Chapter II

The Social Study of Computer Science

Matti Tedre
Tumaini University, Tanzania

ABSTRACT

This chapter introduces the reader to some social research characteristics that are central to the social study of computer science. It introduces research studies that focus on the sociocultural aspects of computing and computer science, explains some of the central characteristics of those studies, and discusses their implications for the computer science discipline. Furthermore, this chapter is aimed at giving the reader a basic understanding of why social studies are important for the discipline of computing, as well as some broad guidelines and pointers towards carrying out such studies in computer science.

Our objective ... is to state precisely and clearly where and why sociological analysis is necessary in the understanding of scientific knowledge. Our main method is to present historical case studies. We then show how sociological analysis applies in these cases, and how it is an essential complement to even the most insightful interpretations derived from other perspectives.

—Barnes, Bloor, & Henry (1996)

INTRODUCTION

Computer science is a relatively new discipline, and it spans across traditional disciplinary boundaries, covering mathematical, engineering-oriented, and scientific traditions (Denning et al., 1989). From the birth of modern (digital, Turing-complete, electronic) automatic computing in the 1940s, those traditions have been essential to the development of the discipline. Modern computer science was born in the 1940s as a result of a number of organizations, a number of top people, many coincidences, a variety of disciplines, an uncommon political situation, a certain culture, unusually liberal funding, and
convergence of a number of technical and scientific breakthroughs (Tedre, 2006:passim).

Since the 1940s, modern computer science has been surrounded and shaped by a vastly complex conjunction of affairs. Due to their rich and colorful history, computer science and computer technologies include plenty of phenomena, the form and functioning of which cannot be explained in terms internal to those phenomena. For instance, one cannot explain the design and the (non-)diffusion of any programming language by referring solely to its technical characteristics (Sammet, 1991). Understanding the design and diffusion of any programming language requires understanding its history and the original motivations for its development in the first place (e.g., Denning, 2003; Rosenblatt, 1984). Similar, one cannot explain the development of GNU/Linux in solely technological terms—several non-technological motives, such as economic, political, ideological, and cultural motives, can be attributed to the development of GNU/Linux (cf. Tedre et al., 2006). Technical characteristics of GNU/Linux that stem from non-technological motives are perhaps better explained in other terms, such as in psychological, sociological, or anthropological terms.

So it is implausible that one could understand the current state, a static snapshot, of knowledge in computer science without understanding the history of computer science. Moreover, one cannot understand why knowledge in computer science is what it is without understanding the history of computer science. In addition to history, one must also understand how society and culture today shape computer science. As computer science is a product of an array of sociocultural forces, any portrayal of computer science is a historically, culturally, and societally specific image. Especially computer science as human activity always happens in some philosophical, historical, and sociocultural framework. That is, of course, not to say that computer science that is situated in a historical, cultural, and societal framework could not be objective. Objectivity can be defined in a number of ways that permit comparisons of socially constructed knowledge (e.g., Searle, 1996:p.8). For example, objectivity can refer to how strong consensus there is concerning a specific statement.

The importance of historical, cultural, and societal self-understanding of computer science are explicitly noted in the ACM/IEEE computing curricula CC1991 and CC2001 (Tucker et al., 1991: p.73; Denning et al., 2001:p.141). Those curricula emphasize the importance of understanding cultural, social, legal and ethical issues; and stress the appreciation of philosophical questions, technical problems, and aesthetic values. It is, however, uncertain how exactly should philosophical questions, technical problems, and aesthetic values be studied. Neither is it certain how the cultural, social, legal, and ethical issues in computing should be approached. One approach that originates from science and technology studies is social studies of computer science— that is, research of computer science itself in its sociocultural context. The focus of social studies of computer science is different from that of social studies of computing as the former is focused on the discipline, whereas the latter is focused on the activity. Social studies of computer science aims at enriching disciplinary self-understanding of computer science by producing meta-knowledge about computer science. That knowledge helps computer scientists to delineate between brute facts (like the laws of nature) and socially constructed facts (like standards and models).

