Semiotically-informed design: a case for semiotic engineering

Clarisse Sieckenius de Souza
Departamento de Informática, PUC-Rio
Rua Marquês de São Vicente, 225
Gávea, Rio de Janeiro, RJ
Brazil, 22451-900
+55 (21) 3114-1500 ext. 4344
clarisse@inf.puc-rio.br

Simone Diniz Junqueira Barbosa
Departamento de Informática, PUC-Rio
Rua Marquês de São Vicente, 225
Gávea, Rio de Janeiro, RJ
Brazil, 22451-900
+55 (21) 3114-1500 ext. 4353
simone@inf.puc-rio.br

Thaís Chiappetta Salgado
Departamento de Informática, PUC-Rio
Rua Marquês de São Vicente, 225
Gávea, Rio de Janeiro, RJ
Brazil, 22451-900

ABSTRACT
The design of organizational systems has recently gained one magnitude of complexity, as social issues have become at least as important as technical ones. The challenge for developers is to integrate knowledge and techniques springing from two different sub-fields in Computer Science, namely Human-Computer Interaction (HCI) and Software Engineering (SE). The task is not easy, since theories, ontologies and values driving research on each sub-field have typically developed independently of each other, yielding unrelated, if not conflicting, results. There are usually two strategies to achieve integration: extending knowledge from any one end, and working towards the other; or reaching for a higher-level integrative framework, where lower-level ends are shown to be parts of the same whole. Work in Organizational Semiotics has explored both strategies, with an emphasis on the first – especially working from SE towards HCI. This paper reports a study in Semiotic Engineering – an HCI theory, and shows the integration of concerns that this theory has allowed us to achieve in the redesign of an online forum. Evidence from this study is discussed against the background of reported work in Organizational Semiotics, showing how Semiotic Engineering can be combined with Organizational Semiotics to develop such valuable support for communities of practice and other knowledge-intensive groups as online forums.

Categories and Subject Descriptors
H.5.2 [Information interfaces and presentation (e.g. HCI)]: User Interfaces – Theory and methods

General Terms
Design, Human Factors, Theory.

Keywords

1. INTRODUCTION
The beginning of this century has witnessed an increase in concern about the social aspects of computer technology, from online communities [32], to social computing [12], to knowledge management [44], and communities of practice [45]. Ackerman has even suggested that the main challenges for technology are social and not technical [1], whereas Dourish proposes that social and tangible approaches to computing are probably the most promising ways to break the barriers of tradition in computation [12]. Organizational Semiotics [22] is a new discipline that aims at providing solid semiotic foundations to integrate information systems development perspectives with communicative perspectives of linguistic and semiotic breeds. The power of models and methods for the development of organizational information systems has been illustrated by Liu [21], Stamper [41], and others.

This paper reports an in-depth study about the design of online help for an electronic forum system (e-forum). The study is based on Semiotic Engineering1 [9, 10], a semiotic theory of human-computer interaction (HCI), and addresses two issues of interest for Organizational Semiotics (OS). First, its theoretical foundations are perfectly akin to work in OS. But, second, unlike most work in OS, it is strictly focused on HCI. Its main contribution lies in providing evidence to contrast the results of two ways to achieve integration of systems engineering with interaction design, namely: [i] extending software engineering models, methods and techniques to include HCI concerns; and [ii] using non-cognitive HCI theories to inform systems design.

Findings of this study confirm those of previous research in Semiotic Engineering. Viewing HCI as metacommunication between designers and users helps developers to gain increased awareness of how their decisions regarding software functionality and architecture affect the users’ experience. This is critically important knowledge for socio-technical systems, social computing, and communities of practice in organizational contexts. The study also shows that Semiotic Engineering is considerably different from OS in many respects, and discusses how such differences can contribute to motivate the generation of higher-level, integrative theoretical frameworks with which major social challenges for computer technology can be addressed.

---

1 The term ‘semiotic engineering’ has also been used by Jorna in [18], although with a different definition and for different purposes than de Souza and co-authors have been using since 1993.
In the next section, we will contrast semiotic engineering with existing HCI design approaches and theories, mostly based on cognitive theory and ethnographic practices. Next, we will introduce OriOn and show how and why it has been redesigned based on the semiotic engineering of the metacommmunication message from designers to users. Then we will discuss the findings of our study in the light of OS. In conclusion, we will express our intuitions or informed opinions about possibilities of bringing Semiotic Engineering and Organizational Semiotics together for the benefit of software quality.

