Available online at www.sciencedirect.com
SCIENCE DIRECT®
Interacting with Computers 17 (2005) 317–341
Interacting with Computers
www.elsevier.com/locate/intcom

Semiotic engineering: bringing designers and users together at interaction time

Clarisse Sieckenius de Souza*

Departamento de Informática, PUC-Rio, Brazil

Received 3 December 2003; revised 4 December 2004; accepted 10 January 2005

Abstract

Semiotic engineering is a semiotic theory of human–computer interaction, where interactive computer systems are viewed as one-shot messages sent from designers to users. Through the system's interface, in many direct and indirect ways, designers are telling the users how they can, should, or must interact with the system in order to achieve a particular range of goals anticipated at design time. *Designers* are thus active interlocutors at human–computer interaction time. Their interactive discourse is delivered implicitly and/or explicitly by the system, which constitutes the *designer's deputy*. The importance of bringing designers and users together at interaction time springs from the intellectual nature of software artifacts. They are the result of human reasoning, choice and decision, rather than the direct effect of universal or natural laws. An adequate understanding of interactive artifacts depends on apprehending and comprehending the human intellect in action. Hence, in addition to *producing* interactive artifacts, designers must also *introduce* them appropriately, as is the case of other intellectual products. In this paper, we show how semiotic engineering can provide substantial theoretic support for viewing and exploring design possibilities brought about by this shift in perspective. We also discuss ontological and epistemological aspects of the theory, and conclude that it can bridge some of the gaps between other fragmented HCI theories and approaches.

Keywords: Semiotic engineering; Semiotic approaches to HCI; Epistemic support for design; Users as designers

This paper is about semiotics and human–computer interaction. It focuses on the intellectual nature of software artifacts and on the way we perceive and use intellectual

* Tel.: +55 21 311 41500; fax: +55 21 311 41530.
E-mail address: clarisse@inf.puc-rio.br

0953-5438/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.intcom.2005.01.007

products. Semiotics is a discipline devoted to the study of signs[1] and how they are used in communication, a crucial process in intellectual activity. One of the hallmarks of semiotic theories of communication is to account for innovation and creative uses of language and numerous other sign systems (Eco, 1976). A semiotic perspective is particularly attractive for HCI because it underlines the fact that every computer artifact necessarily introduces new signs or sign systems in its users' universe. Computer artifacts are the result of their designers' reasoning, choice and decision, rather than the direct effect of universal or natural laws. They are meant to introduce changes in the users' world (e.g. to rationalize organizational processes or to enable new kinds of playful experiences). Therefore, the semiotic universe of users is expanded by every new system they interact with. If this semiotic expansion is expected to allow users to understand what designers mean, designers must *tell* users what they (designers) have in mind. Although it is true that design must rely on *intuition* and *familiarity* to alleviate the users' cognitive loads, it is also true that design can only succeed if users are able to handle what is *unfamiliar* and to see value in what is *new*. Hence, the ultimate goal of HCI design is not so much to *mirror* what users are when they first encounter the system, but to *propose* characters to whom users can relate and whom they may eventually incorporate.

In order to communicate the design vision to users, designers must step forward as legitimate interlocutors in human–computer interaction. Since they are not physically present at interaction time, they must speak *through* the system, which then becomes the *designers' deputy*. As a consequence, even though human communication is permeated with creative use of signs, with figurative speech, and even with signs invented *on the fly* by interlocutors in order to achieve contingent communicative goals, in HCI the communication between designers and users has a different profile. Since, programs require a fixed set of very specific semantic rules in order to respond *systematically* to every input it receives, in HCI signs must be *used* in strict accordance with limited semantic encodings of what they may actually *mean* to users and designers alike. In other words, because the designer's deputy can only talk with users by means of *computable* sign systems, the designer-user communication at interaction time is a new type of human communication, constrained by formal computational factors.

Our aim with this paper is to show how semiotic engineering—a semiotic theory of HCI (de Souza et al., 2001; de Souza, 2005), can contribute to bring about new perceptions and design possibilities in HCI. We start by examining some influential perspectives on design, namely Simon's (1981) and Schön's (1983), and then focus on a particular type of artifact that design can produce—*intellectual artifacts*. We explicitly characterize intellectual artifacts as those that can only achieve their purpose within and by virtue of the particular linguistic encoding of the design vision wherefrom they originate, a characterization that encompasses all computer artifacts.

We use an anecdotal example of interaction with Adobe Acrobat© (Acrobat) to illustrate *why* designers should not only *produce* good interactive systems but also

[1] Anything can be a sign as long as someone takes it to mean something else. For instance, the acronym 'HCI' is a sign for anyone acquainted with research in computer science today, just as the non-English word 'de-sign' is also a sign as long as there is somebody who takes is to mean something (say 'to disassemble the sign') and uses it to communicate such meaning.

deliberately *introduce* to users the product of their intellectual work. We then present the gist of semiotic engineering to illustrate *how* designers can be supported while communicating to users the values, the reasons, and choices of their design. We emphasize some ontological, epistemological and methodological aspects of the theory to show some of the most important implications of viewing designers as legitimate users' interlocutors at interaction time.

We conclude this paper by touching on what may be one of the most serious obstacles for HCI nowadays—scientific fragmentation (Carroll, 2003). In our view, integrative theories like semiotic engineering can help researchers and designers understand their task more clearly, and frame the problems to be solved in new ways. The shift of perspective discussed in this paper is just one particular instance of the changes and effects that we can produce in our field, bridging some of the gaps between competing theories and approaches known to-date.

1. Are interactive software artifacts special?

The word 'artifact' denotes something created by humans. An artifact's meaning, or value, is intrinsically associated to both its creator's *intent* and to its users' interpretation of how, when and where it can be *used*. There is, strictly speaking, no *natural* meaning for artifacts. Artifacts originate from the unpredictable *idea* that sparkles in human mind and triggers the design process that eventually produces them. Some ideas have such power and beauty that they not only inspire the creation of other artifacts of similar kind, but begin to be taught in professional education programs. Thus, they contribute to enhance a designer's perception and understanding of the design process and of the types of design resources that can be drawn upon to create new products.

Herb Simon (1981), in his influential series of MIT lectures about *the sciences of the artificial*, said that 'everyone designs who devises courses of action aimed at changing existing situations into preferred ones' (Simon, 1981: p. 129). Thus, whenever we meet a new artifact whose design coincides with what could have been *our own* preferences in changing the perceived situation, understanding the new artifact is easy. Even if we were to have different preferences, and consequently different ways to change the situation that the new artifact affects, understanding it may still be easy if we can see why and how somebody else would follow another line of reasoning.

1.1. Intellectual artifacts

Because all artifacts are intrinsically associated with the *idea* that started the whole design process that produced them, they all have an intrinsic intellectual dimension. They all:

- encode a particular understanding or interpretation of a problem situation; and
- encode a particular set of solutions for the perceived problem.

The distinctive feature of *intellectual artifacts* compared to other artifacts is that:

- the encoding of the problem situation and the corresponding solutions which they refer to are fundamentally linguistic (i.e. based on a system of symbols—verbal, visual, aural, or other—that can be interpreted by consistent semantic rules); and
- the artifact's ultimate purpose can only be completely achieved by its users if they can formulate it within the linguistic system in which the artifact is encoded (i.e. users must be able to understand and use a particular *linguistic encoding system* in order to explore and effect the solutions enabled through the artifact).

According to this definition, a logic truth table is an intellectual artifact, and so is Simon's book about the sciences of the artificial. The periodic table of elements in Chemistry is yet another example of an intellectual artifact, and so is www.ChemicalElements.com© (Chemical Elements), an online periodic table of elements (Fig. 1). Like www.ChemicalElements.com, all other web-based systems are intellectual artifacts, an indeed all other computer systems.

