines of the company, the IS developer can increase his/her understanding about how different demands affect the system and the commissioning organization. Looking at problems through the viewpoint of the different personnel can also help the developer to create his/her own view of the system.

4.3.3 Forming Project Groups

Respondents noted that common practice was to work in project groups during IS development. They further noted that IS development group members had little involvement in deciding who would be included in the group. This responsibility lay, instead, with the commissioning organization. The reason that certain people were chosen over others, according to respondents,

was their interest in IT or commitment to change. Respondents espoused the belief that customers choose project members who show positive attitudes and ability to work well with others, since no one wishes to begin a project with problems. That is, enthusiasm was seen as a more important criterion than technical competence, organizational relevance/insight, or domain knowledge:

I wonder if it was the Managing Director who had ideas because the ones that were on the project were those who were the most open to change. Of the factory managers, those who were chosen were those most concerned with change. On the sales side of things, they chose a salesperson who… wanted change. (A)

Respondents also noted that when project members were chosen by the customers, relevant actors were not always included. For example, sometimes members representing only one or a few departments may be chosen, and other departments' needs are often not known, forgotten or given lower priority. Problems may also arise when those who are less interested in the process are not allowed to express views regarding requirements of the system to be developed:

There is also the problem that the users who are most interested in the project are chosen by the customer to be included in something like this. "Yes, now we need to set aside 10 non-users to talk with the system developers." Yes, but who do we choose, then? Yeah, those who are interested, which is actually kind of wrong because those who are not interested don't get to be heard. This can cause internal conflicts. (D)

Respondents noted that too often the commissioning organization does not take responsibility to assign the resources needed and the 'right' people for the project group. This seems to shift the responsibility for overcoming (or recovering from) problems related to inadequate team selection onto the members of the team.

4.3.4. Conflicts and Negotiations

All respondents reported that it was helpful for conflicts to arise during development projects and that this was much better than having conflicts surface after the projects had ended. Conflict, while not always pleasant, was seen as the best means to provide all actors the opportunity to learn and consider multiple sides of an issue:

When you get into a conflict, if you can call it a conflict – you have to examine your own reflections about the problem instead of continuing along on your own line of thought; which is probably easiest and most obvious to yourself. That is when you begin to wrestle with the problem, twist and change your argument. Sometimes it is then that you trip yourself up and then you have to discuss issues more, and then you realize that things are going to hell. (J)

In these periods of conflict, developers note there is difference between how leaders and project workers engage. However respondents, for the most part, consider customers to be a relatively homogenous group:

You have, actually, two different types of priorities. Partly, you have the customers' priorities, what is most important and what is less important. But then you also have our internal priorities … how difficult different things are to do and what is dependent on what. (R)

Paradoxically, developers observe that conflicts often reflect ongoing issues within the commissioning organization that are revealed during requirements disputes:

I believe that some of the worst fights are internal, when we are not with them. I believe that things can get pretty hot then. (A)

They [the conflicts within the commissioning organization] I believe are not really taken up in front of us, more internally. If we have a board meeting, they take up problems that they have with us and we take up problems that we have with them. (P)

Most developers believe that it is the customer's responsibility to solve internal problems and disagreements. It is not the job of the IS developer to convince opponents or to attempt to persuade parties within the organization to compromise:

We will be good at telling them how to do things, but it is the customer who must say what and why. It is there that I believe we have to shut our mouths and open our ears. (A)

Others argue that they should participate in resolving conflicts, in case it is otherwise impossible to continue before resolving them, even if it is not the main task at hand. Regardless of their position on resolving intra-customer conflict, respondents emphasized that discussion among the interested parties is the most important factor for creating a mutual understanding. For example, it is important to thoroughly discuss requirement specifications in order to avoid different actors interpreting the specifications in different ways:

You have to sit down and discuss […] you have to have meetings often and make sure you are in agreement often, because if you work too long on the wrong concept, you are just throwing money into the sea. (P)

Developers also observe that compromise is critical. One respondent went on to note:

It becomes cheapest and easiest [to first try to compromise between the two departments' different demands]. Sometimes you can get a bit irritated. Things that you believe are the basis, but this goes both ways and no one agrees. At that point, I usually develop a compromise. Then I throw it out there – then they grumble a bit, but you pretend not to hear it. Sometimes they agree. Otherwise you just do it. (I)

