IT’S NOT A PROPERTY! EXPLORING THE SOCIOMATERIALITY OF SOFTWARE USABILITY

Kai Riemer
The University of Sydney, kai.riemer@sydney.edu.au

Nadine Vehring
The University of Muenster, nadine.vehring@wi.uni-muenster.de

Follow this and additional works at: http://aisel.aisnet.org/icis2010_submissions

Recommended Citation
http://aisel.aisnet.org/icis2010_submissions/215

This material is brought to you by the International Conference on Information Systems (ICIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in ICIS 2010 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
IT’S NOT A PROPERTY!
EXPLORING THE SOCIOMATERIALITY OF SOFTWARE
USABILITY

Completed Research Paper

Kai Riemer
The University of Sydney
Discipline of Business Information Systems, Sydney, NSW 2006, Australia
kai.riemer@sydney.edu.au

Nadine Vehring
The University of Muenster
Department of Information Systems
Leonardo-Campus 3, D-48149 Muenster
nadine.vehring@wi.uni-muenster.de

Abstract

Usability is typically conceived of as a property of a software artifact (“a software product has good/bad usability”) and tested in user experiments. In this conceptual paper we challenge this established view. For doing so, we draw on the concept of sociomateriality and a case study, which illustrates how usability of a workplace telephony software is perceived and treated very differently in usage practices across use contexts. We find that the software interacts with different types of hardware and with institutionalized ideas and norms in the various contexts of user organizations in such a way that it renders defining usability as a property of the software artifact a pointless task. In fact, we question whether it is useful to conceptualize (workplace) software as artifacts with stable set of properties. Rather, we argue that usability manifests in the sociomaterial use context. We discuss methodological implications and identify areas for future research.

Keywords: Usability, Usability testing, Sociomateriality, Workplace Software
Introduction

Software usability is a rather mature concept. It is typically portrayed as a characteristic or property of a software artifact (e.g., Cockton, et al., 2003; Dumas and Redish, 1999; Goodwin, 1987). As such, it describes the capability of software to be understood and used by its users to fulfill particular tasks. In this conceptual paper, we will question this established view. In fact, we will ultimately challenge the very notion that software can be usefully conceptualized, or more precisely individuated, as artifacts with stable sets of properties/characteristics - one of which is usability. For doing so, we will draw on the concept of sociomateriality, introduced by Orlikowski and Scott (2008a, 2008b) as an alternative worldview that questions the so-called entity view of the world. With regards to software, sociomateriality poses that software can only be fully understood when viewed in the context of a social practice (i.e., as technologies-in-use, cf. Orlikowski, 2000).

In order to richly illustrate our idea, we reflect on a research project, which we carried out for a software company: We were asked to assess the usability of a workplace telephony (VoIP) software, targeted at medium-sized companies. We carried out interviews with users in a set of user organizations in order to get to know the ways in which the software is being used and to explore the perception of its usability, with the aim to derive recommendations for improving the usability of this software artifact. However, our project practically failed when judged against this goal. Fundamentally, we had difficulties to define and understand what counts as usability for the users in the various cases, to the extent that treating usability as a property of the software artifact became outright impossible. Our case study illustrates firstly that software interacts with its surrounding hardware in a way that renders defining software without its hardware a difficult task (hardware changes the ways in which the software behaves and is being used). Hence, usability cannot be attributed to the software product alone. Secondly, but much in the same way, the social environment of a particular use context interacts with the bundle of hardware and software, as it has a strong influence on the perception, role and usage of the bundle. Ultimately, usability manifests as an aspect of this sociomaterial use context, with which software and hardware become entangled.

Consequently, we argue that usability can only be encountered, researched and understood in its use context, when seen as embedded in the sociomateriality of user practices. Using our case study we will illustrate how the use context of a particular workplace can have a considerable influence on how a software is perceived, what kinds of features the users draw on, what kind of usage is deemed appropriate and so forth. At this stage, we want to point out that this paper resembles a conceptual study more than an empirical study. As such, the focus of the paper is not on the qualitative research part of the study (the case study), but on the conceptual argument outlined above. Hence, we will use evidence from our case to illustrate our reasoning, but we do not claim to be generalizing our findings from this case.

Our paper proceeds as follows: In the next section we will firstly portray usability and usability testing based on a review of the literature, before we briefly introduce sociomateriality. In the then following section we provide an overview of the case company, our project and interviews, before we discuss in more detail how usability is perceived in the different workplace scenarios. Using the case findings for illustration, we then further outline our line-of-argument. In doing so, we argue that software can only be usefully individuated on the basis of the structure of a use context (with its user practices) and that usability is embedded in this holistic involvement. After that, we discuss implications for usability studies and argue for the need of contextual usability methods. In doing so, we point out opportunities for future research. The last section concludes the paper.

Usability

In this section we will introduce usability as depicted in the literature. Please note that it is not our intention, nor is it necessary for our argument or even feasible to provide a comprehensive review of the very large usability-related body of literature. Rather, we will portray usability as it is conceived of in the most commonly used definitions. We establish that usability is typically treated as a property of a software artifact. We will then corroborate this observation by briefly summarizing methods for usability testing, which are used in practice, i.e. in testing usability for the purpose of software design or as carried out by consultants offering such testing services.
Usability Construct: Definitions and dominant Conceptualizations

Software usability is a mature concept and has been around for at least as long as end user computing. It is seen as an important outcome of systems design, and as a predictor for system use and adoption (Gerlach and Kuo, 1991). It has been argued that “usability determines whether people will accept the use (or purchase) of a computerized system.” (Tractinsky, 1997, 115) Moreover, the efficiency of skilled workers in the workplace is said to be related strongly to the usability of their computer systems (Li, et al., 2001).

Property view of usability

As such, usability has long been recognized as a property of an artifact that complements its functionality (Goodwin, 1987). Usability denotes the quality of software (or any artifact) with regards to its use (i.e., its interaction with the user): “Usability is a key quality of interactive systems, typically encountered negatively as system features and behaviors that obstruct usage.” (Cockton, et al., 2003, 1119) While existing definitions differ in certain details, they mostly focus on the artifact, i.e. a software or system, as an object of which usability is seen to be a property or characteristic, even if it shows itself only in an interaction situation (see appendix for a selection of widely used definitions).