Computer systems share with artifacts, in general, the fact that they encode a particular interpretation of a problem situation and a particular set of solutions for the perceived problem. And they share with intellectual artifacts, specifically, the fact that the encoding of perceived problems and solutions is essentially linguistic, and that they can only be used by people who understand this encoding and who can express themselves through this language system. Pens and pencils, for instance, also encode a particular set of solutions for a range of perceived problem situations. But they do so through physical characteristics and *affordances* (Gibson, 1979) that do not constitute a linguistic system. We can build linguistic explanations *about* their encoding, as well as formulate linguistic descriptions *of* them. Metaphorically, these are *languages of design* (Alexander et al., 1977). But more technically, they are *metalanguages* of design that describe the intellectual dimension of

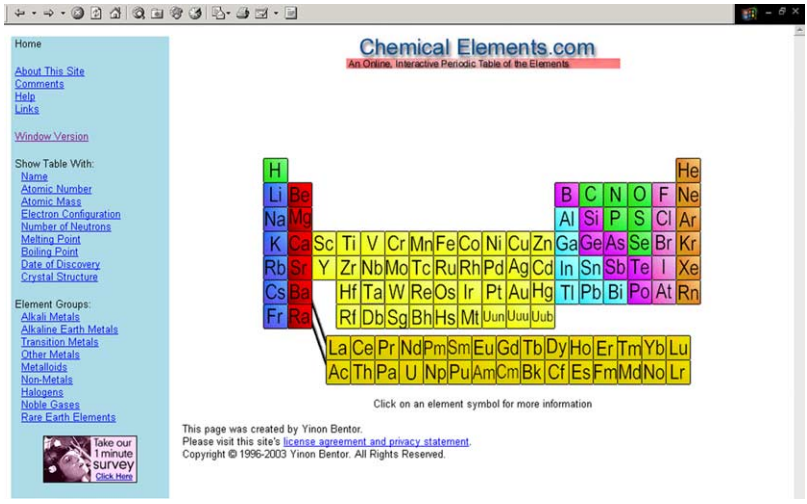

Fig. 1. The homepage of www.ChemicalElements.com© on the Web.

physical artifacts. Pens and pencils achieve their purposes by means of certain manipulations and fittings that are do not depend on any *language*.

Computer artifacts, however, all have linguistic interfaces, which designers create to express and effect their design intent, and which users must necessarily interpret, learn and use in order to achieve their own goals. Nothing can be done with the artifact except through the *language* (symbols, grammars and interpretive rules) encoded in its interface.

1.2. How are intellectual artifacts designed?

Simon's conception of design has inspired many in Computer Science. In recent years, for example, it has been used to formulate the scientific foundations for disciplinary research in computer-supported collaborative work (Ackerman, 2000). Possibly the strongest appeal of Simon's perspective is that in spite of starting from the very broad definition, quoted above, it eventually crystallizes into a much more exact formulation captured in the following passage of his book:

> The artificial world is centered precisely on this interface between the inner and the outer environment. […] There exists a considerable area of design practice where standards of rigor in inference are as high as one could wish. I refer to the domain of so-called 'optimization methods' […]. The logic of optimization methods can be sketched as follows: The 'inner environment' of the design problem is represented by a set of given alternatives of action. […] The 'outer environment' is represented by a set of parameters, which may be known with certainty or only in terms of a probability distribution. The goals for adaptation of inner to outer environment are defined by a utility function […]. The optimization problem is to find an admissible set of values of the command variables, compatible with the constraints that maximize the utility function for the given values of environmental parameters. (Simon, 1981: pp. 132–135)

Simon's framework is, however, hardly applicable to the design of intellectual artifacts. Problems start from the points made by Schön (1983), who was not speaking only about intellectual artifacts, but about artifacts in general. The following excerpt summarizes Schön's view of what is wrong with Simon's perspective (which he calls *technical rationality*):

> [In technical rationality] problems of choice or of decision are solved through the selection, from available means, of the one best suited to established ends. But with this emphasis on problem solving we ignore problem setting, the process by which we define the decision to be made, the ends to be achieved, the means by which they may be chosen. […] Technical rationality depends on agreement about ends. When ends are fixed and clear, then the decision to act can present itself as an instrumental problem. But when ends are confused and conflicting, there is as yet no 'problem to solve'. (Schön, 1983: pp. 39–41)

Schön's critique centers on how technical rationality extensively ignores the role of interpretation in naming and framing the elements of any perceived situation from which a design opportunity arises, and a *unique* problem-setting process yields 'the problem' to be

solved. Standard problems that can be solved by pre-established optimization methods are often 'of relatively little social importance' (Schön, 1983: p. 42). 'Messy but crucially important problems' must be solved by methods springing from an *epistemology of practice* that can help designers deal with the inherent complexity of situations where 'there are more variables—kinds of possible moves, norms, and interrelationships of these—than can be represented in a finite model' (Schön, 1983: p. 79).

Schön's alternative to technical rationality is reflection in action, where designers use epistemic tools very similar to those used in scientific research to *discover* problems and solutions in design. Such tools allow them to 'interact with (a) model, get surprising results, try to make sense of the results, and then invent new strategies of action on the basis of the new interpretation' (Schön and Bennett, 1996: p. 181). Designers thus, shape the situation in accordance to their interpretation and, by means of epistemic tools, they inspect the results of applying certain types of methods to the perceived problem variables. This inspection *talks back* to designers, and as they interpret what the back-talk means designers reflect in action and eventually crystallize reflection in the form of a design product.

Comparing the adequacy of the technical rationality and the reflection-in-action approaches to the design of intellectual artifacts, some features immediately stand out. First, because the utility of intellectual artifacts is determined by the users' ability to master the linguistic encoding of problem and solution, the technical rationality perspective, if adopted, would impose some odd requirements. For instance, it would require a pre-existing set of linguistic encoding types, previously known to designers and users, from which variables and/or constraints could be extracted and then used in the utility function that feeds the optimization methods used in the design process. The utility function should be able to yield a ranking of alternative linguistic encodings in terms of both the designer's and the user's expressive and interpretive needs. The obstacle here is that interpretive and expressive needs, as Schön so well points out, cannot be defined a priori. If not for designers, much less for the users. In other words, in this context, utility functions to be used in the design of intellectual artifacts have highly questionable value in very basic epistemological terms. Second, supposing that we tried to remove the epistemological obstacles by talking about a utility function that can evaluate *any* existing or non-existing linguistic encoding of design problems and solutions, other difficulties come up. For example, there should be a decision process, possibly based on a probability distribution, to differentiate *comprehensible/usable* from *incomprehensible/unusable* linguistic encodings of *any* range of meaningful elements for designers and users. Although one might think that this is a solution for designing intellectual artifacts following the technical rationality perspective, at closer examination this is only moving the problem forward. It rests on the assumption that now the meaning of linguistic *elements* are always determinable. In other words, that they can be always established at design time and that such meanings are definitive for all identical situations. The question is: when is a situation identical to another given the role played by contingent factors in any interpretive process?

In sum, we can say that the adoption of a technical rationality approach to the design of intellectual artifacts is compromised because it presents serious epistemological problems. Either it requires the existence of a fixed set of linguistic encoding systems, so defined that

they can be 'always' contrasted with each other for ranking purposes. Or it requires the existence of such thing as 'the' (known or knowable) meaning of linguistic elements in any actual or potential linguistic encoding system, so defined that such meaning can be used as a basis to evaluate how any two systems compare to each other in terms expressive and interpretive capabilities for all possible use situations.

It is interesting to see that in HCI these two assumptions are usually not examined in depth when we talk about measuring distances between what designers and users mean by interface signs (see for example Hutchins et al. (1986) and a host of quantitative usability studies reported to-date). What these measures are actually saying is not that the designed interfaces *pass* or *do not pass* a definitive usability test but rather that, given the contextual parameters influencing interpretive processes during the test situation, certain discrepancies between the designer's and the user's *meaning* for a particular set of interface signs were or were not observable.