One respondent noted that there is a certain limit to how much you can discuss:

Sometimes you get permission to be a little heavy-handed. We cannot continue to swing back and forth indefinitely. We have to begin from those demands which are specified during the time when you specify demands. At some point you have to say that you are now finished. We cannot

change anything now […] If it becomes too much, stop and discuss with the people and maybe things get delayed. It is a balance thing. (G)

While all respondents reported that it was useful for conflict to arise during development projects, since it provides all actors the opportunity to learn, it is not necessary to find compromises for all conflicts. In fact, a few respondents stated that avoiding conflict can be important, in terms of minimizing delays and reducing the cognitive load that the developers face in turning requirements into a working IS. One developer even went so far as to state that the optimal situation is when you do not even have to ask the customer anything at all.

5. Discussion

Here we build on the findings from the previous section to discuss two outcomes from this work. We look first at organizational and team-level influences which serve to frame the work worlds of IS developers relative to enacting requirements gathering. Then, we examine the set of simplifying assumptions about the stakeholder's work that IS developers use as a means to support a working fiction of homogeneous work environments.

5.1 Organizational and Team-Level Influence on Developer's Work

The developers are aware that an IS development project does not always lead to a finished system, even if the project's resources and scope are finite. The developers we interviewed understand that IS development must be both envisaged and undertaken as an ongoing and indeterminate activity, one that accounts for the changing nature of both the organization and its environment, even though the project-based structure imposes an artificial completion. This supports argument advance by Howcroft and Light (2010) regarding the socially-constructed nature of softare. Moreover, our respondents made clear that during any IS project development, demands are added and existing needs are often changed, supporting the work of Chakraborty, Sarkar and Sarkar, (2010).

Respondents believed that additions and changes to requirements show that, to a large extent, the participating actors are either unsure about their needs or do not clearly express what they require from the IS being developed. Thus, representing and accounting for change is expected, but the ambiguity and incompleteness of requirements are problems that arise from the participating actors (users) rather than the developers. This agency displacement – attributing ambiguity and change not to the IS developers' actions but to the participating actors – reinforces the findings regarding the differential locus of power between users and developers that Kling and Iacono (1984a) identified.

One factor that contributes to this agency displacement is the respondent's reliance on following structured design methods; a deontological approach, as noted by Wood-Harper, Corder and Watson (1996). Respondents used a range of IS development methods to manage demands, so it appears that this use of methods as an instrument of disassociation is not tied to a particular method, but to their uses.

Respondents clearly indicated that domain experience and knowledge about a customer organization's activities are important for IS developers since such experience and knowledge provide a means to frame and often to prioritize discussions with users. Building on this, we note

that developers see discussion as the best way to create a mutual understanding with their participating actors and to learn more about possible system alternatives. We know from the literature that problems associated with requirement specifications often occur when different participating actors either do not understand or choose to interpret things to their own advantage (Bergmann, et al., 2002b; Guinan, 1988). It appears that meeting, discussing and negotiating are seen by IS developers as highly valuable (if not essential). Respondents also note the importance of a system developer being socially competent, since social competency is the main mechanism used to successfully interact with the many different types of people in a customer organization.

These findings reinforce the notion that social competency, often gained from experience, and showcased through ongoing discussions with clients (participating actors), are important parts of the sense-making process on which RE relies. This suggests to us that IS developers use sense-making skills in order to adapt methods to specific situations. By sense-making we mean here the process of taking in complex equivocal information from the environment, attempting to extract meaning from it, and applying what was learned in the future (Weick 1993).

The importance of sense-making is not reflected in many of the IS (or RE) methods used by the developers in this study. It appears that IS developers are conscious of client organizations' demands for constant change. However, while a majority of the respondents see conflicts as having advantages as stimulators of reflection and consideration, this does not mean that they like or seek conflict among users or between participating actors and developers. They note only that when it occurs, conflict can be productive.

