Semiotic Engineering Tools for the Inspection of Meanings Inscribed in Software

Clarisse S. de Souza
SERG / Dep. Informatica
PUC-Rio
R. M. de S. Vicente 225
22451-900 Rio de Janeiro, RJ
clarisse@inf.puc-rio.br

Luiz M. Afonso
SERG / Dep. Informatica
PUC-Rio
R. M. de S. Vicente 225
22451-900 Rio de Janeiro, RJ
lafonso@inf.puc-rio.br

Renato F. G. Cerqueira
IBM Research Brazil
Av. Pasteur, 138-146
22296-900 Rio de Janeiro, RJ
rcerq@br.ibm.com

Rafael R. M. Brandão
Juliana S. J. Ferreira
IBM Research Brazil
Av. Pasteur, 138-146
22296-900 Rio de Janeiro, RJ
{rmello, jjansen}@br.ibm.com

Abstract

In this position paper, we briefly present *SigniFYI Suite*, Semiotic Engineering tools for the investigation of human meanings inscribed in software. From a semiotic perspective, human values are *signified* in software. Our contribution to the debate about human values embedded in software is an overview of this perspective.

Author Keywords

Human meanings inscribed in software; semiotic engineering; SigniFYI.

ACM Classification Keywords

H.5.m. Information interfaces and presentation (e.g., CameraReady

Introduction

All products of design carry not only the imprint of their designers' intent, but also a large collection of human values that are more or less widely shared by individuals, groups, organizations, societies and cultures. With software artifacts, the *carriers* of design intent and human values manifestations are the *meanings* inscribed in them, which can be modified and expanded by those who use or reuse them, as end users, learners or professional developers. Therefore, one of the basic requirements for the study of values embedded in software is the study of meanings inscribed in it.

This position paper presents a bird's eye view of the *SIGNIFYI Suite* of tools for a semiotic investigation of human meanings inscribed in software [7]. *SigniFYI* is founded on Semiotic Engineering [5], which started as a theory of human-computer interaction and has recently been expanded to support the study of semiotic aspects and dimensions of notation-supported conceptual modeling and programming activities in software development.

As a contribution to the debate about human values embedded in software, we propose that the *SigniFYI Suite* can be used in a variety of qualitative, meaning-intensive, research studies. Results can then be used as a theoretically-founded springboard for subsequent research projects, using other theories, methodologies and research designs.

Semiotic Engineering in a nutshell

The study of human-computer interaction (HCI) has been profoundly influenced by cognitive [3,11] and ergonomic [9,13] perspectives. Usability is one of the most important qualities of software products, and it amounts to the software's ability to meet its *users'* needs and expectations within the range of purposes that it has been designed to achieve. In this tradition, the *humans* under study are software systems' *users*, which means that the *values* under consideration in HCI design have been typically those of the users'.

However, since the early ages of HCI, a not-as-popular alternative to cognitive and ergonomic approaches is Computer Semiotics [1]. With different backgrounds and interests, researchers in this group share the view that there are more humans involved in HCI than the users. Interaction is, in their view, a special case of computer-mediated social communication, where software producers and consumers communicate with each other through systems' interface signs [10,8].

Semiotic Engineering [5] is a theory of HCI according to which human-computer interaction is a case of *meta-communication*, that is, the producers' communication (in verbal or non-verbal mode) about how, when, where, why, and to what effects the consumers may or should communicate with a software product. The entire *communication about communication* unfolds gradually as users interact with the system, in the same way as a playwright's *message* to the audience of his or her play unfolds gradually as the audience is exposed to the interactions and dialogs between the characters in the play. The centrality of communication in this perspective defines three classes of HCI 'interlocutors' engaged in social communication: software producers (designers and developers), software consumers (users) and software itself, which represents its producers *vis à vis* the consumers at interaction time [4]. Semiotic Engineering has centered on the study of meta-communication through systems interfaces and has defined its own methods, concepts and models to investigate *communicability*, the counterpart of *usability*. Its *interpretive* methods allow for the study of the emission of the metacommunication message (by producers), as well as for its reception (by consumers) [6]. They center on the investigation of meanings, as expressed by software producers and consumers, during/about interaction supported by interface signs.

The *SigniFYI Suite*

Interface signs are the outermost manifestation of human meanings that contribute to software development. Therefore, in semiotic terms, there is an ontological continuity between outer and inner software signs. The most obvious aspect of it is the relation between the externally perceived and interpreted system *behavior* and the internal program *code* and system *architecture*. This is a causal relation that is affected by other relations with and between additional software development signs, such as those that are present in conceptual models and specifications, requirements and various kinds of design representations, for instance.