Usability thus is seen as “an attribute of every product” (Dumas and Redish, 1999, 4), as “the extent to which a product can be used” (IOS Definition, cf. Bevan, 2001) or as “the ease with which people can employ a particular tool or other human-made object in order to achieve a particular goal”\(^1\). Often, software usability is then further broken down into more detailed lists of characteristics that a software needs to meet to be deemed usable, such as user relevance, efficiency, learnability etc. (Lecerof and Paternò, 1998, 864). For example, Alonso-Ríos et al. (2010), in a recent article, have criticized existing usability definitions as being too brief. Instead, the authors have developed an “exhaustive and thorough taxonomy of the attributes underlying the concept of usability”. In essence, by drawing on the extensive body of usability literature they have synthesized a comprehensive catalogue of artifact properties to describe all facets of usability in meticulous detail. Consequently, usability, while not an atomic characteristic, is seen as describing an artifact, as comprising of a set of quality attributes or properties of that artifact. Ultimately, this leads to formulating statements such as ‘a system’s usability is good/bad’ or ‘a system is usable (or not)’ (see section 2.2). This view also dominates the public perception of usability, e.g. as frequently applied in popular media discussing technology. For example, a recent article in the British Newspaper Guardian, quoting a study on Apple’s iPad (Nielsen, 2010), states that “Apple's iPad has usability problems, and shows an "overemphasis on aesthetics", according to usability guru Dr Jakob Nielsen” (Schofield, 2010).

Interaction with user and task characteristics

Typical definitions acknowledge that usability interacts with other constructs, such as user characteristics or the task context (cf. Karat, 1997). Jordan for example states “that usability is not simply a property of a product in isolation”, but “a property of the interaction between a product, a user and the task” (Jordan, 1998, 7). Hence, some authors put emphasis on the interaction of the product with user and task, because “usability is experienced by each individual person. If the person experiences a system as usable it is.” (Whiteside, et al., 1988, 808) Similarly, Gutwin defines a usable (or “effective”) system as one “that fits the needs and work practices of the users and the task situation.” (Gutwin, 1997, 150). However, these definitions still focus on the artifact and put emphasis on its characteristics in relation to the user (and a typical task for which the system was built). Usability thus is treated as a property of software, which is experienced by a user in carrying out a task.

The role of context in typical usability definitions

Some definitions further add ‘context’ (e.g. workplace culture, situational factors) as another influence factor that impacts on the usability experience. For example, the widely used ISO 9241-11 definition describes usability as “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use” (cf. Karat, 1997, 34). Surprisingly however, context as a factor is mostly overlooked or treated in very abstract ways. For example, Alonso-Ríos et al. (2010) at the end of their literature re-

\(^1\) http://en.wikipedia.org/wiki/Usability
view acknowledge that “usability of a product is not an intrinsic property but depends on the context of use” (p. 73), but the authors nevertheless see the necessity to derive formalized attributes for describing context aspects. Hence, while context is sometimes acknowledged as an influence factor, the call typically is not for a contextualized understanding of usability, but for a more complex set of variables to capture a generalized understanding of usability properties. In fact, when it comes to testing usability, context is commonly associated with methodological challenges of mapping what is observed in a use situation to product properties (cf. Dumas, 2003).

Usability as software design outcome

While some publications acknowledge that context factors can have an influence on (the experience of) usability, “usability has typically taken an engineering approach in an attempt to identify a set of principles and common practices that will insure usability is an outcome of systems design.” (Palmer, 2003, 152) It has been argued that “well-designed systems are thought to (...) enhance usability by providing interfaces and capabilities that are easy to use and conform to users’ preferred perspectives and expectations.” (Butler and Gray, 2006, 220) A design view of usability creation is not new and can be found in early publications in the field (e.g. Bennett, 1987; Kim and Moon, 1998); it manifests itself in practices of usability engineering (Howarth, et al., 2009) and is seen as an outcome of vendors’ strategic product decisions (Chen and Forman, 2006). Others have acknowledged and discussed the role of usability guidelines in systems design (Holzinger, 2005; Spool, et al., 1999; Webster and Ahuja, 2006), such as the well-known Microsoft Usability Guidelines (Venkatesh and Ramesh, 2006).

Usability Testing: Methods and Application in Practice

Following the above notion of usability as a property of an artifact, examining (or testing) usability seems to be a rather straightforward task. In fact, an entire service industry has formed around testing software quality, an important dimension of which is usability (Bevan, 1995). Different kinds of methods for determining software usability can be distinguished: Usability is either judged ‘objectively’ by examining the artifact itself (e.g. by inspection or based on formal models) or be determined in testing the artifact in the interaction between user and task (experimental/empirical testing) (Hollingsed and Novick, 2007). Based on this general distinction, typical textbooks in the field (Jacko and Sears, 2003) distinguish three approaches: 1) inspection methods (Cockton, et al., 2003), 2) model-based methods (Kieras, 2003), and 3) user-based evaluations (Dumas, 2003).

Inspection methods

Usability inspection methods are based on expert judgments; they are typically carried out by “having evaluators inspect the interface.” (Nielsen, 1994, 413) By far the most widely used inspection method is called heuristic evaluation (Rosenbaum, et al., 2000). In heuristic evaluation, an expert user inspects the software artifact using the above-mentioned usability principles or design guideline catalogues to determine usability and identify weak aspects, which subsequently need to be improved by redesigning the artifact. In another method, called cognitive walkthrough, the user’s problem solving process is modeled and simulated by the expert. The aim is “checking if the simulated user’s goals and memory content can be assumed to lead to the next correct action.” (Nielsen, 1994, 413) As it is not our intention to provide a complete method overview in this paper, we would like to refer the reader to (Nielsen and Mack, 1994) for an overview of inspection methods.

Model-based evaluation

Model-based usability evaluation takes the property-view of usability the furthest, in that it claims to completely formalize usability testing based on established, generalizable knowledge of user behavior, which then interacts with formalized artifact attributes, thus making it possible to simulate usability testing without having any human involvement: “Model-based evaluation is using a model of how a human would use a proposed system to obtain predicted usability measures by calculation or simulation.” (Kieras, 2003, 1140) It aims “to develop and apply true engineering methods for user interface design.” (Kieras, 2003, 1141) The existence of such methods signifies the strong belief held in the field that usability is ascribed to the artifact. In practice however, model-based methods seem to play no role. According to Holzinger (2005), by far the most dominant methods are the above-described heuristic inspection methods and empirical user tests facilitated by usability laboratories.
Empirical testing methods

In contrast to the above methods, empirical testing methods are mostly based on the observation and interviewing of users in laboratory experiments, which simulate interaction situations between artifact, task and user. Nielsen stresses that “usability is typically measured by having a number of test users (selected to be as representative as possible of the intended users) use the system to perform a pre-specified set of tasks” (Nielsen, 1993, 27). Usability labs for performing such tests have become a ubiquitous technology in practice and in universities (Dumas, 2003).

