Human - Computer Interaction

Theory and Practice (Part I)

Volume 1

Edited by

Julie Jacko • Constantine Stephanidis

Human – Computer Interaction:

Theory and Practice (Part I)

Volume 1

Human Factors and Ergonomics
Gavriel Salvendy, Series Editor

Bullinger, H.-J., and Ziegler, J. (Eds.) : Human–Computer Interaction: Ergonomics and User Interfaces, *Volume 1 of the Proceedings of the 8th International Conference on Human–Computer Interaction*

Bullinger, H.-J., and Ziegler, J. (Eds.) : Human–Computer Interaction: Communication, Cooperation, and Application Design, *Volume 2 of the Proceedings of the 8th International Conference on Human–Computer Interaction*

Hollnagel, E. (Ed.) : *Handbook of Cognitive Task Design*

Meister, D. (Ed.) : *Conceptual Foundations of Human Factors Measurement*

Meister, D., and Enderwick, T. (Eds.) : *Human Factors in System Design, Development, and Testing*

Smith, M. J., Salvendy, F., Harris, D., and Koubeck, R. J. (Eds.) : *Usability Evaluation and Interface Design: Cognitive engineering, Intelligent Agents and Virtual Reality*

Smith, M. J., and Salvendy, G. (Eds.) : *Systems, Social and Internationalization Design of Human–Computer Interaction*

Stephanidis, C. (Ed.) : *Universal Access in HCI: Towards an Information Society for All*

Stephanidis, C. (Ed.) : *User Interfaces for All: Concepts, Methods, and Tools*

Ye, N. (Ed.) : *The Handbook of Data Mining*

Jacko, J., and Stephanidis, C. (Eds.) : *Human-Computer Interaction: Theory and Practice (Part I)*

Stephanidis, C., and Jacko, J. (Eds.) : *Human-Computer Interaction: Theory and Practice (Part II)*

Harris, D., Duffy, V., Smith, M., and Stephanidis, C. (Eds.) : *Human-Centred Computing: Cognitive, Social and Ergonomic Aspects*

Stephanidis, C. (Ed.) : *Universal Access in HCI: Inclusive Design in the Information Society*

Human – Computer Interaction:

Theory and Practice (Part I)

Volume 1 of the Proceedings of HCI International 2003
10th International Conference on Human - Computer Interaction
Symposium on Human Interface (Japan) 2003
5th International Conference on Engineering Psychology
and Cognitive Ergonomics
2nd International Conference on Universal Access
in Human - Computer Interaction
22 – 27 June 2003, Crete, Greece

Edited by

Julie Jacko
Georgia Institute of Technology

Constantine Stephanidis
ICS-FORTH and University of Crete

Routledge
Taylor & Francis Group
LONDON AND NEW YORK

First published 2003 by Lawrence Erlbaum Associates, Inc.

Published 2021 by Routledge
2 Park Square, Milton Park, Abingdon, Oxon OX14 4RN
605 Third Avenue, New York, NY 10017

Routledge is an imprint of the Taylor & Francis Group, an informa business

Human-Computer Interaction : Theory and Practice (Part I) / edited by Julie Jacko and Constantine Stephanidis.

p. cm.

Includes bibliographical references and index.
ISBN 13: 978-0-8058-4930-1 (hbk) (Volume 1)

ISBN 13: 978-0-8058-4931-8 (Volume 2)
ISBN 13: 978-0-8058-4932-5 (Volume 3)
ISBN 13: 978-0-8058-4933-2 (Volume 4)
ISBN 13: 978-0-8058-4934-9 (Set)

Preface

The 10th International Conference on Human-Computer Interaction, HCI International 2003, is held in Crete, Greece, 22-27 June 2003, jointly with the Symposium on Human Interface (Japan) 2003, the 5th International Conference on Engineering Psychology and Cognitive Ergonomics, and the 2nd International Conference on Universal Access in Human-Computer Interaction. A total of 2986 individuals from industry, academia, research institutes, and governmental agencies from 59 countries submitted their work for presentation, and only those submittals that were judged to be of high scientific quality were included in the program. These papers address the latest research and development efforts and highlight the human aspects of design and use of computing systems. The papers accepted for presentation thoroughly cover the entire field of human-computer interaction, including the cognitive, social, ergonomic, and health aspects of work with computers. These papers also address major advances in knowledge and effective use of computers in a variety of diversified application areas, including offices, financial institutions, manufacturing, electronic publishing, construction, health care, disabled and elderly people, etc.

We are most grateful to the following cooperating organizations:

- Chinese Academy of Sciences
- Japan Management Association
- Japan Ergonomics Society
- Human Interface Society (Japan)
- Swedish Interdisciplinary Interest Group for Human Computer Interaction - STIMDI
- Asociación Interacción Persona Ordenador - AIPO (Spain)
- Gesellschaft für Informatik e.V - GI (Germany)
- European Research Consortium for Information and Mathematics - ERCIM

The 258 papers contributing to this volume (Vol. 1) cover the following areas:

- Design Approaches, Methods and Tools
- HCI Methodologies and Perspectives
- Usability
- Design and Evaluation Studies
- Web Design and Usability
- Learning and Edutainment
- Virtual, Mixed and Augmented Environments

The selected papers on other HCI topics are presented in the accompanying three volumes: Volume 2 by C. Stephanidis and J. Jacko, Volume 3 by D. Harris, V. Duffy, M. J. Smith and C. Stephanidis, and Volume 4 by C. Stephanidis.

We wish to thank the Board members, listed below, who so diligently contributed to the overall success of the conference and to the selection of papers constituting the content of the four volumes.

Organizational Board
Niels Ole Bernsen, *Denmark*
Hans-Joerg Bullinger, *Germany*
Masaaki Kurosu, *Japan*
Zhengjie Liu, *China*
Ben Shneiderman, *USA*
Michael J. Smith, *USA*
Jean-Claude Sperandio, *France*
Hiroshi Tamura, *Japan*

Ergonomics and Health Aspects of Work with Computers
Arne Aaras, *Norway*
Pascale Carayon, *USA*
Barbara G. Cohen, *USA*
Marvin J. Dainoff, *USA*
Martin Helander, *Singapore*
Bentzion Karsh, *USA*
Waldemar Karwowski, *USA*
Peter Kern, *Germany*
Danuta Koradecka, *Poland*
Helmut Krueger, *Switzerland*
Holger Luczak, *Germany*
Aura C. Matias, *Philippines*
Takao Ohkubo, *Japan*
Susumu Saito, *Japan*
Steven L. Sauter, *USA*
Dominique L. Scapin, *France*
Naomi Swanson, *USA*
Gunnela Westlander, *Sweden*

Human Interface and the Management of Information
Lajos Balint, *Hungary*
Gunilla Bradley, *Sweden*
Alan H.S. Chan, *Hong Kong*
Helmut Degen, *Germany*
Michitaka Hirose, *Japan*
Yasufumi Kume, *Japan*
Mark R. Lehto, *USA*
Kee Yong Lim, *Singapore*
Fiona Nah, *USA*
Shogo Nishida, *Japan*
Leszek Pacholski, *Poland*
Jennifer J. Preece, *USA*
Robert W. Proctor, *USA*
Francois Sainfort, *USA*
Tjerk W. van der Schaaf, *Netherlands*
Lawrence M. Schleifer, *USA*
Karel Vredenburg, *Canada*
John R. Wilson, *UK*
Sakae Yamamoto, *Japan*
Li Zheng, *China*
Bernhard Zimolong, *Germany*

Human-Computer Interaction
Albert G. Arnold, *Netherlands*
Sebastiano Bagnara, *Italy*
Nigel Bevan, *UK*
Klaus-Peter Faehnrich, *Germany*
Pierre Falzon, *France*
Xiaowen Fang, *USA*
Sheue-Ling Hwang, *ROC*
Judy Kay, *Australia*
Nancy Lightner, *USA*
Kari Lindstroem, *Finland*
Aaron Marcus, *USA*
Masaki Nakagawa, *Japan*
Celestine A. Ntuen, *USA*
Katsuhiko Ogawa, *Japan*
Kjell Ohlsson, *Sweden*
Gary M. Olson, *USA*
Thomas Rist, *Germany*
Michelle M. Robertson, *USA*
Andrew L. Sears, *USA*
Pentti K. Seppala, *Finland*
Kay M. Stanney, *USA*
Tomio Watanabe, *Japan*
Nong Ye, *USA*
Wenli Zhu, *USA*
Juergen Ziegler, *Germany*

Engineering Psychology and Cognitive Ergonomics
Chris Baber, *UK*
Kenneth R. Boff, *USA*
Guy Boy, *France*
Pietro Carlo Cacciabue, *Italy*
Judy Edworthy, *UK*
James Fisher, *South Africa*
Arthur Fisk, *USA*
Curt R. Graeber, *USA*
Erik Hollnagel, *Sweden*
Kenji Itoh, *Japan*
Peter Jorna, *Netherlands*
Kenneth R. Laughery, Sr., *USA*
Nicolas Marmaras, *Greece*
David Morrison, *Australia*
Sundaram Narayanan, *USA*
Reiner Onken, *Germany*

Eduardo Salas, *USA*
Dirk Schaefer, *France*
Neville A. Stanton, *UK*

Universal Access in Human-Computer Interaction
Julio Abascal, *Spain*
Demosthenes Akoumianakis, *Greece*
Elizabeth Andre, *Germany*
David Benyon, *UK*
Noelle Carbonell, *France*
Pier Luigi Emiliani, *Italy*
Michael C. Fairhurst, *UK*
Gerhard Fischer, *USA*
Ephraim Glinert, *USA*
Jon Gunderson, *USA*
Ilias Iakovidis, *EU*
Arthur I. Karshmer, *USA*
Alfred Kobsa, *USA*
Mark Maybury, *USA*
Michael Pieper, *Germany*
Angel R. Puerta, *USA*
Anthony Savidis, *Greece*
Christian Stary, *Austria*
Hirotada Ueda, *Japan*
Jean Vanderdonckt, *Belgium*
Gregg C. Vanderheiden, *USA*
Annika Waern, *Sweden*
Gerhard Weber, *Germany*
Harald Weber, *Germany*
Michael D. Wilson, *UK*
Toshiki Yamaoka, *Japan*

We also wish to thank the following external reviewers:

Chrisoula Alexandraki, *Greece*
Margherita Antona, *Greece*
Ioannis Basdekis, *Greece*
Boris De Ruyter, *Netherlands*
Babak Farschian, *Norway*
Panagiotis Karampelas, *Greece*
Leta Karefilaki, *Greece*
Elizabeth Longmate, *UK*
Fabrizia Mantovani, *Italy*
Panos Markopoulos, *Netherlands*
Yannis Pachoulakis, *Greece*
Zacharias Protogeros, *Greece*
Vassilios Zarikas, *Greece*

This conference could not have been held without the diligent work and outstanding efforts of Stella Vourou, the Registration Chair, Maria Pitsoulaki, the Program Administrator, Maria Papadopoulou, the Conference Administrator, and George Papatzanis, the Student Volunteer Chair. Also, special thanks to Manolis Verigakis, Zacharoula Petoussi, Antonis Natsis, Erasmia Piperaki, Peggy Karaviti and Sifis Klironomos for their help towards the organization of the Conference. Finally recognition and acknowledgement is due to all members of the HCI Laboratory of ICS-FORTH.