More attuned with what is actually the case with the usability of intellectual artifacts, Adler and Winograd (1992) suggest that 'the key criterion of a system's usability is the extent to which it supports the potential for people who work with it to understand it, to learn it, and to make changes. Design for usability must include design for coping with novelty, design for improvisation, and design for adaptation' (Adler and Winograd, 1992: p. 7). In other words, the evaluation of the linguistic encoding systems of intellectual artifacts cannot rest on any pre-determined set of meaningful elements, because changes in the users' contexts will always bring in new meanings that, by definition, extrapolate the encoded meanings carried by any computer system. Assuming that users can imagine novel applications for a computer system, the distance measured between the users' novel meanings and the design meanings encoded in the system would be infinite. But what would this measure mean? Or, in other words, what would be the (contribution of) utility functions that used or yielded infinite values for ranking alternative choices? Schön's perspective is more promising for those who, like Adler and Winograd, bet on human imagination and creativity. Users are, themselves, designers (in both Simon's and Schön's sense of the word) of perceived improvements or solutions to perceived opportunities or problem situations. Reflection-in-action bets on epistemic tools rather than technical ones, and thus, may more easily account for how users relate to and use intellectual artifacts in general, and software artifacts in particular.

Thus, following Schön's inspiration, we conclude that the design of intellectual artifacts cannot be supported by a technical rationality approach. The next question is: what theoretical basis is needed for providing design with the foundations of an epistemology of practice? Our answer will be presented further, but before that we will talk about semiotics and illustrate how and why semiotic concepts can and should be used in HCI.

1.3. Why semiotics?

Semiotics, as many other disciplines, covers a vast territory of knowledge where numerous perspectives co-exist and dispute prestige. We choose to follow Umberto Eco's theories (Eco, 1976; 1984), according to which this discipline has as its main objective an investigation of *signification* and *communication*. *Signification* is the process through

which certain systems of signs are established by virtue of social and cultural conventions adopted by the users (interpreters and producers) of such signs. *Communication* is the process through which, for an open variety of purposes, sign producers (i.e. signification system users in this specific role) choose to express what they mean by exploring the possibilities of existing signification systems or, occasionally, by resorting to non-systematized signs (that they invent or use in unpredicted ways).

It is easy to see from Eco's definition that semiotics has strong relations with HCI design, even at this very superficial level of analysis. Going deeper into the theory, we find yet other relations and mutual resonance. In this paper we will talk about three of them: the relation between *signs* and computer symbols; the relation between *unlimited semiosis* and algorithms; and finally the relation between signs and computer artifacts. These three suggest that there are considerable advantages in switching the ultimate goal of HCI design from *producing* to *introducing* high-quality interactive systems, and recognizing designers as legitimate interlocutors of users at interaction time.

In semiotic theory *meaning* is discussed in the context of *signs*. A sign is anything that is taken by someone to mean something else. This something else does not necessarily have to exist (Eco, 1976: p. 7). In other words, it may just be a figment of one's imagination, a possibility or impossibility—absolutely anything that the *sign* represents in any given interpreter's perspective. There are three elements involved in any sign: the representation (that is taken to mean something else); the referent (or object that the representation refers to); and the interpretation (or meaning ascribed to the sign by the interpreter). Peirce (1931) called these elements *representamen*, *object* and *interpretant*, respectively. For our purposes, the interest of this definition is that the *interpretant* is itself another sign. In other words, meaning is not a value but a recursive function, an ongoing process of interpretation. The process is temporarily stabilized when the interpreter finds no need or means to continue. It can be resumed as soon as the interpreter finds new needs or means to proceed with interpretation.

Our ordinary life is full of examples that corroborate such conception of the meaning of a sign. And, so is our daily experience with computers. Suppose a user is going through the three-step process of creating a new group in Yahoo Groups (Yahoo!), and in step 2 he sees the screen presented in Fig. 2. After filling out all the required information in textboxes and text area, he clicks on the button labelled Continue. Is the button labelled Cancel also a sign for him? Certainly. What does the cancel button *mean* to him? Suppose that he answers this question saying that 'cancel means interrupt or stop'. Because what he really wants to do is to continue, there is no further need to linger on what cancel *means*. Does he understand what the system is telling him with that screen? Yes, he does: 'cancel' *means* 'to interrupt or stop the process'.

Now, suppose that a few months later this same user decides to create another group, and when he is exactly at the same stage as he was when he was asked about the meaning of 'cancel' he realizes that maybe he should have done something different in the previous stage ('categorize'). So, he decides to go back to the previous stage and make the desired changes. He decides to 'cancel' (interrupt or stop) the current stage of the process, and to go back to the previous stage. However, clicking on cancel leads him out of the whole group-creating process. He must start it all over again. Indeed, this user realizes that 'cancel' means 'interrupt or stop the complete process'.

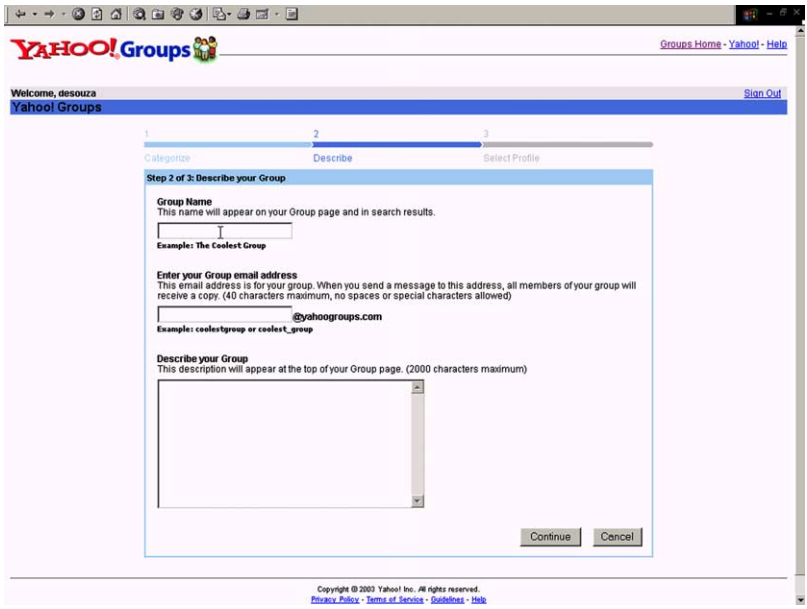

Fig. 2. Signs in Yahoo Group©'s group-creating interface.

This example shows us that in semiotic terms human *meaning* is not a static and definitive entity, but an evolving process that interpreters halt and resume for contingent reasons. Moreover, snapshots of meaning configurations throughout the interpretive process are not necessarily consistent with each other, much less can we pick one and *predict* the next or derive the previous by means of any kind of interpretive rule.

Such conception of meaning is totally different from what is the *meaning* of symbolic expressions processed by computer systems. Computer-processible meanings *must* be either fixed or fully derivable. Therefore, the computer-processible meaning of 'cancel' in the interface shown in Fig. 2 is one and the same every time the button is pressed: a process that causes the group-creation program to end. Regardless of whether computer meanings are defined as an operation to be performed by a physical machine, as a function to be evaluated by an abstract machine, or as an expression in another symbol system, the Turing machine model of computation states that symbol processing must be based on a set of explicit rules. This is one of the fundamental differences between signs for humans and symbols for computers.

Going further in this direction, we see that this continuous and unpredictable meaning-producing process triggered in human minds by the presence of a significant representation challenges one of the fundamental pillars of computation—algorithms. The human interpretive process, also called *unlimited semiosis*, cannot be modeled by algorithms for lack of precisely definable halting conditions. As a consequence, within the scope of a semiotic theory, there cannot be such a thing as a computational model of human interpretation or of human meaning.

Although this conclusion may sound disastrous for all semiotic theories of HCI as we now know it, or as a presumptuous attempt to ruin the theoretical edifice built mainly by cognitive theories in the last two decades, this is absolutely not the case. The positive implication of this conclusion for HCI is that computer systems contain what is possibly the richest encoding (or representation) of a particular meaning configuration produced by a human mind[2]. It contains an encoding of the *design vision* that produced this artifact. In other words, computer artifacts are *signs* themselves. They represent, refer to, and mean what their designers had in mind when they completed their design process.

Unlike all other signs, computer artifacts can do *limited semiosis*, an algorithmic interpretation of symbols that are, themselves, part of the designer's semiosis. For example, the computer-processible meaning of the Cancel button in Fig. 2 is a sign of what the Yahoo Group designer(s) had in mind. And users can query the 'designer's' meaning of this sign by asking the system to reveal this meaning. The system can do this by directly performing the cancel function or by providing hints and explanations about what the sign means. Hints and explanations can come in various forms: as online help content, as tips displayed when the mouse hovers over interface signs, as directives on the screen, and so on. In other words, computer systems can generate (compute) signs that stand for what other signs mean within some unique human vision that produced them as a result of intensive interpretation and reflective design.