THE CONTRIBUTION OF SOCIAL STUDIES OF COMPUTER SCIENCE

Researchers of social studies of computer science often adopt different conceptual and theoretical frameworks, and start from different sets of assumptions. Often those assumptions are in line with the constructionist, contingent, non-relativist, and nominalist viewpoints of science. In other words, social studies of computer science often entails the assumptions that much of people’s knowledge is constructed (rather than absolute), that the history and development of current computer science is one out of an infinite number of possible routes (rather
The Social Study of Computer Science

than an inevitable course), that there is a world of ontologically and epistemologically objective things (rather than only subjective statements about the world), and that many of the observed hierarchies and structures in computer science are constructed in order to give structure to the discipline (rather than being a result of an inherently structured world) (cf. Hacking, 1999).

From a narrow point of view, social studies of computer science informs computer science to the extent that social studies of computer science can contribute to knowledge about the subjects of computer science. From a broad point of view, when one considers disciplinary self-understanding to be a part of a mature discipline (e.g., Barnes et al., 1996: pp. iix-xii), then one should also acknowledge research that can contribute to the meta-theories, meta-knowledge, ontology, epistemology, and methodology of a particular discipline. For example, De Millo et al.’s (1979) research on theory-formation in computer science is a contribution to the meta-theories of computer science, Harel’s (1980) research on theorems that are untested yet widely held contributes to the meta-knowledge of computer science, Brian Cantwell Smith’s (1998) On the Origin of Objects is a study of the ontology and the epistemology of computer science, Kidder (1981) and Suchman (1987) have contributed to the understanding of how computer scientists actually work, and there are numerous examples of research on the methodology of computer science (e.g., Tichy et al., 1995; Glass, 1995; Alavi & Carlson, 1992; Vessey et al., 2002; Palvia et al., 2003; Glass et al., 2004; Randolph, 2006). Other aspects of a broad interpretation of computer science can be considered to be, for example, sociocultural impacts of computing and computing ethics.

Generally speaking, the constructionist paradigm has established a stable status in the discipline of computing. Especially in those topics that are in close contact with the social sciences, humanities, or education field, the human-constructed nature of computing is emphasized (e.g., McGuffee, 2000; Grier, 2002; Siefkes, 1997 in Freksa et al., 1997). Computing has aptly been described as a humanly constructed and constructive endeavor: computational tools are products of human activity and those tools are agents of social change (Naur, 1992: pp. xiii-xiv). The term social studies of computing has been used in connection with research on the impact of computers on society and of society on computers (Kling, 1980); with studies of technological discourse (Agre, 1995); and on studies of virtual society (Woolgar, 2002). There are plenty of studies of sociologically, historically, anthropologically, and philosophically oriented research about different aspects of computing (such as Viller & Sommerville, 1999; Crabtree et al., 2000; Hartswood et al., 2002; Suchman, 1987; Godin, 1997; Olazaran, 1996; MacKenzie, 1993; Bowker, 1993; Forstythe, 1993; Kidder, 1981), and often those studies go under some umbrella term such as STS or SSK.

So computer science researchers, such as those above, often borrow methods and aspects of research from social sciences and humanities. This chapter summarizes the methods and concepts used by social scientists and philosophers, and explores their relevance to computing research. In this chapter, the intellectual contribution of social studies of computer science to the broader field of computer science is portrayed by referring to some aspects of research that social studies of computer science entails. This chapter discusses (1) three different sources of information—phenomena in situ, reports of phenomena, and mute evidence; (2) a linkage to the sociohistorical context; (3) ethnomethodology; (4) ethnographic methods; (5) a non-generalizing focus on cases; and (6) measures of interpretive research. Those six research aspects are here connected with examples of existing studies of computer science in order to show that our common understanding of computing has benefited when researchers have incorporated those aspects of research in their inquiry. Those six aspects of research offer some alternative windows to computing research, alternatives that differ from strictly quantitative research. Still, those six aspects do not replace quantitative methods but complement them—qualitative researchers can utilize quantitative methods, and vice versa.
SOURCES OF INFORMATION IN SOCIAL STUDIES OF COMPUTER SCIENCE

Disciplines such as sociology, history, and anthropology can contribute unique viewpoints to social studies of computer science. The research approaches that are either explicitly discussed in this chapter or implicit in the sources of this chapter can be categorized, according to their source of information, into

1. those that study phenomena in situ, or what people do (for instance, ethnographic observations of computer scientists in their work, or observational field studies at locations that play a part in the innofusion of computer systems—locations such as academia, hardware/software manufacturers, professional associations, government branches, or homes);
2. those that study reports of phenomena, or what people say (for instance, interviews with people in the computing field or discourse analysis of debates about computing); and
3. those that study mute evidence like written texts and artifacts, the creators of which are not alive or cannot be interviewed (for instance, historical records, old or new computational instruments, or statistics).

