

Extreme Designing: Binding Sketching to an Interaction Model in a Streamlined HCI Design Approach

Bruno Santana da Silva, Viviane Cristina Oliveira Aureliano, Simone Diniz Junqueira Barbosa

SERG, Departamento de Informática, PUC-Rio

R. Marquês de São Vicente, 225

Gávea, Rio de Janeiro, RJ, Brasil, 22451-900

+55 21 3527-1500 ext. 4353

{brunosantana, vaureliano, simone}@inf.puc-rio.br

ABSTRACT

This paper presents a streamlined approach to human-computer interaction design called extreme designing. Extreme designing follows on the footsteps of agile methods and is analogous to extreme programming. However, it is not radically committed to “user interface coding” (sketching or prototyping alone), but instead proposes to combine user interface sketches with a more structured representation such as an interaction model. By doing so, it brings together the advantages of sketching and prototyping as a communication tool, and of interaction modeling as a glue that binds together the sketches to allow designers to gain a more comprehensive view of and to reflection on the interactive artifact, thus promoting a more coherent and consistent set of design decisions.

Categories and Subject Descriptors

H.5.2 [Information interfaces and presentation]: User Interfaces – Theory and methods.

General Terms

Design, Documentation, Human Factors.

Keywords

Streamlined approaches to HCI design, Communication-centered design, Interaction design, Sketching, Semiotic engineering

1. INTRODUCTION

Interactive software development processes lie on a continuum between prototype-driven (more agile, streamlined) processes and specification-driven (model-based) processes. From the early days, traditional software development processes have focused on detailed software specifications. But, as business practices become more dynamic and technology evolves ever so more rapidly, in recent years, agile methods have received increasing

attention from the industry and academia alike. Moreover, as software becomes increasingly more interactive and accessible to a wider range of users, human-computer interaction (HCI) concerns have come to play a major role in software development as well, emphasizing the need for user involvement in the development process and intermediate artifacts that promote communication between designers, developers, and users. Scenarios, storyboards and prototypes are increasingly being used in a variety of software development processes, even those who don't follow user-centered design nor consider usability as high priority.

Extreme programming [[6]] is an exemplar of an agile development process that focuses on rapid development and code production, driven mostly by user stories. When seen from an HCI perspective, however, agile methods such as extreme programming lack a coherent vision of the application's emerging behavior, where all user stories should fall into place. Such a vision is essential for building an adequate and coherent user interface, i.e. a user interface that reveals consistent interaction patterns across the supported goals and tasks.

In this paper, we present a streamlined approach to interactive software design that aims to promote the creation of alternative design solutions that are not evaluated in an isolated fashion. Instead, design fragments are put together making up interaction threads that are then evaluated as a coherent whole. The proposed approach combines the agility, power and flexibility of sketching with the structure and a more comprehensive view of the product provided by an interaction model.

This approach is grounded in the semiotic engineering of human-computer interaction [[14]], a theory that explores HCI as communication phenomena, be it among various users, or from designer to user via the user interface.

In the next section, we briefly present the semiotic theory underlying our work, followed by a description of the communication-centered approach to interactive software design. In the fourth section, we briefly present the philosophy of agile methods and how they may fit into HCI-related processes. Next, we present extreme designing, describing how sketches and interaction models may be combined to form a streamlined HCI design approach. We discuss how formative evaluation may take place in this context, and finally conclude with some additional remarks and directions for future work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission from authors.

IHC 2006 – VII Simpósio Sobre Fatores Humanos em Sistemas Computacionais. Novembro, 2006, Natal, Rio Grande do Norte, Brazil. ACM .2006 ISBN 1-59593-432-4/06/0011.

2. THE SEMIOTIC ENGINEERING OF HUMAN-COMPUTER INTERACTION

Semiotic engineering takes on a communicative perspective to HCI, viewing the user interface as a metamessage sent from designers to users. One of the design issues it addresses is the engineering of sign systems that convey what HCI designers and users have in mind and what effect they want to cause in the world of things, practices, ideas and experiences [[14]]. The interface signs constitute a message sent from designers to users, created in such a way as to be capable of exchanging messages with users, i.e., allowing human-system interaction. In semiotic engineering, the high-level message sent from the designer to users can be paraphrased as follows [[14]]:

“Here is my understanding of who you [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 to fulfill a range of purposes that fall within this [my] vision.” (p.84)

In particular, semiotic engineering proposes a change of focus from *producing* to *introducing* design artifacts to users [[13]]. Because semiotic engineering brings to the picture designers themselves as communicators, we need to provide tools to better support them in this communicative process, ultimately via the user interface.