The *SigniFYI Suite* is a set of five Semiotic Engineering tools with which we can trace inner and outer human meanings in software [7]. The first one – ***SigniFYIng Message*** – is a conceptual tool to inspect constituent dimensions of metacommunication. It includes the designers' and developers' beliefs about who is involved in metacommunication, what these people know, what they need to do, what they prefer and expect, where they are, in which potential circumstances, and for what purposes they are using the software. It also includes the designers' and developers' general description of what the software is and how it works, as well as their commentary on other possible contexts of use, adaptations and extensions that they think are compatible with their product's design intent and principles.

SigniFYIng Interaction is a Semiotic Engineering inspection method to probe signs inscribed in systems interfaces. It can be used not only to study the end user interface of a software system, but also the user interfaces of various software development tools, such as IDE's, as well as modeling and documentation tools. This is an important feature of *SigniFYI* because software developers, as users, are exposed to interaction blunders whose consequences may affect their end product's interface. ***SigniFYIng Models*** and ***SigniFYIng APIs*** are two additional constituents of the suite that, as their name suggests, allow us to probe signs of human meanings inscribed in software models and programming packages (which is what we generally refer by *API*s). Because notations are critically important in both cases, these constituents also incorporate the use of the *Cognitive Dimensions of Notations* (CDN) framework [2].

Finally, ***SigniFYIng Traces*** is a conceptual blueprint for a capture and access tool that can be used to support the documentation of critically important "signs" for an investigation of human meanings inscribed in software. It can document (portions of) various kinds of software development artifacts, as well as (representations of) the final end product itself. It can also document software execution and use, as well as discussions and analysis held by developers and other experts about the software behavior, design alternatives, usability, communicability and so on. It can additionally be used to document the users' experience as well as their opinions, perceptions and suggestions in user-centered activity during software design and development. Most importantly, however, ***SigniFYIng Traces*** can be used to build a software design and development community's knowledge base and practice repertoire, and support continued reflection on action and reflection on practice [12].

Since the *SigniFYI Suite* is anchored in Semiotic Engineering, it provides an underlying metacommunication-centered ontology that guides all interpretive studies and inspections of meanings inscribed in software. Furthermore, because its metacommunication message structure underlies the entire suite, it provides a thread to *connect* inner and outer software signs of human meanings inscribed in software.

Conclusion

Semiotic Engineering is a well-positioned theory to support *some* aspects of an investigation of values embedded in software. It provides a systematic and cohesive set of tools to investigate meanings in software systems' design, development and use. Although it is limited by its narrow focus on *metacommunication* between software producers and consumers and by methodological commitments with qualitative research, it can support various steps in larger research projects about values and software. Moreover, metacommunication meanings can support further investigations of values with other stakeholders (*e.g.* software companies, internet legislators), other dimensions (*e.g.* political and ethical), as well as other research designs (*e.g.* predictive studies and mixed method investigations). We thus believe that our approach can be an important asset as our community's interest in values embedded in software gains momentum.

Acknowledgements

This work is partially funded by the Brazilian National Council for Scientific and Technological Development.

References

1. Peter B. Andersen 1990. *A theory of computer semiotics*. Cambridge. Cambridge University Press.
2. Alan Blackwell and Thomas Green. 2003. "Notational systems–the cognitive dimensions of notations framework." In J. M. Carroll (Ed.) *HCI Models, Theories, and Frameworks: Toward an Interdisciplinary Science.* Boston, Mass. Morgan Kaufmann.
3. Stuart K Card., Thomas P. Moran, Allen Newell. 1983. *The Psychology of Human Computer Interaction*. Hillsdale, NJ. Lawrence Erlbaum Associates.
4. Clarisse S. de Souza. 2005a. "Semiotic engineering: bringing designers and users together at interaction time." *Interacting with Computers* 17(3) 317-341.
5. Clarisse S. de Souza. 2005b. *The semiotic engineering of human-computer interaction*. Cambridge, Mass. The MIT Press.
6. Clarisse S. de Souza and Carla F. Leitão. 2009. *Semiotic engineering methods for scientific research in HCI*. San Rafael, CA. Morgan & Claypool.
7. Clarisse S. de Souza, Renato F. G. Cerqueira, Luiz M. Afonso, Rafael R. M. Brandão, Juliana S. J. Ferreira. 2016. *Software developers as users*. Chem. Springer International.
8. John Kammersgaard. 1988. "Four different perspectives on human–computer interaction." *International Journal of Man-Machine Studies* 28(4) 343-362.
9. John Long and Andy Whitefield. 1989. *Cognitive ergonomics and human-computer interaction*. Cambridge. Cambridge University Press.
10. Mihai Nadin. 1988. "Interface design and evaluation–semiotic implications." In H. R. Hartson and D. Hix (Eds.) *Advances in human-computer interaction* 2. 45-100.
11. Donald A. Norman and Stephen W. Draper. 1986. *User centered system design*. Hillsdale, NJ. Lawrence Erlbaum Associates.
12. Donald A. Schön. 1983. *Reflective practitioner*. New York, NY. Basic Books.
13. Brian Shackel and Simon J. Richardson. 1991. *Human factors for informatics usability*. Cambridge. Cambridge University Press.