2. SEMIOTIC ENGINEERING BASICS

Like most other semiotic approaches to HCI (e.g. [3, 19, 26]), Semiotic Engineering views human-computer interaction as a special kind of computer-mediated communication (CMC). Semiotic Engineering, however, carries this perspective much deeper than previous approaches, and provides a full-fledged conceptual characterization of HCI as metacommmunication – communication (from designers to users) about all kinds of (user-system) communication enabled by computer artifacts, and about the effects that can be achieved with it, by design [9]. The theory includes a communication-centered ontology of HCI phenomena that can be used to design and evaluate the users’ experience. It also moves smoothly into programming and computation, by means of representations – a key concept in Semiotics and Computer Science. Thus, useful knowledge about theoretical and technical limits of CMC can be generated at design time, for the benefit of both developers and users.

2.1 Semiotic Engineering compared to Cognitive and Ethnographic theories

The paradigmatic contribution of cognitive theories of HCI is User-Centered Design (UCD), a “winner” approach to designing interaction championed by Norman [29, 31]. The key to UCD is to build a proper system image – a set of physical characteristics associated to a computer artifact’s appearance and behavior. The system image expresses the designer’s conceptual model of tasks, goals, and use situations. Good HCI design amounts to producing system images that will make the user’s conceptual model coincide with the designer’s. In order to do so, designers should concentrate on users’ cognitive abilities, taken to be universal within homogeneous populations of users, such as teenage groups, organization employees, technical professional communities, office workers, and so on. This view of HCI has boosted voluminous research on cognitive factors related to task structuring, interface style, and conceptual metaphors, for example. Usability metrics of all sorts have been proposed to measure distances between designed and perceived features of interactive systems (see early work in [28] and recent work in [36]), based on the assumption that cognitive abilities exhibited by sample populations participating in usability studies are universal, and thus constitute a fixed target for designers to aim at.

The universal character of relevant aspects of the users’ experience has been deeply questioned, however, especially by ethnographers [42, 17], ethnographically-inspired computer scientists and HCI experts [12, 25], as well as by supporters of Activity Theory as an alternative foundation for HCI [27]. In their view, the social (and cultural) context of the users’ activities have unique characteristics that are not properly captured by cognitive theories. Research methods, in particular, differ considerably, since non-cognitive approaches tend to favor qualitative over quantitative methods, reaching for in-depth understanding, rather than fragmented universal knowledge. They have been particularly successful for the analysis of users’ needs and context (e.g. contextual inquiry [7]), but less successful than cognitive theories in proposing alternative interface styles, interactive design patterns, and other generic (although narrowly focused) HCI solutions.

Compared to cognitive and non-cognitive theories to-date, Semiotic Engineering has a number of distinctive features. First and foremost, unlike either, Semiotic Engineering includes designers as active participants at interaction time. Systems interfaces are the designers’ representatives (the designers’ deputies), who speak for the designers, responding to each and every users’ “interactive action” (or communicative act) with logical and systematic interpretations and behavior. Such interpretations and behavior are decided by the designers at design time, based on extensive user studies, domain and task models, and so on. At later stages of systems development, they are encoded in computer programs, and crystallized into fixed or enumerable sets of meanings that can be productively exchanged by users and “systems” in practical situations. Second, because Semiotic Engineering switches the role of HCI designers from back-stage wizards to front-stage (computer-mediated) HCI participants, designing interaction changes from a 3rd-person activity (i.e. one that specifies the parts played by others in conversations that do not involve the specifier) to a 1st-person activity (i.e. conversations that involve references to and decisions made by oneself). For example, one of Microsoft Word® interface dialogues says: “Word has examined the document and picked the sentences most relevant to the main theme.” Of course Word hasn’t done any of this, except by virtue of human expert decisions and programming skills. The effect of switching the ontological perspective from product (Word) to producer (Microsoft design teams), is not necessarily a switch from “Word” to “we, designers”, or anything like that. The most important effect is to prepare designers to talk about what they think users would ask them, not Word. For example, users are much likely to ask: “How do you decide whether a word is relevant to the main theme of this document?” Or, even: “How do you pick up the main theme of this document?” Interestingly enough, the dialogue window where this conversation takes place does not give access to such questions (see Figure 1; paper clip question prompt appears after pressing F1).
conversations, they may prevent users from gaining access to conversations at interaction time. If they do not design for such participation, through their deputy, in explanatory and exploratory cultural aspects of situated use, because they must be prepared to naturally leads designers to reflect on psychological, social and perspective (“why and what for it has been designed”). The latter functional and operational perspective (“how it works”), but also view and understanding of the whole system, not only from a functional and operational perspective (“how it works”), but also and perhaps more importantly from a logical and strategic perspective (“why and what for it has been designed”). The latter naturally leads designers to reflect on psychological, social and cultural aspects of situated use, because they must be prepared to participate, through their deputy, in explanatory and exploratory conversations at interaction time. If they do not design for such conversations, they may prevent users from gaining access to prime insights about technology and becoming creative power users in much shorter time.