The boundaries of these three categories (research of reports, research of phenomena in situ, and research of mute evidence) are not sharp. However, that does not really matter because these categories are not presented in order to define social studies of computer science, but in order to present some practicable conceptual categories for types of social studies of computer science.

STUDYING THE SOCIOHISTORICAL CONTEXT OF COMPUTER SCIENCE

No matter in what terms the shaping of computer science is presented, if computer scientists wish to retrospectively understand the reasons why computer science and computing have shaped as they have, their methodological toolbox must include historical methods. This is because computer science and computing are always situated in some sociohistorical context. A historical study of computer science needs to link aspects of computing with changes of some kind (cf. Lemon, 2003:pp.294-295), be they social, cultural, theoretical, technological, or conceptual changes. A historical study is also always conducted from a specific point of view (Tuchman, 1994).

The narrative form of history-writing is especially dependent on happenings (Lemon, 2003: pp.298-301). For instance, it is not very revealing about the history of programming languages to write, “In the early 1950s machine language programming was popular and in the early 1960s FORTRAN was a popular programming language.” The historian of computer science should also link happenings. For instance, the historian may expound on why there was a shift from machine language programming to high-level programming, who began the shift, what factors contributed to the shift, and how the shift actually happened—the shift certainly did not happen overnight. An analytic history of computer science should not only explicate, for instance, what is “a (statistically) typical 1950s computer scientist” (e.g., a young male, in his 30s, with a background of electrical engineering or mathematics,) but also explain the reasons why typical computer scientists of the 1950s shared those characteristics and how those characteristics affected the development of computer science (cf. Lemon, 2003:pp.295-297; Tedre, 2006:393).

A narrative history of computer science is able to portray a living computer science instead of a gallery of snapshot images. That is important for our common understanding of computing, because far from being discrete steps, many “milestone” concepts and events in computing have been multifaceted issues, and they have formed as a result of controversies, debates, and power struggles. Almost everything that is considered to belong to the core knowledge of computer science today is traceable to a number of controversies or discussions.
Computer science and electronic computing are old enough that a historical study of computer science can include both (1) secondary sources, such as works of communications specialists, literary critics, or historians; reference guides; references of good monographs; or citation indexes; as well as (2) primary sources, found in, for instance, archives, statistics, censuses, letters, diaries, newspapers, or popular literature (see Tuchman, 1994). The authors of primary sources have been eyewitnesses to the reported phenomenon, whereas the authors of secondary sources have not been first-hand witnesses to the reported phenomenon.

The importance of the historical research about computers has been acknowledged broadly (e.g., Zhang & Howland, 2005; Lee, 1996; Lee, 1996b), and the history of computing as a research field is well-established. A prime example of a history of computing journal is the *IEEE Annals of the History of Computing*, which has been published quarterly since 1979. Examples of monographs include, for instance, general histories of modern computers (e.g., Campbell-Kelly & Aspray, 2004; Ceruzzi, 2003), topic-specific books (e.g., Sammet, 1969), works on specific aspects of the history of modern computers (e.g., Flamm, 1988), and works on the general history of computation (e.g., Williams, 1985). Many of these historical accounts offer sociohistorical interpretations of the discipline of computing.

Historical research on computer science, such as the kind of research published in *the Annals*, includes not only the analysis of texts, but also the analysis of other mute evidence, such as devices, parts of devices, blueprints, diagrams, components, and other material traces. The problem of interpretation of mute evidence is that there may no longer be anybody alive to articulate the intentions behind the creation of the material (Hodder, 1994). This problem of interpretation and credence is, however, common to many disciplines. In all types of interactive research the analyst has to decide whether or not to take commentary at face value and how to evaluate spoken or unspoken responses (Hodder, 1994).