Constantine Stephanidis
ICS-FORTH and University of Crete, GREECE

Julie A. Jacko
Georgia Institute of Technology, USA

Don Harris
Cranfield University, UK

Vincent G. Duffy
Mississippi State University, USA

Michael J. Smith
University of Wisconsin-Madison, USA

June 2003

HCI International 2005

The 11th International Conference on Human-Computer Interaction, HCI International 2005, will take place jointly with:

Symposium on Human Interface (Japan) 2005
6th International Conference on Engineering Psychology and Cognitive Ergonomics
3rd International Conference on Universal Access in Human-Computer Interaction
1st International Conference on Virtual Reality
1st International Conference on Usability and Internationalization

The conference will be held in Las Vegas, Nevada, 22-27 July 2005. The conference will cover a broad spectrum of HCI-related themes, including theoretical issues, methods, tools and processes for HCI design, new interface techniques and applications. The conference will offer a pre-conference program with tutorials and workshops, parallel paper sessions, panels, posters and exhibitions. For more information please visit the URL address: http://hcii2005.engr.wisc.edu

General Chair:

Gavriel Salvendy
Purdue University
School of Industrial Engineering
West Lafayette, IN 47907-2023 USA
Telephone: +1 (765)494-5426 Fax: +1 (765) 494-0874
Email: salvendy@ecn.purdue.edu
http://gilbreth.ecn.purdue.edu/~salvendy
and
Department of Industrial Engineering
Tsinghua University, P.R. China

The proceedings will be published by Lawrence Erlbaum and Associates.

Table of Contents

Section 1. Design Approaches, Methods and Tools

Section 2. HCI Methodologies and Perspectives

Section 3. Usability

Section 5. Web Design and Usability

Section 6. Learning and Edutainment

Section 7. Virtual, Mixed and Augmented Environments

Section 1

Design Aproaches, Methods and Tools

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Use Case Maps: A Visual Notation for Scenario-Based User Requirements

A. Alsumait, A. Seffah, and T. Radhakrishnan
Human-Centered Software Engineering Group
Computer Science Department, Concordia University,
Maisonneuve Blvd. W. Quebec, Canada
{alsumait, seffah, khrishnan}@cs.concordia.ca

Abstract

Among their popular and potential applications, scenarios and use cases provide an integrative picture of the user tasks with its context (users and work). Our goal is to build a complete and consistent user interface and usability requirement model that is simple, intuitive, unambiguous and verifiable by extending the prospective standard the Use Case Maps (UCMs). In particular, we provide step-by-step guidance for developing UCMs from scenarios.

1 Introduction and Background

During the last two decades, several HCI research efforts tried to develop appropriate representations to support user interface and usability requirements. Among the various representations, scenarios and use cases were proposed as a working design representation of user experiences with, and reactions to, system functionality in the context of pursuing a task. Other efforts have tried to combine use cases and task analysis as vehicle for bridging the current gaps between user interface/usability and functional requirements. For example, Forbrig (1999) introduced a framework for combining task models, user models, and object models. These multiple dimensions require tools enabling the manipulation of all these models along the different phases of software development (Sutcliffe, 1998; Weidenhaupt, 1998). Related to this work, Constantine and Lockwood (1999) proposed a use case model description that does not contain any implicit user interface decisions; essential use cases. Later on, user interface designers can use these essential use cases as input to create the user interface without being bound by any implicit decisions. Our research aims to complement the scenario-based requirement approach using an extended use case maps (UCMs) notation. We describe the process of building two UCM models that provide an easy way to understand and validate the user interface and usability requirements.

2 The User Interface and Usability Requirement (UIUR) Model

The proposed UIUR model makes use of the prospective standard Use Case Maps (UCMs) to bridge the gap the between users' statements of requirements in an informal natural languages and the target of formal specifications. UCMs are intended to be a useful tool for functional requirement of scenario-based software development. However, their support for designing user interfaces is still acknowledged to be insufficient (Alsumait et all., 2002). Our strong belief that UCMs are powerful for user interface and usability requirements is based on the simplicity of UCMs

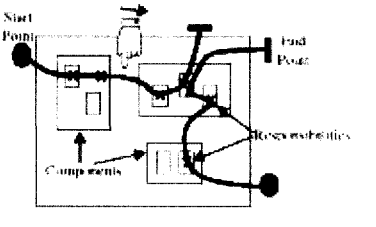

Figure 1: example of a Use Case Map (Buhr, 1998)

notation (Buhr, 1998). The basic UCM notation is very simple (See Figure 1), and consists of *start-points* (filled circles, representing pre-conditions), *responsibilities* (crosses, representing tasks to be performed), *end- points* (bars, representing post-conditions). The responsibilities can be bound to *components*, which are the entities or objects composing the system. The wiggly lines are *paths* that connect start points, responsibilities, and end points. UCMs notation is easily understandable by both the user and the user interface designer. In fact, this helps the designers to handle different users' understanding and expectation of the interface, and bridge the gap by refining the requirements earlier. Moreover, several studies are focusing on how to integrate UCMs to UML and how to formalize UCMs in XML (Buhr, 1998). Finally, we will show that by incorporating the extended UCMs to create our UIUR, we can express user interface and usability requirements much better.

2.1 UIUR Process and Notation

The proposed UIUR model divides the process of transforming requirements stated described in an informal language to formal specifications through a number of iterative phases. Each phase aims at increasing the formality of representation by a small step. The use of UIUR model includes four typical phases: (1) scenario analysis, (2) conceptual use case maps, (3) physical use case maps, and (4) formal specification.

2.2 Scenario Analysis

At this phase, only very limited information is available. A process model of scenario analysis consists of the following iterative and interactive activities: identification of scenarios, elicitation of information and description of scenarios as well as building use cases.

2.3 Build Use Case Maps

By extending the UCM notation, we find that it can be used in the following four dimensions:

- Task Dimension: *represents tasks that are relevant for interactions. Thus, UCMs are used as a simple and expressive visual notation that allows describing task scenarios at an abstract level in terms of sequences of responsibilities and tasks over a set of components.*
- Dialog Dimension: *explains the style of human-computer interaction and also describes the sequence of dialogs that can take place between the user and the system. New notations are introduced to provide UCMs with more expressive power for interaction design (Alsumait et. al, 2002).*
- Structural Dimension: *identifies the objects comprising the user interface, their grouping and specifies their layout, e.g. by indicating approximate placement or by indicating topological relations between groups.*
- Usability Dimension: *Measure the usability of the use case maps by using usability metrics. Different metrics such as task simplicity and task performance can be used to measure the usability of the use case map early in the requirement phase.*

In the second phase, two types of UCMs are created: the conceptual use case map *(CUCM) and the* physical use case map *(PUCM). The CUCM and PUCM together integrate the four dimensions stated above and capture a complete picture of user interface and usability requirements. This helps in analyzing the consistency between requirements of different use cases, and discovers if any conflicts between different types of users, different purposes of use and different operating conditions.*

Conceptual Use Case Map (CUCM)

The CUCM helps provide details on what information will be needed from the user and the system to accomplish a task. Steps to create CUCM are:

- Partition the use cases: *Consider only the use cases with human actors.*
- Create Task Model: *Decompose use cases into tasks and subtasks.*
- Define components of the use case: *Tasks can be bounded to components. This will help in determining the entities and objects that compose the system*
- Create Dialog Model: *Tasks represent decision points in the use case. At any decision point, a dialog will take place between the system and the user.*
- Analysis of the consistency: *Two properties of consistency are examined. The consistency among a set of use cases where the requirements captured by a set of use cases are not in conflict with each other. The consistency of a use case with respect to a requirements model where the information contained in the use case does not conflict with the requirements model and the information contained in the use case is a subset of the information contained in the model. Both properties can be validated in our proposed use case model. If the use cases are inconsistent, it will backtrack to the information elicitation and scenario description (phase 1) to resolve the conflicts*
- Analysis of completeness. *Completeness is the property where all the information contained in a requirement model is covered by at least one use case and does not contain any information that cannot be inferred from the information contained in the use cases. If incompleteness is discovered, it will backtrack to the scenario identification phase to identify additional scenarios that will cover the missing situation.*

Physical Use Case Map (PUCM)

The extended UCMs not only describes the sequence of tasks and dialogs that can take place between the user and the system but also help to understand and reason about the requirements of the user interface, including usability aspects. The PUCM represents the space within the user interface of a system where the user interacts with all the functions, containers, and information needed for carrying out some particular task or set of interrelated tasks. Moreover, successive display of different screens and interactive objects are presented. The PUCM can greatly benefit from the graphical representation of use cases. Steps to create PUCM:

- *Identify the objects that make of the user interface.*
- *Convert the use case maps into an active prototype using a graphical user interface development tool.*
- *Measure the usability of the screens*

Further information on validating CUCM and PUCM can be found in (Seffah and J.A Poll, 2003). Once the CUCM and PUCM are constructed and validated, further requirement analysis and specification activities can start to produce formal requirement specifications according to the requirement models. More research and investigations is required in this area.

3 A Case Study- Movie Recommender Software (MRS)

This section illustrates the description and analysis of the UIUR model using a case study that simulates Movie Recommender Software (MRS) for a PDA. The software should provide recommendations for movies based on preferences of other users.