Given the semiotic considerations presented so far, in our view, the ultimate goal of HCI should not be to capture the users' meanings and to encode them in computer systems (for such snapshots of the users' meanings may not really mean much over time), but to provide users with all possible means to support *their* semiosis at interaction time. Designers should help them probe and interpret interface signs, allowing them to benefit from the unparalleled semiotic richness of computer artifacts. This is why we propose to shift the target of HCI design from *producing* to *introducing* high-quality interactive systems, and encourage designers to step forward as legitimate users' interlocutors during interaction. Section 2 will present a detailed example of what this shift may mean in HCI.

2. Why switching from producing to introducing software?

In order to illustrate the points we want to make, we will present a detailed example of interaction with Adobe Acrobat® 5.0 [Acrobat]. The context of interaction is focused on a reviewing activity where a user, fictitiously named *Karina*, is reviewing a colleague's manuscript. She has been using Acrobat for a while, although she recognizes there is 'much more to Acrobat than she realizes'. We will first narrate her experience and then summarize the HCI design lessons we learn from it.

[2] For a brilliant essay on the semiotic richness of computer signs see Eco's short story *On truth: A fiction* (Eco, 1988).

2.1. The Acrobat® example

Karina has received a colleague's manuscript to review. The manuscript is in PDF format and she uses Acrobat to do the reviewing. She finds Acrobat very useful for this task, because it allows her to use a number of convenient reviewing tools like *highlighting*, *notes*, *striking through*, and others that she has in fact never used. Karina has the habit of commenting her reviews, so that the author being reviewed understands her reading better and is thus, in a better position to accept or reject her suggestions. Karina has also the habit of reading documents online, rather than on paper. She finds it more practical than reading printed versions of documents and then transposing comments to their electronic versions, which are ultimately the ones exchanged between authors and reviewers most of the time.

Karina uses Acrobat's *highlight* tool to signal the part of the document to which her reviewing refers, and always creates a note with a brief explanation or additional information that will help the author see her point more clearly. In Fig. 3 we see a snapshot of Karina's screen while reviewing a draft paper. Karina has highlighted one sentence in the text, and then used the *note* tool to write her message to the author. The little popup window with a text area is opened when Karina creates a note. The *note* icon positioned right above the popup window signals to Karina and other document users that there is a note on this page. Double-clicking on the icon toggles the state of the popup window between open and closed.

One of the reviewing possibilities enabled by Acrobat is to navigate through the various review marks, page-by-page, checking them all online. In Fig. 4 we see how the navigation can be made. On the left-hand side of the screen is a panel where all comments associated

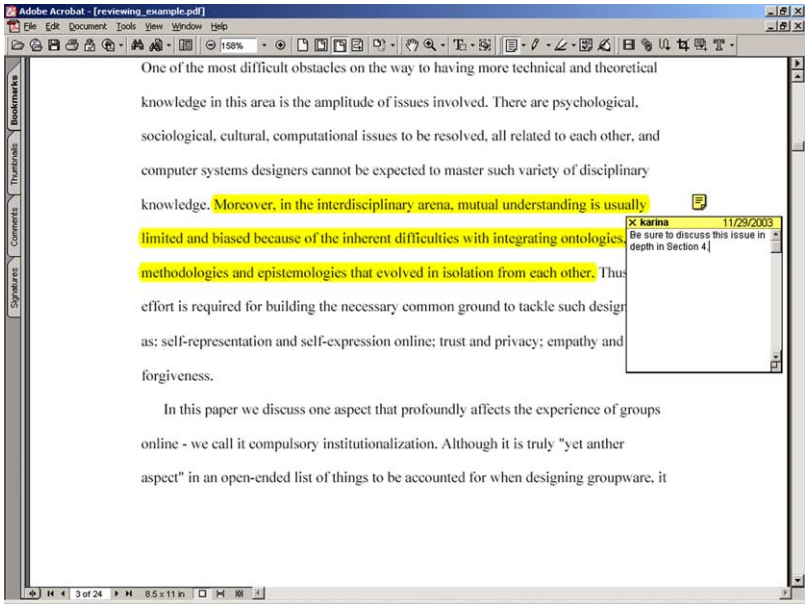

Fig. 3. Effects of the highlight and note tools in Acrobat®.

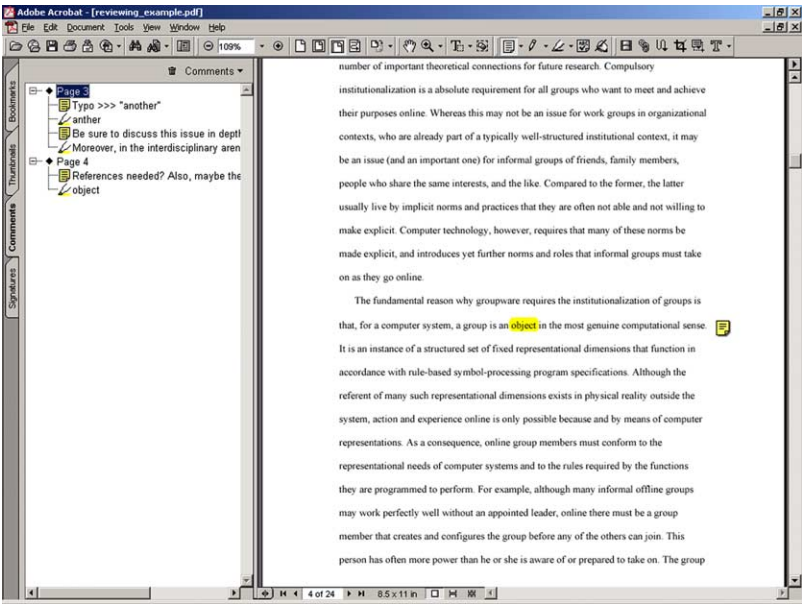

Fig. 4. Navigational support for browsing through all review marks in Acrobat®.

to their respective page can be examined in detail if the user clicks on them. Karina often follows this path to check her review marks.

This is basically what Karina does all the time. She knows Acrobat generates a summary of all review marks, in PDF format, which she finds useful for her records. At times she does reviewing work for periodicals and is asked to write a report with suggestions and requests for authors. She then uses the summary file to copy and paste most of the content of her report. In Fig. 5 we see a snapshot of what the PDF summary file looks like for the comments Karina has made so far.

At closer examination, we can observe that Karina's reviewing strategy seems to carry some potential problems with it. The summary file presents review marks sorted by sequence number. This is the default configuration in Acrobat, and it replicates the order in which review marks were introduced. The connection between Karina's highlights in such printed form (sequence numbers 1 and 3, and then again 1 on 'page 4') and what she wants to tell the document's author about the highlighted contents (sequence numbers 2 and 4, and then again 2 on 'page 4') is totally dependent on creation order. That is, if Karina had highlighted the sentence starting with 'Moreover…' first, had then highlighted the typo on the word 'another' a few lines below, and had then decided to create the notes telling the authors the reason for each highlight, in the printed summary we would first read the highlights (sentence first, word second), and only then would we read the content of the notes. The problem with this slight change in Karina's reviewing strategy is that the readability of the printed summary would be seriously jeopardized. In the specific example we are studying, the content of the note (which we know to correspond to the highlighted sentence starting with 'Moreover…') is this: 'Be sure to discuss this issue in