If there is no way to gather indigenous commentary, material artifacts pose special problems for historians of computer science (cf. Hodder, 1994). It is, in fact, impossible to understand an unknown artifact with certainty without knowing the intentions of the creators of the artifact (Tedre, 2006:p.131). Historians have argued that there is no “original” or “true” meaning of an artifact outside a specific sociohistorical context (cf. Hodder, 1994). Von Neumann’s (1945) *First Draft of a report on the EDVAC* is an example of mute evidence that needs to be situated in its sociohistorical context in order to be understood. It is difficult to capture von Neumann’s intentions, motivations, and meanings and to put them in today’s terms. Even more, von Neumann’s metaphor transfer from neuropsychology to computing technology blurs even the technological parallels between the language of his famous draft and the language of today’s computer science. Artifacts are always produced under certain material conditions embedded within social and ideological systems, and the EDVAC plans were produced in an especially rare social, political, cultural, and economic situation.

**UNDERSTANDING COMPUTER SCIENTISTS’ WORK THROUGH ETHNOMETHODOLOGY**

There are some attempts to prescribe how computer scientists should properly work, yet those prescriptions may not correspond to how computer scientists actually investigate computing, to how they give structure and meaning to computing, or to how they sustain and manage that knowledge. That is, although there are attempts to describe rigorous, official set(s) of methods of computer science, the practices of computer scientists still do not always match the official set(s) of methods. How people give structure and meaning to knowledge and how they sustain and manage that knowledge are the focus of *ethnomethodology* (Holstein & Gubrium, 1994; Denzin & Lincoln, 1994:p.204).

Similar to *methodology*, *ethnomethodology* does not refer to a specific set of methods in any straightforward sense; it is more of a study of specific actions, “people’s methods,” which constitute
The social activities of a group of people (Lynch, 2004; Lynch, 1996). Ethnomethodology has been successfully used in studies of how scientists of different disciplines, including mathematics and natural sciences, create and maintain knowledge (Clayman, 2001). Although scientific methods in computer science can be highly technical, they are specialized instances of the much broader social phenomenon: Scientific methods are instructions that enable computer scientists to reproduce the computing community’s practices (cf. Lynch, 2004). When ethnomethodologists study computer science, the aim is to dig deep into the unexplicated, obscure foundations and features of practices that are not mentioned in the formal methodological prescriptions or reports (cf. Clayman, 2001; Lynch, 2004). Hence, ethnomethodology delivers an especially attractive promise: That of explicating the actual ways of constructing and managing knowledge in computer science. This could be called, for instance, the in situ methodology of computer science or the tacit methodology of computer science.

In other words, ethnomethodological approaches in social studies of computer science benefit computer science (both as an activity and as a body of knowledge) to the extent that they can expose how the technological, philosophical, theoretical, conceptual, and methodological frameworks of computer science are created, maintained, and managed. For instance, ethnomethodological studies are revealing about the manners in which new innovations are conceptualized by groups of computer scientists and other stakeholders. Ethnomethodological research can elucidate the processes through which conceptual consensus is achieved in computer science. It can shed light on how epistemologically subjective results in computer science are communicated, confirmed, adopted, objectified, and institutionalized into epistemologically objective facts. It can explain how knowledge is transmitted. It can reveal how common knowledge about computer science gives meaning to activities of computer scientists and to the results of computer science. It can expound on how activities of computer scientists generate knowledge in computer science. And it can track how both intra-scientific and extra-scientific contradictions are dealt with.

The practical value of ethnomethodology has been recognized in fields such as software engineering, human-computer interaction, and other kinds of research on the relationship of work, users, and computers (e.g., Viller & Sommerville, 1999; Crabtree et al., 2000; Clayman, 2001; Hartswood et al., 2002). Lucy A. Suchman (1987:pp.49-50) expressed her ethnomethodological viewpoint in her book Plans and Situated Actions. On the more theoretical side of computing disciplines there are reports on how epistemologically subjective proofs are created and transformed into epistemologically objective “hard” facts in computer science (e.g., Richard De Millo et al.’s (1979) theoretical review Social Processes and Proofs of Theorems and Programs). Such reports shed light on the very foundations of knowledge creation in the computing disciplines (albeit De Millo et al.’s research can be considered to be only borderline ethnomethodological, because their purpose was to contribute to the formal verification debate in computer science).