The theoretical basis of this mechanism is simple and familiar to all acquainted with semiotics. Semiotic theories of Peircean breed view sign interpretation as an ongoing unlimited process, geared by abductive reasoning, capable of generating indefinitely many meanings (new signs) [13]. The generation of new meanings is affected by the presence of other occurring signs in the same abductive chain. Thus, as soon as HCI designers allow users to have quick and easy access to key design rationale signs, they open completely new meaning-making opportunities for users. Going back to Word’s “Auto Summarize” facility, quick and easy access to the fact that summarization is based on sentence patterns the user selects the appropriate index entry returned by the paper clip (in this case “About automatically summarizing a document”). No answer to any of the two questions in the paragraph above is provided if the user chooses another entry (like “Automatically summarize a document”, for instance). In other words, situating designers at interaction time helps them situate conversation better.

Fourth and finally, for the purposes of this paper, Semiotic Engineering proposes that the content of the metacommunication message from designers to users can be paraphrased as follows:

“Here is my [the designer’s] understanding of who you [the users] 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.” [9: 84]

This formulation clearly motivates, and requires, an integrated view and understanding of the whole system, not only from a functional and operational perspective (“how it works”), but also and perhaps more importantly from a logical and strategic perspective (“why and what for it has been designed”). The latter naturally leads designers to reflect on psychological, social and cultural aspects of situated use, because they must be prepared to participate, through their deputy, in explanatory and exploratory conversations at interaction time. If they do not design for such conversations, they may prevent users from gaining access to prime insights about technology and becoming creative power users in much shorter time.

The theoretical basis of this mechanism is simple and familiar to all acquainted with semiotics. Semiotic theories of Peircean breed view sign interpretation as an ongoing unlimited process, geared by abductive reasoning, capable of generating indefinitely many meanings (new signs) [13]. The generation of new meanings is affected by the presence of other occurring signs in the same abductive chain. Thus, as soon as HCI designers allow users to have quick and easy access to key design rationale signs, they open completely new meaning-making opportunities for users. Going back to Word’s “Auto Summarize” facility, quick and easy access to the fact that summarization is based on sentence patterns might give users the idea to deliberately use some patterns rather than others, in order to optimize summarization.

2.2 Epistemic design tools

The theoretical foundations of Semiotic Engineering have a number of implications for the use of this theory in the design and evaluation of interactive computer artifacts. One of the most important of these is the nature of tools that can be derived from the theory (e.g. models, methods, analytic frameworks, etc.). By subscribing to the view that meaning is not a fixed state or entity, but rather a chain of signs produced by an ongoing process of interpretation (called semiosis), halted and resumed by virtue of contingencies inherent to the abductive essence of human sense-making, Semiotic Engineering cannot produce the same tools as cognitive theories of HCI have produced. For example, there is no way to guarantee that the system image (a sign, whose representation is the system’s physical characteristics and perceptible behavior) will make the user’s model of it (the user’s interpretant for the system sign) coincide with the designer’s. It is not possible to postulate that ongoing contingent sense-making processes, taking place in different minds, different contexts, and for different purposes, should or could ever coincide. Mutual understanding is achieved when interacting semiotic processes feed each other with signs that are immediately appropriated by interpreters, as part of their sense-making or other sign production activities (such as communication, for instance).

By the same token, measuring distances between human interpretations, typically the designer’s and the user’s, requires a reduction of meaning to a fixed state. But choosing end states for measurement is necessarily detrimental to the very nature of human semiosis. Therefore, from a Semiotic Engineering perspective, measures can only refer to idealized users and designers, which defeats the purpose of most usability test measurements in HCI, for instance.

The ultimate consequence of viewing meaning as contingent and evolving, is that every computer system is unique. And because it is unique, there cannot be ready-made design solutions for it. In other words, design is not a generative activity, based on scientific, technical, or even heuristic rules that constitute the foundational knowledge from which applications are derived. Design is essentially an interpretive activity.

