End-user (further) development: A case for negotiated semiotic engineering

Clarisse Sieckenius de Souza
Departamento de Informática, PUC-Rio
clarisse@inf.puc-rio.br

Semiotic engineering

Semiotics is a discipline devoted to studying signs and signification, which includes processes of representation, interpretation, sense making, and – for a number of semioticians – communication[1]. Its object of investigation is thus strongly connected with that of various sub-areas of Computer Science such as: Artificial Intelligence, Human-Computer Interaction, and even Theoretical Computer Science. In HCI, specifically, the most popular, although often superficial and restricted, use of Semiotics has been the famous classification of signs into *icons*, *indices* and *symbols* proposed by Peirce[2]. However, just as Cognitive Psychology has the power to provide the foundations of full-fledged theories of HCI, so does Semiotics. Semiotic Engineering[3] is the first proposed theoretical account of HCI in general based on semiotic theories (mainly Eco's and Peirce's).

The gist of Semiotic Engineering involves the following main concepts: metacommunication, semiosis, signification and communication. All of them are familiar to semioticians, and have not been originally proposed by Semiotic Engineering. What is new, however, is how they can be put together to characterize and explain HCI, to generate HCI research questions and methods, among which some related to end-user development. In fact, EUD holds an important position in this theory for the reasons briefly presented in the following paragraphs.

Metacommunication is classically *communication about/of communication*. Semiotic Engineering views HCI as a specific type of metacommunication, a process where systems' developers communicate to systems' users how they (users) can/should communicate with the system in order to achieve a particular range of intended effects. So, for example, if you are using a text editor to write a position paper, you are in fact getting (and reacting to) the developers' message about all the things you can do with their software in order to create great-looking documents. Of course this developer-to-user message is received progressively by users, as they interact with the communicative agent that represents the developers at interaction time: the system itself, or the designer's *deputy* as we say in Semiotic Engineering terms. Some important shifts of perspective follow from this. First, developers participate in interaction (the system speaks for *them*), which represents a radical change compared to the classical user-centered model of HCI[4], for instance. The change does not take users out of the scene. It includes designers/developers in it, and by so doing expands the topic of interactive exchanges from tasks to design intent, rationale and value. Second shift, problem-solving and cognition do not constitute the focus of investigation in this theory. Communication is the new focus. Thus, problem-solving and cognition are only covered by the theory inasmuch as they constitute the object or purpose of communication. Third shift, developers and users belong to the same ontological category – they are interlocutors in computer-mediated communication. To our knowledge, this is the only theory of HCI (and maybe one of the few, if not also the only, theoretical account that can be used to characterize Software Engineering, in a broad

Dagstuhl Seminar Proceedings 07081
End-User Software Engineering
http://drops.dagstuhl.de/opus/volltexte/2007/1083

sense) where software producers and software consumers, and their respective purposes and activities, can be described in terms of the same ontology.

Semiosis is process through which we generate (interpretive) signs in the presence of something that we take to stand for something else. For example, if you see this on a text page, you are likely to take it to mean *a hyperlink*. So, in your process of interpretation you generate other signs (*i.e.* things that, themselves, stand for various other things, to you). Among the signs you generate in your interpretation it is very probable that you will have a sign representing the expectation that when you put the mouse on the underlined blue text you see this. This sign will be part of your interpretation of this unless your expectation fails. All the signs generated in this interpretive process are part of what the original (base) sign means to you. Very importantly, they are subject to further revision, as the example shows. When expectations or inferences are contradicted by current factual evidence, you *change* your previous interpretation by generating other signs that accommodate the new information you just acquired. This "generate-revise" interpretive process, described and defined by Peirce as *abduction*[5], is continuous, and so we say that semiosis is *continuous*, or *unlimited* over time. Consequently, a semiotic theory of meaning, of Peircean breed, does not view meaning as a *static* entity associated to representations, but as an ongoing process that includes unpredicted and unpredictable signs. The fundamental role of semiosis in Semiotic Engineering is to characterize more precisely the developers-users interlocution at interaction time. Although both developers and users share the same interpretive capacities (that are species-specific for Peirce), computer mediation introduces a radical reduction in the developers' abilities to communicate productively with users during interaction. The system, unlike human beings, is not capable to carry on *unlimited semiosis*. Quite contrarily, it is in the nature of computer representations that, for all practical purposes, they need *grounding*. An examination of the semantics of computer programs can show the various pre-established meanings that developers have associated to the interactive signs that users will be exposed to and will be able to use in order to get the computer to exhibit various types of behavior. So, very briefly, although developers and users are communicating to each other (through the system), and thus are both interlocutors in the same conversation, computer mediation imposes an important limitation for both parties. Developers must realize that they won't have the usual unlimited human capacity to explain and revise what they mean by the kinds of interaction they invite users to have with the system they have designed. And users must realize that what they mean to communicate to the system will only be understood (and effective) if it is consistent with a pre-established range of meanings that have been encoded in it. Users can always explain and will constantly revise (and expand) *their* meanings, of course. And this is the fundamental link between Semiotic Engineering and End User Development.