In addition to the aforementioned studies in which ethnomethodology has been used to study the users of computational systems, there is also research on the methods and practices used in computer science. There is research on, for instance, how rhetorics in discourse have influenced technological decisions (e.g., Godin (1997) studied the rhetorics surrounding the innofusion of a health technology). There is research on how contingent social elements affect the closure of scientific debates (e.g., Olazaran (1996) studied how Minsky’s and Papert’s proofs and arguments were interpreted as showing that neural nets were not a fruitful approach to artificial intelligence). There is research on how some mathematical parts of computer science are negotiated, rather than deduced (e.g., MacKenzie (1993) examined how the IEEE standard for floating-point arithmetic arose as a result of negotiation). There is research on what kinds of rhetorical strategies have been used in arguing for the universality of computing technology (e.g., Bowker (1993) showed the ways the practitioners of cybernetics argued that they were producing a new, universal science). And there is research on how knowledge engineers’ epistemological stances are reflected in artificial intelligence technology (e.g., Forsythe’s (1993) study
in which he drew on ethnographic material about knowledge engineers’ work, showed that building a knowledge-based system necessarily involves interpretation and selection, and suggested that knowledge engineers should be trained in qualitative social science).

Ethnomethodological investigations can be conducted with a variety of methods. The knowledge construction processes in computer science have been mostly examined using analytical methods or reflection (e.g., De Millo et al., 1979; Crabtree, 2004; Hartswood et al., 2002). If the organization of social interaction is the focus of an ethnomethodological investigation, ethnomethodology is often coupled with conversation analysis (Holstein & Gubrium, 1994; Lynch, 2004). Usually, however, ethnomethodological research is ethnographic and pays especially close attention to how the people in the study setting communicate and interact (Holstein & Gubrium, 1994). The ethnographic kind of ethnomethodological studies require extensive participant observation in specialized work settings (Lynch, 2004). However, whereas in traditional ethnographic research on computer science one might assume that the language of computer science is a neutral conduit for description, in ethnomethodologically oriented research on computer science, descriptions, accounts, or reports should be treated not merely as being about the social world of computing as much as being constitutive of the social world of computing (Holstein & Gubrium, 1994).

Although it cannot be said that there is an ethnomethodological tradition in the field of computer science, ethnomethodology is not unknown to computer scientists, either. In the computer science literature there are studies in which the ethnomethodological approach has been made explicit, as well as studies that can be characterized as ethnomethodology. The majority of the ethnomethodological studies in computer science literature report on the users of information technologies (they study the end users’ methods) and are aimed at informing, for instance, system designers, interface experts, and software engineers. Also present are ethnomethodological investigations in which the practices and behaviors of computer scientists are studied, yet those studies are more commonly aimed at informing sociologists than they are aimed at informing computer scientists.

**ETHNOGRAPHIC METHODS**

Around the mid-1900s new technical systems grew too large and complex to be designed and maintained by single individuals. The complexity of systems today necessitates broad approaches to understanding system development. Suppose, for instance, that one wants to explain the ontological, epistemological, methodological, or material assumptions, decisions, foci, or compromises that system design may incorporate. Nowadays it is not enough to study individual actors and their surroundings because systems are no longer designed or managed by individuals; studying groups is necessary. When explicating the design decisions behind a complex system collective or multiple perspectives need to be accounted for. Ethnographic methods offer researchers of computer science a unique way of understanding the processes and dynamics behind, for instance, computer architecture design. Instead of historical studies, which are conducted in retrospect, ethnographic methods are studies of the present—studies of computer science in the making.

The term *ethnography* has been used in a large variety of meanings. One characterization is that ethnography is the “art and science of describing a group or culture” (Fetterman, 2004). Originally the term ethnography referred to the book-length record of anthropologist’s observations and analysis about his or her involvement in a community (Agar, 2001). The data of ethnography are derived from the direct observation of behavior in particular groups (cf. Conklin, 1968 in Sills, 1968). As a verb, doing ethnography merely means the collection of data that describe (some parts) of a culture (e.g., Bernard, 1995:p.16; Agar, 2001 in Smelser & Baltes, 2001; Conklin, 1968).
Roughly speaking, a researcher using the scientific method seeks universal laws, emphasizes control of the research process, preserves the initial assumptions throughout the study, relies on linear models, and represents data with numbers (Agar, 2001). By contrast, again roughly speaking, a researcher using ethnography seeks local particulars, emphasizes adaptability in the course of study, develops new concepts over the course of the study, relies on systemic and processual models, and represents data more often with words than with numbers (Agar, 2001). The promise of ethnography in social studies of computer science lies in the extent to which ethnography succeeds in eliciting the perspectives and realities of computer scientists, that is, the insider’s or emic perspectives or reality (cf. Fetterman, 2004). In other words, the promise of ethnography lies in the extent to which ethnography can explain how the activities of computer scientists create the body of knowledge of computer science.

