Semiotic Engineering – A New Paradigm for Designing Interactive Systems

Clarisse Sieckenius de Souza

SERG – Semiotic Engineering Research Group
Departamento de Informática, PUC-Rio
Rua Marquês de São Vicente, 225
22453-901 Rio de Janeiro, RJ – Brazil
clarisse@inf.puc-rio.br
http://www.inf.puc-rio.br/~clarisse

Abstract. This paper presents semiotic engineering – a semiotic theory of HCI. The theory has the advantage to integrate *back end* and *front end* design and development perspectives into a single metacommunication process that affects the user's experience and, ultimately, the success of any system. By means of illustrative examples, we show the kinds of effects that can be achieved with the theory, and discuss why a semiotic perspective is relevant for the future of information systems.

1 Introduction

This paper presents the gist of semiotic engineering, a semiotic theory of human-computer interaction (HCI) 0. Back in 1980, Ives et al. proposed a model to organize Information Systems (IS) research 0. They structured the world of IS in three layers: the external environment, the organizational environment, and the IS environment. In their view, as a discipline, IS should investigate three embedded environments within the IS environment, namely: the user environment, the IS development environment, and the IS operation environment. Although at the time HCI did not exist as a discipline, in retrospect we see that in the last two decades, the contribution of HCI to IS research comes from complementary perspectives. From *inside* the IS environment, HCI sets out to discover, organize and instrumentalize knowledge about the user environment. From the *outside*, HCI sets out to provide knowledge about how the whole IS environment interacts with the organizational environment and the external environment.

The specific contribution of semiotic engineering to IS design and evaluation is twofold. First, it has the ability to integrate the perspectives of back end and front

end development activities, and to let the users know and enjoy the benefits of all the intellectual efforts that eventually crystallize into *software artifacts*. Second, it has the ability to frame the users' experience within increasingly broader contexts of communication – from basic *user-system* dialogue, to contemporary *user-in-cyberspace* activity. Bearing on key concepts drawn from the work of such semioticians as Peirce 0 and Eco 0, this theory views all instances of HCI as involving a particular case of computer-mediated communication (CMC). In it, the producers of interactive technology talk to users through the interfaces of the artifacts they build. Although this CMC perspective is the hallmark of all semiotic approaches to HCI and IS [5-8], semiotic engineering is different because it is a theory, not a semiotic analysis, of HCI. Hence, it has its *own* ontology, from which specific models and methods to support design and evaluation can be derived [1,9].

One of the difficulties for practical collaboration between HCI and IS researchers is that mainstream HCI is heavily influenced by cognitive theories, like Norman's cognitive engineering and user-centered design (UCD) 0. A study of about research in Computer Science (CS) found that HCI publications are *outliers* compared to others. For example, they do not have Mathematics as a reference discipline, they do not use mathematical methods of analysis, and don't aim to formulate processes or algorithms 0. Because these three features are predominant throughout CS, there is a gap between disciplines, which makes it difficult to turn HCI contributions into a scientific and practical asset for both CS and IS. Semiotic engineering, however, may constitute an important step for bridging this gap.

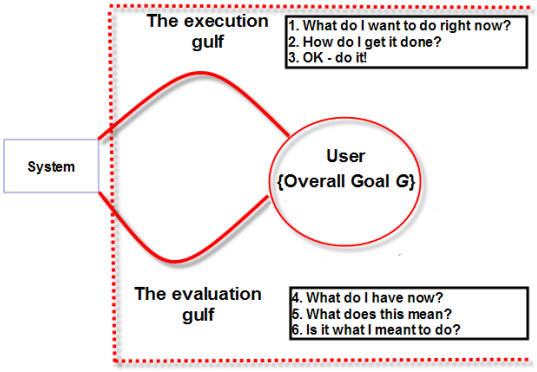


Fig. 1. Norman's execution and evaluation gulfs

A brief illustration of the kinds of contrast between semiotic engineering and Norman's influential cognitive engineering 0, for example, helps to show why IS design and evaluation can benefit from what we propose. Cognitive engineering views human-computer interaction as the traversal of two gulfs (see Figure 1). All interaction is dominated by the user's overall goal. Given this goal, interaction starts by *the user* establishing her immediate intent (*e.g.* playing back her favorite CD in her new laptop), then planning how to achieve it as result of various software functions, and finally executing the plan by activating interface controls. These three