Dumas and Redish (1999) describe typical characteristics of usability tests: 1) the goal is to determine and improve the usability of the software, 2) participants represent real users, 3) the users do real tasks, 4) in doing so they are observed and behavior is recorded, and finally 5) the data is analyzed to recommend design changes. In doing so, such testing requires that usability properties be measured effectively (Bevan, 1995; Tractinsky, 1997). Hence, there is much concern with regards to improving the validity of testing and the generalizability of results (e.g. Dumas, 2003). Consequently, scholars have called for the need to develop more rigorous metrics for measuring software usability (Palmer, 2003), by decomposing usability in measurable sub properties of the system in question. Ultimately, the goal is “to find an absolute measure of usability, that is, a numerical measure of the usability of a product that is independent of its relationship to any other product.” (Dumas, 2003, 1094)

Moreover, as usability is ascribed as a property to the software artifact, such experiments need to be representative in relation to the actual application context, in order to achieve generalizable, valid results. Such representativeness is then mostly reduced to selecting the ‘right users’ and to engage these test users in ‘typical tasks’. While some authors have acknowledged the importance of contextual usability testing (e.g. Whiteside, et al., 1988), e.g. observations in the workplace context, such observations have generally been characterized as inferior to the typical laboratory tests. The main reasons are that “it is difficult to infer causality while observing any behavior” and that “the observer is unable to control when events occur.” (Dumas, 2003, 1096) In other words, what is observed in a workplace context cannot be mapped easily to the properties of the artifact.

In summary, we conclude that usability is predominantly defined as describing an artifact, i.e. as a characteristic or property thereof, and that it is to be tested in task-interaction situations with representative user groups and subsequently improved by redesigning the artifact. We will challenge this view and argue that usability is in fact not a property of an artifact, but should be conceived of as embedded in the sociomaterial use context of the workplace. Scholars in neighboring communities (e.g. in human factors research) have long argued for a more socialized view of software design. But proponents of more contextualized usability methods have been marginalized in the field so far. We will argue that such methods might be the only ones that can truly capture usability as experienced by users in their every-day workplaces, which is after all what matters to these users. After discussing our case from a sociomateriality viewpoint in the next sections, we will discuss the role and merit of contextualized usability methods in the discussion and implications sections.

Sociomateriality

In order to inform our argument we draw on sociomateriality, which was introduced by Orlikowski and Scott (2008a; 2008b) as an alternative way of viewing the world of technology and its interactions with organizations. Please note that we will not provide a full account of sociomateriality and its origins, but will introduce its core argument to inform our line of reasoning.

Sociomateriality has been proposed as an alternative worldview to the long established, traditional way of viewing the world as consisting of individual, distinct entities with stable, largely unambiguous properties. In their paper, Orlikowski and Scott begin by portraying this established worldview, referred to as the individualist ontology. This individualist view, also termed entity view or artifact view, sees the world as consisting of “discrete, self-standing entities” (Orlikowski and Scott, 2008b, 27; Suchmann, 2007, 263). The authors further argue that by doing so, this view “objectifies the technical and sequesterates the social” (Orlikowski and Scott, 2008b, 27). Consequently, the associated stream of Information Systems research is portrayed as reflecting “an ontological commitment to a world of individual entities that have some inherent and relatively stable characteristics.” (Orlikowski and Scott, 2008b, 6) Under this notion, “technology is treated as a specific and relatively distinct entity that interacts with various aspects of the organization.” (Orlikowski and Scott, 2008b, 7) Unquestionably, the above-introduced understanding of software usability is very much a product of this ontological position. Usability is treated as a property or characteristic
of the individual software artifact. The software artifact in turn can be described in terms of material features and other more qualitative characteristics (such as usability).

However, as others have argued before, focusing on the individual artifact and its characteristics might lead to applying an inappropriate materialism and technological determinism (Barley, 1988). Technological determinism refers to a view, whereby the artifact is conceptualized as an independent variable with certain characteristics that impacts on the social world (organizations, groups, individuals) and subsequently leads to certain effects (modeled typically as dependent variables). Under this notion, technology (and its characteristics, such as usability) determines certain outcomes in the organizational context. The above notion implies that usability can indeed be tested in experiments and improved by way of design, which will ultimately yield improvements in the intended outcomes in context, such as user productivity.

In their paper, Orlikowski and Scott strongly argue against an individualist/entity view of the world. It is worth mentioning that they are at the same time equally critical of the opposite view, an overtly social view of the world in which the social determines the technical. Rather, Orlikowski and Scott argue that the technological and the social are inherently inseparable. This worldview stresses the embeddedness of technology in its social context, or what the authors term ‘constitutive entanglement’, whereby the technology becomes entangled with the social context and institutional ideas of applying the technology. The focus in this worldview is on the “dynamic interactions between people (or organizations) and technology over time.” (Orlikowski and Scott, 2008b, 13)

In this view “entities have no inherent properties, but acquire their form and attributes only through relations with others.” (Orlikowski and Scott, 2008b, 21) Essentially, values, identities, cultural notions, and organizational ideas become entangled with the use of software. To this effect, the same technology may be perceived entirely differently and used in ways so different that conceptualizing the artifact independent of this entanglement becomes unfeasible. Consequently, the authors stress that in order to research and understand technologies one must examine and understand the sociomaterial configurations of work practices in context. In fact, they argue that work practices “are inherently sociomaterial, and so to understand work, we must understand its sociomaterial configuration.” (Orlikowski and Scott, 2008b, 40) Ultimately, this worldview poses that one should reconsider “the status of technology (based on) a grounding in relationality” (Orlikowski and Scott, 2008b, 41), i.e. relative to the material and social context of application and use. Or as Roberts and Grabowski (1996) have argued, generalized accounts of technology and their effects are no longer appropriate, as one loses the “possibility of seeing the technical and social as inextricably intermingled.” (Orlikowski and Scott, 2008b, 27).

We will draw on this notion in arguing that usability should be treated not as a property of an artifact but as being embedded in a sociomaterial use context, as software becomes entangled with the material (hardware) and social (ideas, norms etc.). In the next section we introduce our case company, the usability project we were asked to undertake and the software. We then proceed by illustrating how this software becomes entangled in the sociomaterial use contexts across different organisations.

**The Case Project**

In order to illustrate our argument we draw on experiences from an industry project we undertook with the aim to “enquire about and improve the usability” of a software solution. The software in question is an IP-based telephony software product for small and medium-sized companies. The authors were approached by the software company to help with examining and improving the usability of the client component of this software solution, i.e. the software with which end users interact in the workplaces. Our argument is informed by this project; we will briefly introduce the company, its product and the project.