Ethnography is often misunderstood as being purely qualitative research, but in reality it can include quantitative aspects, too. Usually, however, ethnographic research shares the features of (1) exploring phenomena rather than testing hypotheses; (2) emphasizing unstructured data instead of analytic categories; (3) focusing on cases in detail instead of large populations, and (4) explicitly interpreting the meanings and functions of human actions (Atkinson & Hammersley, 1994). Ethnographers in computing fields should live with the group they study for an extended period of time (ideally about 2 years), they should actively participate in the daily life of the group members, and they should carefully observe all aspects of the group members’ life as a way of obtaining material for their study (cf. Tedlock, 2005; Atkinson & Hammersley, 1994; Bernard, 1995:p.78).

Tracy Kidder’s (1981) The Soul of a New Machine is one of the early examples of an ethnographic-type participant observation in the field of computing. Kidder observed a group of engineers at Data General from 1978 to 1980—the whole period of a design, implementation, testing, and release of a new 32-bit minicomputer (which became the Data General Eclipse MV/8000). In his book, Kidder described the company work environment and the machine, concentrating on not only technological decisions, but also on things such as the engineers’ emotions, the birth of innovations, bottom-up management, the dedication and motivations of the engineers, the pressures caused by tight schedules, disappointments, engineering ethos, and engineering artistry. Kidder discussed how architectural design is actually done, the challenge of designing a new 32-bit architecture while maintaining downward compatibility to legacy architecture, decisions concerning microcode, instruction set, registers, diagnostics, input/output, types of components used, and so forth. A competent computer scientist can get acquainted with the architecture of Data General Eclipse MV/8000 computer by studying its blueprints and specifications. Kidder’s study offers some viewpoints on why the Eclipse MV/8000 architecture is what it is.

Ethnographic methods in social studies of computer science aim at describing and interpreting social phenomena such as ways of working, group relationships, communication, metaphors, and tropes in computing community (cf. Atkinson & Hammersley, 1994). Since ethnographic methods emphasize understanding phenomena in their rich sociohistorical contexts, ethnographic methods can be utilized in order to examine, for instance, patterns of production of scientific results, innovation, and standards in computer science. They can also be utilized to study mechanisms of technological production, design, adoption, rejection, diffusion, non-diffusion, and so forth. Scientists today have a unique opportunity to examine and document the early formation of the discipline—modern computer science is no older than 60 years, and it can be argued that many parts of computer science, such as information systems and software engineering, are still at the pre-paradigm stage of scientific development (e.g., Wernick & Hall, 2004).

**FOCUS ON CASES**

Often computer science research aims at generalizability; computer scientists often argue that
The results from one set of data are applicable to all similar data. In research where generalizations are made, the significance of single cases is often downplayed. By contrast, in those studies in social studies of computer science that aim at contributing to knowledge about computer science by explaining how and why computer science has taken its current form, single cases are important. Single, non-generalizable studies, such as Donald MacKenzie’s (1993) study of the negotiation of floating-point arithmetic, are important because they can offer information about the hows and whys of technoscience (yet single cases can also contribute to generic theories about technoscience).

The term case study can be understood as a method or a research strategy (Yin, 2002) but here it is understood as the focus of a specific study. When case study is understood as an indicator of the focus of the study, case studies can be quantitative or qualitative, although many studies that are labeled as case studies are qualitative (Stake, 1994). The driving question behind case studies is, “What can be learned from the single case?” (Stake, 1994). However, it is typical of case studies that the researcher is ultimately interested in a process, or in a population of cases (Denzin & Lincoln, 1994: p.203; Denzin & Lincoln, 2005:p.380). Case studies in social studies of computer science aim at a deep understanding of phenomena in computing; and they can be of an individual, a group, a site, a class, a policy, an institution, or a community (Ary et al., 2006:p.456). Case studies aim at rich, detailed descriptions of phenomena, and they often use several different research instruments or methods.