steps help the user traverse the execution gulf that spans between user and system. Next, *the user* must perceive the signal corresponding to the system's reaction (*e.g.* the button ▸ turns from grey to black), interpret what it means (she can press it to start the playback), and evaluate her success. These three steps help the user traverse the evaluation gulf. *User-centeredness* springs from the fact that all relevant activity for HCI is enacted by *the user*, even if as a response to what the system suggests. Norman's original theory does not include *the system* as a partner in HCI, which represents a radical shift from the once traditional view that user and system play equal (or, more often than not, *unequal*) parts in interaction. By the same token, his theory makes it difficult for IS researchers and developers to connect to HCI.

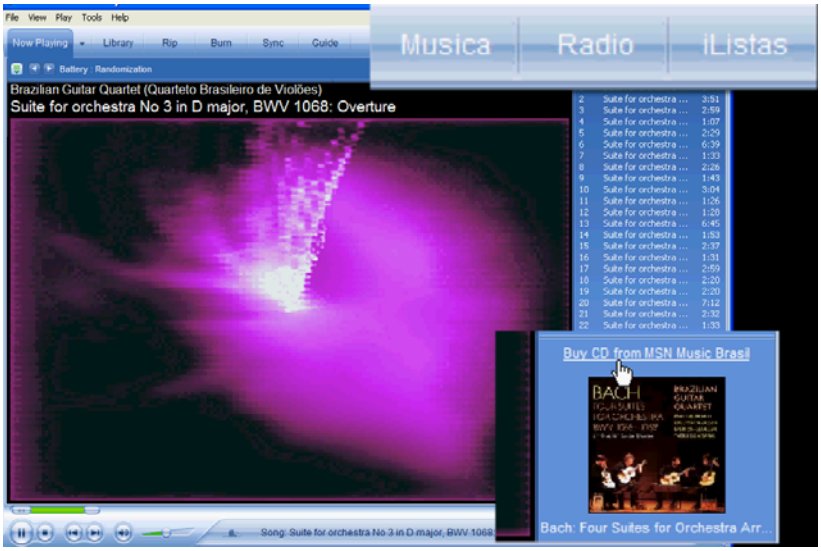


Fig. 2. A screen shot of Windows® Media Player®

The kinds of design concerns that the theoretical foundations of UCD help address are eminently cognitive: How difficult is it for the user to know what to do or expect? How difficult is it to learn something new? How difficult is it to retain and recall it? Which analogies and metaphors can be used to accelerate the appropriate framing of concepts to be learned? As a result, voluminous research and valuable techniques based on cognitive theories helped designers *make interaction easier* for users, and account for most that is meant by *usability*. Nevertheless, usable technologies have to exhibit qualities other than cognitive. For example, what other sorts of theories will explain (or support) design choices and decisions in the Windows Media Player® interface (see Figure 2)? How does a designer integrate *being connected* (and takes full advantage of it) into the user's experience while she is listening to her CD? Which theories and techniques support choices about what information to display on screen, what links to offer (for further details about artists and music), what related activities to enable (buying other CD's or chatting with other fans)? And how should all these things be expressed – through words, images,

sounds, movement? The best answers to each of these and other related questions determine the success of technology, and they are not always easy to find. Notice, for example, that in Figure 2, although the preferred interface language is *English*, an effect of automatic customization (based on the user's IP being located in Brazil) incorrectly introduces linguistic miscellanea in the user's experience (the three rightmost tabs magnified in Figure 2 contain words in Portuguese), but correctly directs the user to "MSN Music Brasil".