The movie Recommender software allows the user to carry out the following scenarios: (1) View a list of the latest movies by selecting (Any Genre, Action, Comedy, Drama), (2) Search for movies of (any genre, For Action, For comedy, For drama), (3) View or add a movie to the (To See) list: a list of movies to be seen in the future, (4) Rate a movie. On choosing the options 1 or 2, the user can get detailed information of a movie generated by the Recommender system

As shown in Figure 2, the MRS system can be decomposed into five main tasks, (1) view latest movies, (2) search for movie recommendations, (3) create user to see list, (4) rate a movie, and (5) view movie detailed information. Different scenarios can be drawn, and details are delayed to sub-UCMs. Figure 3 depicts the second level of the requirement model when a user rates a movie. A *dialog* notation illustrates that a conversation between the user and the system is taking place. The CUCM should be validated against the scenarios by analyzing whether the use cases are consistent with each other. Examples of such validation are shown in (Seffah and J.A Pool, 2003).

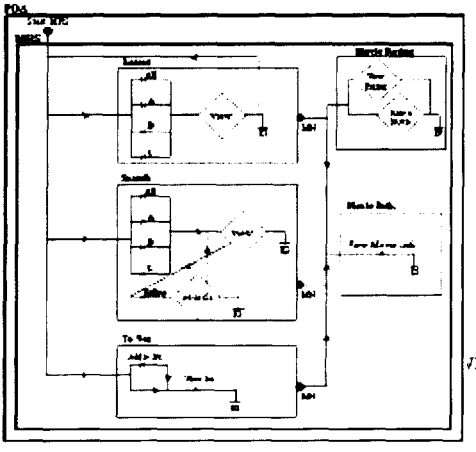

Figure 2: **CUCM for the movie Recommender**

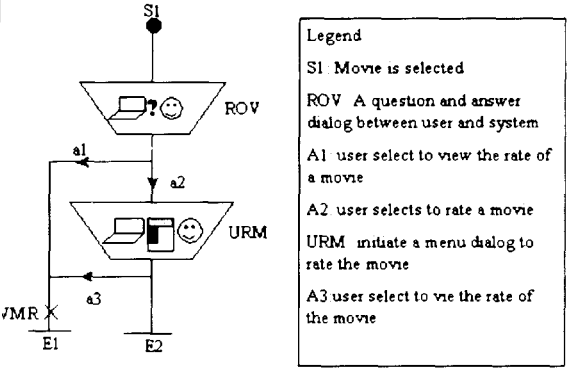

Figure 3: **UCM for movie rating**

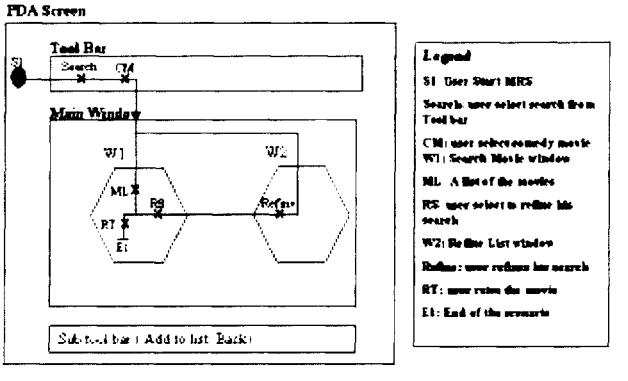

Figure 4: **PUCM for movie Recommender software**

The PUCM in Figure 4 describes the scenario where a user searches for a certain comedy movie and rates the movie. The user interface consists of a tool bar, Main Menu Window, and a sub-

toolbar. Next, the PUCM can be converted into a prototype. Both the PUCM and this prototype can be used by usability expert to predict the usability of the interface.

4 Conclusions

Our research effort leads to the development and analysis of a complete model for user interface and usability requirements (UIUR). This model ensure that: (1) a consistent and complete requirement specification can be captured using scenarios, (2) the specification is a valid reflection of user requirements, (3) the derivation of early design artifacts such as low fidelity prototypes is possible. The new model extends the UCMs to accommodate four dimensions of requirements including the task, the dialog, the structure, and the usability of the user interface. New notations are introduced to provide UCMs with more expressive power for interaction design. As a consequence, the extended UCM helps not only describe the sequence of tasks and dialogs that can take place between the user and the system but also assists to understand and reason about the requirements of the user interface, including usability aspects. Future work includes more investigation on formalizing the requirements according to the UIUR model.

References

Alsumait, A.; Seffah, A.; Radhakrishnan, T. (2002), Use Case Maps: A Roadmap for Usability and Software Integrated Specification, In: *Proceedings of IFIP World Computer Congress 2002*, August 25-30, Montreal, Canada.

Buhr, R. (1998), Use case maps as architectural entities for complex systems, *IEEE Transactions on Software Engineering*, 24(12), 1131-1155

Constantine, L.; Lockwood, A. (1999), Software for use: a practical guide to the models and methods of usage-centered design. Reading, Massachusetts: Addison-Wesley.

Forbrig, P. (1999), Task and object-oriented development of interactive systems - how many models are necessary? In: *Proceedings of Design Specification and Verification of Interactive Systems Workshop-DSVIS'99*, Braga, Portugal, 225-237.

Jarke, M. (1999), Scenarios for modeling. *Communications of the ACM*, 42(1), 47-48.

Seffah A., J.A, Pool Combining UCMs and Formal Methods for Representing and Checking the Validity of Scenarios as User Requirements, to appear in interact 2003. Human-Centered Software Engineering Technical Paper 01-0203, Concordia University

Sutcliffe A.G., Maiden N.A., Minocha S. & Manual D. (1998), Supporting scenario-based requirements engineering. *IEEE Transactions on Software Engineering,* 24(12), 1072–1088.

Weidenhaupt K., Pohl K., Jarke M. & Haumer P. (1998), Scenario usage in system development: a report on current practice, *in Proceedings of the International Requirements Engineering Conference (ICRE '98).*

A Socio-centric Model of User Interactions

David Ambaye

Interaction Design Group (IDC),
Middlesex University, London, UK
d.ambaye@mdx.ac.uk

Abstract

The investigation described in this paper is motivated by the paucity of socio-centric models of user behaviour in relation to complex interactive systems such as collaborative technologies. It argues that existing approaches to modelling users, such as those underlying task based techniques, do not take sufficient account of the psychological and social issues surrounding the usability of such systems. This fundamental weakness means that user-interface design and evaluation are currently less holistic and effective than they ought to be. Two key premises are thus presented. The first is that such holistic models form a proper basis for creating more effective evaluation tools. The second is that the construction of such models needs to be rooted in empirically derived understanding of interactions which occur between users and collaborative systems. The paper reports on results from an empirical study of users interacting with collaborative technologies known as groupware, where social and psychological issues are often as important as other factors. It describes the research framework underlying a multi-method ethnographic case study and summarises the observations.

1 Introduction

Few will dispute that collaborative software implementations such as groupware commonly fail to fulfill both end-user and organisational expectations. Where failures have occurred, it is not unusual to find that commentators and users alike point to root causes that have multifarious socio-technical and psychological dimensions. This is not surprising in light of the fact that collaboration or teamwork is a social phenomenon, and that collaborative technologies are aimed at enriching activities that occur within such a context. This paper argues that such difficulties are largely the result of a paucity of understanding of how users are really likely to behave when interacting with such systems. Poor understanding makes it difficult to create effective collaborative systems, in terms of users' and organisational needs. The result is that organisations resist such technologies because they feel them to be constraining, instead of augmenting and supporting their 'natural' communication and co-ordination processes. Moreover, systems which appear to have few usability problems during design and evaluation, can prove awkward to use and learn when placed in real work contexts (Pinelle & Gutwin, 2000). The importance of getting to grips with this issue cannot be underestimated. The drivers within the business world for the development and utilisation of collaborative technologies are unlikely to lessen with time. Organisations are likely to continue to restructure using a variety of team constructs, and globalisation is unlikely to decrease. Consequently, organisations will come to rely on such technologies as critical technological levers for augmenting the creative and productive potentials of teams and individuals. Such technologies will also be useful in the transitional role, enabling organisations to cope with the downsides of rapid changes, which are increasingly becoming the norm. Sudden change in structure or environment can have a chaotic effect on existing intra-team co-ordination and communication strategies. Inspite of early hopes, the simpler types of

groupware such as e-mail, video-conferencing and shared diaries appear to have gained significant foothold within the workplace today. In sharp contrast, technologies which have a more rigourous dependence on principles of team collaboration, such as work-flow systems, group decision support systems, shared document editing and application sharing systems appear to be much less successful.

2 Evaluation Implications

There are two main conclusions to be drawn from the above discussion. The first is the pressing need to develop new approaches and techniques which are better suited to assessing the usability of technologies oriented towards teams. The second is that such assessment approaches and techniques must be firmly rooted on improved understanding and identification of the critical social, psychological factors and mechanisms that can contribute to the success or failure of such implementations. Success in this respect will mean that designers and usability experts can feel confident that they are designing *team-aware* technologies which fulfill functional, psychological and social needs of teams and individuals in organisations (Baker, 2002). The next section describes results from an investigation which attempts to assist the aforementioned objectives. In particular, it describes the 3 Phase model which characterises observed interactions between team-base users and a variety of collaborative technologies.

2.1 The research framework

Users in three organisations were observed over a period of four years. An ethnographically derived analytical framework known as 3D was utilised to structure the observations. In brief, 3D attempts to relate system usability to individual and team effectiveness (Ambaye 2002). The framework postulates the existence of three usability components: Individual-empowerment, Productive-empowerment and Cultural-empowerment. Taken together, these represent psychological, productivity and social impacts of technology intervention on the effectiveness of team-based users. This made it possible to categorise observed usability issues into one or more of these components. The insight gained from applying the 3D framework has led to the identification of three key emergent properties about how users may interact with collaborative technologies. That is, the bulk of interactions observed in this investigation exhibited three key characteristics:

- Team attributes were transformed in different ways through the intervention.
- The interaction between groupware and users occurred in temporally differentiated phases which were directional, inter-dependent and cyclic.
- Perceived net-benefit was the main motivation that sustained and encouraged user commitment towards a system.

2.1.1 Property 1: Team Attributes are Transformed

When collaborative technology is introduced into a team, team members and the team as a collective may begin to undergo a process that changes many existing attributes, as well as causing the development of new ones. For instance, it may necessitate team members learning new skills, re-negotiating existing working relationships, adopting new norms, communicating differently and generally experiencing changes to the nature of their work activities.