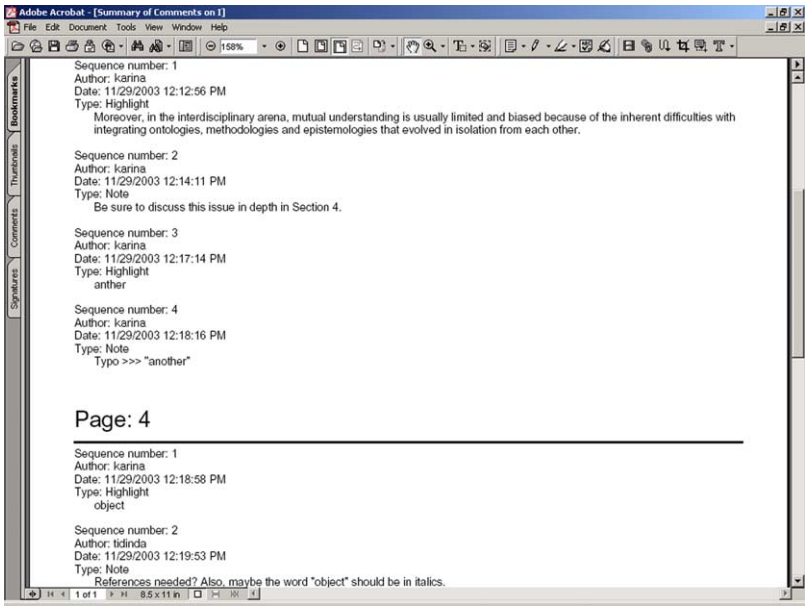

Fig. 5. Contents of Acrobat®'s summary file for all review marks in a document.

depth in Section 4'. How would we be able to interpret the meaning of the word 'this' in Karina's note if her text appeared anywhere else but immediately contiguous to the text span in the document to which it refers?

A similar problem actually exists in the graphic depiction of reviewing marks online. In both Figs. 3 and 4 we see that the spatial contiguity of the *note* icon with the text span it refers to is crucial to interpreting deictics like 'this', 'here', and similar ones. But Karina has learned about the contiguity requirement by making some mistakes along the way. Once she deleted a note associated to a highlighted text and created a new one to replace the deleted one with. On the screen, everything was fine, but when she generated the summary file she realized that her editing had altered the sequence numbers of review marks on that page, and it took her a while to understand what the summary file was showing. The comments and their corresponding highlighted text appeared many lines apart from each other. This is why she *never* deletes notes anymore. She edits them whenever needed, but she pays close attention not to alter the sequence number order of her review marks.

Today Karina sets out to finish reviewing her colleague's manuscript, but in her rush to do it before noon she inadvertently forgets to comment on one of her highlights in the document. She realizes this a bit too late, and gets totally confused about the order of remarks that she should have obeyed so that the printed summary (which her colleague has explicitly asked her to send him) shows the note next to the text it refers to. After trying to find a solution probing Acrobat's interface here and there, she searches Acrobat's online help and discovers that she can configure Acrobat to display sequential numbers with each comment. This looks exactly like what she needs to know in order to restore the reading

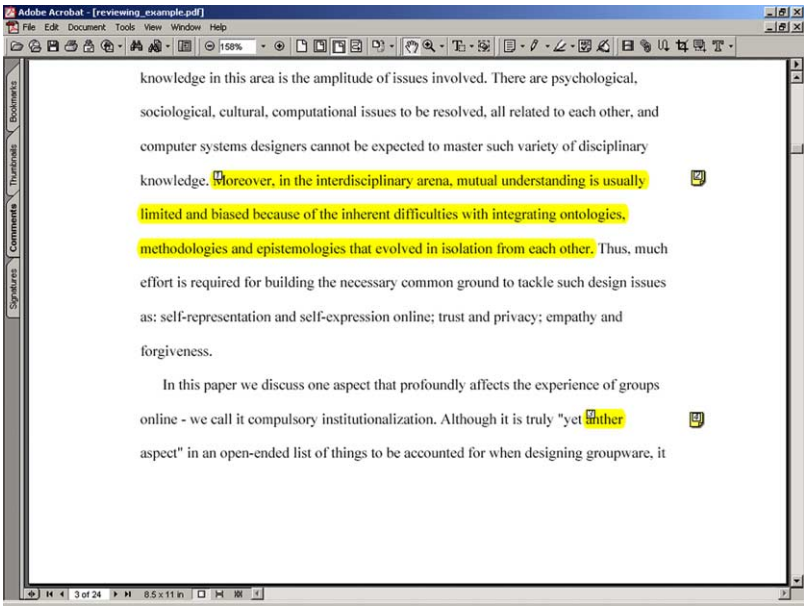

Fig. 6. Screen layout with sequence numbers shown for comment in Acrobat®.

order in the summary file. The help content even mentions that 'This is useful when used in conjunction with the summarize comment feature'.

Full of confidence, Karina sets the 'show comment sequence numbers' parameter to on, and then what she gets is a screen like the one shown in Fig. 6. She can now see the numbers on each comment but… Wait! Is every *highlight* a 'comment'? What does that mean? A highlight is not a 'comment', is it? A *note* is a comment, alright. But a *highlight* is not, is it?

Karina clicks on both highlights she sees on the screen, and is surprised to see that—indeed—every highlight *is* a comment that contains the very text over which the highlight extends. As she stops to think about it, she realizes that she should have known it all the way, because the summary includes every highlighted text in the document. Of course!

Karina realizes that this is in fact a great finding. If she can add her explanations and suggestions to the highlight text *directly* she will need only half as many review marks as she now has in reviewed documents. Instead of highlighting the text she wants to comment on and creating a note with the comments themselves, all she needs to do is to highlight the text and add her comments directly beneath the contents of the highlight, already inserted in the comment. She tries this newly found strategy and is delighted to see how nicely it works (Fig. 7).

She checks the summary file with all her review marks and is even more enthusiastic with her finding than she was before. As shown in Fig. 8, the readability of her review report increased enormously. Just as she had guessed, the file contains only half as many comments as before.

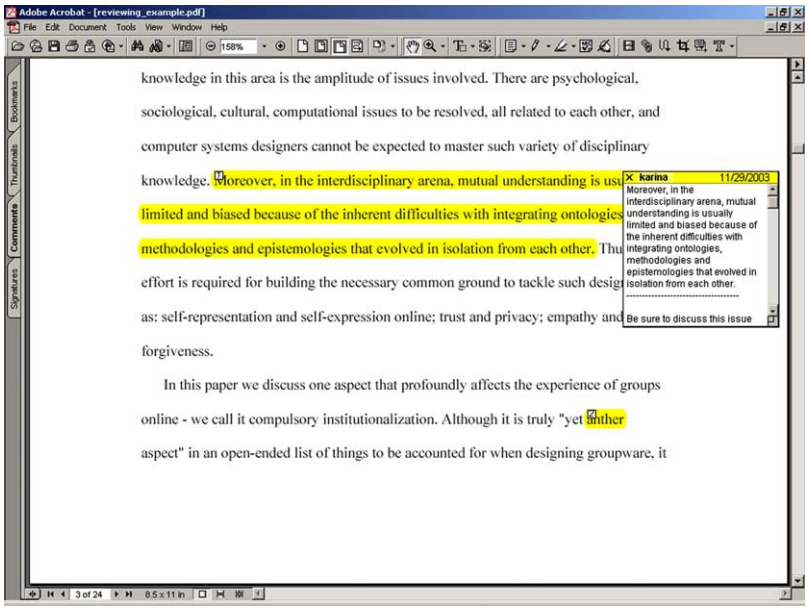

Fig. 7. Content of comments associated to highlighted text in Acrobat®.

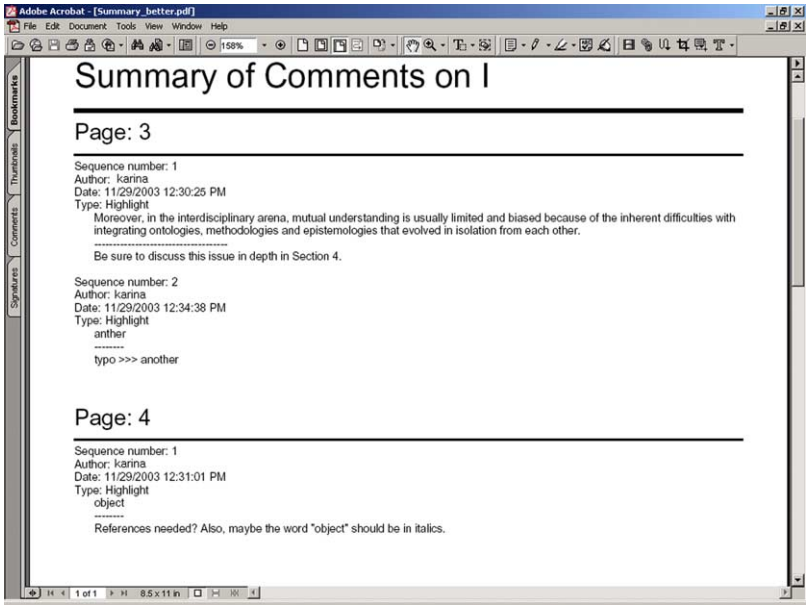

Fig. 8. Summary with comments directly associated to highlight contents in Acrobat®.