Instrumental case studies are conducted because the researcher believes that a particular case may provide insight into an issue, theory, concept, technology, or such; and collective case studies extend to several cases (cf. Stake, 1994). Also historical narratives—such as Thomas Hughes et al.’s reports on relational databases, the Internet and the World Wide Web, theoretical computer science, artificial intelligence, and virtual reality—can be considered to be a form of case study which is aimed at uncovering the actions of stakeholders (NSR Computer Science and Telecommunications Board, 1999). Such narratives allow analogies to be drawn between events that occurred decades apart; narratives can accommodate complexity more easily than can a tightly-structured analytical essay, and can “present finely nuanced accounts that convey the ambiguities and contradictions common to real-life experiences” (NSR Computer Science and Telecommunications Board, 1999:p.3).

EVALUATION OF STUDIES OF SOCIAL REALITY

The crux of social studies of computer science is not numbers and proofs. Unlike the subjects of mathematics and logic, the subjects of disciplines such as sociology, history, anthropology, and philosophy are often not well-structured, logical, coherent, or well-defined. If one were to agree that changes in technology follow from choices that mirror the social relations of innovation and diffusion of technology, it would be an error to assume that having exposed the choices and their motifs, one could simply deduce the rest of reality from them (Noble, 1999). Researchers should not assume that by unearthing the social, cultural, economic, institutional, personal, and other human variables they can converge on the “true” state of affairs. That is because models and tools—such as classification systems, conceptual frameworks, data structures, or computational models—are influenced by the researcher’s existing knowledge about the domain, as well as by the epistemological and methodological commitments of the researcher. Even in the field of statistics, differences in classification systems and their changes over time are actually seen today as phenomena that deserve to be examined in their own right (Desrosières, 1996 in Hantrais & Man- gen, 1996).

In many kinds of research validity and reliability form the stone base of confirmation. But in qualitative research there is less agreement on confirmation procedures than in quantitative research. It has been argued that the “twin struts of confirmation” are coherence and correspondence (Hodder, 1994). Internal coherence is the degree to which the parts
The Social Study of Computer Science

of an argument do not contradict each other and to which the conclusions follow from the premises (Hodder, 1994). However, it must be noted that what is considered to be a credible argument may differ between disciplines and between individuals. For example, what is considered to be a credible argument in biology may differ from what is considered to be a credible argument in sociology. Ultimately, the audience interprets and judges every argument in interpretive research. External coherence refers to the degree to which the interpretation of research material fits theories, models, or interpretations accepted within and outside the discipline (Hodder, 1994) (See Figure 1).

Correspondence between theory and data is an essential part of a coherent argument. “Correspondence between theory and data” does not imply an absolute and independent link between theory and data, but it rather embeds the fit of data and theory within coherence (Hodder, 1994). The data are made to cohere by being linked within theoretical arguments (Hodder, 1994). Similarly, the coherence of the arguments is supported by their fit to data. The more robust fit there is between data and theory, the better correspondence they can be said to have.

The concepts of correspondence and coherence are portrayed in Figure 1 (cf. Tedre, 2006:p.407). Correspondence expresses how well sets of data cohere within the selected theoretical framework in the study, internal coherence expresses how well arguments and conclusions follow from the data and theory, and external coherence expresses how well the conclusions and arguments resonate with other pieces of research and theory.

In addition to the traditional measures of validity; such as instrument validity, face validity, data validity, and criterion validity (see, e.g., Bernard, 1995:pp.38-42); there are a variety of less frequent measures of validity, such as contextual validity, dialogic validity, and self-reflexive validity (Saukko, 2005). Contextual validity refers to the thoroughness and defensibility of the analyses of social, historical, political, or economic processes and structures (Saukko, 2005). Dialogic validity refers to the extent to which research is able to expose tacit, experienced, emotional, and embodied knowledge and understanding (Saukko, 2005). Self-reflexive

Figure 1. Correspondence and coherence
validity is based on the critical reflection of how social discourses shape or mediate people’s self-experiences and the experiences of their environment. Self-reflexivity refers to the extent to which the researcher is aware of the discourses that guide the research analysis itself (Saukko, 2005). Although those validity measures are subjective, one must remember that any and all validity measures are subjective. Ultimately, the validity of a concept depends on two things: the utility of the device that measures it, and the collective judgment of the scientific community (Bernard, 1995: p.43; also Tedre, 2006: p.408).

On lines similar to Thomas Kuhn (1977: pp.321-322), three other criteria to the success of research have been proposed (Hodder, 1994): fruitfulness (how many new directions, new lines of inquiry, new perspectives are opened up), reproducibility (the extent to which other people, perhaps with different perspectives, come to the same results), and intersubjective agreement (on a science that balances between a number of disciplines, the adequacy of the results to those disciplines). Research that attacks an obstacle that hinders progress in a number of topics often turns out to be especially fruitful. That is, there are certain obstacles that, after they are overcome, allow a number of topics to be pursued.