**Swyx and its product**

Swyx (www.swyx.com), founded in 1999, is a typical technology company. The company has developed an IP-based telephony software product for small and medium-sized companies. The authors were approached by the software company to help with examining and improving the usability of the client component of this software solution, i.e. the software with which end users interact in the workplaces. Our argument is informed by this project; we will briefly introduce the company, its product and the project.
SwyxServer component contains the telephony logic and handles all phone traffic. It carries out all the functions of a traditional PBX (public branch exchange), which connects a company phone network to the public phone networks (i.e., traditional PSTN (ISDN) networks).

More important for this paper is the end-user software client, called SwyxIt!. This is a MS Windows client, which turns a PC into an advanced telephone (called a softphone). This component is the one with which the users interact when placing, receiving or diverting calls. It is also used for advanced features, such as setting up complex call routes for diverting incoming calls based on caller ID or time of day. SwyxIt! allows to assign frequently used contacts to quick-dial buttons. When doing so users are provided with presence information: a contact is displayed as available (green), currently engaged (red) or not logged in (grey) depending on their softphone status.

Importantly, the client software connects and interacts with different types of hardware, such as: 1) a headset (combination of headphone and microphone), 2) a handset (essentially a simple phone receiver without dial pad), both connected to the computer via USB, or 3) a native/stand-alone IP telephone, which can work independently of a computer, but be linked using computer telephony integration (CTI).

The Usability project

The starting point of our project was Swyx’s desire to enhance the functionality and most importantly the usability of its SwyxIt! client. Not too long before embarking on this project, a consulting firm had tested the usability of the client software by carrying out user experiments. This test took place in a laboratory setting, where test users had to solve predefined tasks. The results of this test led to design changes in the software. However, according to the company, many of these changes, especially the more visible ones, had not been received well by the majority of users at the time. Consequently, the company received rather mixed feedback from the marketplace.

At this stage it is worth noting that we will not illustrate the results of these tests nor the particular changes made to the software, as this is not the focus of this paper. However, the experiences from this project prompted the company to approach us to gain a better understanding of the reasons for these complaints and how software usability could be improved.

As a consequence, we decided to go out into the field and interview and visit actual users of the software in their workplace contexts. Ultimately, the aim of the interviews was to examine software usability, as perceived by the users, and to report back a verdict to the company, based on the analysis of data collected in those interviews. However, when judged solely against this original goal, our project clearly has to be regarded a failure, as we were not able to establish any coherent understanding of the usability of the client software. After visiting seven user organizations and interviewing 29 users in their workplaces, we had been confronted with such a variety of viewpoints that establishing a notion of usability as a characteristic of the software turned out an impossible task. In fact, we were unable to even establish what accounted for as the relevant artifact our study needed to focus on, as the software interacts with different kinds of hardware and is perceived very differently across the cases.

Only by drawing on the notion of sociomateriality as described above, did we progress in making sense of and interpret what we experienced in our encounters with the users in their workplaces. Ultimately, it leads us to argue why usability as a concept viewed across the cases is problematic. Before we elaborate on our case findings, we will briefly describe our initial data collection method and how we made sense of the data collected.

Data collection and analysis

As part of our usability case study, we interviewed users from different industries in their daily work contexts. In total, we conducted 29 interviews with users from seven companies. These interviews lasted between 45 min and 90 min and were all tape-recorded and transcribed. Besides this, we used a range of other data sources to inform our understanding of the software, its features, the company’s strategy etc. In doing so, we had access to a version of the software itself, user handbooks, various design screenshots, Powerpoint presentations, and the company website with materials for clients and business partners. We also conducted a workshop with company representatives, where we were presented with the software, the company’s intentions for future developments and its aims in carrying out this study. Equipped with a rich background understanding, we went out into the field.

Our user interviews were based on an interview concept map, which we derived to comprise all aspects that were deemed relevant for enquiring about usability. This included ways of using the software, perceived usability of dif-
different features and concepts (i.e., properties), existing problems in using the software, user suggestions for future developments, as well as an overall assessment of the SwyxIt! client software.

The data derived from the interviews was then analyzed with regards to deriving usability suggestions for the company. Specifically, we analyzed the interview transcripts and notes taken of what we perceived in the users’ workplaces (e.g. the users demonstrating use) with regards to answering two main questions for the company: 1) which elements and functionalities are used in each case (and in which way)? 2) What is the perceived usability of the different concepts and functionalities?

However, our findings remained on the one hand very general and were more related to suggestions for new features; on the other hand they included a list of specific problems encountered by the various individual organizations. Hence, while we presented some results to the company, we were far from deriving a coherent picture from the interviews. The variance of observed software use and what users told us about how they perceived the solution was quite staggering\(^2\). In the following, we will present findings from five selected user organizations. We decided to limit our discussion to five cases due to space restrictions and because the two cases we left out were quite similar to two of the cases presented below. As it is not our intention to provide a complete picture but to point out the variety of use and variance in configurations in context, this approach seems appropriate.

Illustrating the Sociomateriality of SwyxIt!-in-use

Swyx sees its SwyxIt! client as the centre piece with which the users interact. Hence, its focus in creating and improving the user experience (i.e., usability) is on this software artifact. However, our interviews with both the Swyx representatives as well as the users across organizations have shown that the software cannot be viewed independent of the hardware with which it interacts, as the hardware impacts considerably on what software functionality can be used and how the software is perceived and used. More importantly, our experiences gained from the field study show that the actual configuration of hardware and software adopted by a particular organization is also highly context dependent, i.e. influenced by organizational culture and the institutional ideas surrounding the role and nature of telephony in the organization. Hence, in much the same way as the software interacts with hardware, which as a particular impact on functionality, perception and use of the solution, the bundle of hardware and software further interacts with the social context. The context influences not only the selection of a particular software/hardware bundle, but also how it is used. In the following we will firstly elaborate on the interaction of software with hardware, before we discuss the software/hardware bundle in the context of different social contexts as represented by our user organizations.

Entanglement of Software with Hardware

As mentioned above, a range of hardware devices can be connected to or used in conjunction with the SwyxIt! client for enabling IP-based telephony. We have identified four different configurations, which not only represent different hardware/software bundles, but also differing ideas of telephony itself. At the one end of the spectrum there is a purely software and computer-based solution, which represents a new way of relating to telephony (as a multi-faceted services with a range of new possibilities). At the other end of the spectrum, IP telephony can be used without a computer by way of using a stand-alone IP-enabled desktop phone. This configuration represents the old telephony world adapted to the new reality of the IP-based backend. In between these two, we identified another two configurations (see table 1). Depending on the configurations, different features are available to the user.