Not only the size and readability of the summary is much improved when she uses this new strategy, but the problem of visually contiguous notes and strict sequencing of creation is gone! It is not a problem any longer. It fact, it might have never been a problem, if only she had seen it that this is how reviewing like hers should be done in the first place. This is why they say in help content that 'This is useful when used in conjunction with the summarize comment feature'. But how would she know?

2.2. Lessons learned from the example

Karina's story is about interactive strategies and important usability issues that may escape an HCI designer's consideration at design time (as seems to be the case with Acrobat). Nielsen's *Ten Usability Heuristics* (Nielsen, 1994) are among the most popular resources for usability-related decisions during HCI design. In Nielsen's own words, they are:

- Visibility of system status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

- Match between system and the real world

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

- User control and freedom

Users often choose system functions by mistake and will need a clearly marked 'emergency exit' to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

- Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

- Error prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

- Recognition rather than recall

Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

- Flexibility and efficiency of use

Accelerators—unseen by the novice user—may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

- Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

- Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

- Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Our own interpretation of Nielsen's heuristics focuses on the underlined parts of each explanatory text below the heuristics' phrasing. The purpose of reproducing them in this paper is to call the reader's attention to the fact that none of them seem to account for the problem encountered by Karina. In fact, to many faithful followers of usability engineering, Karina did not have a problem. Her story is one of gaining expertise with the tool, a natural path followed by all users of all systems. As Nielsen himself puts it: 'Accelerators—unseen by the novice user—may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users.'

We argue that this *is not* the case, and that Karina did have a problem. As she realized herself, she had been spending twice as much time with her reviewing than was needed, she was handling contiguity problems that might not have existed to start with, and she was generating poorly readable review summaries that she could only use to copy and paste content from. Now that she knows a much better reviewing strategy that she can use with Acrobat, she will also be able to send the automatically-generated summary report to authors, with no additional cost.

Karina's was not an operational problem. It had nothing to do with: interpreting current system states; understanding system terms; being able to use emergency exits; having to deal with words that mean different things in different parts of the system; getting incomprehensible error messages; having to remember too much information; lacking accelerators and customized access to frequently used functions; cluttered visual layouts; knowing how to recover from errors; and finding help information. All of these were fine, and the fact that she needed to search online help to solve the problem that led her to a new understanding of how Acrobat should be used only goes to show that Acrobat design—in this particular context-does not show evidence of failing to follow usability canons.

The kind of usability problem experienced by Karina is one of 'flexibility and efficiency of use', keeping with Nielsen's terms, but it is not about actions and operations—it is about goals and strategies. And the lesson we learn from this example is that HCI is about communicating 'the concrete steps to be carried out' in order to complete a range of 'tasks', of course, but it is also about communicating the value of innovative problem-solving strategies supported by the system. Not just about *producing* mechanisms to support such strategies (which Acrobat designers did). It is about *introducing* them to the users (which Acrobat designers failed to do), during interaction.

3. What can semiotic engineering bring into the picture?

Semiotic engineering (de Souza, 2005) is a semiotic theory of HCI that brings together under *the same* communicative context the three sources of interpretation and communication involved in the design of interactive computer artifacts: designers, users and computer systems. Resting on explicit ontological, epistemological and methodological premises, semiotic engineering provides an account of HCI that sharply contrasts with many widely adopted views in this area. It views HCI as a specific type of twofold computer-mediated communication, in which computer systems designers send systems users a *one-shot message*. The message's *representamen* (i.e. the perceptible sign that stands for the message) is the system itself and it tells the users how to communicate with it in order to accomplish a certain range of effects. It is a *one-shot* message because, at all times, it carries a complete and immutable computer-processible content encoded in and made available by the system's interface. The message content can be paraphrased as:

> Here is my understanding of who you are, what I have learned you want or need to do, in which preferred ways, and why. This is the system that I have, therefore, designed for you, and this is the way you can or should use it in order to fulfill a range of purposes that fall within this vision.

(de Souza, 2005).

We see that it is a message *about* other messages, a piece of communication *about* other communications, which characterizes a *metacommunication* process. The 'I'—first person of discourse—in the content paraphrase above is the system designer (or a spokesperson for the system design team). The metacommunication can only be completely successful if users communicate with the message and, by so doing, they *get* what the designer is trying to tell them. *Getting* what someone is telling you does not always involve (actually we can wonder if it ever involves) generating the exact same meanings as this person generates for each of the signs he or she is using in communication. Communication is a far more subtle and richer process than just matching meaning components between interlocutors. Rather, communication is a meaning-negotiation process where an efficient use of resources and strategies is the key to success.

One of the crucial differences introduced by semiotic engineering in the HCI picture is thus, that HCI designers are *present* at interaction time. That is, unlike more traditional views where HCI is viewed as communication between users and systems, in our view

designers are communicating with users at interaction time through a specifically designed space of communicative exchanges that users can have with the system.

Another peculiarity of semiotic engineering is that it is fully committed to the view that software is an intellectual artifact. Just as Norman's cognitive engineering (Norman, 1986) underlined the fact that cognition is central to the *user's* activity, we are now underlining the fact that cognition is also central to the *designer's* activity. Both application designers and HCI designers are building artifacts that reflect their particular understanding and reasoning about a number of goals, constraints, opportunities and challenges related to using computer technology to affect human life. Additionally, because the nature of software products is representational, both application designers and HCI designers can be said to aim at *representing* (or signifying) their understanding and their intent in such a way that the users of their products can see what they *mean*.

The HCI designer is of course more closely involved with making users understand what he or she means. And the way to accomplish this task is through metacommunication. The designer's one-shot message to users is progressively unfolded and interpreted by the users as they communicate with the system. For metacommunication to proceed consistently and cohesively, the system must *speak for* the designer. If the system does not speak for the designer, the designer's message is lost and metacommunication itself is canceled. The system must be the *designer's deputy*—a communicating agent that can *tell* what the designer means, and thus, participate in the meaning-negotiating processes that characterize more precisely what communication is in our daily experience.

The presence of a deputy is what brings together users and HCI designers, and what restores the complete range of communication taking place in HCI. Many existing applications, such as Microsoft Excel® (MS Excel), for example, show interface evidence that designers intuit that they can or should speak to users during interaction. In Fig. 9, for instance, the user's interlocutor is not exactly *the system*. Otherwise, why would the system be referring to *itself* as 'Microsoft Excel'? It is the system's designer who is addressing the user.

In Fig. 10 MS Excel is somehow 'negotiating' the meaning of '=' and '−' with its users, and telling them what alternatives they can use to express what they mean. But we do not know who exactly is speaking—the system or the designer. Consequently, we do not know whom we are talking to when we say 'Ok' or 'Help'.

Further evidence provided by MS Excel shows a number of apparently simultaneous messages sent to users from disparate interlocutors. As a result, a consistent history of meaning negotiation, which would allow users to know what is the current common ground with *the* interlocutor they are talking to through MS Excel's interface, is difficult to

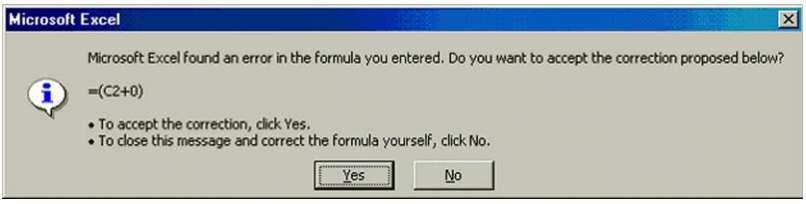

Fig. 9. Who is speaking to whom through Microsoft Excel® interface?.

