Design of Visual Interactive Systems: a Multi-Facet Methodology

Daniela Fogli[1], Andrea Marcante[2], Piero Mussio[2], Loredana Parasiliti Provenza[2]
[1]Dipartimento di Elettronica per l'Automazione
Università di Brescia, Via Branze 38 - 25123 Brescia, Italy
Te. +39 030-37 15 666 Fax +39 030-38 00 14
[2]Dipartimento di Informatica e Comunicazione, Università degli Studi di Milano,
Via Comelico 39/41 - 20135 Milano, Italy
Tel. +39 02-503 16 290 Fax +39 02-503 16 373

fogli@ing.unibs.it, marcante@dico.unimi.it, mussio@dico.unimi.it, parasiliti@dico.unimi.it

ABSTRACT
The paper presents a multi-facet methodology for interactive software creation which melts holistic, participatory, meta-design approaches to obtain usable, co-evolutive domain specific applications. The proposed methodology is based on the definition of three Interaction Languages, each one permitting to express the conceptual model of a Visual Interactive System in the system of signs of the stakeholder (Software Engineering expert, Human-Computer Interaction expert and domain expert) participating in the design.

Author Keywords
Participatory design, Holistic design, Meta-design, Interaction Language, Visual Language, Communication gap, Co-evolution.

ACM Classification Keywords
H.5.2 Information interfaces and presentation (e.g., HCI): User Interfaces - Theory and methods

INTRODUCTION

In the last years, software engineers are increasingly required to design and develop interactive systems that are understandable, acceptable and usable by end users and that adequately support them in performing their activities in a real context, meeting their expectations, and smoothly augmenting their capabilities. The design of such systems is a complex design problem that overcomes the knowledge of a single discipline and depends on the end user culture, on the context in which the system is used and on the task to be performed.

As a contribution to the emergence of a Science of Design, this paper presents a holistic, participatory, meta-design methodology for the creation of usable, co-evolutive and domain specific interactive applications, which aims at satisfying these needs. The methodology stems from several experiences in the development of domain-specific interactive environments [6,7,19]. It is aimed at overcoming some of the phenomena affecting interactive system development, namely the communication gap among designers and users [2,5,14,17,18], and the co-evolution of systems and users [3,8,21]. The methodology we propose is *holistic* in that it takes into account the whole system constituted by the human user and the software application in use, considered as a deputy of the designer [9]. Moreover, it focuses strongly on the visual appearance of the interface and its behaviour, as it is perceived by the users [22], but it also insists on the existence of multiple interpretations of the visual appearance. The methodology is *participatory* [23] in that representatives of the users (called domain experts) participate in the design of the system collaborating with Software Engineering (SE) and Human-Computer Interaction (HCI) experts. It is a *meta-design* methodology in that design environments are provided to designers of different cultures permitting them to create and shape application environments from their points of view [7,11,12].

According to our methodology, the design process is performed by an interdisciplinary team including different experts – SE, HCI and domain experts – each expert being a stakeholder that evaluates the system and proposes solutions from his/her perspective. However, communication gaps arise among the different stakeholders participating in the design because of their different cultural background, experience and view of the problems at hand [14,17]: SE, HCI and domain experts adopt different approaches to abstraction and follow different reasoning strategies to modeling, performing and documenting the tasks to be carried out in a given application domain; additionally, each of them expresses and describes such tasks adopting his/her own language and jargon.

To overcome these communication gaps, the methodology is *multi-facet* in that it allows every stakeholder involved in the design process to reason on the problems of the system from the perspective of his/her own culture. This is analogous to an observer looking at a light beam with colored glasses which only permit the perception of one spectral component of the light: in this sense, our methodology provides each stakeholder with adequate

"cultural glasses". In other words, as a prism that allows a beam of light to break itself up into its component spectral colors, our methodology breaks the interaction process into different perspectives and allows each stakeholder to observe the whole system according to his/her cultural abilities. In this way, the conceptual model of the interactive system can be expressed in each stakeholder system of signs [10].

Based on this view, we have proposed in [6,7] the Software Shaping Workshop approach to provide each stakeholder with an interactive environment allowing him/her to reason within his/her system of signs and within his/her context of work. These environments are organized in a network so that the different stakeholders can communicate each other their findings. In this way, the approach allows to relate the different perspectives to permit the synthesis of the whole process.

In this paper we focus on the description of the linguistic facets of the interaction process and their relationships, while the details of the development and organization of the software environments can be found in [6,7].