Generally, the purely computer-based solution breaks with the traditional telephony view of being associated with a dedicated artifact. Telephony is used as a software-based service on the computer. The second configuration is similar in many ways, but retains the handset as the main material artifact associated with telephony. Consequently, users in our cases are drawn towards the handset for initiating and receiving phone calls (lifting the receiver will bring the software to the front, the user then needs to dial via the computer keyboard) rather than the software client. In both configurations, dialing a number needs to be done via the software/computer keyboard.

\(^2\) Please note that we do not believe that we would have been able to get a more a coherent picture had we visited more companies. It is likely that variance would have been even higher.
The third configuration attaches to the software client a full-featured IP telephone (a range of models exist) by way of computer-telephony-integration (CTI). In this configuration, the users can access a range of advanced telephony functions from both the IP phone and the software. Also, software and phone act in unison: existing calls can be seamlessly transferred between computer and phone and both devices provide similar functionality. But this configuration comes at a cost, in that CTI, for technical reasons, disables the software’s application sharing and call recording features. Moreover, as we will show below, it very much depends on a particular social context how this configuration is used, e.g. whether the user’s focus is on the telephone hardware or the client software. In any case, an interaction exists between the hardware attached and the kinds of features used in the software.

For example, in cases where users use an IP phone in connection with the software, day-to-day routines such as switching on/off of the call diversion or voice box features, checking the call register, or simply initiating and receiving a call are quite often carried out on the phone and not in the software. This raises the question, how one can realistically evaluate software usability, when the range of features used and their importance and role for the user depends on the hardware linked with the software, and more importantly the social context in which usage evolves.

### Table 1: Variation in software/hardware configurations.

<table>
<thead>
<tr>
<th>Software/Hardware bundle</th>
<th>Metaphor</th>
<th>Impact on functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW client + USB/wireless headset</td>
<td>“Pure play new world”</td>
<td>No physical dial pad, or other telephony features.</td>
</tr>
<tr>
<td>SW client + USB handset</td>
<td>“New world, old feel”</td>
<td>Physical handset to initiate and receive phone calls.</td>
</tr>
<tr>
<td>SW client + CTI w/ IP phone</td>
<td>“Mix and match”</td>
<td>Wide range of features on telephone. Application sharing &amp; call recording disabled. Independence from computer.</td>
</tr>
<tr>
<td>Stand-alone IP phone³</td>
<td>“Old world resemblance”</td>
<td>No software functionality.</td>
</tr>
</tbody>
</table>

### Entanglement of Software/Hardware bundle with social context

We have selected five user organizations to illustrate the striking differences in hardware/software configurations and, more importantly, in perception and use of the SwyxWare solution across contexts. Table 2 provides a brief summary of these use cases and their particularities. The five cases can be characterized by the differing role hardware plays in relation to the software part of the bundle. While users in some cases (e.g. the City Council case) have a strong hardware focus, mainly drawing on telephony features implemented in the desktop phone, in other cases (e.g. the Multimedia Company), users have appropriated telephony as a software service with a headset to enable speech transmission. We will describe each case, albeit in differing detail to cater for their particularities.

**City Council**

This is a typical city council organization in a small German country town. SwyxWare is servicing all civil servants, those in back office roles as well as those dealing with clients (e.g. receivers of social security payments). Decision makers have opted for a full roll-out of IP phones for every workplace, with some having stand-alone IP phones, while most workplaces have the SwyxIt! client software on their computer, connected to the IP phone via CTI.

The case poses interesting insights into the social acceptability of IP-based telephony, in particular the use of headsets and with it a more software-focused configuration of the solution. All interview partners stress the fact that headsets are “not acceptable socially” in this environment, which has been described as traditional and conservative. This can best be illustrated with the workplace situation of one of the front line employees in the social service section. The employee needs to use the phone often and extensively in order to talk to or enquire about his clients. In doing so, he often needs to walk to a drawer to fetch a file or to skim through piles of papers while on the phone. Hence, he would very much like to use and certainly benefit from a wireless headset. However, as a considerable

³ Please note that we will not elaborate in detail on this option as this is out of focus, as no software client is needed.
proportion of his daily work involves meeting with clients in his office, a headset is out of the question as this would, in the interviewees’ words, “increase the (social) distance” to his clients, ultimately interfering with his work. Much in the same way, the wearing of headsets was seen by all interviewees as socially not acceptable in the city council environment (“…because with a headset I would feel like working in a call center”).

The adopted software/hardware bundle and the rather conservative organizational culture was subsequently reflected in the usage practices reported in the interviews and observed during our visit: The user focus is clearly on the desktop phone to the extend that for most users the software only serves as a panel with quick dial buttons and for the occasional change of a phone setting. Quick dials were reported to be a valuable, but for some users the only function the SwyxIt! client served. Moreover, while user acceptance of headsets did not exist, much to the contrary the phone itself was said to serve as a “status symbol”, with better phones (with larger displays) being a visual representation of user position in the organizational hierarchy.

IT Distributor

This company has also implemented a combination of IP desktop phones and SwyxIt! software client (with CTI). This CTI-based configuration is embedded in user practices that mostly evolve around the desktop telephone, with call initiation and receiving generally realized through picking up the receiver. Also, users tend to use the quick dial buttons on the phone, and some of the more advanced call management tasks are also realized using the hardware. However, unlike in the first case, users occasionally switch to the headset during longer conversation and some move their focus seamlessly between phone and software, so that the software plays a greater role in this case. When being asked questions with regards to their perception of the telephony solution (i.e., its role), users still perceive it largely as a telephone, albeit as a modern one.

Communications Technology Company

In this company, users have been given CTI-connected IP phones, while most users also actively use wireless headsets. While in this case the hardware/software bundle is essentially the same as in the previous case, it has been adopted and incorporated in the user practices in a very different way, as the users’ focus is clearly on using the software and its (advanced) telephony features. In fact, the reasons for adopting this particular bundle lie in the specific context of the company. Reportedly, the main reason for using CTI is for the phones to serve as the phone bell, as this frees the computers’ soundboards from having to play ringtones, since users depend on the availability of this piece of hardware for doing their jobs. For the users, the solution goes beyond a traditional telephone and is perceived more in terms of its software characteristics.

Not surprisingly, the comparison with the IT Distributor case shows that the same software/hardware bundle can be perceived and appropriated rather differently when placed in different social context. The implications again are that software usability in these contexts, as users draw on rather different sets of features with varying frequency, will eventually mean very different things.

Private University

This organization is a provider of tertiary business education (e.g. BBA and MBA). It has rolled out SwyxWare throughout the entire organization. One reason was to seamlessly connect the various German campuses with one solution. Almost all users have access to a computer with a USB handset; headsets are not commonly used. Full-featured IP phones with CTI phones have not been acquired for two reasons, the first is cost, the second is that CTI disables certain software features. In particular, the school’s employees frequently need to use call recording when carrying out research interviews over the phone.