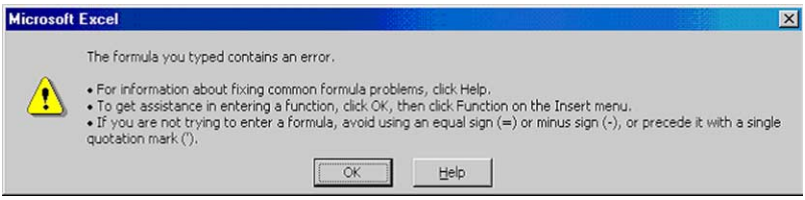

Fig. 10. Negotiating meanings through Microsoft Excel® interface.

build. In Fig. 11 we see a very confusing *design* of metacommunication, which suggests that designers have not explicitly settled to design conversations for meaning and/or initiative negotiation. Non-verbal signs like spatial distribution, font type and choice of interface objects like check boxes or pull-down lists do not consistently organize the conversation between the user and the designer's deputy (or the system, since, we cannot claim that the designer's deputy belongs to MS Excel's design ontology). A guiding principle to encode the 1st ('I'), 2nd ('you') and 3rd person of discourse ('it'), along with the turn-taking cues (i.e. when/where each interlocutor, or person of discourse, is expected to say something in an ongoing conversation), is hard to identify and, at any rate, very different from the ones shown in Figs. 9 and 10.

A semiotic engineering approach to designing HCI should help MS Excel's designers to pay closer attention to such factors as:

1. How is conversation organized in MS Excel?
   a. Who are the interlocutors during interaction?
   b. How are they represented to each other? How is their interlocution cued?
   c. What interlocution codes and patterns may or must be used? When?

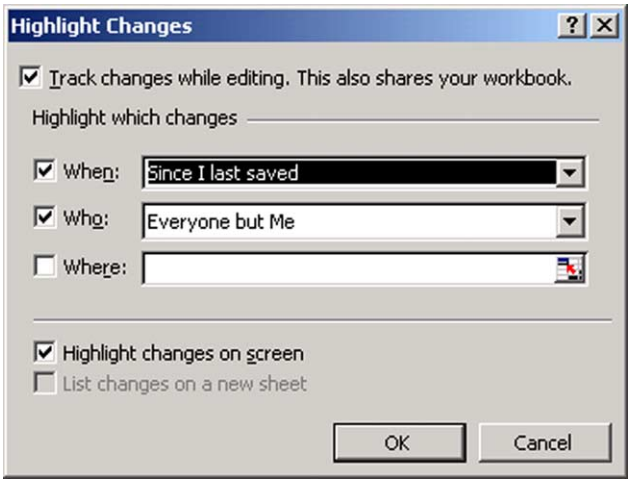

Fig. 11. Confusing communication design at Microsoft Excel® interface.

2. What kinds of meaning negotiation take place in this application?
   a. Is MS Excel prepared to take on the user's meaning? When? How?
   b. When should the user take on MS Excel's meanings? Why?
3. Which conversational patterns can be systematically used to communicate the opportunity, the process and the result of meaning negotiation?
   a. How are explanations about the negotiation required, provided and distinguished from the negotiation itself?
   b. How are presuppositions (often expressed as *default* values) signaled to each interlocutor during meaning negotiation?
   c. What is the scope (duration) of negotiated meanings? Can they be recalled by both interlocutors? Can they be revised? How?

The examples presented in Figs. 9–11 show that users cannot always draw useful inferences about their interlocutor's pattern of communication, and, therefore, are prone to not knowing how to communicate or how to interpret communication.

By defining HCI as a metacommunication process through which designers are *telling* users their design vision, and by so doing enabling users to learn, explore and apply this vision to a wide range of predicted and unpredicted situations, semiotic engineering brings together three otherwise conflicting views invoked in this paper: (a) Simon's view that everyone is a designer who tries to change a problem situation into a solution one; (b) Schön's and Adler and Winograd's views that design should be prepared to support creative interpretations; and (c) that meaning-negotiation plays a fundamental role in HCI. As we learn from semiotic theory, briefly explained in Section 2, communication is a process whereby interlocutors explore existing signification systems in order to express what they *mean*. They are not, however, forced to use only the signs encoded in the signification system they explore. They may and very often do create new signs, or use known signs in creative ways (like metaphors), to add expressive power to what they want to tell to others.

At this point we can see clearly the causes and consequences of aiming at *introducing* computer artifacts to users rather than settling for just *producing* them. Designing how to introduce the design product is the way to support meaning-negotiation processes in an unpredictable range of situations where users will try to devise solutions for situations that need some kind of improvement.

As a theory, semiotic engineering carries, however, some of its challenges and limitations on its sleeve.

At the ontological level, semiotic engineering splits meaning into two very diverse categories. Human meanings (that of designers' and users') are produced and interpreted in accordance with the unlimited semiosis principle. Computer meanings (human meanings encoded in programs), however, cannot be so produced and interpreted. Theoretical constraints on algorithmic computation cannot support the notion of unlimited semiosis, and thus, set the borderline between human and computer semiotics. Systems and people 'interpret' symbols/signs in different ways. People produce interpretations by: bringing in significant elements that have unpredictable relations with the sign which they are trying to make sense of; and working with provisional meanings that stand till they need, wish, or happen to resume interpretation for also unpredictable reasons. In other

words, the ingredients that determine a halting condition in human continuous semiosis cannot be specified a priori. This situation is in contrast with the notion of algorithms, the very basis of all classical theories of computation. Algorithms are symbol-processing procedures for which precise halting conditions must be defined. Therefore, whereas we can, in theory, expect users to eventually predict and foresee how all signs pertaining to an interactive system's interface will be processed by a computer (the equivalent of the user's having full mastery of the system's interface), there is no possibility that a system will eventually predict and foresee how all of its signs will be interpreted by the user in situated interaction. The Yahoo Groups' interface example in Fig. 2 further back in this paper is a good illustration of the continuing and contingent interpretation process that makes it impossible to predict and foresee what any particular sign *means* to the user. So, semiotic engineering constantly deals with both algorithmic and non-algorithmic meaning processes, explicitly.

At the epistemological level, some important constraints must also be tackled. First, as we said before, in a semiotic perspective, there is no room for a purely objective and stable account of meaning, whether the designers' or the users'. Meaning carries inherent subjective and evolutionary ingredients determined by unlimited semiosis, which casts a shadow of doubt upon the idea that the users' context, requirements and capacities can be fully captured by any human interpreter at any given time. Even when we let go of the idea of capturing *all* meanings, the choice of which are the relevant ones to be captured is equally prone to biased judgments and incompleteness.

In order to preserve the theory from being declared anarchic and ultimately useless, for scientific purposes in general and for HCI design in particular, this epistemological challenge must be counterbalanced by sound methodological choices. They must enable researchers to identify the limits of subjectivity, the power of cultural determination, the conditions of social contracts in communicative phenomena, and, in the particular domain of HCI, the effects of computer technology upon human signification and communication processes. Sound choices will protect researchers (and professional practitioners) against adopting a naïve idealized view in which interactive computer-based artifacts can be built according to laws that are similar to the ones that allow civil engineers to build bridges. And they will also protect them against adopting a nihilist view in which, because users can never be fully understood and systems can always be built, any approximation of the actual use situation is as good as any other.

In sum, assuming that computer artifacts are indeed intellectual artifacts, whose language plays a different role than pattern languages in architecture (Alexander et al., 1977), and whose elements carry meaning and intent instead of affordances (Gibson, 1979), the quality of interaction with them is profoundly determined by the quality of communication between those who produce them and those who consume them. Because one of the key elements in productive communication is meaning-negotiation, the ultimate goal in HCI design is more likely to be that of introducing high-quality interactive systems than just that of producing them in hopes that its meaning will be obvious and intuitive in all situations where users are trying to design solutions for the situations they perceive themselves to be in.