OVERVIEW OF THE METHODOLOGY

The methodology we propose in this position paper consists of the definition of three Interaction Languages – the *Interaction Trace Language*, the *Direct Manipulation Language* and the *Finite State Machine-Interaction Language* - for describing the behaviour of a Visual Interactive System (VIS for short) to each different community of stakeholders – SE, HCI and domain experts. VIS behaviour is defined by describing VIS evolution during the interaction process. Stakeholders from each community describe such evolution according to their own "cultural glasses".

Each Interaction Language defined in the methodology is based on a Visual Language. The term "Interaction Language" denotes here a language whose sentences provide an explicit description of the interaction process between the user and the VIS, whereas by the term "Visual Language" we refer to a language whose sentences are visual entities to be used in communications among humans, in human reasoning and in human-computer interaction [1,20]. These languages are the tools by which each member in the design team belonging to a specific community expresses his/her reasoning on VIS requirements, problems and solutions according to his/her points of view and to his/her cultural and linguistic background.

The proposed methodology also foresees the study of the relationships among the above languages and the definition of adequate translation methods.

The three Interaction Languages rely on a common interaction model describing a VIS in terms of its component entities, referred to as *virtual entities* (ves) [6]. This interaction model evolves and refines the model for WIMP (Window, Icon, Menu, Pointer) interaction proposed in [4]: it models the HCI process as a cyclic process, in which the user and the interactive system communicate by materializing and interpreting a sequence of messages. Users interpret the messages by applying human cognitive criteria, while the system applies criteria embedded in an underlying program P. In WIMP interaction, the messages exchanged between the user and the interactive system are the entire images represented on the computer screen, formed by texts, pictures, icons, etc. Humans look at the screen and interpret the visual message - the image - currently shown by the computer within the context of their activity, by recognizing *characteristic structures*, css, i.e., sets of pixels representing functional or perceptual units for the humans. On the machine side, each cs is the physical manifestation of a virtual entity, which exists because the computer interprets a program P specifying its appearance and behaviour.

The VIS is a virtual entity itself composed by other virtual entities interacting each other and with the user through the I/O devices. The user sees the VIS as a whole ve, whose computational state is materialized at each instant as an image i on the screen. This association between the VIS materialization and its computational meaning is called *visual sentence* (vs) [4] and specifies the state of the whole VIS at each instant.

Users interpret a cs of a component ve of the VIS, such as a selected menu, within the image i on the screen and manifest their intention to the VIS by performing an action on the cs through the input devices available in the computer at hand, i.e., clicking on the mouse button when the mouse pointer is over the cs. The VIS reacts by changing its state, i.e., moving to a new visual sentence which manifests itself to the users as a new image on the screen. More details about virtual entity theory can be found in [4,13].

A MULTI-FACET SPECIFICATION OF THE VIS BEHAVIOR

Let us examine the three Interaction Languages to highlight the linguistic facets of the interaction process with respect to the different stakeholders' culture.

The Interaction Trace Language (ITL) is specified for domain experts: it is the set of the *interaction traces*, defined as sequences of the form: initial image i_0 followed by a finite sequence of pairs <action, new image>. ITL is specified by the pair $<i_0$,VCL>, where i_0 is the initial image associated with the initial state of the VIS and VCL (Visual Command Language) is a finite set of *visual command*s. A visual command expresses through visual and textual expressions familiar to end users how the user has to interact with the VIS to execute a task and how the VIS visually reacts. VCL is a Visual Language based on a Pictorial Language (PL), whose sentences are the images

the user sees on the screen and reasons about during the interaction.

The Direct Manipulation Language (DML), which is specified for HCI experts, describes the same interaction process specified by ITL, by maintaining explicit the visual part of the interaction process, but explicitly using VIS states and direct manipulation user actions. It is the set of the *interaction process instances*, defined as sequences of the form: initial visual sentence vs_0 followed by a finite sequence of pairs <action, new visual sentence>. DML is specified by the pair $<vs_0,TR>$, where vs_0 is the initial state of the VIS and TR is a set of *transformation rules* - the Visual Language of the transformation rules [5,13], based on the same PL introduced above. A transformation rule describes, at a high level of abstraction, the process through which a visual sentence is transformed into another one as a reaction to a user action. In other words, given a transformation rule, its application to a visual sentence vs_1 results in a *transformation* of vs_1 into another visual sentence, let's say vs_2, as the reaction to a user action. With respect to a visual command, which describes the interaction with the VIS to execute a given task, the correspondent transformation rule also includes a computational part specified through an adequate description of the state of the programs generating the ves composing the VIS, which depends on the adopted programming technique. DML shares with ITL the pictorial part, in that both their specifications (based on TR and VCL, respectively) are built on the same PL of the images appearing on the screen.