Usage practices evolve around the software client and its (advanced) features, as the USB handset only serves as the speech transmission piece. The solution is perceived by its users as “a telephony replacement”, or a “database” (of contact details), which “offers more features” than a telephone. While the users still compare the solution with a traditional telephone, they acknowledge that it is different and goes beyond traditional telephony both in its range of features, as well as in the ways of using it.
Multimedia company

The last case company again has adopted a computer-based solution, but without any material reference to a traditional telephone, e.g. there are no artifacts on the users’ desks reminiscent of a telephone. The majority of users use wireless headsets. The organizational context of this company was described as innovative, technology-enabled, with people being highly computer savvy. This is reflected in the practice of personalizing the client software’s appearance, as a number of users made use of the client’s advanced customization features. Some have amended its visual appearance to better suit their needs; many users have assigned telephone features as shortcuts to certain parts of their computer keyboards.

All in all, it became clear that this company had broken with the traditional telephony world, as represented in the respective material artifacts. The nature of the computer-based communication solution seemed to be in unison with the organization’s culture. Accordingly, interviewees explained that the solution “is a software, which can act like a phone, … but it is not a telephone. I never perceived it that way, … it is much more than a telephone.”, or: “It is a piece of innovation, which simply allows to do more than a telephone.” Not surprisingly, from our sample users in this company used the widest range of advanced telephony features.

<table>
<thead>
<tr>
<th>Table 2: Comparison of configuration, perception and use across the user organizations.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Context</strong></td>
</tr>
<tr>
<td>Configuration</td>
</tr>
<tr>
<td>Software/hardware bundle</td>
</tr>
<tr>
<td>Use patterns</td>
</tr>
<tr>
<td>Perception as:</td>
</tr>
</tbody>
</table>

Discussion

The presentation of the adoption, use and perception of the Swyx telephony solution in the above cases serves the purpose of illustrating that software cannot be usefully conceptualized as a bundle of features with certain objectively agreed upon properties, one of which is usability. The adoption of several different configurations of software/hardware bundles across the cases, and their interaction with particular institutionalized ideas about telephony and the associated acceptability of certain practices, reveals that what might count as usability is certainly very different across such contexts and not necessarily focused on the software artifact.

**Contextual, non-individualistic individuation of software**

Based on our observations we argue that software can only be understood in context and in a non-individualistic way. We have shown that software is not experienced by its users independently from the connected apparatus of telephony hardware, as this can change considerably what features are available and how they might be used. Hence, we argue that a piece of software by itself should not be specified/conceptualized as an artifact in its own right, be-

---

4 Except for those who need to use Apple hardware, who are forced to use a stand-alone IP phone as there is no Mac client software yet. As no software client is involved this group was excluded from our sample.
cause, with regards to its use, it does not possess a stable set of characteristics independent of its hardware settings. Secondly, and much in the same way, we argue that a software/hardware bundle should not be viewed as an artifact, because (with regards to its use) it does not possess a stable set of relevant characteristics independent of the social setting with its institutional ideas. Consequently, if we cannot establish objective and stable artifact properties, we likewise cannot establish an objective set of usability measurements and instruments to be applied independent of the sociomaterial use context.

Such a position is consistent with prior works on the social construction of technology, which poses that software artifacts inhibit a certain interpretive flexibility (Pinch and Bijker, 1987), which gives an artifact the ability to represent “different things to different actors” (Law and Callon, 1992, 24). As such, interpretive flexibility is manifested in the variety of ways in which people think of, interpret, or inscribe meaning to a software artifact across different social contexts (Doherty, et al., 2006). While interpretive flexibility as a concept can suitably be applied in the above cases, our argument goes one step further than assuming that varying interpretations of the same artifact might exist. We question altogether the usefulness of conceiving of software as distinguishable artifacts with regards to understanding usage. As discussed above, sociomateriality suggests that software becomes inseparably entangled with the use context, i.e. other tools in the user’s workplace, as well as social ideas, norms and work practices.

Of course, an entity view of the world in general and of software in particular is often appealing and sometimes necessary. For example, software vendors need to identify and portray their software as a marketable entity with properties that can be compared with other/similar software products. Designers who are building software need to specify their product in terms of requirements and features. And even users need to sometimes individuate certain pieces of software in the workplace, e.g. when discussing problems or identifying wishes for change. However, our argument is that such an individuation of software as an entity can only happen in a non-individualistic and situated manner, relative to and on the basis of the background of other software, hardware, ideas, norms and practices, i.e. the sociomateriality of a concrete workplace, and not in absolute and generalizable ways.

The issue of non-individualistic individuation of tools has been recognized in other fields, such as history and archaeology, where people are confronted with the task to interpret the function of artifacts from long-gone cultures. It has been argued that tools, as human-made artifacts, generally cannot be fully described and understood on the basis of their inherent properties alone. Rather, tools derive their significance and meaning from their place within human practices. Preston, in criticizing long-standing believes with regards to tool use, has argued that, in order to understand any artifactual object, “you have to be acquainted with supra-individual structures of the social and physical world in which the object originated.” (Preston, 1998, 521) In essence, the author argues that tools cannot be made intelligible based on their properties alone, but that one has to know what it is used for, how it is being used, and the purpose it serves. In our Swyx case, any individuation of the software artifact very much depends on the situational context. In doing so, it becomes obvious that the software product is perceived very differently by the software vendor on the one hand, and the various users in the various use contexts on the other hand. In essence, how this software is encountered (indeed, what it is) very much depends on the situational ideas of telephony, norms of communication, typical forms of relating to and the social acceptability of telephony artifacts (e.g. the use of headsets).

We conclude that workplace software cannot be understood by itself, it cannot be individuated individualistically; it is entangled in the sociomaterial use context and can only ever be suitably individuated within such contexts and in very situated ways. Hence, we believe that, while sometimes necessary, an individuation of software as artifacts is always a simplification and not without risk, as ways of using the software cannot be derived from its properties. In the same way, usability cannot be generalized from user-task-interactions as tested in a laboratory setting.

**Manifestation of Usability in context**

With regards to usability our case study reveals the following: in cases where the user-focus is more on the hardware (e.g. case 1), usability of the telephony solution is very much an issue of hardware usability, while in turn in some cases (e.g. case 5) usability in context is very much geared towards software. Moreover, due to the differences in perception, adoption and user practices, users draw on different features in different ways, which renders assessing usability as a universal property of the artifact a pointless task.