This view has been extensively discussed and defended by Schön [37], who proposed that the main activity in design was “reflection in action”. Naming the elements and framing the problems in design – two semiotic processes par excellence,
although Schön never resorts to semiotics in his work – constitute the initial stages of the process. From there, designers proceed much in the same way as researchers do in scientific investigations: looking for alternative solutions, testing them, refining them, and learning from them, for all ulcer iterations of the same cycle of naming, framing, solving, testing, refining, and learning. Thus, general design knowledge does not abide in solutions, but in what Schön calls “an epistemology of practice” – sound knowledge-generating and knowledge-critiquing principles.

Semiotic Engineering, supported by Schön’s perspective on design, provides epistemic tools, for HCI researchers and practitioners. Epistemic tools are not meant to stamp out ready-made solutions (nor even solution types) for known classes of a priori problems. Epistemic tools are meant to help problem solvers name and frame problems, and then find, test, refine and learn from unique solutions for them.

The Semiotic Engineering characterization of HCI as designer-to-user metacommunication, for instance, is an example of an epistemic tool. This concept does not tell designers how to design interfaces for text editors or group decision support systems. But it certainly helps them frame the design problem (by using the metacommunication message content paraphrase, for example), to search for alternative solutions, to analyze them, and compare their efficacy in conveying what they mean to users. At the end of the design cycle, designers have typically gained knowledge, especially reflective (self-referenced) knowledge, which is naturally incorporated to the designer’s ongoing semiotics for all ulcer interpretive processes.

3. DESIGNING METACOMMUNICATION FOR AN E-FORUM

Electronic forums (e-forums) are among the most typical CMC applications. From a Semiotic Engineering perspective, CMC applications present some of the most challenging cases of metacommunication, because designers have to make decisions not only about user-system communication, but also about user-user communication through systems. Their piece of metacommunication to users must then include conversations about their vision of the kinds, the modes, the purposes, the contexts, the possibilities and limitations of conversations that users may have among themselves, using the product of an elaborate design process.

3.1 The Orion e-Forum

Online discussion forums were first created to provide the same functions as physical bulletin boards, in which people post messages that others will be able to read later, when passing by. Since then, some features were added, such as multiple simultaneous discussions, identification of the messages’ authors, restricted access, and the ability to search for some text within the discussions. More importantly, e-forums allow users to keep a history of the discussions, because they can be kept for indefinitely long periods of time.

OriOn is an e-forum designed to support a specific kind of CMC: scientific debate in academic research groups [6]. To alleviate some of the burdens of scholarly debate online, especially the difficulty to manage information overload in parallel conversation tracks, the design team needed to go beyond traditional forum design, whose structure is typically based on threads.

In thread-based forums, each thread is defined by a higher level message, sometimes called a forum’s topic or discussion. All messages in a thread refer to its topic, and are usually shown in the order in which they were posted. Some tools allow only one thread per topic (Figure 2a), whereas other tools allow for a hierarchical structure, in which a new thread of discussion can start from any previously-posted message (Figure 2b) [5].

![Figure 2. Typical structure of thread-based discussions: (a) flat (one thread per topic), and (b) hierarchical (multiple threads per topic).]

To support long and elaborate scientific debates, it was necessary to better characterize (i) interrelated topics, and (ii) how a message relates to the one it replies to.

Instead of having a flat list of discussion topics, making up a discussion–messages structure, OriOn added one additional level: it provided a discussion, which could contain different subjects, and each subject then would be composed by a message thread (discussion–subjects–messages structure illustrated in Figure 3).

![Figure 3. Discussion–subjects–messages structure in OriOn.]

In traditional forums, there is only one kind of relationship between messages: “replies to”. OriOn users needed a tool which provided different kinds of relationships, indicated by different rhetorical markers. For instance, some markers necessary to support argumentative debates are: “strongly agrees with”, “somewhat agrees with”, “somewhat disagrees with”, “strongly disagrees with”, and “is indifferent to”. Other markers include: “elaborates”, “explains”, “asks”, “answers”, “comments on”, “continues”, and so on. The assumption was that, by using these rhetorical markers, the users’ interventions (i.e., the messages they posted) would be more precisely expressed, and the resulting discourse threads would be more easily read and understood.

Figure 4 illustrates a discussion in OriOn. The discussion title is indicated by the number 1. The discussion subjects are indicated by the local links in 2 and the grey cells in 3. The message structure is characterized by the hyphens in 4, which indicates...
the indentation of each message. Finally, the labels in indicate the rhetorical relationships between the messages. It is important to note that the rhetorical markers provided by OriOn make it possible for users to explicitly structure their utterances into discourse. This facilitates the reading and interpretation of the discourse by the other forum participants, especially after long periods of time.

Figure 4. A discussion depicted in OriOn (Discussion contents, originally in Portuguese, translated into English by the authors).