2.1.2 Property 2: The Existence of Temporally Differentiated Phases

The quality of interactions between the technology and users appears to be dependent on, not only the impact a system has on users, but also when it has this impact. For instance, the response time

of a video-conference system under extreme conditions (such as large number of users, or intensive use) is unlikely to be of paramount importance to first-time users. Rather, high in their minds will be concern about issues such as: learn-ability, ease of use, video and audio quality, as well as the consequences the system will have on the way they communicate with each other. Performance under high load conditions, as influential as it can be, will more likely become important at a later time when:

a) users are in such a position as to exploit and appreciate the full potential of its capabilities
b) there is sufficient load on the system.

However, this is not meant to imply that such factors cannot be important in more than one 'phase' of an interaction. Rather, it is simply a matter of priorities. Issues such as reliability, underlying task-model or flexibility are important throughout the lifetime of a system, but may have their most profound impact at particular times and less so in others. This suggests that such interactions may progress through various inter-dependent, but temporally differentiated, phases. More specifically, it is suggested that there may be an initial phase, where users learn and explore the system's characteristics. In subsequent phases, users will evaluate the extent to which they can rely on the system for augmenting or performing mission-critical tasks and will begin to exploit it according to their conclusions.

2.1.3 Property 3: Perceived Net-Benefit is a Key Driver of Entrainment

The notion of **entrainment** is introduced here to describe the process of inclusion of and reliance on, a system for accomplishing routine work. The above examples suggest that the net benefit a system provides has a great influence on the extent of its entrainment. That is, it greatly influences users' level of desire to use a system and whether they will continue to rely upon it for performing routine work. So in the case of a shared diary system, initial usage difficulties (perhaps because of poor training, policy deficiencies etc.) might prevent a majority of users from exploring and using the system. In turn, this may lead to a situation where the system cannot gain the needed critical mass of users (Greenberg, 1991) and by implication critical mass of information. Not surprisingly, the remaining users find that a shared diary system that is used by only a small proportion of users (and infrequently), is unlikely to contain information of value to them. Eventually, they too become non-users.

3 Three Phases of Entrainment (3P)

Taken together the three emergent properties described above form the basis of the 3P interaction model proposed in this paper. 3P conceptualises user interactions as progressing through three key phases of development (property 2 above) that are temporally differentiated, directional and have cyclic characteristics. These are described below.

From system developers' perspective, and in an ideal case, the level of entrainment of a system and the resulting user experience in Phase 3, will generally be greater than for the other two. Conversely, the Orientation phase will reflect a stage when systems have the lowest level of entrainment. Each of these phases is described below. It should be noted that the term 'user' is utilised here to mean primarily team-based users.

3.1 An Orientation Phase

Johansen (1988) observed long ago that a major obstacle to the adoption of groupware is the lack of understanding of its purpose, or how it can be exploited. The Orientation phase in many ways conforms to this observation. This is because it is an impressions phase, very much characterised by whatever initial concerns, questions or opinions team members may have regarding a system.

In this phase, it is not uncommon to find that team member's differing roles, backgrounds and experiences, lead users to react differently to a given system. As a result the issues of concern may also differ between users. If a system is being introduced for the first time, then the Orientation phase reflects first impressions. This may have been gained by observing the way that others have used the system before, information they have read or heard about the system (such as product reviews in trade journals) or through an organisational orientation program (i.e. training). These issues may have even come to mind by simple analogy to previous similar systems users may have been exposed to. If the system has been in use for some time, this phase largely reflects impressions based on experiential evidence that users may have gained from previous contact with the system, in the same or similar roles.

3.2 An Exploration-Evaluation Phase

3.2.1 Phase 2.1: Exploration

One element of Phase 2, Exploration, is intrinsically an experiential stage. It is a stage, where team members actively investigate in greater detail whether the general impressions and concerns which were gained in the Orientation phase are well grounded. If users do not perceive any benefit they are unlikely to enter the Exploration phase. Through this exploration, users may attempt to determine whether there are any benefits or consequences which have been overlooked. Many of the issues that users will be eager to investigate further, will be those that were identified as being important during Orientation. New issues may also emerge that were not considered previously. The means for attempting to gain this understanding involve, exploratory use, a training program, or even careful observation of how others are currently implementing the system. In this phase, users also consider how they could adapt a system's attributes to suit their own work process. They question what would be involved in terms of maintenance and ease of use of the system.

3.2.2 Phase 2.2: Evaluation

A process of evaluation is also important in the second phase. Here, team members may make decisions about what level of entrainment is warranted. This will be mediated in relation to the experiential and other evidence collected about the system. In this sense, users weigh up the costs and benefits of entraining the system (if at all), after having appraised how the variety of system attributes will impact their work. For instance, two important considerations could be: the degree of changes required to the way that existing tasks are performed and the way that users might have to re-organise their work life. Based on the costs and benefits identified users make a decision as to how deeply they will entrain the system into their work. If they are already using the system, they decide whether they should increase or decrease the level of entrainment. Generally this exploration and evaluation of a system occurs simultaneously.

3.3 An Exploitation Phase

Here team members begin the process of entraining a system into their work processes according to the 'contract' details of the preceding phase. Users will begin to exploit whatever attributes of the system they have decided to entrain. This phase can be characterised as follows:

• Users interact with the groupware as per the 'contract' they concluded in Phase 2.
• System is used here as a part of routine work processes.
• Users input just enough effort to maintain the contract type agreed (relative to Phase 2).

To summarise, each of the three phases described above are depicted together in figure 1. When the interaction progresses in time from orientation towards exploitation, the directionality of the phases is such that user understanding and experience of the system increases in each phase. Moreover, knowledge, experience and attitudes acquired in one phase are likely to influence others which follow it. Entrainment is highest near the outer circle and lowest at the inner circle. The outer and inner circles can also be viewed as representing high and low levels of commitment, respectively. The dimension of time is depicted by the clockwise arrow. Users will cycle through each phase (clockwise) on a continuous basis, so long as users continue to entrain the system, even at a minimum level. This cyclic characteristic also implies that the usability of a system can vary substantively within the life cycle of a system. That is, each time users interact with a system, ever changing circumstances will lead to it having a changed level of usability. Results have demonstrated that representation to discuss and represent conditions under which this cyclic trend can grow, decrease or terminate.

4 Conclusions

The 3P framework consolidates much of what has been learned from observations of users, in real contexts. Its potential value as a conceptual and practical analysis tool for gaining insight into the possible factors and mechanisms surrounding usability problems is currently the focus of a number of researchers. At a base level, the framework attempts to provide a plausible description of the essential elements of how users interact were observed to behave when interacting with groupware. It models these interactions as interactive processes that progresses through several temporally differentiated phases, that are directional and cyclic. Future work will concentrate on exploring the generalisability of the key assumptions underlying the framework. In particular, more work needs to be done in understanding whether the three emergent properties that form the basis of the 3P framework are plausible. Work also will focus on how to derive practical design and evaluation tools from the framework.

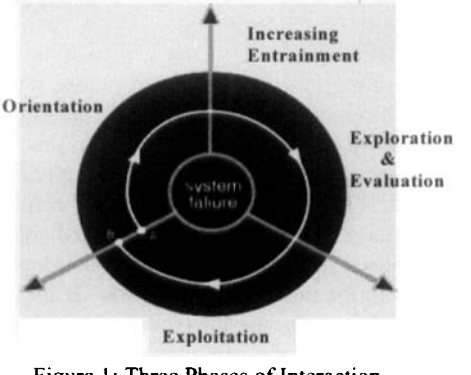

Figure 1: Three Phases of Interaction

Tools for task-based interaction and collaboration analysis

Nikolaos Avouris[1], *Georgios Fiotakis*[1], *Nikolaos Tselios*[1], *Vassilis Komis*[2]

[1]Electrical & Computer Eng. Dept, HCI Group, [2]Early Childhood Educ. Dept.,
University of Patras, 26500 Rio Patras, Greece
{ N.Avouris, Fiotakis, NiTse }@ ee.upatras.gr, komis@upatras.gr

Abstract

An innovative framework of interaction and collaboration analysis is proposed, jointly with tools to support the process. The proposed framework is based on a *collaboration analysis* first and *individual task analysis* subsequently, approach. The objective of this framework is to facilitate understanding of the group's and individual user's tasks and goals and associate the artifacts used with usability problems. An innovative aspect of the framework is the association of tasks to artifacts (tools) engaged by the users during the activity. The typical use of this framework is in interactive systems evaluation and design. The framework and the tools functionality are described in the paper. The framework is inspired by the *Activity Theory* perspective, which recognizes the importance of artifacts, actors and the context in which an activity takes place.

1 Introduction

Tools and techniques to support interaction and collaboration analysis have been proposed in the frame of usability evaluation studies for many years now (Dix et al., 1998, Nielsen 1994). These techniques involve analysis of data that are collected from field studies in various forms. Stream media like audio and video, logfiles, as well as notes and comments of observers are used in these studies. Discrete data items, like files containing solutions to problems, drawings, etc. may also be used. All these data need to be correlated and processed in order the researchers to extract useful patterns of behaviour of the actors involved, identify usability and conceptual flaws in the design of the artifacts used and evaluate the approaches that have been pursued. This analysis process has become tedious, since the high volume of data has made it more time-consuming and complex. The need for adequate tools to support the analysis has therefore increased recently. A framework of interactive systems analysis is proposed in this paper, which involves two complementary views over an activity involving multiple actors, supported with relevant tools:

(a) The first one concerns analysis of collaboration, which involves collection of field data, annotation of these observations and building of an abstract description of observed collaborative task execution. A tool has been built for annotation of data and building of abstract multi-level annotated views. This tool (Collaboration Analysis Tool, ColAT) bears interesting characteristics among which the support for various annotation schemes, the capability to commend and annotate at various levels of abstraction a sequence of events, interrelation of multiple media (video, audio, log files, snapshots) to the multilevel annotation and to the cognitive structures built through the second view.

(b) The second view is a cognitive one, which involves building of typical task structures, as anticipated by the designers of the artifact (e.g. the computer tools that are used in the activity), subsequently relating the observed task execution to the anticipated model by individuals who participate in the activity. A tool has been built to support this task-based approach, the Cognitive Modelling Tool (CMTool), also described in (Tselios & Avouris 2003). Analysis characteristics of

task models are stored in the CMTool database, together with qualifying information. The possibility of analysing further the observed system usage according to a number of dimensions permits evaluation of the artifacts both in terms of usability and effectiveness.
The proposed framework and analysis tools have been applied in non-routine task domains (Tselios & Avouris 2003). The findings reported indicate the effectiveness of this structured technique in identifying usability and interaction design problems. In the reported studies, we focused on analysis and evaluation of collaborative tasks, involving a number of individuals and artifacts engaged in problem solving either at a distance or in the same place. The ethnographic methodological approach used in the evaluation studies of this nature proved to be compatible to the proposed theories underlying the framework and the tools.