5.2 The Wish for a Homogeneous Environment

Respondents regard differences in interpretation of participating actors' needs and concerns as arising collectively from the customers. This leads to developers seeing differences as occurring mainly between themselves and these participating actors – homogenizing both the differences among IS developers and tremendously simplifying the variety of organizational perspectives which they encounter. Both are problematic. However, the desire among IS developers to perceive the staff of a client organization as a homogeneous group, all of whom are striving to meet the same objectives, is puzzling (and may be deeply unhelpful) because it directly conflicts with their experience when working with participating actors. It may be that this is, unwittingly, abetted by other actions. For example, and as noted earlier and discussed below, typically only a small group of potential users in the customer organization are engaged in RE, which is likely to reduce the range of perspectives on use and need.

However, regarding the staff of customer organizations as a homogeneous group simplifies the approach to RE (essentially by ignoring the complexity and messiness of needs of some users during a negotiation process). Essentially, IS developers use RE as a means to downplay the importance -- and messy details -- of negotiating. However, it is unlikely that an IS developer will effectively manage demands based on a negotiation perspective if he or she is not able to understand the range of wishes co-existing within the commissioning organization. Even if the IS developers agree that there are different opinions, it seems that the wish to simplify prevails over the reality of complexity.

This contradictory simplification seems to be further magnified by the process of choosing participating actors. Findings show that people chosen to engage in RE work by the commissioning organization are often selected by management or are those who take initiative for the system. The chosen project members are often people naturally disposed to a pattern of change, often share similar opinions about the aims of the project, and may even be self-selecting. In other words, these main actors may be somewhat more homogeneous and perhaps not representative of the commissioning organization's broader and possibly more divergent views.

Creating a more diverse group of participating actors can be difficult since those who choose the members naturally do so in an attempt to shape the outcome of the system. It can be seen as less inviting to include members of the customer organization who, from the outset, have different opinions about how the system should work. There may also be perceptions that the project will run less smoothly and be less easy to control if the participating actors do not agree about most aims from the beginning.

If the participating actors who form the customer organization's project group is both homogenous and aligned in terms of power perspectives (such as reflecting a managerial, and not primary user, orientation to the IS being designed), alternative opinions are not likely to be raised in the negotiation process. The desire to reduce the heterogeneous nature of the organization and its member's needs, as we have observed from our respondents, reduces the prerequisite conditions for engaging in negotiations to resolve conflicts among those with differing views of the IS and to capture the evolving patterns (and idiosyncrasies) imbued in the organization.

6. Theorizing Requirements Engineering as a Social Construction

Our data show that there is a certain awareness among IS developers that different opinions about the nature and needs of new IS/requirements exist within commissioning organizations. However, a desire to simplify this complexity and (seemingly) disregard these complications too often prevails amongst the developers. There are at least two reasons for this. First, the selection of participating actors is too often focused on minimizing heterogeneity – albeit for very understandable reasons. Second, the requirement gathering and representation methods are more focused on establishing a set of actionable needs than on identifying the full range of needs – again, for very understandable reasons.

In focusing on these organizationally-desired goals, the IS developers do not attend to the full diversity of needs or accommodate the certainty of changes over time. Both points illustrate not only the developer's desires to simplify the complexities of RE practices, but also that requirements are not things that exist or are clearly articulated in a form that can be readily gathered in an organization. Rather, requirements are generated through processes of negotiation and conflict resolution among the actors involved in the project.

The conceptual basis of SCOT provides us with an analytical lens that is sensitive to these complexities, explaining technological change by reference to processes of interpretation, negotiation, and closure by different actors and social groups. RE practices, from such a perspective, are thus understood as social constructions that can only be explained by focusing

on the social processes that have constructed them. Understanding RE practices as socially constructed through processes of negotiation will increase the chances of developing requirements that meet the client organization's needs and of IS development projects delivering useful IS.

This second point is further highlighted by the awareness of IS developers that changes over time are both likely and important. Nevertheless, these same developers draw on methods that do not engage with changes in needs over time. We further note that IS developers are keenly aware that their social competency is both the vehicle used to enact methods and the means by which they engage participating actors.

These findings clearly show that IS development methods, particularly requirement development methodology, must move beyond thinking of participating actors as homogenous groups with similar needs to more carefully address the social construction of requirements. To improve IS development, requirements elicitation activities must create both awareness and an understanding of the possibilities that opposing arguments can be beneficial (Urquhart, 1999). This is not news, since both those who advocate participatory design, and those who advocate the social design (and particularly power) perspective of IS, have long drawn similar conclusions (e.g., Ehn, 1993; Kling and Iacono, 1984a; 1984b; Markus, 1983). These issues are, however, highly relevant in relation to today's IS practices. For instance, when looking at the recent development towards agile methods we can note how the issues raised in this paper are highly relevant in relation to recent developments in agile methods. With agile methods being routinized and infused in the adopting organizations, one of the most pressing issues is the need to develop a better understanding of negotiations, and how to balance negotiation processes building on the competence bases found in both the adopting organization and among the developers. This issue is too often overlooked in today's agile practices (Abrahamsson et al., 2009).