3.2 Engineering metacommunication for OriOn’s help system

The original OriOn e-forum didn’t provide online help to its users. As mentioned before, in semiotic engineering the help system is viewed as a valuable opportunity for the designer to explicitly communicate to users his intentions underlying the application’s design and to explain how he planned for them to interact with the application. Silveira explored the idea of creating this explicit designer-to-user metacommunication and developed a model-based method for the construction of online help [38, 39]. The system built on existing research on minimalist instruction, in which small pieces of content were created to clarify specific users’ doubts. In order to express their doubts more precisely, users could make use of the communicability utterances [9] to access help.

For instance, when in doubt of what a user interface element means, the user may ask “What’s this?” in the context of the element. The designer’s deputy will then attempt to clarify the user’s doubt and, if necessary, the user may continue to question the deputy through additional communicability utterances in the context of the answers provided. Figure 5 illustrates Silveira’s help mechanism.

According to Silveira, the help designer should work with designers and developers from the beginning of the project, to collect information necessary to build explicit metacommunication. Silveira has gathered evidence that this approach to building online help also serves as an epistemic tool. As designers try to communicate the essence of their work more consistently and coherently to users, they gain new insights and knowledge to improve the quality of the designed product itself.
Figure 5. Help responses provided by the designer’s deputy: first to a “What now?” user’s doubt, and later to a “How do I do this?” doubt over the previous response.

As stated previously, however, OriOn didn’t have a help system at first. After a few months of OriOn’s usage, the research team decided to build a help system following Silveira’s architecture. What was not followed, however, was the recommendation that the help system be designed during the application development. In this case, OriOn’s help system started to be designed when OriOn was already in production. Besides, the help system’s designer was not part of OriOn’s development team. Therefore, this help system was built from the interpretation of the designer-to-user metacommunication, through a deep exploration of the user interface and of the system’s behavior. This is why the whole semiotic engineering of this help system turned into a study, a semiotic engineering “case” from which lessons were learned.

3.3 Findings and Lessons Learned

The interpretation of the metacommunication had been explored earlier in [34]. In that study, undergraduate students needed to build a help system for ICQ®, based on Silveira’s approach. That small-scale experiment showed evidence that the process could provide valuable input for an application redesign as a whole.

During the help design for Orion, the issues under investigation were the feasibility, usefulness, costs and benefits of following the approach for building online help systems a posteriori, i.e., after the application has been built. Although it isn’t the ideal situation, it is certainly very common in many software development circumstances.

The first observation is related to the cost of reconstructing all design information that was available during the application development phases. It was necessary to re-explore the application domain, understand its users, needs and concepts. When the help system and the application are built part passu, the cost of looking for this kind of information is greatly reduced.

On the other hand, the study corroborates Salgado and de Souza’s findings in [35], which suggests that even a posteriori the approach leads the designers to reflect about the application, its context of use, its users and the solutions that are offered to them. This reflection may result in questioning some aspects of the application and in finding potential or actual metacommunication problems. This way, help design would, even a posteriori, serve as an epistemic tool for redesign.

The main difference between Silveira’s original method and a posteriori help design, however, is in the nature of the information captured for the help system. Silveira claimed that the help system should be a means through which the designer could explain to users his intentions when developing the application. In this case study, however, the help designer did not have access to those intentions. Therefore, OriOn’s help was based exclusively on the interpretation of the designer-to-user metacommunication as directly encoded in the user interface, through interface sign inspection and observations of users during interaction. Instead of communicating the designer’s intentions underlying the application, the help designer sought to communicate the result of his global interpretation of such design intent, based on evidence provided by interface inspection and user observations.

The main disadvantage found in the a posteriori help design is loss of the application design rationale: why the application was built that way, why a certain icon was chosen, why a certain task was assigned this or that name, and so on. One of the greatest advantages, however, was the possibility of basing the help content on concrete data about the software usage, such as frequent doubts and user errors, for instance.

4. DISCUSSION

Our study about the semiotic engineering of explicit metacommunication for Orion’s help system raises a number of issues for discussing how our theory compares to and can be articulated with OS. This discussion is structured around the following issues:

1. Strategies for integrating systems engineering and interaction design
2. The semiotic engineering of explanatory metacommunication as a reengineering tool
3. Engineering without universals

4.1 Strategies for integrating systems engineering and interaction design

HCI as a discipline has existed for only 25 years or so. Its origins can be traced to systems engineering, where user interfaces were treated as a system module, and users played mainly two fundamental roles: that of providing information for system analysis stages; and that of driving the system for the specified purposes. Mismatches in users and developers perspectives, purposes, values, and even vocabulary, often led to the design of inadequate interfaces. HCI problems, at the time, were typically diagnosed as either a failure in requirements analysis, or more frequently as a sign of the users’ inability to deal with the complexities of computer systems. Both diagnoses were correct, by the way, except that the users’ inability to deal with the complexities of computing were not their fault, but the consequence of a poor understanding of what people can (and wish to) do with computers. The effort to broaden and strengthen this understanding gave rise to a new discipline – human-computer interaction.