Finally Semiotic Engineering uses two definitions from Eco's Semiotics[6]: signification and communication. **Signification** is the process by which certain *contents* are systematically assigned to certain *expressions* as a result of deep and strong cultural conventions. **Communication** is the process by which interlocutors *explore* the signification systems within their reach in order to produce signs meant to achieve an unlimited range of purposes and effects. They can not only pick up culturally established signs in the process, but they can also (and *extensively do so*) invent new expressions and/or use the signification system in innovative ways. The beauty of human communication is that just as sign producers are prepared (and actually inclined) to express themselves innovatively, sign consumers are equally well-equipped to interpret creative expressions, exactly because they are naturally born with the ability to think abductively. So, human communication is a negotiation of meanings, where abduction plays a central role, allowing interlocutors to revise constantly their assumptions and expectations about each other's understanding and intent. For the purpose of this discussion about EUD, Semiotic Engineering draws two important consequences from these notions. One is that it is *natural* for users to use any computer-encoded signification system in innovative ways. The other is that *usable* technologies must necessarily support revisions of the computer-encoded signification systems they support[7].

Designing at interaction time

Semiotic Engineering provides theoretical arguments to support what Liberman and co-authors express in the opening chapter of *End user development*: "We think that over the next few years, the goal of human-computer interaction (HCI) will evolve from just making system *easy to use* (even though that goal has not yet been completely achieved) to making systems that are *easy to develop*"[8]. Natural human communication is extensively innovative compared to the expressions that can be systematically derived from any given signification system. Innovation can focus on the expression side of the system (*e.g.* new expressions or expressive modes can be used to convey well known content), on the content side (*e.g.* a well known expression can be used to refer to a modified version of its previously known corresponding content), or on both (*e.g.* new expressions can be instantly produced to signify new content, or new expression/content correspondences can be created to achieve particular effects in communication). Using innovative forms of communication always requires additional interpretive efforts from interlocutors, who will typically engage in adbductive reasoning processes to interpret what they are being told. However, the efficacy and efficiency of communication can be very positively affected by precisely such innovations. At one end of the spectrum, it is clear that without them no evolution (of culture, society, science, and even personal lives) would be possible. At the other, it is also clear that communication constrained by perfectly ordinary situational factors, such as lack of time or space, wouldn't be possible otherwise. For instance, if this example worked for you, it's because you undertand innovative communication. If not, I need more time and space to explain it to you. But once you get the idea, you will be able to use this yourself to communicate things you mean.

To communicate is thus to *design* forms of expression that will effectively and efficiently cause your intentions to be fulfilled. Some of the EUD-related challenges for HCI within a Semiotic Engineering perspective are to: (i) let users communicate more naturally (hence, more effectively and efficiently) with systems; (ii) let developers communicate more effectively and efficiently to users *the limitations imposed by computer mediation* to their mutual understanding; (iii) help developers design various ways computer-encoded signification system manipulations that users can choose to explore interface languages in order to communicate innovation; and (iv) develop theoretical concepts and models to explain, characterize and expand the connections between HCI and EUD. Because the success of HCI for Semiotic Engineering is measured by the developers' ability to get their metacommunication message across to users[9], it is important that the gist of this message be preserved at least as a reference for further developments. Revisions of meanings encoded in an application's signification system must not destroy the original developers' message. Thus, the kind of EUD that Semiotic Engineering is prepared to deal with only involves negotiating meaning revisions with the designers' deputy at interaction time. This particular case of EUD might best be named end user *further* development.

SERG's related research publications

de SOUZA, C. S. ; BARBOSA, S. D. J. (2006) A semiotic framing for end-user development. In: Henry Lieberman; Fabio Paternò; Volker Wulf. (Org.). *End User Development: Empowering people to flexibly employ Advanced Information and Communication Technology*. New York: Springer, 2006, v. 9, p. 401-426

de SOUZA, C. S. (2005) Semiotic engineering: Bringing designers and users together at interaction time. *Interacting with Computers*. Vol. 17, n. 3, pp. 317-341.

BARBOSA, S. D. J., de SOUZA, C. S. (2001) Extending software through metaphors and metonymies. *Knowledge Based Systems*. Vol.14, n.1-2, pp.15-27.

de SOUZA, C. S., BARBOSA, S. D. J., SILVA, S. R. P. (2001) Semiotic Engineering Principles for Evaluating End-User Programming Environments. *Interacting With Computers*. Vol.13, n. 4, pp.467-495.

BARBOSA, S. D. J. (1999). *Programação via interfaces*. [Title in English: Programming via interface]. Ph.D.Thesis in Portuguese. Presentation: 23/12/1999. 109 p. Advisor: Clarisse Sieckenius de Souza.

[1] Eco, U. (1984) *Semiotics and the philosophy of language*. Indiana University Press.
[2] Houser, N. and Kloesel, C. (Eds.) (1992-1998) *The essential Peirce*. Vols. I, II. Indiana University Press.
[3] de Souza, C. S. (2005) *The semiotic engineering of human-computer interaction.* The MIT Press.
[4] Norman, D. A. (1986) Cognitive Engineering. In *User Centered System Design* (Norman & Draper, Eds.). Lawrence Erlbaum.
[5] See note 2.
[6] Eco, U. (1976) *A theory of semiotics*. Indiana University Press.
[7] For a discussion about usability and creative use see Adler, P. & Winograd, T. (1992) *Usability: Turning technologies into tools*. Oxford University Press.
[8] Lieberman, H.; Parternò, F.; Klann, M.; Wulf, V. (2006) End-user development: An emerging paradigm. In *End-User Development* (Lieberman, Paternò and Wulf, Eds.). Springer. p. 1.
[9] Prates, R. O., de Souza, C. S., and Barbosa, S. D. 2000. Methods and tools: a method for evaluating the communicability of user interfaces. interactions 7, 1 (Jan. 2000), 31-38.