Consequently, we argue that usability of workplace software simply cannot be understood nor tested in artificial laboratory settings. Such products become deeply entrenched in the sociomateriality of user practices in context. In doing so, people find creative ways of working with a software (Orlikowski, 2000), very much as the context shapes the perception and use of the software and its embedded concepts (e.g. telephony). Consequently, what accounts as
usability within a concrete use context will inevitably differ from what a user perceives who interacts with software for the first time, e.g. when trying to solve a task in an experimental situation. For example, while users might not be able to fulfill the assigned tasks in the experiment, as they might not know the best way of using the software, in their day-to-day practices in the workplace users might have already acquired the necessary skill to work with the software in very effective ways and thus perceive matters of usability vastly differently. Whiteside et al, who have advocated a more contextual view of usability, state: “In the laboratory, subjects perform tasks prescribed by the experimenter. In the workplace, people perform tasks important to their careers and livelihood. What shows up and what is concealed against these different backgrounds?” (Whiteside, et al., 1988, 806) However, usability testing of routine-use of software in the workplace is vastly underrepresented in the field, as has been acknowledged by (Dumas, 2003); usability testing largely focuses on initial use, by applying lab testing methods.

All in all, we conclude that the above cases and their respective software appropriations and usage practices serve as a good example of sociomateriality in action. Drawing on the above findings allows us to argue that, since it is not possible to suitably individuate software as an artifact with a stable set of properties, it is hence not possible to measure/define in certain terms the usability of that artifact. Consequently, we further argue that the unit of study with regards to usability should really be the ICT-enabled work practice in context. What is needed is a situational understanding of usability, which has been described in the literature before, but largely been marginalized so far. Hertzum, in describing different images of usability, describes a situational view as being “about the complete use situation (…), situational usability implies a distinctive socio-technical focus.” (Hertzum, 2010, 4) Obviously, such a situated and relational view of usability has implications for how usability studies can usefully be carried out and how they can feed into software design.

**Methodological Implications**

Our findings suggest that usability testing of workplace software in laboratory experiments does not work and might actually be counterproductive, as it might produce results that are detrimental to the ways in which usability manifests in the sociomaterial use context. In fact, software vendors who want to make strategic product decisions based on the outcomes of such de-contextualized usability tests risk to develop their products in directions that do not correspond with the ways in which users embed the product in their everyday work practices. A practical dilemma resides in the observation that, while our analysis points to a need for more contextual usability methods, at the same time, the most widely used methods treat context as a source of bias and as a threat to the validity and generalizability of their test results.

**Context in traditional usability testing methods**

Experimental usability testing methods aim to objectify and generalize their findings in the name of validity (Dumas, 2003). To this end, the selection of tasks and test users is done to achieve representativeness and neutrality, so as to ensure that what is measured is a product of the artifact and not of specific characteristics of the user, the task situation or the context in which usability is being measured. Consequently, all of these components are seen as potential sources for introducing bias to the test, hence compromising its validity. For example, experienced users are often seen to introduce bias to usability studies (Dumas, 2003), in that they might be able to work with the software effectively in spite of existing “usability problems”. According to our line-of-argument, such thinking points in the exact opposite direction of what is required to do the nature of usability justice. In our view, usability should be treated as a distinctly contextual phenomenon.

Moreover, a number of recent studies (e.g. Hertzum and Jacobsen, 2003; Molich, et al., 2004) have found that experimental usability methods tend to be very unreliable. In the above studies, a number of professional usability labs tested the same artifact with typical users. In one study all seven labs together identified a total of 310 usability problems, but only one problem was identified by all seven labs, while 75% of problems were only identified by one lab (Molich, et al., 2004). While such results are very discouraging for proponents of laboratory testing methods, the authors however largely attribute the observed variance in results to methodological challenges, which leads to a call for more rigorous method development in the name of validity. The variance is seen as resulting from a lack of rigor in method design and application. However, our findings suggest that these problems might have a much deeper cause, in that the randomness of results might be a product of the decontextualised test situation, which is deprived of all everyday meaning and relevance to the users.
Need for contextual usability methods

Our findings point to a need for usability enquiries in the workplace context, in a way that enables doing justice to the entanglement of the software in the everyday use practices as a way to elicit a more holistic picture of usability. Proponents of contextual methods have long argued that laboratory studies “may not accurately represent how a system is used in the work environment.” (Kukla, et al., 1992, 45) Contextual usability studies on the other hand offer “a much more complete understanding of how users do their work.” (Kukla, et al., 1992, 45) For example, it has been argued that ethnographic fieldwork can enable usability enquiries in context (e.g. Sperschneider and Bagger, 2003).

Considering context in that respect has been described as taking a user-centric perspective to “step into their shoes, and to take a rich user perspective.” (Stappers and Visser, 2007, 2) This perspective however needs to be inherently social and not individual, since usage practices (cf. Orlikowski, 2002; Reckwitz, 2002) are social and shared by people in a workplace: “what people do takes on its meaning and can only be recognized within a social context.” (Brown and Duguid, 1992, 172) We suggest that future research needs to explore to what extent existing methods in related domains, such as the Industrial Design field, might be applied to the usability phenomenon. One such technique is called contextmapping and describes a way to elicit concrete context attributes as inputs for design activities (Stappers, et al., 2004; Stappers and Visser, 2007). It might be possible to combine such methods with ethnographic enquiries to elicit a more grounded understanding of usability that might then generate valuable input for software design activities.

However, while we see contextual usability methods as key to capture a sociomaterial understanding of usability, such methods have largely been marginalized today. For example, Hertzum (2010) in describing situational usability only draws on references from the late 1980s and early 1990s. The question arises why the usability mainstream today is dominated by such a decontextualized view, given that this was not always the case. Enid Mumford’s account of the evolution of socio-technical design might offer an explanation (Mumford, 2000; Mumford, 2006). Mumford describes a strong shift away from socio-technical design philosophies, which are inclusive of user and contextual settings and which were quite common in the 1980s. According to the author, this shift away from socio-technical design was largely driven by increases in efficiency concerns across most industries, due to the competitive pressures of the 1990s. Against the backdrop of such developments, laboratory methods appear as both more rigorous and less costly than contextual usability methods, which seems to appeal to practitioners in a world characterized by time and cost constraints.

It remains to be seen, if the recent sociomateriality movement in the Information Systems discipline can become a vehicle to revitalize a contextual view of usability. While our findings certainly point to a need for applying more contextualized methods, a new strand of research will have to further develop and refine such methods both with regards to novel software technologies, as well in the face of practical applicability, since contextual methods are perceived by many to be cumbersome, costly and less rigorous as laboratory methods. Henceforth, design oriented, method-developing research is needed to find pragmatic ways to acknowledge that usability is embedded in context, while still leading to useful, applicable results.