CONCLUSION

In this chapter, a number of research aspects have been discussed; aspects that are useful in social studies of computer science, yet that do not necessarily belong to the traditional computer scientists’ toolbox. These aspects of research are an example of the new viewpoints that a humanities and social sciences-based approaches have brought into understanding computer science and computing. Research that can be considered to be social studies of computer science is nowadays often conducted by trained specialists in fields such science and technology studies, interface design, management information systems, and history of computing.

But also the modern computer scientist-as-a-bricoleur ought to be cognizant of different research approaches; the computer scientist working with social studies of computer science needs a full toolbox and the knowledge of how to use those tools appropriately.

The social studies of computer science can be considered to be research that situates and investigates computer science in its social, historical, cultural, linguistic, political, economic, institutional, personal/individual, and other socially constructed frameworks—including its scientific and technological frameworks. The social study of computer science aims to provide an image of computer science as knowledge and as activity; an image in which more than merely the technological aspects are considered to be influential. It can reveal how computing as a discipline is continuously re-created and maintained, and explain how scientific statements in computing are externalized, objectified, internalized, and reified—that is, explain why many things that computing professionals produce are afterwards perceived as something other than human products. The social study of computer science can explicate unspoken assumptions, shared attitudes, and tacit knowledge.

In the social study of computer science, historical methods are utilized in order to retrospectively appreciate the reasons computer science has become shaped as it has. Sociohistorical understanding offers important “lessons learned”; it can trace changes in technology and science to challenges, controversies, and discussions; it can discover parallels and analogies between previous, current, and future technology; and it links happenings and assesses the reasons for those happenings. Ethnomethodological research is a social study of computer science that offers insight into the present of computer science. It is aimed at portraying the actual processes of constructing and managing knowledge in computer science. It attempts to uncover the ambiguous and complex practices of, for instance, creating, maintaining, using, abusing, proving, refuting, negotiating, accommodating, appropriating, and contextualizing knowledge in computing.
In social studies of computer science, ethnographic methods are used to explore and portray the realities, perspectives, and group dynamics of computer scientists. Those aspects of computer science are interesting in themselves—but they may also reveal some aspects in the practices of computer scientists that have direct consequences on the common knowledge in computer science. Often different methods are combined together to case studies; studies which aim at an in-depth investigation and analysis of phenomena. Case studies attempt at portraying phenomena and at explaining the hows and whys concerning those phenomena. Multiple methods in case studies can give rich insight into social as well as technical aspects of computer science. But the aims of the social study of computer science are not solely descriptive. In the sense that social studies of computer science offer alternative explanations of concepts, theories, instruments, techniques, methods, or designs of computer science, they can have normative aims too—that is, those studies can also aim at changing the content, processes, hierarchies, or other aspects of computer science.

REFERENCES


**KEY TERMS**

**Ethnomethodology:** The study of the ways (such as conventions, practices, and codes) through which people make sense as well as create their social reality and which underlie social interactions between people.

**Ethnography:** The term ethnography is used in various meanings, but as a set of methods, it refers to observational and participatory methods.
that focus on the life of some particular group of people; their culture, behavior, social interactions, and other aspects of their everyday life.

**Case study:** The term case study can refer to a method, research strategy, or focus of study. In the latter meaning case studies aim at finding out what can be learned from a single case. Case studies can have quantitative and qualitative aspects, and they aim at giving rich, detailed descriptions of a phenomenon.

**Method:** The term method refers to a means or a procedure for accomplishing something, like measuring the execution time of a task, interviewing a group of people about an interesting phenomenon, or comparing the execution times and output sizes for a given input with two computer algorithms.

**Methodology:** The term methodology refers both to a specific set of methods and to the study of usage patterns, procedures, principles, and assumptions that underlie such set of methods.

**ENDNOTE**

1. *Ontology* refers to the study of the nature of being and reality. It deals with questions such as what kinds and types of things exist, what does the existence of some things depend on, and what does it mean if one says that something exists. *Epistemology* refers to the study of the nature of knowledge. It deals with questions such as what is knowledge, when is knowledge justified, and how do people acquire knowledge.