Finally, the third Interaction Language, the Finite State Machine-Interaction Language (FSM-IL), is the set of the possible *interaction paths* that can be performed on the Finite State Machine (FSM) recognizing the DML sentences, which start from the initial state corresponding to the initial visual sentence of the VIS. Such FSM is specified by SE experts through a further Visual Language, the State-Chart Language (SCL) [15,16]. The next state function and the output function of the FSM are specified through the transformation rules in TR. Being specified on the basis of TR, also FSM-IL sentences are built on the same PL on which ITL and DML are based.

A UNIFIED VIEW OF THE INTERACTION PROCESS

The three Interaction Languages are not independent: the set of such specifications links the user views and jargons to the SE and HCI views and jargons (and vice versa) and bridges the communication gaps arising in the design process, thus allowing the three different stakeholder communities to discuss, test and use the VIS according to their specific cultures. The three languages share the same PL of the pictorial elements appearing on the screen, which constitute the boundary objects on which each stakeholder in the design team reasons and which s/he interprets according to his/her own semantics.

Figure 1 shows the three Interaction Languages we have defined and the relationships among them. A stakeholder uses his/her own language to describe and reason on the VIS behaviour; s/he uses the different procedures to communicate his/her findings to the other stakeholders or to receive their observations.

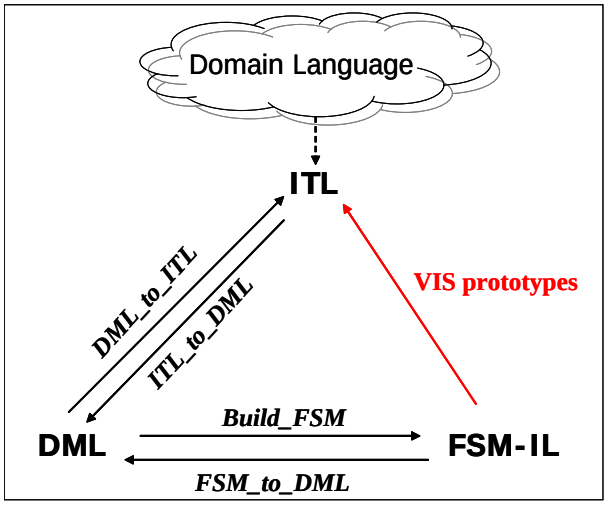

Figure 1. The relationships among the three Interaction Languages

The Interaction Trace Language is derived from the end users domain-specific language. Domain experts use it to describe the system behaviour in accordance with the user perspective. Domain experts communicate their findings to HCI experts feeding their sentences in this language. A procedure *ITL_to_DML* has been defined, which translates these sentences into DML sentences. During this translation activity, ambiguities and non deterministic definition of ITL can be found. Therefore, HCI experts may decide to go back to the ITL definition, and thus to its specification through VCL, and, with the help of domain experts, revise it in order to remove these faults. In this case, they follow procedure *DML_to_ITL*, defined to translate DML sentences into ITL sentences. In this way, domain experts always reason on ITL sentences while HCI experts always reason on DML sentences. HCI experts can additionally communicate with SE experts: algorithm *Build_FSM* has been developed to derive the FSM from DML specification and procedure *FSM_to_DML* has been defined to translate FSM-IL sentences into DML sentences, whenever SE problems emerge, such as incompleteness or faults in VIS definition, which ask for FSM revision.

The use of the three languages permits the creation of the VIS, which represents the unified view of the three perspectives on the holistic process of interaction. This is synthesized in Figure 1 by the arrow labelled "VIS prototypes" meaning that the result of the development activity according to the FSM specification is a VIS supporting users in performing some tasks in a given context.

Note that the procedures mentioned above require a certain amount of human decision-making, therefore they are not rigorous algorithms and cannot be fully automatized. The process is iterative and the developed prototypes are evolutionary [22], in that they are constructed, evaluated and evolved continually throughout the VIS life cycle. In this way, co-evolution is taken into account explicitly: each new evaluation-evolution cycle requires the revision of ITL, DML and FSM-IL. Moreover, it requires to balance the usability requests, the need of satisfying end user requests on the expressiveness of the messages and the technological constraints. More details on the three languages and their relationships can be found in [13].

ACKNOWLEDGMENTS

The present work is partially founded by the the 12-1-5244001-25009 FIRST grant of the University of Milan (Italy).