The RE community appears to have continued to focus on the artifact, and thus failed to exploit the potential value of social constructivist approaches, which are highlighted by our findings. There seems to be a clear need for more of a social process perspective, and awareness of the need for multiple perspectives of IS requirements. We find that developers see negotiation, conflict and reflection as critical means to gather insights, but they readily take opportunities to simplify their work by opting for limited and too-often heterogeneous views of IS needs. In this manner their actions illustrate the constructivist view that technological change is best explained by reference to a number of technological controversies, disagreements and difficulties that involve different actor groups. In this case the developers engage in strategies to shape RE practices according to their own plans by means of simplification.

We note that the evidence suggests that this simplification, this black-boxing of difference into a homogeneous view, is neither successful RE nor useful as an IS design approach. We argue that in order to understand and develop requirements for systems that will be useful and capable of evolving properly over time, IS developers must engage more fully with heterogeneous environments. This requires developers to engage more proactively in acquiring diverse perspectives and to see the gathering and representing requirements as a process of social learning (e.g., Stewart and Williams, 2005; Suchman, 2002).

To do this well will require two changes. First, IS developers need to have better analytical approaches to help them evaluate and prioritize domain and systems challenges (e.g., Ackerman, 2000). Second, both developers and participating actors need to develop social skills for successfully engaging in negotiation, conflict management and the other social competencies through which requirements (and methods to gain them) are developed. The comforting but myopic focus on the artifact (a set of requirements) promotes underestimation of the importance of the dynamics of change and the need to reflect dynamism and emergence (e.g., Truex et al, 1999). This demand for increased social skills does not, however, mean that we are suggesting that it would be ok to diminish an IS developer's technological or analytical skills. Quite the contrary: we are articulating that an IS developer must have both, and that there seems to be evidence that the social skills are not yet emphasized in ways that translate directly to practice.

It should be noted that a shift towards acknowledging the socially-constructed nature of requirement raises its own problems. One is that requirements development is often tied up with internal conflicts among the user community within the commissioning organization. It may be that in such cases the IS developers serve more as focusing agents, raising issues for the commissioning organization's potential user community to address. Project, conflict and role demands are not easy to address (and disentangle) in such politically charged environments, and this is likely to pose great challenges for project leaders (e.g., Rowlands, 2008). A second is that greater emphasis must be placed on developing shared understanding (and if possible some consensus) of the various perspectives on IS requirements – the specific goal of social learning approaches (cf. Stewart and Williams, 2005; Suchman, 2002). Since it is often during these conflict situations that alternative solutions and new ideas can arise, IS developers can add valuable information when they participate in these conflicts, but may not be able to sort out what they have learned.

There are limits on how much the different actors can discuss: eventually certain choices must be made in order to begin the development of the IS. Also, attempting to access and understand too many different opinions, and then resolve the resulting conflicts, can be time-consuming (perhaps incommensurately). One reality is that neither the commissioning organization nor the IS development firm have unlimited time or resources. We see the challenge partly as encouraging negotiation, but at the same time not letting the negotiations take over. Clearly, RE is a balancing act, but the balance cannot be achieved by simply denying the centrality of negotiation in defining requirements.

More broadly, we have argued through this paper for the need to shift towards adopting a more complex, negotiated, approach to developing requirements. It is our argument that such a shift calls for a rethinking of current requirement engineering practices to consider social learning and the social construction of requirements. We have highlighted this argument by drawing on empirical evidence from a study of how IS developers engage in requirements engineering and how they conceive its value. Factors we found to be important in this process included: the changing needs of the organization, the ways in which structured IS methods are enacted via experience and social competency, the formation of project groups, and finally engagement in inter-personal conflict and negotiations.

References

ABRAHAMSSON P, Conboy K and Wang X (2009) Lots done, more to do': The current state of agile systems development research, *European Journal of Information Systems,* **18(4)**: 281-284.