Today, most software development methods recognize the importance of addressing the users’ needs and expectations. It is thus natural to try to bring systems engineering and HCI together. The alternatives for this are essentially two: (i) working from one end towards the other; or (ii) finding a higher-level framework where both ends fit as integral parts.

The OS community has produced alternatives of type (i). For example, de Moor [8] has proposed to combine OS and language-action perspective methodologies to improve socio-technical specifications of multi-user systems. Together, these methodologies add fundamental insights about how people relate to each other according to social norms, and express such relations in linguistic form during communication. By bringing such insights into the system engineering effort (through specifications), de Moor is advancing some critical HCI issues, such as the need for supporting the social evolution of communities of users. HCI, however, includes various other issues that are not addressed by either social norms or speech acts.
that perform the social procedures embedded in norms. For instance, the choice of interface signs (in the form of menus, buttons, labels, icons, etc.) is a critical topic in HCI. By examining de Moor’s proposal, one realizes that the design and engineering of signification systems to support the expression of all possible interactions with socio-technical systems must be informed by other theories and approaches, typically HCI theories. The valuable contribution of de Moor is then to adopt strategy (i), and inform software engineering with important HCI concerns.

Another attempt to integrate HCI and systems engineering coming from the OS community is the work of Sjöström and Goldkuhl [40]. The authors resort to Information System Actability Theory [2, 15] to “elaborate on user interfaces of information systems as a means to understand socio-pragmatic and communicative aspects of IS use”. The foundation of their work in Actability Theory can be interpreted as an instance of strategy (ii), especially because they aim to evaluate IS use (just like Agerfalk in [2]). Evaluation of use is typically an HCI activity, and the notion of systems actability spreads over aspects of both HCI and systems engineering. However, Sjöström and Goldkuhl admit to prioritize business model communication over HCI communication. Comparing their work to Semiotic Engineering, they say that:

“de Souza [and colleagues] focus on the communication between designer and user, while we argue that it is more important to focus the business communication going on: Users of the IT system actually communicate with each other, using the artifact as a medium for communication. This is actually pointed out by de Souza [and colleagues], but it is only discussed in relation to specific types of multi-user applications (e.g. groupware). We want to stress this kind of communication since we find this to be the core of an information system. A communicative perspective means that information systems are regarded as systems for technology mediated business communication.”

It thus seems to be the case that even if there are elements in Actability Theory work to advance an integration according to strategy (ii), the values driving the work reported here are committed to IS engineering, and fall back to strategy (i). They do not achieve a balance between IS and HCI values, which will fairly account for relevant phenomena at both ends.

In this paper, we also present a study showing an attempt at strategy (i) integration. The difference between our work and that originating in the OS community is our radical commitment with values and our semiotic ontology of HCI. In other words, we are using the same type of strategy, but working in the opposite direction.

The tradeoff is manifolded, and fair evaluations can only be weighed in view of encompassing frameworks and theories of type (ii) strategies, or in the presence of extensive empirical data. Only they would allow for balanced comparisons. In the lack of such resources, we would nevertheless like to comment on a couple of aspects.

First, although Semiotic Engineering is not an integrated theory of HCI and IS engineering, the metacommunication message from designers-to-users is an integrative message. When designers are led to introduce and explain what they have done, why, what for, for whom, and how it works, they are prompted to tell an integrative story of everything that affects the users’ experience. So, for instance, if there are many voices who speak through the interfaces, many stakeholders like HCI designers, system developers, system owners, marketing directors, government regulators, communities of users, advocacy groups, and so on, the designer’s deputy discourse must have some mechanism of floor control and signal who is speaking to whom, when, how, and for what purposes. Recent empirical research with website users carried out by Light and Wakeman [20] has provided convincing evidence that users sense the presence of a speaker, though in many cases an indefinite speaker whom they refer to as “they”.