2 Analysis of observed collaborative activity

The first phase of analysis involves collaboration analysis study. A new integrated environment of analysis, the *Collaboration Analysis Toolkit* (ColAT), which integrates multiple sources of behavioural data of multiple logging and monitoring devices is used in this phase. The main emphasis of the ColAT environment is on the analysis of situations involving more than one actors. Special attention has been put on scenarios of synchronous computer-supported collaborative problem solving, in which the actors are spatially dislocated, a factor which imposes additional complexity in the analysis task.
The most important phase of analysis relates to the interpretation and annotation of the collected data, as well as generation of aggregate data of interpretative nature. An innovative feature of the ColAT approach is the support for creation of a multi-level structure that describes and interprets the logfile events. In figure 1 the concept of the multi-level logfile is shown.
The original sequence of events contained in the master logfile is level 1 (*events level*) of this multilevel structure. The keystrokes or raw observations are included in this level. An example is the event "User X selects option Y from the menu" or "User Z said …."in case of a dialogue event. A number of such events can be associated to an entry at *level* 2 by the analyst. Such an entry can have the following structure:

< *ID, User, entry_type, comment* >

where *ID* is a unique identity of the entry, *type* is a classification of the entry according to a typology that has been defined by the researcher, followed by a textual comment or attributes that are relevant to this type of task entry. Examples of entries of this level are:" User X inserts a link in the model", or "User Y contests the statement of User Z". In a similar manner the entries of the higher levels are also created, which describe the activity at the strategy level as a sequence of interrelated goals of the actors involved.
An implication of this approach is that the associated stream media are related to this multi-level view of the activity and therefore the user of ColAT can decide to view the activity from any level of abstraction he/she wishes. This approach results in a powerful analytical tool, since the possibility of viewing a process from various levels of abstraction supports its deeper understanding and interpretation.
A "*ColAT project*" is stored in a database to facilitate processing and navigation of the source data and annotations. The integrated logfile can be exported in XML form to other applications and data processing tools for further analysis. The main activity of this phase involves creation of the higher-level logfile entries. In these higher levels of the logfile the typology of the Object-Oriented Collaboration Analysis Framework (OCAF), see (Avouris et al. 2003a), has been used. This framework is particularly suitable for analysis of collaborative activity, which involves interleaving of actions and dialogue. OCAF puts emphasis on the objects of the jointly developed solution. Every object is assigned its own history of events (actions and messages) related to its

existence, as a sequence of events according to the following functional types: I = Insertion of the item in the shared space, P= Proposal of an item or proposal of a state of an item, C= Contestation of a proposal, R= Rejection / refutation of a proposal, X= Acknowledgement/ acceptance of a proposal, T= Test/Verify using tools or other means of an object or a construct.

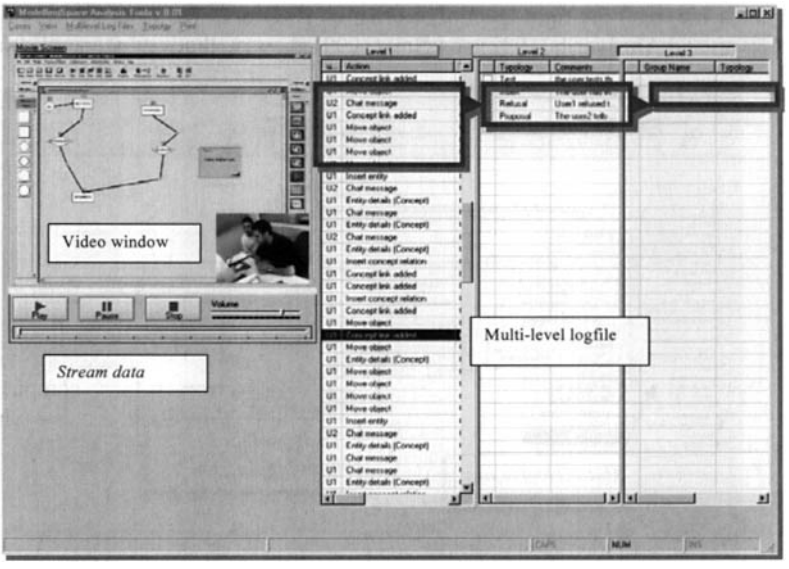

Figure 1: Overview of the ColAT data navigation environment

As an example of an OCAF event, the introduction of a new object in the shared space, is indicated as *Object (X)*= I_{UI}, i.e. User 1 inserted the object (X) in the shared space.
The ColAT environment that supports navigation of the constructed multilevel logfiles is shown in figure 1. A video window permits viewing of streaming data in association to selected events in any level of the logfiles. There are different modes of use of this environment: In the first mode, *navigation is controlled through the video*. When a position of the video file is selected, the corresponding event of the log hierarchy that the video is related to, is highlighted. In the second mode, *navigation is controlled from the logfiles*. In this case the user can select any event in the first level of the log file and the video starts from that event onwards.
The user can hide the levels of abstraction he/she wishes to ignore, thus defining the desired view over the field data. The ColAT navigation tool has been proven particularly useful in analysing the data of reported experiments (Avouris etal. 2003b), following the OCAF framework.

3. The task analysis of individual actors

During the second phase, each individual actor is studied in relation to the tasks she undertook and the goals she attempted to accomplish. The task analysis method adopted is the Hierarchical Task Analysis (HTA), proposed by Shepherd (1989), accordingly modified, as discussed here. The intention is to build a conceptual framework reflecting the way the user views the system and the tasks undertaken in order to accomplish set goals. So the main objective of our analysis is to reflect on the observable behaviour of users and identify bottlenecks in the human-system dialogues rather than explicitly understand the internal cognitive processes as one performs a given task. So we include in our analysis, even incorrect problem solving approaches, caused

either by limitations in the provided tools design or by conceptual misunderstandings related to domain knowledge and use of available tools. These errors may lead to not satisfactory solutions to a given problem, but often do contribute to better and deeper understanding of concepts.
Through the proposed task analysis technique, the typical or expected use of the software environment by each individual user is reduced, to a sequence of tasks. However, this task analysis can be achieved if the right level of abstraction is used, as fed from the previous phase of the study. Kirwan and Ainsworth (1992) and Boy (1999) show methods of accomplishing this.

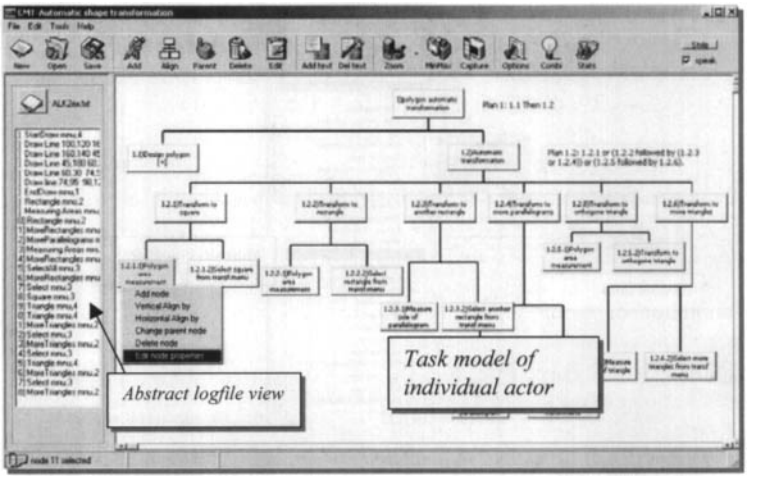

Figure 2: The Cognitive Modelling Tool (CMTool) Environment

One important aspect of our analysis framework is the classification of observed unexpected or incorrect user behaviour, see Tselios et al. (2002) and Tselios & Avouris (2003). Through the application and analysis of this technique, we have identified and classified five main categories of errors: *Severe syntactic error, Minor syntactic error, Conceptual error, Inconsistency, Design principle violation.* These errors can be identified during observation and analysis of problem-solving activity during this phase. Also in *plans* (Shepherd, 1989), simple expressions are included such as (!) and (x!) to denote subtasks containing errors, and { } to indicate occurrence of unforeseen tasks according to the original designer model.
Display of an annotated log file at selected level of abstraction, shown on the left of fig. 2, next to the task model structure, is supported. So both interaction details and cognitive goal hierarchies are displayed simultaneously to the user of CMTool. Task models are stored in a relational database, grouping the various tasks analysed, with additional identification information (designer's model or revised designer's task model (DTM) or user's task models (UTM)). In addition, quantitative analysis tools are supported to extract useful metrics related to the analysed tasks. Examples of these metrics are the number of keystrokes required to achieve a specific goal or sub-goal, the mean time required and the interaction complexity of specific user model, compared to the original designer's expectations or to the revised and adapted model.
In CMTool, the evaluator can select parts of a task structure representing a specific problem solving strategy, which can be stored for future reference or comparison with other users' strategies. In addition, the possibility to analyse system's usage in five dimensions is a contribution of the tool to the evaluation process. These are: (i) High level tasks, (ii) users, (iii) specific strategies, (iv) tools used, (v) usability problems detected. This process is carried out through a visual information query environment. Field experiments could be analysed across any possible element of the five different dimensions discussed (e.g. "Show all encountered problems

related to tool X", etc.). Complex analysis can be carried out according to any of these dimensions, supporting study and analysis of encountered problems.

4 Conclusions

This paper describes the main functionality of tools to support multilevel analysis of field data collected during evaluation studies of group collaboration. Both *ColAT* and *CMTool* have been recently applied in evaluation studies of group problem solving activities. The theoretical underpinning of this approach is Activity Theory (Kaptelinin etal. 1999). This is a conceptual approach, that considers as the unit of analysis the activity, consisting of a subject (an individual or group), an object or motive, artefacts, and sociocultural rules. Understanding thus human activity requires a commitment to a complex unit of analysis. The multi-view approach proposed covers this aspect. In the proposed framework we move from the group level analysis to the individual human-computer interaction study in a smooth way. The multi-level annotation scheme described here permits change of point of view and relates the observational data to the annotations. These annotations can comply with a typology imposed by a methodological framework, like the OCAF scheme used in our example. The framework permits shifting from bottom up annotation of group activity data to top-down task level description of the observed human-computer interaction

The concepts and tools discussed here are relevant to researchers who are involved in analysis and evaluation of complex collaboration-support activities, in design and evaluation of new tools and in support of users' meta-cognitive activities.