ACKERMAN M (2000) The Intellectual Challenge of CSCW: The Gap Between Social Requirements and Technical Feasibility. *Human-Computer Interaction,* **15(2-3),** 181-205.

AVGEROU C and CORNFORD T (1993) A review of the methodologies movement. In *Proceedings of the first European conference on information systems*, Henley-on-Thames, pp. 278-289.

AVISON D and FITZGERALD G (2003) Where Now for Development Methodologies?, *Communications of the ACM*, **46**(1), 79-82

BERGER P L and LUCKMANN T (1966) *The Social Construction of Reality.* New York: Doubleday.

BERGMAN M, KING J, LYYTINEN K (2001) Large Scale Requirements Analysis as Heterogeneous Engineering. *Scandinavian Journal of Information Systems*, **12(1)**, 37-56.

BERGMAN M, KING J, LYYTINEN K (2002a) Large-Scale Requirements Analysis Revisited: The need for Understanding the Political Ecology of Requirements Engineering. *Requirements Engineering*, **7(3),** 152-171.

BERGMAN M, KING J, LYYTINEN K (2002b) Large Scale Requirements Analysis as heterogeneous Engineering. In *Social Thinking - Software Practice,* (FLOYD C, KLISCHEWSKI R, Eds), pp. 357-386, MIT Press, Cambridge, MA.

BIJKER W E, HUGHES, T P and PINCH T J (1987) The social construction of technological systems. New directions in the sociology and history of technology. Cambridge, MA: MIT Press.

BIJKER W E (1995) Of Bicycles, Bakelites, and Bulbs: Toward a Theory of Sociotechnical Change. Cambridge MA: MIT Press.

BIJKER W E and LAW J (1992) Shaping Technology/Building Society: Studies in Sociotechnical Change. Cambridge MA: MIT Press.

BOEHM B (1981) *Software Engineering Economics.* Prentice-Hall, New York.

BROOKS F (1987) No silver bullet: Essence and accidents of software engineering. *Computer,* **20,** 10-19.

CHABRABORTY S, SARKAR S, SARKAR S (2010) An Exploration into the Process of Requirements Elicitation: A Grounded Approach, *Journal of the AIS **11(4)**, 212-249.*

EHN P (1988) Work-Oriented Design of Computer Artefacts. Almquist & Wiksell, Stockholm.

EHN P (1993) Scandinavian design: On participation and skill. In *Participatory design* . (SCHULER D and NAMIOKA A, Eds), pp. 41-77. Lawrence Erlbaum, Hillsdale,New Jersey.

EISENHARDT K M (1989) Building Theories from Case Study Research, *Academy of* Management Review **14(4),** 532-550.

GHEZZI C and NUSEIBEH B (1998) Guest editorial – Managing inconsistency in software development. *Transactions on software engineering*, **24(11),** 906-907.

GLASER B and STRAUSS A L (1967) The discovery of grounded theory: Strategies for qualitative research. Chicago: Aldine.

GOLDEN-BIDDLE K and LOCKE, K (1993) Appealing work: An investigation of how ethnographic texts convince. *Organization Science*, **(4)**, 595-616.

GRINT K and WOOLGAR S (1997) The Machine at Work: Technology, work and organization. Polity, Cambridge.

GUINAN, P J (1988) Patterns of Excellence for IS Professionals. An Analysis of Communication Behavior. ICIT Press, Washington.

HANISCH J and CORBITT B (2007) Impediments to requirements engineering during global software development. *European Journal of Information Systems*, **16**, 793-805.

HANSEN S, BERENTE N, and LYYTINEN K (2009) Requirements in the 21st Century: Current Practice and Emerging Trends. In K. LYYTINEN, P. LOUCOPOULOS, J. MYLOPOLOUS and W. ROBINSON (Eds.) *Design requirements: A Ten-Year Perspective.* Springer, Berlin, 44-87.

HOLMSTRÖM J (2005) Theorizing in IS research: What comes first and what comes after? *Scandinavian Journal of Information Systems*, **17(1)**, 167-174.

HOLMSTRÖM J, WIBERG M and LUND A (2010). *Industrial Informatics Design, Use and Innovation.* Hershey, PA: IGI Global.