Traditional software development processes assume that the conversations between users and designers occur in two stages: early in the process, in analysis activities, and later when there is a prototype or product, in user testing. Semiotic engineering proposes that this conversation continues during interaction, through the system [[13]]. The theory actually introduces an ontological element called the designer’s deputy, who “speaks” to users on behalf of the designer during their interaction with the system. The theory also proposes a set of tools for designing help systems, extensible applications, and multi-user applications. These tools, however, are of a singular nature: they are not defined mostly as construction tools used directly for generating a software solution. Instead, semiotic engineering puts forth *epistemic tools*, elaborated mainly to help designers reflect about the product at hand [[14]].

The theory is in line with Schön’s view of reflection-in-action, in which the activities of framing and naming a problem are considered essential for first understanding it from different perspectives, before trying to work a solution for it [[31]]. In our work, the elements for reflection are communicative in nature.

3. COMMUNICATION-CENTERED DESIGN

Within semiotic engineering, the communication-centered design approach emerged as an attempt to ensure that domain concepts to be communicated to users are well represented and understood by every team member¹ before proceeding to later design stages

[[3]]. It argues for the need to promote the shared understanding among the team members (for instance, by representing domain concepts and their interrelationships), and to allow designers to represent communication-centered concerns developed for improving designer-to-user communication during interaction [[13], [14]].

In order to address the communication-oriented concerns, Barbosa and co-authors have used scenarios and sketches representing user interface and interaction fragments, combined with an interaction model that provides structure to these fragments, making up a global view of the application’s apparent behavior, i.e., a blueprint of the user-system interaction possibilities [[3]].

The basic assumption of the communication-centered approach to design is that, in order to increase the chances of engineering adequate signs at the user interface to convey the designers’ vision and thus properly introduce the design artifact, this vision must first be established and communicated effectively among team members themselves (Figure 1). In other words, if designers are unable to convey their vision to each other and to every stakeholder, they will hardly succeed in conveying it to users (through the user interface).

If, on the other hand, they succeed in promoting designer-designer communication via communication artifacts, they will be better equipped to communicate with users through the user interface, i.e., to engineer the user interface sign systems.

In this paper, we advocate a streamlined approach to communication-centered design. The designers’ vision is to be elaborated incrementally and in short cycles, supported by sketches and interaction diagrams. In line with other agile approaches, the use of detailed specification models is secondary to the rapidity and brevity of the iterative design and development cycles. Unlike most agile approaches, however, we maintain enough documentation so as to create the designer-to-user metamessage. This way, we take one step towards an agile communication-centered approach to interactive software design and development.

¹ By “team members” we mean the project stakeholders and designers (members of the development team from various

disciplines, such as software engineering, human-computer interaction, graphics design, linguistics, psychology and so on).

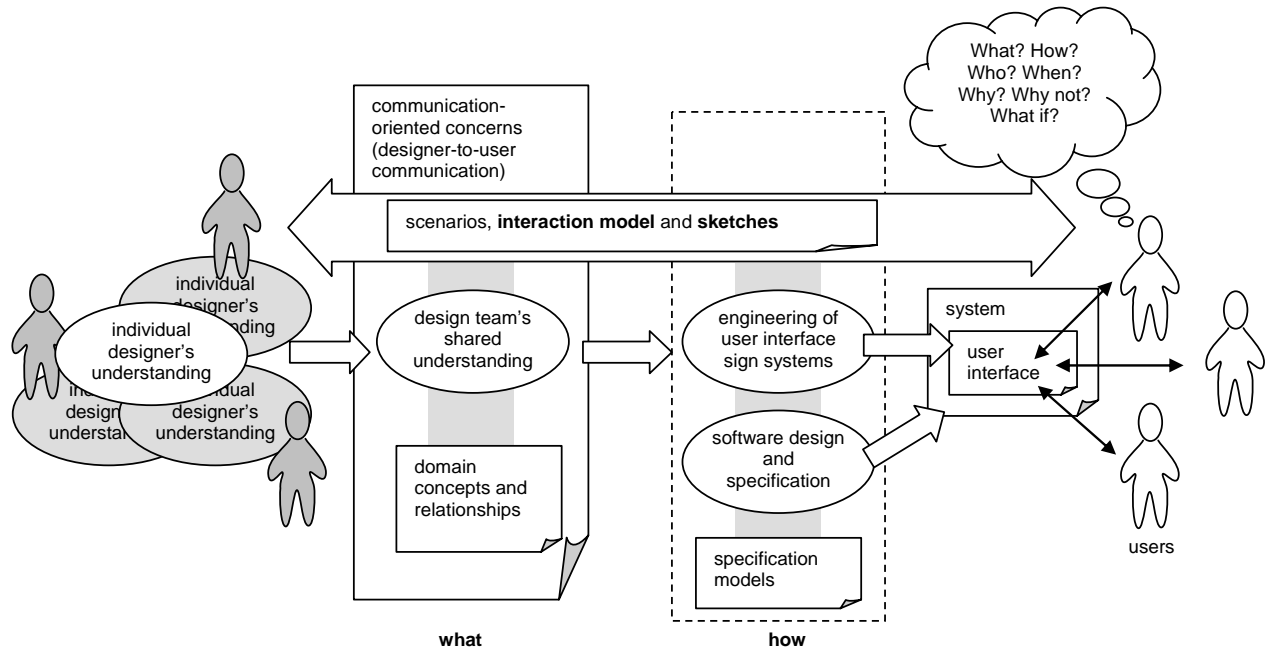


Figure 1: Communication-centered design (adapted from [[3]]).