Their finding is in line with previous research carried out by Clifford Nass and colleagues [43, 33], who report experiments where they found strong evidence that users, with various levels of computer literacy, including the savviest ones, anthropomorphize systems – they assign psychological and social attributes to systems as they communicate with each other. These studies allow us to postulate that users unify interface discourse, and perceive communication as coming from one source at a time. This should not be confused with communications conveying one’s perspective at a time. Any speaker may communicate others’ perspective in his or her discourse. This is the essence of reported speech, with which we are all familiar.

So, the designer’s deputy, postulated in the Semiotic Engineering ontology of HCI, is the producer of unified discourse reporting on as many different perspectives as are involved in systems development. The main criteria for such discourse production is that it be guided by a radical commitment to the users’ ability and motivation to participate in this kind of communication.

Thus, our work also differs from that of Baranauskas and co-authors’ [4], because for Semiotic Engineering the dialogue between designers and users is not “just one aspect of […] communication” in the whole process of system development – it is a prime aspect. One of the advantages of giving a differentiated status to this sort of communication is to accommodate emerging approaches of CMC, which gained interest with the rise of e-commerce applications such as Fogg’s perspective on computers as persuasive technologies [14]. The whole idea of this approach is to design interaction in such a way that users are persuaded to develop attitudes and exhibit behavior that will fulfill the system producer’s intent. Once again, the system producer must be depicted as a unified communicator, lest users will probably fail to perceive the intent or react accordingly.

In sum, compared to OS research reported above, Semiotic Engineering adopts the same strategy of integration, but takes the opposite direction. And although it doesn’t provide an integrative framework to account for HCI and systems engineering concerns in a balanced form, the metacommunication engineering activity is in itself an integrative task, except that integration is radically committed with making users understand the integrated view of the system, and adopt the technological solutions embedded in it.
4.2 The semiotic engineering of explanatory metacommunication as a reengineering tool

Our study of building Orion’s help system has actually provided extensive information for the system’s redesign. Even in the absence of the original designers' explicit metacommunication message, the intensive interpretation process required to formulate the answers to all the questions embedded in it resulted in an integrated view of the system. The integration revealed a number of faulty front-end and back-end solutions [35]. Front-end (or interface) problems ranged from the choice of terminology to the lack of interface tools to format one’s message (i.e. the lack of a simple HTML editor, for instance). Back-end (or system) problems ranged from a proper database model (which would support better coupling with the online help module, for instance) to the system’s architecture itself (to reflect the perceived design rationale more directly).

Liu, Alderson and Qureshi [22] have shown how a semiotic approach can help the recover legacy systems requirements as parts of a reengineering process. In spite of the critical importance of many such systems in contemporary organizations, their reengineering and/or integration to fit other technological environments cannot always count on appropriate documentation and access to original developers. Their OS approach to recovering the systems requirements is based on the assumption that “business rules, or norms, govern behavior and operations” of organizations. By analyzing the context of operation, the semantic rules of the domain, and user-system interaction patterns, the authors generate “semantic process models which describe the system functions, relating to the meanings of business operations”. At a later stage, a model of dynamic behavior of the system, including screen displays is generated. Based on these models, the systems requirements are derived.

Compared to the kind of reengineering enabled by metacommunication design reported in this paper, Liu and colleagues’ OS approach provides another instance of how OS and Semiotic Engineering work in opposite directions, although moved by the same belief in the importance of integrating HCI and IS engineering perspectives. Just like HCI in this OS approach is apparently condensed in “an interaction model [that] describes human-machine interfaces”, the backend system in our HCI perspective is cast in terms of a domain model, a task model and an application model [38]. The OS commitment with business rules is comparable with our commitment with the users’ semiosis. In the interest of completeness, we should say that combining both approaches should broaden considerably the spectrum of concerns in systems development, and almost certainly improve the quality of design solutions.

4.3 Engineering without universals

In section 2 we have mentioned the Semiotic Engineering commitment to Schön’s perspective on design. This perspective challenges most engineering perspectives on systems, where the availability of universal principles is crucial to building a solid generative base for problem solutions. We have justified our choice on theoretical terms, and begun the discussion of universal meaning postulates in modeling and building interactive systems.

As we turn to OS research, we find that the norms analysis activity carried out in widely used approaches to organizational semiotics makes use of deontic logic to specify norms [41, 21]. According to them, norms involve the following elements: condition, state, actor (or agent), modality (deontic operator), and action. These norms are usually established through domain analysis, but they can also be recovered through the analysis of legacy systems [22].

In their approach, the actors are "acting" entities, but their interpretive and semiotic processes lie outside the model of norms, which ultimately form the basis for the user interface engineering in their approach. By omitting the interpretive and semiotic processes from the specification of norms, OS gets closer to cognitive approaches to HCI design in their considerations of universal principles and concepts, ideal users, and fixed goals [9].