Holistic understanding of ‘designing-for-usability’

Some scholars in the software engineering domain, e.g. in fields such as participatory design (e.g. Kensing and Blomberg, 1998; Muller and Kuhn, 1993) or human factors (e.g. Salvendy, 2006), have argued for contextual design approaches, where software is developed together with users and thus entangled and contextually embedded at the same time. Hence, such approaches treat usability in much the same way as we have argued, because it is an outcome of a joint, contextual design process. However, contextual design seems only feasible for custom-built software; it is hardly possible for software products such as SwyxWare, which aim to be applied across a range of contexts. At the same time however developers of such software products need to derive some form of design input from usability studies, even if lab-based studies are deemed inappropriate for the above-discussed reasons. At this point it needs to be pointed out that we do not question the influence of software design on usability as experienced by users in context. The fact that usability is not a property of the artifact does not mean that software design does not influence usability at all; it is however not the sole driver, as usability is a relational and contextual phenomenon. Again, more method developing research is needed to make findings of contextual usability studies more accessible for developers.
Moreover, the above discussion points to another important insight into how usability might be achieved in context. We have established that usability, while influenced by it, is not a direct outcome of the software design process. Rather, it should be treated more holistically as an outcome and hence an important goal of the implementation/roll-out process in context. This is especially true for packaged software products such as SwyxWare, which need to work across different contexts. It is thus important to entangle software with both hardware and social ideas in such a way that usability manifests for the users in working with the software/hardware bundle in their social work practices. This will require knowledgeable people and an active rollout process, which treats the application of workplace software as a participatory (sociomaterial) design endeavor, even if it concerns the rollout of packaged software. Hence, a holistic understanding of designing-for-usability perceives usability as the outcome of a sociomaterial intervention in context, which has to focus on both the material (hardware/software) and the social (work practices) at the same time. Again, more research is needed, as the sociomateriality concept does not yet appear to be mature and detailed enough to provide guidance for such design-oriented endeavors.

Limitations

Our paper has certain limitations. First, it needs to be noted that we only focus on one particular type of software, i.e. workplace software. We believe that our argument applies well to such software that is being used on a regular basis, i.e. software solutions that become part of user practices in the workplace. On the other hand, a range of software exists, which is intended to work in one-off situations, in which users need to understand and work with the software immediately, without having the possibility to learn, experiment and entangle. Certainly, for this type of software (e.g. many web-sites) well-designed user experiments will present useful and applicable insights for improvements solely on the basis of design. However, this presents opportunities for further research in terms of finding useful software typologies, e.g. with regards to the degree of property stability and required entanglement.

Second, we illustrated our argument with one software product only, albeit drawing on a range of user organizations. While we believe that such a case study is a valid approach for illustrating our argument, of course, more work is needed to look at different kinds of workplace software in order to further explore the notion of sociomateriality and entanglement of usability with use contexts. In essence, more research is needed to test our theory in different contexts. We do believe that conceiving of usability as not being a stable property, but being embedded in the sociomaterial use context does indeed have merit, but it will be necessary to further explore the boundaries of that theory, e.g. by carrying out case studies on different types of artifacts and across various use contexts.

Conclusion

In this paper we have argued that, at least for a certain type of software (i.e., software, which is used in the workplace on an ongoing basis as part of user practices) it is neither useful nor sensible to conceive of usability as a property of that software artifact. In fact, we have questioned the usefulness of individuating (workplace) software as artifacts with stable sets of properties for that matter. Moreover, we believe that there might be merit in extending our notion beyond the above type of software, but further empirical research is needed to explore the transferability of our argument.

We have used a case study, with a set of user organizations, which have illustrated our argument as they revealed striking differences in interactions of the software with hardware settings and social/institutional ideas and norms across the use contexts. Usability in these cases means rather different things, it results from very different elements of the software/hardware bundle and it can consequently only be understood within each of the particular use contexts, i.e. against the sociomaterial background of social norms and work practices. Hence, usability is to be perceived as embedded in the sociomaterial use context, according to the notion introduced by Orlikowski and Scott (2008a, 2008b). Overall, with our paper we hope to make a contribution to a better understanding of the multifaceted nature of software, be it perceived as an artifact or as an integral part of a sociomaterial use context.

Finally, we see further needs to improve our understanding of sociomateriality, since the concept has only recently made inroads into the Information Systems field. While generally useful to inform arguments such as ours, it needs to be noted that it is not yet a well-specified concept. With regards to future theory-developing research in this field, the concept will have to be further explored and described.
Acknowledgements

The authors would like to thank Swyx for generously providing access to user organizations and for their support in undertaking this research, as well as the interviewees in the seven user organizations for sharing their insights with us. We also want to thank Prof. Dr. Robert B. Johnston, University College Dublin, for his valuable ideas and input, which helped tremendously in shaping our line-of-argument.

Appendix A: Selection of commonly used usability definitions

<table>
<thead>
<tr>
<th>Table 3: Usability Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Definition</strong></td>
</tr>
<tr>
<td>“Usability: the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use”</td>
</tr>
<tr>
<td>“… usability issues can be thought of as pertaining to how easy a product is to use, i.e. they are to do with the ‘user-friendliness’ of a product.”</td>
</tr>
<tr>
<td>“Usability is a term used to denote the ease with which people can employ a particular tool or other human-made object in order to achieve a particular goal. (...) In human-computer interaction and computer science, usability often refers to the elegance and clarity with which the interaction with a computer program or a web site is designed.”</td>
</tr>
<tr>
<td>“Usability is a quality that many products possess, but many, many more lack.” (p.3)</td>
</tr>
<tr>
<td>“(...) when a product or service is truly usable, the user can do what he or she wants to do the way he or she expects to be able to do it, without hindrance, hesitation, or questions.” (p.4)</td>
</tr>
<tr>
<td>“Usability is a quality attribute that assesses how easy user interfaces are to use. The word &quot;usability&quot; also refers to methods for improving ease-of-use during the design process.”</td>
</tr>
<tr>
<td>“It is important to realize that usability is not a single, one-dimensional property of a user interface. Usability has multiple components and is traditionally associated with (...) five attributes: Learnability, Efficiency, Memorability, Errors, Satisfaction”</td>
</tr>
<tr>
<td>“Usability is an attribute of every product – just like functionality. Functionality refers to what the product can do. Testing functionality means making sure that the product works according to specifications. Usability refers to how people work with the product. Testing usability means making sure that people can find and work with the functions to meet their needs.”</td>
</tr>
<tr>
<td>“Usability is most often defined as the ease of use and acceptability of a system for a particular class of users carrying out specific tasks in a specific environment.”</td>
</tr>
</tbody>
</table>
References