4. AGILE METHODS

Software development has come a long way from traditional waterfall lifecycles. In the 80's, Boehm and co-authors called attention to the need for more prototyping over specification, when it comes to highly interactive software [[10]]. In the late 90's, several methodologies were proposed that emphasized "close collaboration between the programmer team and business experts; face-to-face communication (as more efficient than written documentation); frequent delivery of new deployable business value; tight, self-organizing teams; and ways to craft the code and the team such that the inevitable requirements churn was not a crisis." [[1]]

The agile movement is best illustrated by the Manifesto for Agile Software Development, which states that [[7]]:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions** over processes and tools
- Working software** over comprehensive documentation
- Customer collaboration** over contract negotiation
- Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

Perhaps the most famous "product" of the agile movement so far is the extreme programming approach [[6]], which comprises four values —communication, simplicity, feedback, and courage— and five basic principles — rapid feedback, assume simplicity, incremental change, embracing change, and quality of work.

Agile methods pay significant attention to users and their needs. They bring users to the development process from the early stages, when the requirements are discovered, and give them an opportunity to speak up and say what they really need and want. Is this "listening to users" in the agile methods enough to address HCI issues?

4.1 Agile Methods and HCI

There have been many attempts to integrate software engineering and human-computer interaction [[32]]. Recently, agile methods have also come to the picture [[9], [12], [18]]. Blomkvist has discussed both agile and user-centered design (UCD) principles, and some ways in which they could be integrated [[9]]. In fact, UCD has always advocated, by definition, a closer contact with the user throughout the development process. Moreover, many HCI processes are typically more iterative and prototype-based than traditional non-HCI software development processes. As such, most HCI approaches are in line with the agile principles that focus on "individuals and interactions" and "customer collaboration" (this one addressed more intensely by participatory design methods).

However, agile methods usually make users responsible for identifying and specifying requirements, like in the writing of user stories in extreme programming. When requirements are related to the user interface, users end up responsible for also designing the user interface. We agree with Blomkvist [9] when he states that:

"Customers/users participate in agile development by writing and prioritizing system features (know as user stories in XP) and specifying accepted tests. Users can express what they need to a certain extent, but on their own, it is difficult for them to actually design a new

system. User stories or use cases, which are often used to specify user needs, fail to capture many aspects of user interaction. (...) By letting the user write their user stories themselves, there is a risk that the developers will transfer the responsibility of the system's usability to the users/customers" (pp. 232–233)

5. EXTREME DESIGNING

In a way analogous to extreme programming, our work proposes a new approach called extreme designing. Extreme designing is not as radical as extreme programming, in the sense that we propose to use a simple representation to relate and integrate different portions of the software represented in sketches into a coherent whole, as well as to incrementally answer design questions that are fundamental to communication-centered design. Detailed specification, however, is not an ultimate goal of extreme designing.

Extreme designing is in line with the Agile Manifesto, but with a major difference: our focus is on design and not development. Therefore, we needed to change the concept of “working software”. We advocate the use of a sketch-based prototype where user-system interaction may be simulated, so that the portions of the prototype may be evaluated with users before coding begins. This is important for highly-interactive systems, in which the users’ response to prototypes may greatly influence (for the better) the resulting artifact [[10]].

In line with extreme programming, we also advocate the use of user stories or short scenarios as a starting point for design. Instead of moving from user story directly to code, however, extreme designing inserts the activities of sketching, incremental lightweight model building, formative evaluation and assessment with users, before moving to code.

Moreover, instead of having user stories and sketches as the only design documentation, extreme designing proposes the use of a streamlined structuring representation that binds together the stories and sketches into a more coherent whole. When building this representation, designers reflect on important design issues posed by semiotic engineering and described in the communication-centered design approach [[3]]. We argue that, without the connections provided by such a representation, the designer-to-user communication may suffer, because conflicting signs and messages may emerge from the fragments of the application. By having a structuring representation that acts like a glue language relating individual user interface sketches or fragments, extreme designing provides a better resource for designers to evaluate their message to users from various perspectives, to create a coherent and consistent message, and to avoid communication breakdowns and conflicts. Figure 2 illustrates the extreme designing approach.

It is important to note that, in extreme designing, the user, task and context analyses are also conducted incrementally, *pari passu* the design efforts. This is made possible by an intensive user involvement and participation throughout the process.

Due to the grounding of communication-centered design in semiotic engineering, we inspected the semiotic engineering design models and decided to use, as a structuring representation, MoLIC (Modeling Language for Interaction as Conversation) diagrams. The need for a structured interaction model like MoLIC in HCI design has been argued elsewhere [[3], [4]]. In this paper, we only claim that the elaboration of such a model must be driven by user stories and design questions incrementally and iteratively, in short cycles, so as to not get in the way of more rapid design.