Acknowledgement

Partial funding of project IST-2000-25385 ModellingSpace, by the EC is acknowledged.

References

Avouris N.M., Dimitracopoulou A., Komis V., (2003a). On analysis of collaborative problem solving: An object-oriented approach, Computers in Human Behavior, 19 (2), March, pp. 147-167.

Avouris N., Komis V., Margaritis M., Fiotakis G., (2003b). Tools for Interaction and Collaboration Analysis of learning, Proc. Conf. CBLIS 2003, July, Nicosia, Cyprus.

Boy, G. A. (1998). Cognitive Function Analysis. Ablex Publ., Greenwood, Westport, CT, USA.

Dix A., Finlay J., Abowd G, Beale R., (1998), Human-Computer Interaction, Prentice Hall

Kaptelinin, V., Nardi B., Macaulay C., (1999). The Activity Checklist: A Tool for Representing the "Space" of Context, Interactions, July, 27-39.

Kirwan, B. and Ainsworth, L.K. (1992). A Guide to Task Analysis. Taylor & Francis, London.

Nielsen, J., (1994). Usability inspection methods. In J.Nielsen, R.L. Mark (Ed.), Usability Inspection Methods, John Willey, New York.

Paterno', F. (2000) Model-based design and evaluation of interactive applications, Springer Series in Applied Computing, Springer-Verlag, London.

Shepherd, A. (1989). Analysis and training in information technology task. In Diaper, D.(Ed.) Task Analysis for human computer interaction. Ellis Horwood Limited, 15-55.

Tselios N., Avouris N., (2003), Cognitive Task Modeling for system design and evaluation of nonroutine task domains, E. Hollnagen (ed.) Handbook of Cognitive Task Design, LEA, 307-332.

Tselios N., Avouris N., Kordaki M., (2002), Student Task Modeling in design and evaluation of open problem-solving environments, Education and Information Technologies, 7:1, 19-42.

The Semiotic Engineering Use of Models for Supporting Reflection-in-Action

Simone D. J. Barbosa, Clarisse Sieckenius de Souza, Maíra Greco de Paula

Departamento de Informática, PUC-Rio
Rua Marquês de São Vicente, 225
Gávea, Rio de Janeiro, RJ
Brasil, 22453-900
{simone, clarisse, mgreco}@inf.puc-rio.br

Abstract

Diverse models have been proposed to cope with the complexity of HCI design and increase software usability, most of them based on cognitive theories. In this paper, we argue for the need of complementary theories of HCI and present an alternative set of HCI design representations and models based on Semiotic Engineering. These comprise extended scenarios and task models, and a novel semiotically-based interaction model. In the latter, signs and communication breakdowns handle cohesion and coherence issues for prospect conversations between users and the *designer's deputy* – an entity capable of participating in all and only the designed conversations. Our goal is to empower designers, supporting reflection about their interactive solutions.

1 Introduction

Diverse models have been proposed to represent HCI design, taking into account users' characteristics, preferences and needs. Most of them are based on cognitive theories (Norman & Draper, 1986). They focus mainly on the individual interacting with an application, considering his/her physical (motor and perceptive) and mental processes. Their main goal is to understand how these processes take place during interaction, in order to predict and avoid possible cognitive problems. But software applications are intellectual artifacts, resulting from designers' understanding, reasoning, and decision-making processes. Thus, we should support not only the designers' intellectual processes (Schön, 1983), but also the expression and usage of such processes' results through the user interface. This double enterprise is the object of study of Semiotic Engineering (de Souza, 1993; de Souza, in preparation). In this paper, we describe how a specific set of representations and models can support HCI design. We are concerned with building a coherent and cohesive overall interface message, attempting to maximize the chances that intellectual artifacts such as software will be interpreted as meant. As is the case with all interaction about what goes on in one's mind, HCI should support *conversation*, even if non-textual codes are extensively used to convey the designers' ideas and the products' affordances.

Semiotic Engineering views the interface as a designer(s)-to-users message, representing the designers' response to what they believe are the users' problems, needs, preferences and opportunities. In this message, they are telling users, directly or indirectly, what they have in mind. In fact, user–system interaction is viewed as a *conversation* between the user and the "designer's deputy" – a *programmed* representative of the designer. Since designers are no longer there at the time such conversations are carried out, they must anticipate every possible conversation. Thus, the designer's deputy is capable of participating in all and only the designed conversations.

The semiotic engineering of human-computer interaction is focused on building the designer's deputy, which will be implicitly or explicitly represented in the application's interface. And this imposes certain additional requirements on design representations and models. For example, we need to represent the nature of signs that conversational parties may use, the remedial actions for interaction breakdowns, as well as the potential threads of conversation that can be followed during interaction. We propose and use a classification of *signs* to describe every piece of information relative to some concept in the domain or in the application itself. Signs are extracted from scenario representations and provide a thread that binds all design models and representations with the actual user interface. Also, because we take *communication breakdowns* as an inherent part of HCI, we aim at helping designers to tell users not only how to perform their tasks under normal conditions, but also how to ask for help or take remedial action in mistaken or unsuccessful situations. To this end we propose a classification of communication breakdowns, to be used in extended task models.

2 Extending scenarios and task modelling

Scenarios can be used throughout the development process, starting from the analysis of the domain and the users' tasks and characteristics (Carroll, 1995). By means of scenarios, designers not only learn about the users' vocabulary and thought processes, but they have a chance to make this knowledge explicit in a language that is accessible to users. When writing scenarios, the designer should have in mind what the purpose of each scenario is, allowing for user involvement in: investigating certain aspects of a domain or socio-cultural environment; proposing a task structure or an interaction solution; contrasting alternative task structures or interaction solutions; and so on.

Our extension to scenarios is one of adding a semiotic dimension to their use. Designers should carefully extract from scenarios the domain and application *signs* that are meaningful to users. While these signs are usually treated as data, in Semiotic Engineering they acquire a new status. They are analyzed and classified according to the degree of familiarity users are expected to have with them. *Domain* signs are those directly transported from the users' world, such as "full name". *Application* signs are those that have been introduced by the application and had no previous meaning to the user, such as "your login account". Finally, *transformed* signs are those derived from an existing sign in the world that has undergone some kind of transformation (e.g. based on an analogy or metaphor) when transported to the application, such as "backup files".

Different kinds of signs may require different kinds of sense-making support. In general, a domain sign would only require explanation if specific constraints are imposed by the application. For example, the concept of "full name" is clear to users, but a restriction about the number of characters allowed in the corresponding piece of information might not be, and thus need some clarification from the designer's deputy. A transformed sign would require an explanation about the boundaries of the analogy. For example, a folder in the desktop metaphor might require an explanation about its "never" getting full, except when the hard disk which contains it lacks space. Finally, an application sign may require a complete explanation about its purpose, utility and the operations that can manipulate it. An example might be a sign for "zooming" in a graphics application. There are of course some signs that can be classified in either group. In these cases, it is the designer's responsibility to choose, based on the analyzed characteristics of users and their tasks, the amount of support to provide in order to have users fully understand each sign.

From such semiotically-enriched scenarios, users' goals are identified and may be structured in a hierarchical goal diagram. Each goal may be further decomposed into tasks that are necessary to achieve it. We made simple adaptations to the structure chart notation in order to represent the

hierarchical decomposition of tasks, some task properties and relations between tasks, and discrimination among sequential, alternative, and iterative tasks. We can thus represent tasks that do not follow a predefined sequence, tasks that may be performed anywhere within the goal, preconditions for performing a certain goal or task, optional tasks, and stereotypes as a reuse mechanism.

Our approach is somewhat similar to Hierarchical Task Analysis (Annett & Duncan, 1967), but a major difference is that our adaptations are specifically suited to represent signs and potential breakdowns involved in each task. We allow designers to classify and represent the kind of support they intend to provide in each problematic situation: passive prevention (PP: problems that are prevented by documentation or online instructions); active prevention (AP: errors that will be actively prevented by the system); supported prevention (SP: situations which the system detects as being potential errors, but whose decision is left to the user); error capture (EC: errors that are identified by the application and that should be notified to users, but for which there is no possible remedial action within the application); and supported error handling (EH: errors that should be corrected by the user, with application support).

3 A Semiotically-based Interaction Model

Task models describe the structure and organization of the tasks that will be supported by the system, but more is needed to promote reflection about how design sense-making and decisions will be conveyed to users through the interface, for example if explicitly or not, in detail or not.

The interaction model described here was conceived to be applied between the initial analysis and the interface specification phases. It has shown to be useful in helping designers grasp a global view of the conversations that comprise the application (Paula, 2003), and thus design a coherent designer-to-users message, keeping in mind the communicative role of the designer's deputy. It has a diagrammatic view coupled with a textual specification. The diagrammatic view gives designers a global perspective on the interactive discourse, promoting reflection by representing all of the potential user–system conversations. The textual specification delves into the details of each user–system dialogue, and sets the stage for formal user interface specification.

The interaction model comprises scenes, system processes, and transitions between them. A scene represents a user–deputy conversation about a certain matter or topic, in which it is the user's "turn" to make a decision about where the conversation is going. This conversation may comprise one or more dialogues, and each dialogue is composed of one or more user/deputy utterances. In a GUI, for instance, it may be mapped onto a structured interface component, such as window or dialog box, or a page in HTML. Figure 1 illustrates the representation of a scene called Search documents, which contains two dialogues: [inform search criteria] and [choose presentation format].

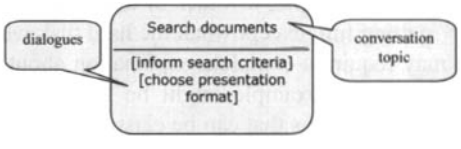

Figure 1: Diagrammatic representation of a scene called "Search documents".

The signs comprised in dialogues and scenes are included in a textual specification. When a sign is uttered by the deputy, i.e., presented to the user, it is represented by the sign name followed by an exclamation mark (e.g. `date!`). A sign about which the user must say something, i.e., which must

be manipulated by the user, is represented by a name followed by an interrogation mark (`login?`, `password?`). Apart from signs, the remedial action for each predicted breakdown situation is also represented. Figure 2 illustrates the textual specification of a "Search documents" scene[1]:

```
Search document {
        [inform search criteria:
                title?, author?, date? ([PP] dd/mm/yyyy)]

        [choose presentation format:
                format?(table, report)]
}
```

Figure 2: Textual representation of a scene, including its corresponding signs.