HOWCROFT, D and LIGHT B (2010) The Social Shaping of Packaged Software Selection, *Journal of the AIS **11(3)**, 122-148.*

KLEIN H K and MYERS M D (1999) A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly*, **(23)**, 67-93.

KLING R and IACONO S (1984a) Computing as an occasion for social control. *Journal of Social Issues,* **40(3),** 77-96.

KLING R and IACONO S (1984b) The control of information systems developments after implementation. *Communications of the ACM,* **27(12),** 1218-1226.

LAMB, R. and KLING, R. (2003) Reconceptualizing Users and Social Actors in Information Systems Research, *MIS Quarterly*, **27(2)**, 197-235.

LOUCOPOULOS P and KARAKOSTAS V (1995) *System Requirements Engineering*, McGraw-Hill Book Company Europe, Berkshire, UK.

MADSEN S, KAUTZ K and VIDGEN R (2006). A framework for understanding how a unique and local IS development method emerges in practice. *European Journal of Information Systems* 15, 225–238.

MALCOLM E (2001) Requirements acquisition for rapid applications development. *Information & Management*, **39**, 101-107.

MASON J (1996) *Qualitative researching*, Thousand Oaks, CA: Sage.

MARKUS M (1983) Power, politics, and mis implementation. *Communications of the ACM,* **26(6),** 430-444.

MATHIASSEN L and NIELSEN P A (2000) Interaction and transformation in SSM. Systems Research and Behavioral Science, **17**, 243-253

MATHIASSEN L and STAGE J (1992) The Principle of Limited Reduction in Software Design. *Information Technology & People,* **6(2-3)**, 171-185.

MILES M B and HUBERMAN A M (1994) Qualitative Data Analysis: An Expanded Sourcebook (2nd ed.), Thousand Oaks, CA: Sage.

MUMFORD E (2003) *Redesigning Human Systems*, Idea Group, Hershey, PA.

NUSIBEH B and ESTERBROOK S (2000) Requirements Engineering: A Roadmap, In *Proceedings of the 22nd International Conference on Software Engineering,* June 4-11, Limerick Ireland. ACM.

ORLIKOWSKI W J and BAROUDI J J (1991) Studying information technology in organization: Research approaches and assumptions. *Information Systems Research,* **(2)**, 1-28.

ORLIKOWSKI W J and GASH D C (1994) Changing Frames: Towards an understanding of information technology and organizational change. *ACM Trans. Inform. Systems,* **2(2)**, 174-207.

PAUL R J (1993) Dead paradigms for living systems. In *Proceedings of the first European conference on information systems*, Henley-on-Thames, pp. 250-255.

PINCH T J and BIJKER W E (1984) The Social Construction of Facts and Artefacts: or How the Sociology of Science and the Sociology of Technology might benefit each other, *Social Studies of Science*, **14**, 399-441.

ROBEY D (1994) Modeling Interpersonal Processes During System Development: Further Thoughts and Suggestions. *Information Systems Research*, **5(4),** 439-445.

ROBINSON W and VOLKOV, S (1998) Supporting the Negotiation Life-Cycle. *ACM, Communications of the ACM*, **41(5),** 95-102.

ROOKSBY J, SOMMERVILLE I and PIDD M (2006) A Hybrid Approach to Upstream Requirements: IBIS and Cognitive Mapping. In A. DUTOIT, R. McCALL, I.MISTRIK and B. PAECH (Eds.) *Rationale Management in Software Engineering*, Spring, Berlin, pp. 137-154.

ROWLANDS, B. (2008) The Enactment of Methodology: An Institutional Account of Systems Developers as Social Actors, *Scandinavian Journal of Information Systems*, **20(2)**, 21-50.

SAWYER S (2001a) Information Systems Development: A Market-Oriented Perspective. *Communications of the ACM*, **44(11),** 97-102.

SAWYER S (2001b) Effects of Conflict on Packaged Software Development Team Performance*. Information Systems Journal,* **11(2),** 155-178.

SOMMERVILLE I and SAWYER P (1997) *Requirements Engineering - A Good Practice Guide*, John Wiley & Sons, England.

STEWART J and WILLIAMS R (2005) The wrong trousers? Beyond the design fallacy: social learning and the user. In D. HOWCROFT et al (Eds.) *Critical IT Handbook*, Edward Elgar, London, pp. 195-221.