The next subsections briefly review the role of both sketching and MoLIC in HCI design.

5.1 Sketching User Interfaces

The benefits of sketching in design activities are widely known [[31]]. Far from being quick-and-dirty representations, user interface sketches represent an overall arrangement and organization of the user interface elements and widgets, focusing more on content and structural aspects than on the look-and-feel and visual details of the final product.

Some researchers have gone one step further and provided software tools to support interactive sketching [[20], [21], [22]]. Plimmer and Apperley have conducted a study that shows that interacting with digital sketches adds new characteristics to the user interface design process [[28]]. Digital sketching allows designers to make a larger number of revisions with the produced drawings, once they are easily edited. In contrast with paper sketches, pieces of a digital sketch can be more easily moved around and reused in other sketches via copy-and-paste mechanisms, instead of having to be redrawn entirely. Moreover, in some of the tools, it is also possible to simulate the behavior of the produced sketches by using storyboards.

We claim that sketches alone provide fragmented views of the user interface, focusing on a single task or few tasks at a time. When putting some or all users’ goals together, however, there are many interdependencies among goals that emerge and that may affect the user interface sketches. Moreover, the interaction paths connecting the sketches are not always clear, and it is difficult to assess their “completeness”.

Although no one can guarantee completeness of a representation, we do need to provide means to inspect whether the user interface sketches fully represent the range of goals and interactions that users may have with the system. In extreme designing, we attempt to do this by relating the sketches and user stories to a more structured representation: an interaction model.

5.2 MoLIC: An Interaction Modeling Language

As we have said, in line with Schön’s approach to design [[31]], semiotic engineering focuses on *epistemic* tools which aim to help designers name and frame an interactive problem, elaborate or search for solutions to them, refine, test, and reflect on the unique solution to the given problem, instead of generating or pointing out solutions for types of known problems [[14]]. However, from the conceptual solution to the concrete representation of the solution, many decisions may be made that

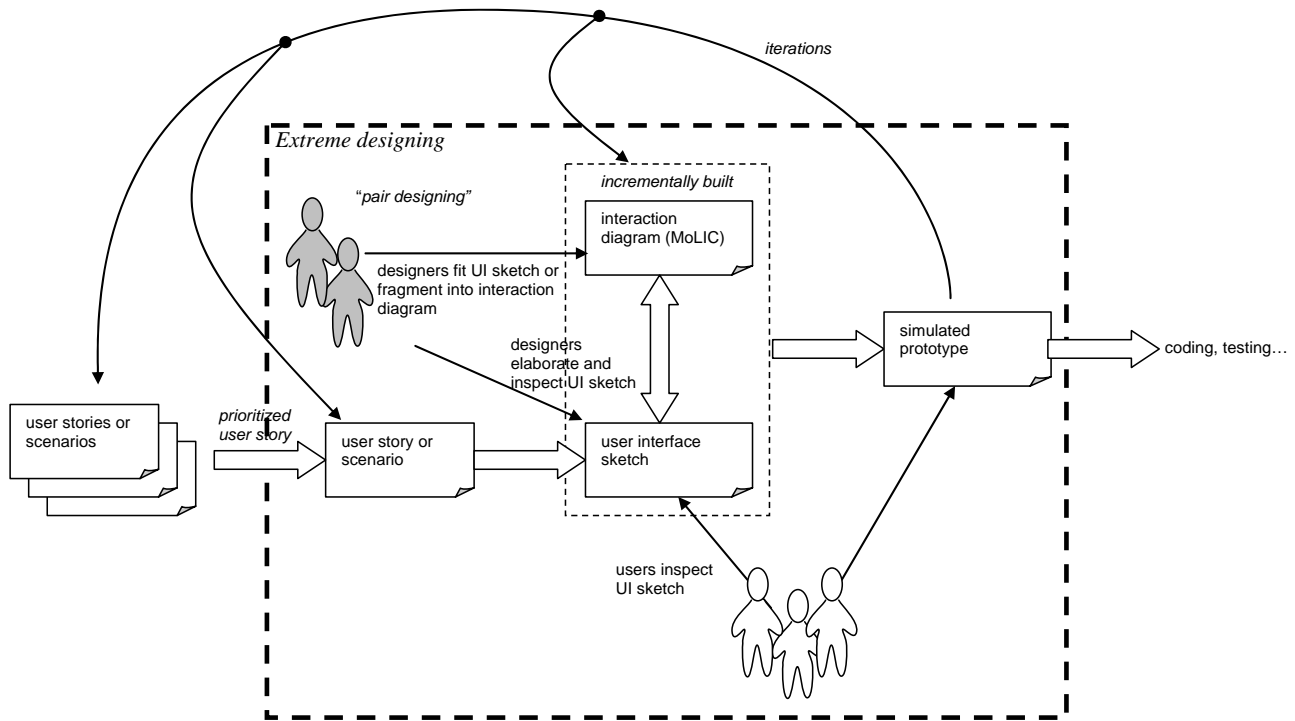


Figure 2: The extreme designing approach.

go unnoticed or unrepresented. Thus, the working solution may not reflect the designers' intentions when conceiving it.