Going back to the diagrammatic view, in it a system process is represented by a black rectangle. By using "black-boxes" for representing something users do not perceive directly, we encourage the designer to carefully represent the deputy's utterances about the results of system processing, as the only means to inform users about what goes on during interaction.

Transitions are changes in topic, and are represented by labeled arrows. An outgoing transition from a scene represents a user's utterance that causes the transition, whereas an outgoing transition from a process represents the result of the processing as it will be "told" by the designer's deputy. In case there are pre- or post-conditions, they should be represented within the transition label, following the keywords pre: and post:, respectively.

Some scenes may be accessed from any point in the application, i.e., from any other scene. The access to these scenes, named ubiquitous access, is represented by a transition from a grayed scene containing an identifier in the form U<number>. Moreover, there are portions of the interaction that may be reused in various diagrams. Figure 3 presents a diagrammatic representation of part of an interaction model corresponding to a goal of searching documents.

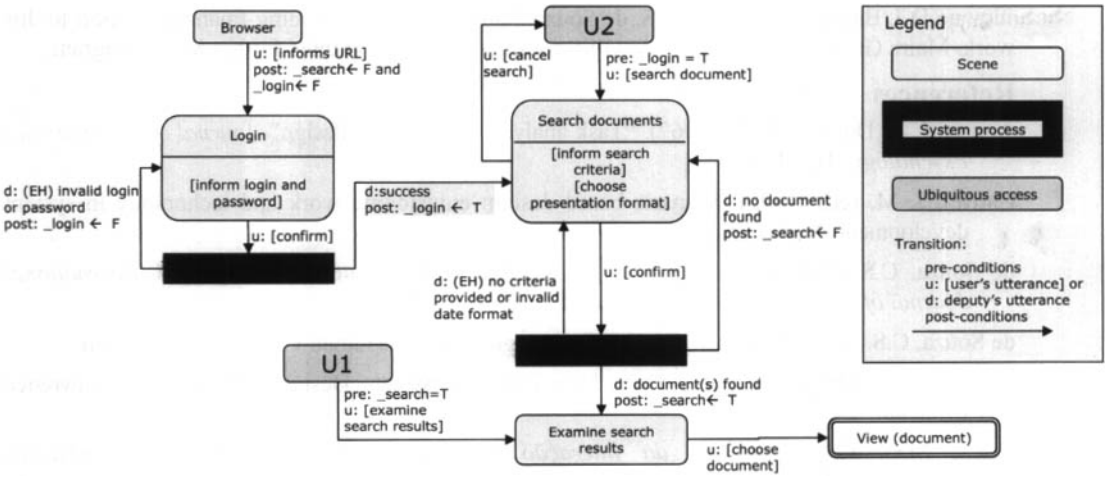

Figure 3: Sample interaction diagram.

[1] The complete definition of the textual specification may be found in (Paula, 2003)

4 Concluding remarks

The semiotically-enriched scenarios and the proposed interaction model are epistemic design tools. They support Schön's *reflection in action* epistemology (Schön, 1983) in that they explicitly represent dimensions of semiotic classification for communication-centered design. The centrality of communication and the importance of conversational affordances are justified by the intellectual nature of software, be it in the traditional format of desktop applications or in contemporary applications for mobile devices. In both, *signs* (visual, textual, gestural, aural, etc.) must be used to convey ideas, create conventions and facilitate usage. The proposed epistemic tools are so formatted that the construction of storyboards and prototypes should be as straightforward as in the case of other popular notations.

Semiotically-enriched scenarios support communication with users, allowing for the identification, exploration and verification of the application's purposes as far as designers and users are concerned. Signs bring attention to what is known and unknown, what remains the same and what is changed, what new language(s) or conventions must be learned, and so on. The extended task model incorporates elements that are crucially important for sense-making, namely the kinds of actions that handle breakdown situations. And the interaction model expands a blueprint of the potential conversations that may be carried out between the user and the designer's deputy during interaction. They help designers gain a sense of what are *all* and *the only* conversations (or communicative exchanges, to incorporate non-textual interaction) that the designer's deputy is equipped to entertain.

The design models under investigation have been used in the design of some Web and Windows applications, and in undergraduate class assignments. We have gathered some informal indications that they succeed in promoting the designer's reflection and as a result help produce better interactive applications. However, further studies are necessary to provide stronger evidence.

Acknowledgments

Simone D.J. Barbosa and Clarisse S. de Souza thank CNPq for providing financial support to this work. Maíra Greco de Paula thanks CAPES for the scholarship granted for her M.Sc. program.

References

Annett, J., & Duncan, K. D. (1967). "Task analysis and training design". *Journal of Occupational Psychology*, 41, 211-221.

Carroll, J. M. (ed) (1995). Scenario-based design: envisioning work and technology in system development, New York, Wiley.

de Souza, C.S. (1993). The Semiotic Engineering of User Interface Languages. *International Journal of Man-Machine Studies*, 39, 753-773.

de Souza, C.S. (in preparation). The Semiotic Engineering of Human-Computer Interaction.

Norman, D. e Draper, S. (eds., 1986) User Centered System Design. Hillsdale, NJ. Lawrence Erlbaum.

Paula, M.G. (2003) *Projeto da Interação Humano-Computador Baseado em Modelos Fundamentados na Engenharia Semiótica: Construção de um Modelo de Interação* (in Portuguese). Master's dissertation. Departamento de Informática, PUC-Rio, Brazil. Available online from http://www.serg.inf.puc-rio.br/

Schön, D. (1983). The Reflective Practitioner: How Professionals Think in Action. New York: Basic Books.

Levels of Guidance

Andrew Basden

Information Systems Institute, University of Salford,
Salford, M5 4WT, U.K.
A.Basden@salford.ac.uk

Abstract

Many sets of guidelines have been offered for, for example, web site design. But how complete are they? Often their categories overlap or lack an ordering principle. This paper shows how philosophy (that of Hart and Dooyeweerd) can be used to identify guidelines and their categories for web site design.

1 Introduction: Categories of Guidelines

Sets of guidelines for designing user interfaces, web sites, etc. abound. But, after using them for some time, we find ourselves wondering whether they cover everything important, we find the categories within some sets overlap or seem haphazard. This paper suggests a framework that can help us critique and refine categories of guidelines that is based on philosophy. We focus on the UI found in web sites, but at the end suggest how the framework may be extended to guidelines for other types of software.

The Evidence-Based Guidelines on Web Design and Usability Issues by the National Cancer Institute (NCI, 2003), contains 60 guidelines in the categories: Design process, Design considerations, Titles and headings, Page length, Layout, Font/text size, Reading and scanning, Links, Graphics, Searching, Navigation, Software and hardware, Accessibility. But these categories exhibit problems by virtue of having been assembled from empirical findings and the preferences of individuals found in other sources. Considerable overlap is evident, e.g. between Links, Navigation and Search, and some guidelines, e.g. "Establish a high-to-low level of importance for each category ..." occurs several times. The categories are not homogeneous (properties of site mix with user and design activity) and their order appears haphazard. We need categories in which the ordering principle is clear, not only to minimize overlap, but to allow us to judge, as guidelines proliferate, whether new categories are needed or categories may be merged or split.

The Web Content Accessibility Guidelines 2.0 (W3C, 2003) contains 21 guidelines under the categories: Perceivable, Operable, Navigable, Understandable, Robust. Though there is no overlap, their content is unbalanced in scope and depth. Navigable, for example, has six guidelines ('checkpoints') of general applicability while Operable has only three, all relevant only to specialised conditions. We also wonder whether other categories should be included, covering such issues as aesthetics, quality of content and cultural acceptability. We need categories that take on their intuitive meaning, are balanced in scope and depth, and cover all important issues of web site design.

2 Philosophy

Categorization involves ontological commitment. Hart (1984) has argued that, only philosophy, which concerns itself with the diversity and coherence of our experience, can provide us with both wide coverage and an ordering principle, and explores the philosophy of the late Dutch thinker, Herman Dooyeweerd (1955). We cannot rehearse his arguments here, but will examine how this philosophy can provide categories of guidelines. Dooyeweerd's ideas are interesting because he radically questioned the presuppositions underlying Western thinking and constructed a philosophical approach based on Meaning-oriented presuppositions that, while still in need of some refinement, is breathtaking in its scope and diversity.

2.1 Aspects

Perhaps the best known portion of Dooyeweerd's thought is his notion of irreducible aspects. Each aspect has a set of laws that enable meaningful functioning. Based on long reflection on both day to day living and scholarly writing, Dooyeweerd offered the following suite of aspects (though he made it clear his suite was to be criticised and refined):

- Quantitative aspect, of amount
- Spatial aspect, of continuous extension
- Kinematic aspect, of flowing movement
- Physical aspect, of energy and mass
- Organic aspect, of life functions and maintenance of organisms
- Sensory/psychic aspect, of sense, feeling and emotion
- Analytical aspect, of distinction, abstraction
- Formative aspect, of history, culture, creativity, achievement and technology
- Lingual aspect, of symbolic meaning and communication
- Social aspect, of social interaction, relationships and institutions
- Economic aspect, of frugality, skilled use of limited resources
- Aesthetic aspect, of harmony, surprise and fun
- Juridical aspect, of 'what is due', rights, responsibilities
- Ethical aspect, of self-giving love, generosity, care
- Pistic aspect, of faith, commitment and vision.

While the earlier (at least first four) aspects have determinative laws, the later aspects are normative, enabling freedom in our functioning but acting as guides for it. Guidelines may be seen as verbal expressions of selected norms in the aspects. To Dooyeweerd, human activity and life involves, in general, every aspect, in rich coherence. This includes using a web site. He claimed that human activity works well when we function well in all aspects. Thus the web site works well only if the user functions well in every aspect when using it. No aspect, nor its laws, may be reduced to any other. This gives multiple levels of design freedom. But it also means we cannot assume that good functioning in one aspect will automatically engender good functioning in any other. So designers must consider each and every aspect in which the user will function and, since much of our aspectual functioning is tacit (Polanyi, 1967), care must be taken not to overlook the less obvious aspects.

The first proposal in this paper is that guidelines for web site design can be based on the norms of the aspects of the user's use of the web site, which also provide useful categories for grouping them. Being irreducible, each aspect provides a distinct level of analysis.