REFERENCES

1. Bianchi, N., Bottoni, P., Mussio, P., AND Protti, M. Cooperative visual environments for the design of effective visual systems. *JVLC*, 4, (1993) 357–382.
2. Borchers, J., Fincher, S., Griffiths, R., Pemberton, L., and Siemon E. Usability Pattern Language: Creating A Community. *AI & Society Journal of Human-Centred Systems and Machine Intelligence*, 15(4), (2001), 377-385.
3. Bourguin, G., Derycke, A., Tarby, J.C. Beyond the Interface: Co-evolution inside Interactive Systems - A Proposal Founded on Activity Theory, *Proc. IHM-HCI* (2001).
4. Bottoni, P., Costabile, M.F., and Mussio, P. Specification and Dialogue Control of Visual Interaction through Visual Rewriting Systems, *ACM TOPLAS*, 21(6), (1999), 1077 - 1136.
5. Carrara, P., Fogli, D., Fresta, G., and Mussio, P. Toward overcoming culture, skill and situation hurdles in human-computer interaction. *Int. Journal Universal Access in the Information Society*, 1(4), (2002), 288-304.
6. Costabile, M.F., Fogli, D., Mussio, P., and Piccinno, A. End-User Development: the Software Shaping Workshop Approach. In Lieberman, H., Paternò, F., & Wulf, V. (Eds), *End User Development*, Dordrecht, The Netherlands, Springer, (2006), 183-205.
7. Costabile, M. F., Fogli D., Lanzilotti, R., Mussio, P., and Piccinno, A. Supporting Work Practice through End User Development Environments. *Journal of Organizational and End User Computing*, 18(4), (2006), 43-65.
8. Costabile, M. F., Fogli D., Marcante A., Piccinno A. Supporting Interaction and Co-evolution of Users and Systems. *Proc. AVI 2006, Venice, Italy*, 2006, 143-150.
9. De Souza, C. S., Semiotic engineering: bringing designers and users together at interaction time, *Interacting with Computers*, 17 (3), Elsevier, 317-341.
10. De Souza, C. S.,Barbosa, S.D.J., A Semiotic Framing for End-User Development, In Lieberman, H., Paternò, F., & Wulf, V. (Eds), *End User Development*, Dordrecht, The Netherlands, Springer, (2006), 401-426.
11. Fischer, G., Giaccardi, E. Meta-Design: A Framework for the Future of End User Development. In H. Lieberman, F. Paternò, & V. Wulf (Eds.), *End User Development*, The Netherlands, Springer, Dordrecht, 2006, 427-457.
12. Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A.G., Mehandjiev, N. (2004). Meta-Design: a Manifesto For End-User Development, *Communications of the ACM*, 47(9), (September), 33-37.
13. Fogli, D., Marcante, A., Mussio, P., Parasiliti Provenza, L., and Piccinno, A. Multi-facet Design of Interactive Systems through Visual Languages. In *Visual Languages for Interactive Computing: Definitions and Formalization, ed. F. Ferri*, (Idea Group Inc. Publication, in print).
14. Folmer E., van Welie M., and Bosch J. Bridging patterns: An approach to bridge gaps between SE and HCI. *Journal of Information and Software Technology*, 48(2), (2005), 69-89.
15. Harel, D. Statecharts: A Visual Formalism for Complex Systems. *Sci. Computer Prog.*, 8(3), (1987), 23-274.
16. Horrocks, I. *Constructing the User Interface with Statecharts*. Addison-Wesley, 1998.
17. Lauesen, S., *User Interface design - A software engineering perspective*. Addison-Wesley, 2005.
18. Majhew, D.J. *Principles and Guideline in Software User Interface Design*. Prentice Hall, 1992.
19. Mussio, P., Pietrogrande, M., and Protti, M. Simulation of Hepatological Models: a Study in Visual Interactive Exploration of Scientific Problems. *JVLC, 2*, (1991) 75-95.
20. Narayanan Hari, N., and Hubscher, R. Visual Languages Theory: Towards a Human-Computer Interaction Perspective. In Merriot, K., & Meyer, B. (Eds). *Visual Language Theory*, New York, NY: Springer, 1998.
21. Nielsen, J. *Usability Engineering*. San Diego, CA: Academic Press, 1993.
22. Preece, J. *Human-Computer Interaction*. Addison-Wesley, 1994.
23. Schuler and Namioka (eds.). *Participatory Design - Principles and Practices*. Hillsday, N.J: Lawrence Erlbaum Associates, 1993.