This gap was discussed by Frederick Brooks [11], in what he called, borrowing Aristotle's terms, the *essence-accident* gap in software engineering. More recently, this gap was revisited by Dan Berry in what he called "the inevitable pain of software development" [8], where methods and models adequate for one side of the gap make the traversal to the other side harder and more painful.

In order to help bridge the essence-accident gap, Barbosa, Paula and Silva devised a modeling language that explicitly encourages the reflection on the designers' deputy's discourse and a partial representation of this discourse in an interaction model that follows an "interaction as conversation" metaphor. The representation language for the model is called MoLIC, which stands for "Modeling Language for Interaction as Conversation" [4], [26], [34]. Besides encouraging the designers' reflection on the interactive solution, MoLIC also serves as a concrete resource for the interactive software construction.

When interaction is viewed as conversation, an interaction model should represent the whole range of communicative exchanges that may take place between users and the designer's deputy. In these conversations, designers establish *when* users can "talk about" the signs we extracted from the user stories. The designer should clearly convey to users *when* they can talk about *what*, and what kinds of *response* to expect from the designer's deputy. Although designers attempt to meet users' needs and preferences as learned during user, task and contextual analyses (through

iterative cycles), designing involves trade offs between solution strategies. As a consequence, users should be informed about the compromises that have been made. For instance, MoLIC allows the representation of different ways to achieve a certain result, criteria to choose one from among them, and of what happens when things go wrong. In fact, MoLIC was devised to encourage designers to reflect on the communicative needs of users at interaction time. And these communicative needs become more relevant when "something wrong" occurs [24].

MoLIC supports the view of interaction as conversation by promoting reflection about how the design decisions made at this step will be conveyed to users through the interface, i.e., how the designers' decisions will affect users in their perception of the interface, in building a usage model compatible with the designers', and in performing the desired actions at the interface. This model has a diagrammatic representation used to define all of the potential conversations that may take place between user and system, giving designers an overview of the interactive discourse as a whole.

6. FORMATIVE EVALUATION IN EXTREME DESIGNING

With sketches related to MoLIC diagrams, designers may make various kinds of inspections about the interactive discourse under design, before spending time in developing a running prototype. First, MoLIC allows designers to check whether the minimum cycle of interaction is being preserved: "the application [designer's deputy] 'says' something to the user; the user 'says'

something to the application (that triggers an action); the application [designer's deputy] 'replies' to the user" [[15]].

Because of MoLIC's metaphor of interaction as conversation, we may also resort to linguistic research in order to find tools that prove to be interesting in the interaction design context. In pragmatics, we find Grice's Cooperative Principle (CP) particularly relevant. CP states that [[17]]:

"Make your conversational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged." [[17], p.45]

This principle is supported by a number of conversational conventions, or maxims:

Maxim of Quantity: Make your contribution as informative as is required (for the current purposes of the exchange). Do not make your contribution more informative than is required.

Maxim of Quality: Do not say what you believe to be false. Do not say that for which you lack adequate evidence.

Maxim of Relation: Be relevant.

Maxim of Manner: Be perspicuous. Avoid obscurity of expression. Avoid ambiguity. Be brief (avoid unnecessary prolixity). Be orderly.

In addition, Grice proposes that a maxim such as "Be polite" is also normally observed.

A MoLIC diagram may be inspected to check whether Gricean maxims are being observed in the deputy-user conversation. The maxim of quantity may be checked with respect to the number of utterances in a topic, the number of signs in an utterance, and also the number of outgoing utterances in a scene (number the choices presented to the user for proceeding with the conversation). Superfluous utterances should be relocated or removed from the diagram. The maxim of quality is applied mainly to the deputy's utterances resulting from system processes, i.e., the system feedback must accurately reflect the status and result of the processing. This maxim is especially important to promote the users' perception of privacy and trust.

The maxim of relation may be observed intra-scene, checking whether the utterances in a scene are closely related to the scene topic, and between scenes, to check for abrupt changes in the conversation thread, which may raise doubts or cause users to hesitate.

Observing the maxim of relation is important to provide the answers to the users' doubts in a timely fashion. We can use the results from communicability evaluation research [[27]] to identify where, in the interaction, certain types of users' doubts [[2], [33]] may emerge and thus design appropriate deputy's utterances in response.

The maxim of manner is related to the content of the deputy's utterances, and thus need to be observed when defining the final expression of every piece of content uttered by the deputy, ranging from explicit instructions and messages to field labels and tips.