Various HCI approaches have helped address the above questions and issues. Some examples are: activity theory 0, the language-action perspective (LAP) 0, social computing 0 and online communities 0. Language and communication-centered approaches like LAP, in particular, have been praised as a relevant *alternative* approach to IS design. According to Hirschheim and co-authors, they "point the direction which some important IS research will likely take in the future to strengthen the interpretive and critical traditions […] within the field" 0.

Compared to both cognitive approaches and LAP, semiotic engineering is different because it emphasizes the communicative role of *designers and developers* in HCI, and brings them up into *the user environment* of IS research. It promotes *intent* (users' and designers') to first-class citizen in HCI, and centers around the necessary communicative settings that will bring designers and users together *at interaction time* 0, to negotiate the scope and evolution of shared meanings encoded in software. Cognitive approaches, for instance, deny the presence of designers at interaction time. And LAP, in spite of its explicit account of IS as communication systems, typically focuses on IS-enabled *communication among users*, and not on designer-user communication.

In the remainder of this paper we will: briefly outline the profile of semiotic engineering; present an example of the kinds of account and epistemic tools that the theory can provide; and discuss the advantages of semiotic engineering as a means to bridge the gap between HCI and IS.

2 Semiotic Engineering

Our theory centers around two fundamental concepts: metacommunication and meaning. Metacommunication is "communication *about* communication". It is the main process taking place in user interfaces and, ultimately, in HCI: interfaces and interaction enable designer-to-user communication$_{(i)}$ about all designed types of system-user communications$_{(ii)}$ and their corresponding effects. Top-level communication$_{(i)}$ is a one-shot comprehensive message that can be paraphrased as:

> *Here is my understanding of who you are, what I've 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.*

The addresser "I" in the message is the artifact's designer (or a spokesperson for the design team), and the addressee "you" is the user (or the users). The content of the message is strongly referenced to the context of design (where the design vision is elaborated and imprinted in the design product). It is unfolded through the process

of interaction, just like the content of a book is unfolded through reading. Hence, ontologically, designers and users belong to the same category – they are interlocutors at interaction time. This is one of the major differences between semiotic engineering and prevailing HCI theories. It underlines the fact that the legitimacy and consistency of the message ("this is the systems that I have build for you") depend on the design intent being shared and restated at every single stage of the software development cycle. In other words, all developers must understand the metacommunication message, agree with it, and contribute to making it clear and useful to the end user. Another difference, related to the second central concept in the theory, is taking meaning to be a culturally-determined constantly evolving process, rather than a fixed target to be captured, encoded and met. Most implicit or explicit theories of meaning supporting computation and software engineering postulate that, just like computer (program) symbols each have their established (enumerable) meaning(s), human meanings occurring in various domains of activity are also fully determined *a priori*. However, human meanings like human life evolve in both predictable and unpredictable ways. In other words, "the user's meaning" is a moving target, and we as developers or designers can never claim to have fully captured it. But we can and do capture relevant parts of it, which are encoded in programs that inexorably compute and predict their occurrence according to well-specified semantic rules. They encode our interpretation of users' meanings in a finite range of possible contexts. The better the job we do at the initial stages of design (through user studies), the greater our chances to communicate and share our understanding with users. When the unavoidable step to outside the boundaries of encoded meanings is made, and the user begins to mean things that are "knowable" but not "known", user's satisfaction will be more dependent on "communicability" than on "usability" 0. Here is a plausible e-commerce scenario to illustrate this.

Scott has been using E-Store for a number of years and different purposes: buying books and computer supplies; buying music CDs, DVDs and cooking books; buying gifts for friends and family of all ages. E-Store uses sophisticated recommendation systems and powerful customization techniques. So, Scott has "his" E-Store, that is nothing like his wife's. Hers looks a different locale – a department store, whereas Scott's feels like a huge music and entertainment warehouse. Based on his purchasing habits over the years, Scott's E-Store puts together recommendations for HCI books back to back with others for children books. Likewise, the organization of store sections gives the same priority access to computer supplies as to flowers (which he often sends to his Mom). He understands this, but he really doesn't like it that much – he'd rather not get the flowers in the way when he visits E-Store for professional purposes, not get recommendations for Dr. Seuss books when trying to check the latest design guidelines for cell phone browsing.