4. Concluding Remarks

John Carroll suggested that the scientific fragmentation in the field of HCI is one of the important barriers to be transposed if we want to make progress (Carroll, 2003). Because there are too many approaches, too many techniques, too many technologies, too many theories, seeing the big picture is extremely difficult. As a consequence, when building interactive software artifacts we are not always sure (and maybe not even aware) of where and why old or new knowledge applies. This situation decreases our ability to make appropriate and justifiable choices in HCI design, and increases the feeling expressed by some that scientific theories cannot help us build quality artifacts any better than sheer individual talent and professional practice. It also jeopardizes the value of crucially important scientific findings in HCI, because it allows equivocal expectations to be built about the scope of what such findings can account for. The major drive behind our proposal to view designers as legitimate interlocutors during interaction is exactly to try and achieve a more integrated perspective in this fragmented landscape. This paper has presented only the highlights of a more profound discussion of how and why a shift like this, and the theory associated to it, can help us make some progress in HCI (for a comprehensive argumentation see de Souza (2005)).

Scientific fragmentation can be improved by an appropriate epistemology. But to-date, because of its interdisciplinary character, the epistemology of HCI—if explicitly addressed at all—has itself been a patchwork of epistemological borrowings from psychology, computer science, anthropology, engineering, and design. The result of this patchworked epistemology is that sorting scientific issues from non-scientific ones, distinguishing between essence and circumstance, and even identifying the permanent goals of the discipline has been extremely difficult. We have used a shift in perspectives about HCI design goals to hint at a number of such distinctions and the importance of being able to make them.

What we propose with semiotic engineering may perhaps be felt as a threat. First and foremost, we embrace the view that human meanings cannot be fully known, and consequently not be completed (or adequately) encoded in computer systems. This is threatening because it shakes the foundations of many design and evaluation methods and practices, where the ultimate match between system meaning and user meaning determines the goal of design and the quality of software. If the user's meaning is an evolving unlimited process of interpretation, halted and resumed for contingent reasons, and shaped by unpredictable factors that emerge during the process itself, the aura of rationality that HCI has so strongly struggled to preserve seems to be suddenly gone.

In defense of semiotic engineering, we can invoke two main arguments. One is that aiming at capturing *the* users' meanings and fixating them into computer systems does not change the reality of what users actually do. This reality is often disconcerting for any one who takes time to sit and watch how users interact with software, or time to talk to users about what they think they and the system are doing. The other is that humans are perfectly equipped to handle unlimited semiosis. We do this all the time. So, maybe the trouble with computers is not really that they cannot capture and account for all that we actually mean while we interact with software, but rather that this is not what they should be doing in the first place. Maybe computer systems are more like story tellers than looking glasses.

Every story has an author, and listeners naturally know this. They may identify themselves with one or more of the characters, or perhaps with none of them, and still think that the story is a great one.

A second threat is that semiotic engineering, the very theory which we use to illustrate a new perspective on HCI design, may be mistaken as a self-defeating theoretical alternative. It rests on the assumption that users are interested in knowing more about the artifacts that they use. But since, people do not usually care to read online documentation and prefer trial and error to using help resources that come with applications, working to let them know more about the *ideas* that led to the construction of the systems they interact with may seem like a waste of time. However, if we look at human communicative behavior, the economy of conversation in goal-oriented situations suggests that interlocutors do go into very sophisticated meaning-negotiating processes under certain basic conditions. They do so in situations like: if they believe that getting the right meaning is needed for the achievement of their goals; if they believe that the negotiation will not cost them much; and if they believe that the benefits of negotiation outweigh the risks of misunderstandings. Trial and error is a communicative strategy in HCI and elsewhere. It suppresses meaning-negotiation processes, and replaces it with repeated attempts at guessing the meaning of things instead of asking what they are and trying to resolve disparities. But this strategy may turn into a problem when guesses fail one after the other, or when guesses do not reach far enough (e.g. when only part of the meaning is understood, and the part that is not understood leads to other communicative breakdowns in situations that are not perceived to be related). Our Acrobat example suggests that if only Karina *knew* that there was a much more efficient revision strategy to follow, she would not have wasted her time trying to prevent or correct mistakes that could be avoided in the first place. But how could she have known this from the interface as it is designed? How could she have guessed that highlights are themselves comments for Acrobat's designers? How if not if designers told her so in a much more efficient way than they did? We can see that reasoning about a waste of time, for designers and users, is difficult when working only with short-term and closed-world perspectives.

Introducing design ideas to users through a careful engineering of efficient semiotic systems and interactive rhetoric (that does not need to be verbose at all) can help designers share responsibility for successful interpretive tasks with users. Just like successful communication is a shared task between interlocutors in natural settings. So, perhaps surprisingly, semiotic engineering is in fact a promising alternative that alleviates the design anxiety of having to hit a moving target-that which users have in mind. The target becomes one of using the computer *medium* and all of its unique semiotic richness to talk to users about good design ideas that they can apply to their own design tasks.

Acknowledgements

The author is thankful to many readers who have asked her *tough questions* about the *whys* and *hows* of semiotic engineering. In particular, she is thankful to all of her students at PUC-Rio, and to Don Norman, Jenny Preece and Ben Shneiderman. She also wants to thank Bonnie Nardi for suggestions on how to introduce alternative theories of HCI, as

well as Raquel Prates and Simone Barbosa for their invaluable research contributions and emotional support. Thanks also go to the anonymous reviewers of the original version of this paper for valuable advice on how to improve it and to CNPq, the Brazilian Council for Technological and Scientific Development, for supporting this research.

References

Ackerman, M., 2000. The intellectual challenge of CSCW: the gap between social requirements and technical feasibility. Human–Computer Interaction 15, 179–203.

Acrobat—Adobe®, 2001. Acrobat© 5.0.5. Adobe Systems, Inc.

Adler, P., Winograd, T., 1992. Usability. Turning Technologies into Tools. Oxford University Press, New York, NY.

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S., 1977. A Pattern Language. Oxford University Press, New York, NY.

Carroll, 2003. Chemical Elements—www.ChemicalElements.com. Copyright© 1996–2003 Yinon Bentor

de Souza, C.S., 2005. The Semiotic Engineering of Human–Computer Interaction. The MIT Press, Cambridge, MA.

de Souza, C.S., Barbosa, S.D.J., Prates, R.O., 2001. A semiotic engineering approach to user interface design. Knowledge Based Systems 14 (8), 461–465.

Eco, U., 1976. A Theory of Semiotics. Indiana University Press, Bloomington, IN.

Eco, U., 1984. Semiotics and the Philosophy of Language. Indiana University Press, Bloomington, IN.

Eco, U., 1988. On truth: a fiction. In: Eco, U., Santambrogio, M., Violi, P. (Eds.), Meaning and Mental Representations. Indiana University Press, Bloomington, IN.

Gibson, J.J., 1979. The Ecological Approach to Visual Perception. Houghton Mifflin, Boston.

Hutchins, E.L., Hollan, J.D., Norman, D.A., 1986. Direct manipulation interfaces. In: Norman, D., Draper, S. (Eds.), User Centered System Design: New Perspectives on Human–computer Interaction. Lawrence Erlbaum and Associates, Hillsdale, NJ.

MS Excel—Excel© 2003 Microsoft Corporation

Nielsen, J., 1994. Heuristic evaluation. In: Nielsen, J., Mack, R.L. (Eds.), Usability Inspection Methods. Wiley, New York, NY (Updated heuristics online at: http://www.useit.com/papers/heuristic/heuristic_list.html).

Norman, D.A., 1986. Cognitive engineering. In: Norman, D., Draper, S. (Eds.), User Centered System Design: New Perspectives on Human–Computer Interaction. Lawrence Erlbaum and Associates, Hillsdale, NJ.

Peirce, C.S., 1931. Collected Papers. Harvard University Press, Cambridge, MA (1931–1958).

Schön, D.A., 1983. The Reflective Practitioner. Basic Books, New York, NY.

Schön, D.A., Bennet, J., 1996. Reflective conversation with materials, an interview with Donald Schön by John Bennett. In: Terry, W. (Ed.), Bringing Design to Software. Addison-Wesley, New York, NY.

Simon, H., 1981. The Sciences of the Artificial, second ed. The MIT Press, Cambridge, MA.

Yahoo—Yahoo! Groups. © 2003 Yahoo! Inc. (http://groups.yahoo.com/)