By following an interaction-as-conversation metaphor, we may also benefit from conversation analysis to assess the quality of the interaction. Issues like turn-taking, adjacency pairs (such as greeting-reciprocation, summons-acknowledgement, request-compliance, assertion-agreement, question-answer, and so on), inserted sequences, sequential placement, reformulations, markedness of utterances, to name a few concepts, may further illuminate interaction design using MoLIC [[19]] and sketches.

Grice's principles can also be applied to sketches: the quantity of signs in each sketch for instance, regards the maxim of quantity. All the necessary goals should be depicted, but superfluous signs should be avoided.

The maxims of quality and manner could be applied to the adequate choice of signs to convey both information and instructions at the user interface. The maxim of relation regards the relations between the set of signs in a sketch and the signs necessary for the user to achieve the corresponding (sub)goal.

Another way to guide a communication-based formative evaluation is to resort to previous investigations on communication problems that users commonly experience when interacting with an application. These problems may be expressed by their frequent doubts and needs for instructions and information, i.e. help content. In the literature about help systems, we find that users would like to receive answers to their most frequent doubts, as summarized in Table 1 [[2],[33]].

Table 1. Taxonomy of users' frequent doubts.

Types of Questions	Sample Questions
Informative	<i>What kinds of things can I do with this program?</i>
Descriptive	<i>What is this? What does this do?</i>
Procedural	<i>How do I do this?</i>
Interpretive	<i>What is happening now? Why did it happen? What does this mean?</i>
Navigational	<i>Where am I? Where have I come from? Where can I go to?</i>
Choice	<i>What can I do now?</i>
Guidance	<i>What should I do now?</i>
History	<i>What have I done?</i>
Motivational	<i>Why should I use this program? How will I benefit from using it?</i>
Investigative	<i>What else should I know? Did I miss anything?</i>

For designers to elaborate the designers' deputy's discourse, and thus elaborate a MoLIC diagram, they need to incrementally form an understanding of the domain and of the effects of their design decisions on the final product (i.e. the user interface), considering the user as an interlocutor in the conversation. By using these potential user questions, we help designers to reflect while they make important design decisions, engaging in reflection-in-action [[31]] about user-system communication and thus enhancing system's accountability [[16]]. In future work, we would want to

encourage the representation of these design decisions, thus building the design rationale of the envisaged application.

From the users' point-of-view, we make use of communicability and help utterances that allow users to better express their doubts during interaction [[30]] (Table 2). By anticipating users' doubts during design, designers will be better equipped to deal with the users' communicative needs, either by designing applications that avoid interaction breakdowns altogether, or by giving users better chances for circumventing them [[36]].

Table 2. Communication-oriented utterances related to users' doubts during interaction breakdowns.

Original Communicability Utterances	(Additional) Help Utterances
What's this?	How do I do this? (Is there another way to do this?)
What now? (What can I do?	
What should I do? Where can I go?)	What is this for? (Why should I do this?)
What happened?	Whom/What does this affect?
Why doesn't it (work)?	On whom/what does this depend?
Oops!	
Where is it?	Who can do this?
Where am I?	Where was I?
I can't do it.	

Silveira and co-authors have described how draft answers to some of these questions may be generated from design models [[34]]. We propose to analyze whether the answers to these questions are clearly conveyed in the sketches, or across sketches in the interaction model. In extreme designing, this is done in the inspections carried out by both designers and users.

7. CONCLUDING REMARKS

In this paper, we have taken an alternative view to both specification-based and prototype-based design processes.

By combining sketches with a global interaction model, a communication-centered approach may promote agility and coherence in design. Agility is promoted by shorter iterations and by elaborating documentation in a "need-to-document" basis. Coherence is promoted by the application blueprint provided by MoLIC. By facilitating the communication among the design team members, the combination of these two representations allows user representatives in the design team to contribute not only with operational and tactical knowledge about the application (i.e., how it works), but also strategic knowledge. This allows the whole design team to acquire a deeper understanding of the application's reason of being and thus clarify the design team's intentions to be conveyed at the user interface. This knowledge may also promote design creativity and exploration of various contexts to improve designer-to-user communication.

The semiotic engineering theory of HCI is an overarching theory that accounts for a range of communicative processes during interaction. Relying on the theory's ontological stance, an interaction model was devised to allow designers to reflect on and build interactive solutions to users' problems, focusing on the

content of the designer-to-user communication. The expression of that communication is provided by the sketches, an efficient representational tool to evaluate alternative ideas and present them to users for discussion.

MoLIC has been used in the past few years in undergraduate and graduate HCI courses, and a number of applications have been designed, both for GUI (a neural network system, a hierarchical plan editor, and a location-based instant messaging application for mobile devices) and for the Web (a bulletin board, a discussion forum, an annotation system, a conference-management system, and a web content publication system).