He then decides to use his email strategy to get around the annoyance. Just like he has 5 different emails accounts for different purposes, he chooses to create specialized *personas* for E-Store. He clicks on the 'If you are not Scott click here' link to create "Skip", his "professional clone". All works fine till "Skip" decides to purchase his first lot of professional books. As he provides his credit card and billing address information, a red light flashes online: "The information you provided is apparently that of another user. Our Customer Service Department will get in touch with you, both electronically and through

regular mail. If you want to save the data you have provided so far, click on 'Save information for future use'. We are sorry for the inconvenience."

A number of meaning-related aspects are illustrated by this scenario. First, Scott is happy that *E-Store* provides recommendations and customized shopping experiences. Second, Scott understands how recommendations and customization work. The only annoyance is that over time the mixed types of purchases he makes online mess up *Scott's E-Store*. But, third, no problem: he thinks he knows how to get around the issue, using knowledge from his online culture. Fourth, *E-Store* supports his strategy for a while, but breaks down when it comes to finishing Skip's first purchase online. Why? Because parts of information provided by customers as they conclude purchasing processes online are used as identity keys. As it is usually the case in social life, identity is a *unique* image of you. No two people can share the same *identity*. But online we can have multiple identities, often confused with multiple roles by both users *and* designers.

So what about *evolving meanings* as opposed to fixed meanings? When *E-Store* was developed, all agreed that identifying the users was a key need if security and trust were expected to qualify the users' experience. They decided to identify users by means of a particular tuple of data extracted from the most reliable and valuable piece of information they provide – their credit card information and their personal name and address. Nothing wrong with that. But nobody could predict (not even Scott, if you asked him) that *E-Store* customers would ever wish or need to create clones online. So, designers and developers feel justified with respect to their original choices, Scott feels justified with respect to his current needs. And an interactive problem is in place.

The issues involved in the above scenario are not only strictly pertinent to *the user environment* and how it interacts with the E-Store organizational environment, but also to how the *user environment* relates to the *external environment*, helping users engage in existing social and cultural practices. Lyytinen 0 remarks that sociotechnical approaches to IS bring together "features of the information system, user, and organizational environment". However, he says, because they focus on the *technical* aspects of IS, they may miss important factor lying beyond these (*e.g.* factors that belong to the external, socio-cultural, environment). Semiotic approaches are promising because Semiotics can be viewed as *the logic of culture* 0.

Taking into account cultural signs and practices, and the way they are communicated, semiotic engineering can explain why Scott is right and angry, even if *E-Store* designers and developers are also right and justified. It can also call the designers' and developers' attention to the fact that the systems they produce will necessarily be used in *different* scenarios than the ones they thought of. Therefore, the more efficiently and effectively designers and developers communicate the key concepts of design rationale to users, the more efficiently and effectively users will work around the demands of context evolution and situational change. In this particular case, although the creators of E-store may be excused for not anticipating Scott's specific problems and preferences, merely signaling at the interface that the user's name, address and credit card info combine and constitute his identity should have put Scott's imagination on a more productive path. He would probably realize that his email strategy would not work for E-Store.

Semiotic engineering also explores the design and development consequences of the fact that *computer* meanings have *human* origin and destination. In spite of all formal verification procedures that can prove symbols to be consistently computed one after the other through all layers of software programming, semantic adequacy and relevance actually depends fully on human judgment. It takes a human mind to ascertain that any particular computation is, for all practical purposes, *semantically adequate*, and ultimately *useful* **Feil! Fant ikke referansekilden.**. Hence, although HCI research does not use Mathematics as a foundational discipline or mathematical analysis as a method, knowledge about human meanings should contribute to both CS and IS.