Such OS norms are similar to what Donald Norman called methods, in his essay about the importance of communicating about design, i.e., communicating to users the reasons underlying the system image [29]. Norman acknowledges Semiotic Engineering when he says that "If the designer explains the reasoning behind the model, the user will find the task of uncovering the conceptual model much easier. In other words, what we need to provide to people is reasons, not just methods."

Moreover, the context of application of each norm is subsumed in the state and condition elements, which may not always capture the broader context of situated action [42].

Still in line with Semiotic Engineering, Norman proposes the elaboration of "stories to explain the occurrences we experience". He continues to defend that, in the absence of the story (explanation), misunderstandings and confusion are bound to occur, and users will have the same difficulty every time they encounter the problem situation. This illustrates an important difference between Semiotic Engineering and OS, and illustrates how each approach positions itself with respect to the universality of principles.

OS seems to assume that users have the same awareness of the domain as the system’s analysts and designers, that they are in agreement with the business rules expressed in the norms, and that they will not make mistakes (or question the norms) during interaction. In other words, users will not ask themselves (or the application) why the application was designed that way and they will interact with it as intended by the software developers. This approach may work well with systems that emphasize control and standardization, but may fall short of supporting creative uses of technology.

Conversely, metacommunication is crucial for semiotic engineering. Semiotic engineering assumes that users will benefit from learning more about the application’s underlying rationale and the encoded strategies for problem-solving, and not only how to do things with it. Regarding the levels of affordance described in [9], OS focuses on the tactic and operational levels of affordance, whereas semiotic engineering insists on including strategic affordances as well.

5. CONCLUDING REMARKS

In this paper we have reported a study in the semiotic engineering of the designer-to-user metacommunication message. The discussion above has highlighted similarities and differences
between some selected approaches to OS and our own. It is clear, from the very name of the topics under discussion – Organizational Semiotics and Semiotic Engineering – that both share an extensive set of concepts and perspectives on Information Systems and computer artifacts in general. However, Semiotics is not a unified field. Quite contrarily, it is populated with a myriad of perspectives, approaches, theories, methods, and ontologies. So, we should not be misguided by the fact that there may be – as is actually the case – some fundamental theoretical differences between both the approaches.

The most salient of differences is certainly the profound influence of Peircean Semiotics on Semiotic Engineering. The major consequence of such influence is the perspective of meaning as a process, rather than a static abstraction (which is the case of most semiotic theories following the Saussurean approach, for instance). This influence should not, however, be carried too far. Semiotic Engineering is not commensurate to Peircean Semiotics. First of all, the object of investigation is not the same. Whereas Peircean Semiotics takes as its object signs of all sorts, Semiotic Engineering is restricted to signs pertaining to human-computer interaction alone. Second, although human sign-interpretation in Semiotic Engineering is cast as a process, in the Peircean sense, human communication is mediated by symbol-processing machines. This mediation introduces an algorithmic constraint on semiosis, which has the practical effect of crystallizing semiosis in an arbitrated point of the ongoing semiotic process of designers and users. Thus, designer-to-user communication must go through the filter of computation, which deprives their (human) meanings from a wide range of specification possibilities. So, in the end, there is a fixed point in HCI – the finite, or at least enumerable, set of meanings generated by computers as they mediate communication among people. If this is the most difficult challenge for HCI design, on the one hand, it is also, on the other, the key to systems engineering. All systems compute on an arbitrarily segmented portion of human semiosis. Making users understand such arbitrary behavior (in the sense of arbitrated, as used by de Saussure [9], not fortuitous) and explore the possibilities of playing by the rules embedded in the system is the goal of HCI design in Semiotic Engineering. Hence the shift from universal cognitive primitives to an elaboration of plausible and useful “stories” about rational players, which the designers’ deputies ultimately are.

Computer mediation introduces the logical form representation of the HCI scenario. It does not reveal the logic of human behavior, for human behavior, in a semiotic perspective, is the matrix of ongoing semiosis that actually extrapolates the individual level, and permeates culture and society, over time. It is our belief that a combination of OS business norms and Semiotic Engineering metacommodation may yield important insights about how and why computer systems work as they do, and are used as they are.

ACKNOWLEDGMENTS
Authors would like to thank CNPq, the Brazilian Council for Scientificent and Technological Development, for financial support for this research, from 2003 to 2006.

REFERENCES
International Association for Semiotic Studies, Berkeley, USA, June, 12-18, 1994. (internal rapport)