Our experience using sketches linked to a MoLIC diagram has provided evidence of some important benefits. First, the approach enhances the understanding of the solution from design team members of different disciplinary backgrounds. MoLIC has served as a common representation for HCI designers, psychologists, graphics designers, system analysts and programmers to reflect and discuss about the solution being designed [[5]].

MoLIC has been used to generate usage scenarios to promote communication with users and draft UML diagrams as a first step towards system architecture and specification [[28]]. It has also been used for representing interaction design patterns that are both rich in contextual information (as the original patterns) and useful for representing the design solution (as typical software design patterns) [[27]].

An interaction solution elaborated according to the extreme designing approach may also be evaluated following a paper prototyping method [[36]]. The test scenarios could be directly extracted from MoLIC, as it provides an association between users' goals and interaction paths. Also, by using MoLIC as the tissue that holds together the individual sketches, designers have an indication of when a partial solution is sufficiently defined to be used in user evaluation with paper prototyping, i.e., when a range of interrelated goals is fully defined. This way, the evaluation of certain parts of the application using paper prototyping may be carried out as early as the designers deem that it makes sense to do it, informed by the MoLIC diagrams.

We are currently developing a tool to support extreme designing. One of the motivations for developing the tool may be borrowed from Constantine and Lockwood's statement: "We draw diagrams only when we have to or when drawing them is faster than not drawing them." [[12], p.5]. The tool will let designers incrementally build a MoLIC diagram and associate each element of the diagram to one or more sketches, document alternatives and the criteria for choosing among the alternatives. To enhance communication between designers and users, the tool will let users simulate the behavior of the application by navigating through the sketches in the ways specified in the interaction model. As for future work, we aim to integrate existing design patterns into the tool, by creating a catalog integrating user interface and interaction design patterns, as well as to derive UML diagrams to better assist software engineers in building the application.

8. ACKNOWLEDGMENTS

The authors thank CNPq and CAPES for the financial support to their work. We thank our colleagues and students at the Semiotic

Engineering Research Group for interesting discussions on various HCI issues and topics.

9. REFERENCES

- [1] *Agile Alliance*, online. Available online at: <http://www.agilealliance.org/intro> (last visited on June 2006).
- [2] Baecker, R. M. et al., *Readings in Human-Computer Interaction: toward the year 2000*. San Francisco: Morgan Kaufmann Publishers, Inc. 1995.
- [3] Barbosa, S. D. J.; Silveira, M. S.; Paula, M. G.; Breitman, K. “Supporting a Shared Understanding of Communication-Oriented Concerns in Human-Computer Interaction: a Lexicon-based Approach” In R. Bastide, N. Graham, J. Röth (eds.) *Proceedings of EHCI-DSVIS 2004*, Schloss Tremsbüttel, Hamburg, Germany, 2004.
- [4] Barbosa, S. D. J.; Paula, M. G. “Designing and Evaluating Interaction as Conversation: a Modeling Language based on Semiotic Engineering” In J. Jorge; N. J. Nunes; J. Falcão e Cunha (eds.) *Interactive Systems Design, Specification, and Verification – 10th International Workshop, DSV-IS 2003*, Funchal, Madeira Island, Portugal, Lecture Notes in Computer Science, Vol. 2844, 2003. pp. 16–33.
- [5] Barbosa, S. D. J.; Paula, M. G.; Lucena, C. J. P. “Adopting a Communication-Centered Design Approach to Support Interdisciplinary Design Teams”. *ICSE 2004 Workshop Bridging the Gaps II: Bridging the Gaps Between Software Engineering and Human-Computer Interaction*. Edinburgh, Scotland. 2004.
- [6] Beck, K. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 1999.
- [7] Beck, K. et al. *Agile Manifesto*, 2001. Available online at: <http://agilemanifesto.org/> (last visited on June 2006).
- [8] Berry, D. M., “The Inevitable Pain of Software Development: Why There Is No Silver Bullet”, Monterey Workshop 2002, *Radical Innovations of Software and Systems Engineering in the Future*, Venice, Italy, 2002.
- [9] Blomkvist, S. “Towards a Model for Bridging Agile Development and User-Centered Design”. In A. Seffah, J. Gulliksen, M. C. Desmarais (eds.) *Human-Centered Software Engineering – Integrating Usability in the Development Process*. Springer, 2005. pp. 219-244.
- [10] Boehm, B. W., Gray, T. E., and Seewaldt, T., “Prototyping vs Specifying: A Multi-Project Experiment,” *IEEE Transactions on Software Engineering*, May, 1984, pp. 290-303.
- [11] Brooks, F. P., “No Silver Bullet—Essence and Accident in Software Engineering “, *Proceedings of the IFIP Tenth World Computing Conference*, pp.69–76. 1986.
- [12] Constantine, L. “Process Agility and Software Usability: Toward Lightweight Usage-Centered Design”. In *Information Age*, Aug/Sep 2002.
- [13] de Souza, C. S., “Semiotic Engineering: bringing designers and users together at interaction time”. *Interacting with Computers*, vol. 17, Issue 3, May 2005, pp.317–341.
- [14] de Souza, C. S., *The Semiotic Engineering of Human-Computer Interaction*. Cambridge, MA: The MIT Press. 2005.
- [15] de Souza, C. S., Barbosa, S. D. J., Silva, S. R. P. “Semiotic engineering principles for evaluating end-user programming environments”. *Interacting with Computers*, 13 (4), 2001, pp.467-495.
- [16] Dourish, P., Accounting for System Behavior: Representation, Reflection, and Resourceful Action. In M. Kyng and L. Mathiassen (eds.), *Computers and Design in Context*. Cambridge, MA: The MIT Press, pp. 145-170. 1997.
- [17] Grice, H. P. “Logic and Conversation”. In Cole, P. and Morgan, J. L. (eds.) *Syntax and Semantics*, vol. 3, Speech Acts. New York, NY: Academic Press, 41-58. 1975.
- [18] Gulliksen, J. Göransson, B.; Boivie, I.; Persson, J.; Blomkvist, S.; Cajander, A. “Key Principles for User-Centred Systems Design”. In A. Seffah, J. Gulliksen, M. C. Desmarais (eds.) *Human-Centered Software Engineering – Integrating Usability in the Development Process*. Springer, 2005. pp. 17-36.
- [19] Hutchby, I. and Wooffitt, R., *Conversation Analysis*. Oxford: Blackwell. 1997.
- [20] Landay, J. A. and Myers, B. A. “Interactive Sketching for the Early Stages of User Interface Design”. *Proceedings of CHI 1995*, pp.43-50.
- [21] Landay, J. A.; Myers, B. A. Sketching Storyboards to Illustrate Interface Behaviors. In *Proceedings of CHI 1996*, 1996, pp. 193-194.
- [22] Landay, J. A.; Myers, B. A. “Sketching Interfaces: Toward More Human Interface Design”. In *IEEE Computer*, vol. 4, no. 3, 2001, pp. 56-64.
- [23] Nielsen, J., *Usability Engineering*. Academic Press, 1993.