Semiotic engineering proposes to connect both ends by postulating that designers/developers, systems, and users, *all* belong to the same ontological category: they are all interlocutors whose conversations are inter-related. Designers are brought onto the stage of human computer interaction, where *they* communicate what they have done, how, and why, to the users of the artifact they have designed. The system's interface *speaks for designers* at interaction time. The interface conveys all communication that designers must and wish to exchange with users, effect all the expected results, exhibit all the expected behavior, and make all the possible sense of the conversation with the users. They are the legitimate representatives of the designer's *mind*. To design a system thus amounts to designing a rational mechanic spokesperson that will tell users what sense (predicted or not) to make of the artifact 0. Most importantly, it also amounts to designing the remedial sense-making and meaning-negotiating dialogues, which will give users resourceful signs to reason upon and recover from communicative breakdowns and misunderstandings.

Implicit in the above is the fact that the system, as the *designers' deputy*, must be able to explain itself to end users, to disclose the essence of its logic and rationale, in case of interactive breakdowns and/or system repurposing. This can only be achieved with sound underlying models that are comprehensible and satisfactory for all members of the development team. If the semantics of the design rationale, as intended by the designer, is tweaked or misinterpreted by developers, the users will suffer the consequence of nonsensical interface discourse.

3 Communication and Metacommunication with an Online Store

Like many other online stores, Amazon.com® makes extensive use of recommendation and customization techniques. Figure 3 shows a piece of the page *Clarisse* gets as she goes to http://www.amazon.com.

Notice the explicit conversational style of interaction ("Hello, Clarisse"), reinforced in numerous dialogues as, for example, in the "Frequently Asked Questions" about recommendations (see Figure 4). FAQ techniques introduce a "user's deputy" in the conversation, one that speaks for the user (*e.g.* "Are my recommendations saved so I can look at them again later?"). The use of "I", "you", "we" establishes a speaker/listener structure, and even signs of persuasive rhetoric are present in the dialogue (*e.g.* "We wouldn't want you to miss something you

might enjoy!"). However, this natural conversation feeling is shaken when the user's deputy asks: "How do I turn off recommendations?" The advice is: "Simply click the link on our home page that says 'If you're not (your name), click here.' Then, leave the e-mail and password spaces blank and click the 'Amazon.com' tab. This will remove our recommendations for you until you sign in again."

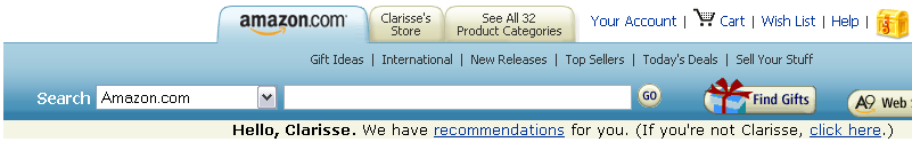
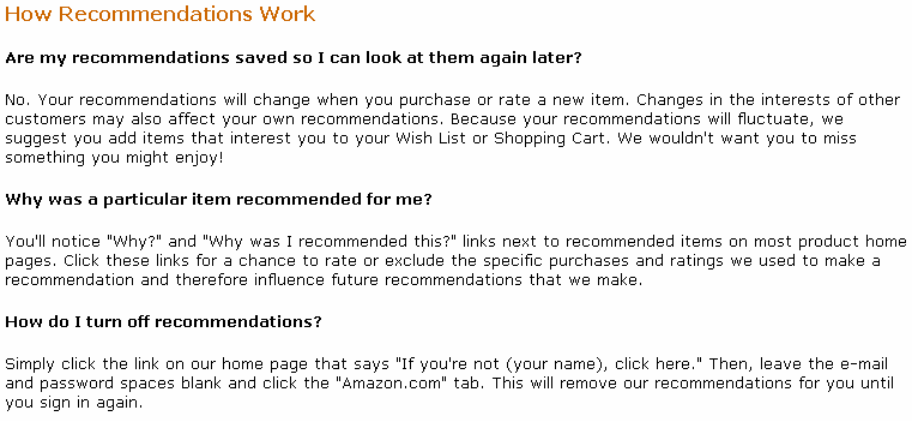


Fig. 3. A detail of Clarisse's customized entry page at Amazon.com



How Recommendations Work

Are my recommendations saved so I can look at them again later?