STOLTERMAN E (1992) How system designers think about design and methods. *Scandinavian Journal of Information Systems*, **4,** 137-150.

STRAUSS A and CORBIN J (1990) Basics of Qualitative Research: Grounded Theory Procedures and Techniques. Newbury Park, CA: Sage.

STRAUSS A and CORBIN J (1994) Grounded Theory Methodology: An Overview. Thousand Oaks, CA: Sage.

SUCHMAN L (2002) Practice-Based Design of Information Systems: Notes from the Hyperdeveloped World. *The Information Society, 18*(2): 139-144.

THAYER R H and DORFMAN M (Eds) (1990) *System and Software Requirements Engineering*, IEEE Computer Society Press, Los Alamitos, CA.

TRUEX, D P, BASKERVILLE R, and KLEIN H (1999) Growing Systems in Emergent Organizations, *Communications of the ACM,* **42**, 117-123

URQUHART C (1999) Themes in early requirements gathering: The case of the analyst, the client and the student assistance scheme. *Information Technology & People,* **12(1)**, 44-70.

VAUGHAN D (1992) Theory elaboration: The heuristics of case analysis. In: *What is a case? Exploring the foundations of social inquiry* (RAGIN C C and BECKER H S (Eds), 173-202, Cambridge University Press, Cambridge, MA.

WALSHAM G (1993) Interpreting information systems in organizations. Chichester: Wiley.

WASTELLS D G (1996) The fetish of technique: Methodology as a social defence. *Information Systems Journal,* **6(1)**, 25-40.

WEICK K E (1993) The collapse of sensemaking in organizations: The Mann Gulch disaster. *Administrative Science Quarterly,* **38**, 628-652.

WEICK K E (1995) What theory is not: Theorizing is. *Administrative Science Quarterly,* **40,** 385-390.

WOOD-HARPER T, CORDER S V and WATSON H (1996) How we profess: the ethical systems analyst. *Communications of the ACM*, **39(3)**, 69-77.

WOOLGAR S (1991) The Turn to Technology in Social Studies of Science. *Science, Technology & Human Values,* **16(1)**, 20-50.

ZAVE P (1995) Classification of research efforts in requirements engineering, In *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, March 27-29, IEEE Computer Society Press, CA.

ZAPPAVIGNA M and PATRICK J (2010) Eliciting Tacit Knowledge about Requirements Analysis with Grammar-targeted Interview Method (GIM), *European Journal of Information Systems,* **19(1)**, 49-59.

Appendix 1: Coding Example

Here we provide an example of our coding effort. As background, we proceeded on a three-step coding effort. First, Open coding was done by focusing on identifying codes supported by two or more text segments, leaving us with 55 initial codes. During this initial stage, we drew on SCOT for guidance, and focused on identifying the nature of interactions among the two relevant social groups (developers and customer/users). Second, during the axial coding stage we consolidated codes that were conceptually similar, reducing to 19 from 55. Third, during the selective coding we strove to integrate the axial codes in ways that connected to SCOT principles and formulate a storyline that offered a coherent and insightful account of the RE practices.

For example, and drawing from the material we used in section 4.3.4, here is how we proceeded:

Step 1: Axial Coding:

These two text fragments speak to internal-to-the-commissioning organization conflicts among project personnel, what we called '*host conflict*' as an initial code. We were looking for evidence of conflict and negotiation because of the prior literature on this and because this sort of behavior is core to SCOT. The letters in parentheses are codes that allow us to distinguish among respondents while preserving their anonymity.

I believe that some of the worst fights are internal, when we are not with them. I believe that things can get pretty hot then. (A)

They [the conflicts within the commissioning organization] I believe are not really taken up in front of us, more internally. If we have a board meeting, they take up problems that they have with us and we take up problems that we have with them. (P)

Step 2: Open Coding

We identified other text fragments that highlighted conflicts among customers and developers ('social group conflict'), among the developers ('role difference'), and about particular issues ('issue conflict). These seemed to be distinct forms of conflict, but for open coding we linked them to the shared concept: conflict.

Step 3: Selective Coding

We used SCOT principles to guide our selective coding. SCOT's principle of conflict and negotiation seems to be reflected in our coding of '*host conflict/conflict.*' So we link these and represent this in our analysis.