- [24] Norman, D. *HCD harmful? A Clarification*. jnd.org, Available online at http://www.jnd.org/dn.mss/hcd_harmful_a_clari.html [last access in September 2005]
- [25] Norman, D. and Draper, S. A. (eds.) *User-Centered System Design*. Hillsdale, NJ: Lawrence Erlbaum and Associates, 1986.
- [26] Paula, M. G. *Projeto da Interação Humano-Computador Baseado em Modelos Fundamentados na Engenharia Semiótica: Construção de um Modelo de Interação*. Dissertação de Mestrado. Departamento de Informática, PUC-Rio, Março de 2003.
- [27] Paula, M. G.; Barbosa, S. D. J. “Bringing Interaction Specifications to HCI Design Patterns”. *CHI 2003 Workshop Perspectives on HCI Patterns: Concepts and Tools*. Florida, USA, 2003.
- [28] Paula, M. G.; Barbosa, S. D. J.; Lucena, C. J. P. “Relating Human-Computer Interaction and Software Engineering Concerns: Towards Extending UML Through an Interaction Modeling Language”. *Interact 2003 Workshop Closing the Gaps: Software Engineering and Human-Computer Interaction*. Zürich, Switzerland, 2003.
- [29] Plimmer, B.; Apperley, M. “Evaluating a sketch environment for novice programmers”. *CHI Extended Abstracts 2003*: 1018-1019.
- [30] Prates, R. O., de Souza, C. S., Barbosa, S. D. J. “A Method for Evaluating the Communicability of User Interfaces”. *ACM Interactions*, 31–38, Jan-Feb 2000.
- [31] Schön, D. A. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books. 1983.
- [32] Seffah, A.; Gulliksen, J.; Desmarais, M. C. (eds.) *Human-Centered Software Engineering – Integrating Usability in the Development Process*. Springer, 2005.
- [33] Sellen, A.; Nicol, A., “Building User-Centered On-line Help”. In B. Laurel (ed.) *The Art of Human-Computer Interface Design*. Reading, MA: Addison-Wesley. 1990.
- [34] Silva, B. S. *MoLIC Segunda Edição: revisão de uma linguagem para modelagem da interação humano-computador*. Dissertação de Mestrado. Departamento de Informática, PUC-Rio, Agosto de 2005.
- [35] Silveira, M. S.; Barbosa, S. D. J.; de Souza, C. S. “Model-Based Design of Online Help Systems”. In R. Jacob, Q. Limbourg & J. Vanderdonck (eds.) *Computer-Aided Design of User Interfaces IV*. Kluwer Academic Publishers, 2004, pp. 29–42.
- [36] Snyder, C. *Paper Prototyping*. Morgan Kaufmann, 2003.
- [37] Winograd, T. and Flores, F. *Understanding Computers and Cognition: A New Foundation for Design*, Addison-Wesley, Reading, MA. 1986.