No. Your recommendations will change when you purchase or rate a new item. Changes in the interests of other customers may also affect your own recommendations. Because your recommendations will fluctuate, we suggest you add items that interest you to your Wish List or Shopping Cart. We wouldn't want you to miss something you might enjoy!

Why was a particular item recommended for me?

You'll notice "Why?" and "Why was I recommended this?" links next to recommended items on most product home pages. Click these links for a chance to rate or exclude the specific purchases and ratings we used to make a recommendation and therefore influence future recommendations that we make.

How do I turn off recommendations?

Simply click the link on our home page that says "If you're not (your name), click here." Then, leave the e-mail and password spaces blank and click the "Amazon.com" tab. This will remove our recommendations for you until you sign in again.

Fig. 4. A detail of Amazon.com explanations about how recommendations work

By framing HCI as metacommunication and shifting the traditional user-system interaction to a user-designers' deputy conversation, semiotic engineering provides conversational models and methods for designing critical parts of such metacommunication. For example, it calls the designer's attention to the importance of observing factors like topical structure in conversation. Notice that the user (through his deputy) is asking about "turning off recommendations". But the designer's deputy responds with apparent *non sequitur* discourse: "Click on the link that says 'If you're not (your name), click here'". Why raise a question of identity for turning off recommendations? Even worse: Why advise the user to belie her own identity? What other kinds of convenient side effects may this socially serious misbehavior cause? Why not simply have a link saying "Turn recommendations off", or another saying "Visit the store anonymously"?

Following Schön's reflection-in-action design paradigm 0, semiotic engineering bets on epistemic tools, which fire the designer's semiosis along certain structured paths for reflection. Thus, when the interaction style is explicitly and prominently conversational as with Amazon.com, the designer is led to ask himself questions about fundamental issues for productive verbal interchange: the consistent identification of interlocutors, topic and purpose of each message, turn-taking and rhetorical structure, and so on. More than that, in the spirit of semiosis, the designer is prompted to ask further questions, and explore the design space, taking semiotic engineering on board as an *epistemic* resource theory.

In order to highlight the relevance of such issues for both front end and back end design and development activities, note that semiotic engineering has the power to raise issues of *reuse* in the Amazon.com example. Epistemic tools for designing communication in multi-user applications like groupware, online communities and others capture the design rationale and use it extensively for building the designer's deputy discourse 0. Thus, in the process, a designer is not likely to explain and justify a change of identity as a rational solution for turning off recommendations. This would be in obvious contradiction with one of Grice's famous maxims for a *logic of conversation* 0, which we integrate to our tools.

Although the same range of effects caused by a given *program* may be interpreted in a number of different ways and serve many different purposes (*e.g.* allow for anonymous visits to an e-store and momentarily clear the recommendation list of long-time customers), there must be a differentiation of expression and representation when humans are engaged in interpretive processes for which such differences matter. This is the case not only of end users, but also of maintenance programmers and technical documentation writers. They must all know what the system can and cannot do, no matter how extensively reuse techniques have been applied to accelerate its development cycle or optimize its size and performance.

Just for illustration, one of our tools walks the designer through the communication design space, asking questions like: Who is speaking? To whom? What is the speaker saying? Using which code and medium? Are code and medium appropriate for the situation? Are there alternatives? Is(are) the listener(s) receiving the message? What if not? How can the listener(s) respond to the speaker? Is there recourse if the speaker realizes the listener(s) misunderstood the message? What is it?

A walkthrough of FAQ-style interaction reveals some interest facets. Is "the user" really speaking? Are the words in the questions phrasing really hers? And if they aren't, should "you" and "yours" be used instead of "the customer"?

One last aspect that is somewhat related to identity, but more precisely to legitimate agency, can be seen on the snapshot in Figure 5, on the FAQ about "the page you made". Curiously, the designer's deputy is telling the user that she is (unknowingly and perhaps unwillingly) *making* an HTML page as she navigates through the store. But *she* isn't. The *system* is automatically assembling this page on the user's behalf, which raises issues of control and legitimacy in the whole cycle of interaction. Users may end up asking themselves what *they* are doing, and even who they *are*, given that the system is apparently taking the user's identity and doing unsolicited things along the way. Some may be pretty nice, some may not. Can the

user always trust this system then? The ethical implications of such choices are all very likely to emerge in the designer's semiosis along the design process.

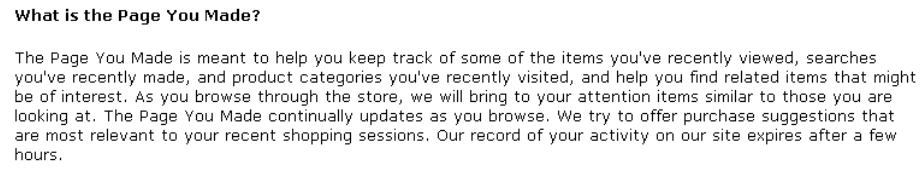


Fig. 5. A detail of Amazon.com explanations about the page you made

4 Concluding Remarks

Although semiotic engineering is firmly established in the HCI camp, it sheds light on universal semiotic processes that occur throughout the development cycle, and on the kinds of commitments and consequences that one is expected to assume when it comes to producing useful, pleasurable, high-quality information technology.

Among the points we've raised in this paper, we want to highlight the following. First, we reject the view that meaning is a fixed ideal value that designers can elicit from users and hopefully encode into a system. The expectation that well done user studies will capture *the* user's meaning and *entail* satisfaction denies the intrinsic creative and evolutionary character of human nature. In terms of the future of IS research, this point suggests that abductive reasoning systems 0 and even evolutionary computing 0 may provide radically different conceptions of computation, and consequently broaden the spectrum of meanings that can be exchanged between the internal components of the IS environment, and between the IS environment and the socio-cultural environment (not only the sociotechnical environment).

Second, semiotic engineering favors model-based design and development, although for a somewhat different purpose than is usually the case in literature 0. Instead of using models to generate implementations of specifications automatically, we propose to use them to generate explanations about design and implementation. These explanations should be primarily used to elicit and negotiate interpretations and meanings throughout the development cycle, with a positive effect on the user's experience at the very end of the process chain. That these models can be used for program generation or transformation is the object of formal methods investigation. The semiotic engineering point is that the semantic adequacy of representations used for specifications and programming is the object of *human* judgment, and not of automatic syntactic manipulations of symbols.

Third, because the fundamental process in HCI is metacommunication of design rationale, it is of prime importance that this rationale be not undermined by programming practices (like the case of reuse, in our example) that cannot guarantee the consistency of the designer's deputy's *discourse at interaction time*, and hence

make sense to users. The role of contingency and context in intelligible communication 0 challenges the idea that design and implementation components can be *reused* without problems in communicative situations other than the ones they have been originally used for. The reuse ideal is fundamentally dependent on fixed and universal meanings. In terms of the future of IS research, this point suggests that software and design components, objects or patterns should include representations of the metacommunicative meanings with which they are thought to be associated. This *integrative* view of back end and front end issues is one of the strengths of semiotic engineering.

Fourth and finally, because meanings evolve in unpredictable ways, allowing users to customize and extend applications (broadly covered by the term *end user development* 0) deserves high-priority among development techniques that are in line with our theory. Users should be able to incorporate contingent meanings to the technology, achieving *evolutionary computing* in a very particular way. Viewed from the perspective of the *external environment*, the *IS environment* would evolve on demand.

Other theories and approaches to HCI usually don't bring all the above issues together. They tend to focus on one or another aspect only, contributing to the feeling that IS and HCI belong to worlds apart. The separation creates tension and favors independent initiatives that try to take care of the user environment within IS based on ontologies and models that exclude some of the most fundamental aspects of the users' experience. Because of its semiotic foundations, whereas other theories seek to provide tools and methods that generate *answers* to design problems, semiotic engineering's tools and methods are meant to generate *questions*. As *epistemic* tools, they are not intended to replace other tools, neither is the theory intended to replace other theories. This may frustrate IS developers and researchers, who would like to get answers for long-standing questions in the field. But, as Hirschheim and co-authors say 0, alternative IS development approaches, including those based on language and communication, are important because they represent useful scientific counterparts of orthodox views. Altogether, we strongly believe that semiotic engineering is a useful bridging theory for bringing together IS and HCI.

Acknowledgment

The author thanks CNPq, the Brazilian Council for Scientific and Technological Development, for giving continued financial support for her research.

References

1. C.S. de Souza. *The semiotic engineering of human computer interaction*. Cambridge, MA. The MIT Press (2005).

2. R. Hirschheim, J. Iivari, and H.K. Klein. A Comparison of Five Alternative Approaches to Information Systems Development. *Australian Journal of Information Systems*, Volume 5(1). (1997).
3. C.S. Peirce. *Collected papers of Charles Sanders Peirce, Vols. 1-8*, C. Hartshorne and P. Weiss. Cambridge, MA. Harvard University Press (1931-1958).
4. U. Eco. *A theory of semiotics*. Bloomington, IN. Indiana University Press (1976).
5. P.B. Andersen, B. Holmqvist, and J.F. Jensen *The computer as medium*. Cambridge. Cambridge University Press (1993).
6. J. Kammersgaard. Four Different Perspectives on Human-Computer Interaction. *International Journal of Man-Machine Studies* 28(4) pp. 343-362. (1988).
7. K. Liu. *Semiotics in Information Systems Engineering*. Cambridge. Cambridge University Press. (2000).
8. M. Nadin. Interface design and evaluation. In R. Hartson, D. Hix (Eds.) *Advances in Human-Computer Interaction, Vol. 2*. Norwood, NJ. Ablex Publishing Co. (1988).
9. C.S. de Souza. Semiotic engineering: bringing designers and users together at interaction time. *Interacting with Computers*, 17 (3) pp. 317-341. (2005)
10. D.A. Norman and S.W. Draper. *User-centered system design.* Hillsdale, NJ. Laurence Erlbaum (1986).
11. V. Ramesh, R.L. Glass, and I. Vessey. Research in computer science: an empirical study. The Journal of Systems and Software 70 (2004) pp. 165-176
12. B.A. Nardi. *Context and consciousness.* Cambridge, MA. The MIT Press. (1996).
13. T. Winograd and F. Flores. *Understanding computers and cognition*. New York, NY. Addison-Wesley (1986).
14. P. Dourish. *Where the action is*. Cambridge, MA. The MIT Press (2001).
15. J. Preece. *Online Communities: Designing Usability and Supporting Sociability*. New York, NY. John Wiley & Sons, Inc (2000).
16. K. Lyytinen. Different Perspectives on Information Systems: Problems and Solutions. *ACM Computing Surveys*, Vol. 19, No. 1, March 1987. pp. 5-46 (1987).
17. D. A. Schön. *The Reflective Practitioner*. New York, NY. Basic Books (1983).
18. H.P. Grice. Logic and Conversation. In: P. Cole & Morgan (eds.), *Syntax and Semantics 3: Speech Acts*. New York, NY. Academic Press (1975).
19. J.R. Josephson and S.G. Josephson. *Abductive inference: computation, philosophy, technology.* Cambridge. Cambridge University Press (1994).
20.D.B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Piscathaway, NJ. IEEE Press (1995).
21. F. Paternò. *Model-Based Design and Evaluation of Interactive Applications.* Heidelberg. Springer (1999).
22. L.A. Suchman. *Plans and Situated Action.* Cambridge. Cambridge University Press (1987).
23. Lieberman, H.; Paternò, F.; Wulf, V. *End-User Development*. Human-Computer Interaction Series, Vol. 9. Heidelberg. Springer. (2006).