3 Identifying the Aspects of Web Site Use

But how do we identify which these aspects are? Normally, we use a web site for some purpose in life, such as finding legal advice so that we might challenge genetically modified crops. We must separate the aspects that are contingent on the purpose (which in this case include the juridical, biotic and economic) from those of web browsing itself, those of searching, scanning, reading, understanding, etc., which pertain across all purposes. (The set of contingent aspects, as we will call them, might overlap with the set of aspects of web browsing.) To identify the aspects of web browsing itself, we make use of three further elements of Dooyeweerd's thought.

First, in many human activities, especially specialised activities carried out for wider purposes as web browsing is, one aspect is of primary importance; Dooyeweerd called this the qualifying aspect. It is the lingual aspect that qualifies web browsing (and indeed most texts, communications, etc.) because, whatever its purpose, its role is to convey meaning via symbols.

Second, Dooyeweerd claimed that proper functioning in any aspect depends on and involves proper functioning in earlier aspects. The lingual aspect, therefore, depends on the formative, analytical, sensory/psychic aspects. (We start at the sensory because it is where medium differentiates into visual, aural and haptic, but a more detailed treatment would continue back at least to the physical aspect.) These aspects correspond approximately with the levels found in linguistics: semantics, syntax, lexics and sensory.

Third, each aspect contains 'echoes' of all the others that anticipate later aspects and retrocipate earlier ones. Often, an aspect anticipates most strongly the one that follows it, paving the way for it. Thus the lingual aspect strongly anticipates the social; so cultural context is important in interpreting text etc.

Our second proposal is that in the case of web sites the relevant aspects for which we must provide guidelines, the 'levels' of guidance, are the sensory/psychic, analytical, formative, lingual and social, and the contingent aspects taken as a single category.

4 Aspects of a Web Site: Levels of Guidance

The sensory/psychic aspect is about sensing (seeing, hearing) and feeling (emotion). This offers us guidelines about matching UI devices used with the sensory capabilities of people. Thus both NCI and W3C sets contain guidelines about colour, colour blindness, sound, flicker, etc. Hardware considerations can be of this aspect, since it is phenomena that emerge from hardware that we sense. Long download times, while sometimes having economic repercussions, is mainly an annoyance to users, so guidelines about this are usually expressions of the norm of feeling. Page layout has a sensory aspect, but is an example of a multi-aspectual guideline that speaks of how one aspect of the web site facilitates another; see below.

The main norm of the analytical aspect is that we should be able to make clear distinctions that are meaningful. Thus the W3C guideline, "Ensure that foreground content is easily differentiable from background for both auditory and visual presentations" expresses this norm, as do the NCI guidelines, "... Headings provide strong cues ..." and "Always use underlines or some other visual indicator ... to indicate that words are links." Guidelines about font size, graphics, screen resolution, etc. where the thrust is about helping the user distinguish things express this norm. Guidelines about helping the user to identify structures in meaning anticipate the next aspect.

The formative aspect is about formative power. Part of this focuses on structure. To understand the site, users of a web site form their own structured mental model of its content, and this is facilitated if the syntactic structure of the site supports its content. The NCI guideline "Put as much important content as close to the top of the hierarchy as possible" is an expression of this. So are those about splitting into paragraphs, sections and subsections (for which the headings are analytical-aspect cues), about how pages are linked, etc. Part of this aspect is to do with achievement. A major area of achievement in web site use is finding relevant knowledge. Therefore guidelines about helping the user find the knowledge they want are expressions of the norms of this aspect.

Dooyeweerd maintained that though the aspects are distinct, they 'resist' being separated from one another, because they were meant (by their Creator) to form a coherent spectrum of Meaning. So it is often difficult to clearly identify under which aspect a guideline belongs because it seems to have several aspects. Such guidelines concern how earlier aspects provide the means by which later aspects may be implemented, and how later aspects provide the purpose for implementing it in the earlier aspects. As mentioned above, page layout has several aspects. Sometimes it is aesthetically important, but most guidelines about it focus on how layout (itself of the sensory aspect) can help the user distinguish what is important to them (analytical aspect) and thus reflect the structure of the content (formative aspect). Recognising the various aspects in even a single phenomenon like this helps us in design of that phenomenon.

The lingual aspect concerns meaning conveyed by symbols, and is the primary aspect of web sites. As one guideline (NCI) says, "Content is the most critical element of a Web site." Some general guidelines exist, such as NCI's "Use only graphics that enhance content or that lead to a better understanding of the information being presented" and W3C's "Write as clearly and simply as is appropriate / possible for the purpose of the content." But the norm of conveying meaning goes further. Content should be accurate, consistent, interesting, relevant, polite, coherent, etc.; some of these are Grice's (1967) maxims. Such quality issues are seldom expressed as guidelines, but perhaps they should be.

The social aspect concerns social interaction and institutions. While web sites might help towards these, the aspect's importance in web site design lies in it being strongly anticipated by the lingual aspect. Cultural assumptions, expectations, etc. of the reader determine how the reader interprets content, and are often of a tacit nature, so misunderstandings or offence can occur. The norm for site designers is "Consider the person who will read this; do not be satisfied with just conveying information." This aspect urges us to be careful and creative in use of humour (thus anticipating the aesthetic aspect).

NCI offers guidelines about site goal, such as "Provide useful and usable content that supports the Web site goal on each page." But does this refer to a communicative goal of what information is conveyed or the goal of the user in accessing the site? Though usually not made clear, it is a crucial difference. A communicative goal is covered by the aspects above, but the user's goal is covered by the contingent aspects. Guidelines need to be made for each contingent aspect. When the designer knows the purpose(s) for which the site will be used, then the contingent aspects can be identified, and guidelines devised that will express their norms. For example, for a site that will be used to give legal advice to campaigners, juridically-oriented guidelines might be useful, such as "Because it will be used in courts of law, be particularly careful about the accuracy and completeness of the content." But for many sites the uses may be diverse and not clearly known in

advance. So contingent-aspect guidelines will be more general, such as, for the ethical aspect, "As far as possible, steer the user towards ethical rather than anti-ethical use of the information."

5 Discussion

We have shown how Dooyeweerd's (1955) theory of aspects can help identify categories of guidelines for web site design, and also how, by considering each aspect in turn, we can be stimulated into devising specific guidelines for aspects we might have overlooked. Our two-part proposal is that guidelines are expressions of the norms of relevant aspects, and that we can identify these as: the qualifying aspect, some of its preceding ones, its immediate successor, and the set of contingent aspects considered as one.

By doing this, we avoid the problems identified earlier. The clear principle behind the categories is to include relevant aspects that form the framework of Meaning for human activity, and the order among the categories is that of the aspects. Minimizing overlap between categories is ensured by the irreducible distinctness of the aspects. Some guidelines are multi-aspectual, so may be included in several aspects, but there is now a rationale for such duplication. Categories based on aspects are all of wide scope, none being applicable only in very specialised situations, but the aspects offer us a framework for considering specialised situations. And, because Dooyeweerd claimed that the kernel meanings of the aspects may be grasped intuitively, the categories take on their intuitive meaning. Finally, by reference to the aspects, we have a scheme that can 'tap us on the shoulder' when we have overlooked something important.

A similar approach might be used to provide guidelines for other types of software: devise guidelines that express norms of its qualifying aspect, and those nearby. For example, computer games are qualified by the aesthetic aspect, of surprise, play and fun, which is preceded by the economic aspect of limited resources (as seen in the importance to many games of limiting players' resources of ammunition, stamina, time, etc.). The approach outlined here shows considerable promise and is worthy of further research.

References

Dooyeweerd, H. (1955). A New Critique of Theoretical Thought. Ontario: Paideia Press.

Grice, H.P. (1967). Logic and conversation. In Cole, P., Morgan, J. (Eds) *Syntax and Semantics* Vol 3. New York: Academic Press.

Hart H, (1984), Understanding Our World: An Integral Ontology, University Press of America.

NCI, (2003). Evidence-Based Guidelines on Web Design and Usability Issues. National Cancer Institute, Retrieved January 31, 2003 from http://usability.gov/guidelines/.

Polanyi, M. (1967). The Tacit Dimension. London U.K.: Routledge and Kegan Paul.

W3C. (2003). Web Content Accessibility Guidelines 2.0. Retrieved January 31, 2003 from http://www.w3.org/TR/WCAG20/.

Guidelines and Freedom in Proximal User Interfaces

Andrew Basden

Information Systems Institute, University of Salford,
Salford, M5 4WT, U.K.
A.Basden@salford.ac.uk

Abstract

There is a tension between guidelines and freedom. It arises from certain philosophical assumptions that we do well to question. An alternative view of freedom as 'ability to respond appropriately to the diversity of the situation' is considered. It is shown how this can work out within the guideline principles of Proximal User Interface for three classes of software.

1 Introduction: Guidelines and Freedom

Guidelines for the design of user interfaces (UIs) - such as style guidelines - can provide a coherent look and feel to software across a platform, so that novices quickly become familiar with new software. But they can be restrictive, constraining the design in unnecessary and inappropriate ways. This is especially so when the user progresses from being a novice to an everyday user. Somehow, we need to relate freedom to guidelines in such a way that both have appropriate power and both are meaningful.

There are two main ways to approach the issue of freedom: 'freedom from' and 'freedom to'. The 'freedom from' view is epitomised by the British thinker Hobbes' stress on personal freedom from interference and the German thinker Habermas' norm of emancipation from unwarranted constraints, as adopted by the critical system community (Lyytinen & Klein, 1985). In this view, guidelines can be seen as interference and constraint, and thus inherently opposed to freedom. But it often results in antinomy; as seen in Hobbes' monarchial Leviathan and in the paradox of enforced emancipation (cf. Wilson, 1997).

We adopt the alternative notion, of 'freedom to'. This is found, for example in Milton's idea of freedom as not being beholden (to lords) and thus being able to be what we should be. More recently the notion has been worked out more precisely by the Dutch thinker, Dooyeweerd (1955), who developed an idea of freedom as the ability to respond meaningfully to the diversity around us. In this view, guidelines can enable and encourage such freedom rather than constrain - but only high quality guidelines do so.

What are high quality guidelines? To Dooyeweerd, the ability to respond is made possible by the 'law side' of reality, which is a framework of Meaning in which everything exists and occurs. He accounts for diversity by suggesting that this framework of Meaning comprises a number of irreducibly distinct aspects, each with a different kernel meaning. Each aspect has a distinct set of laws that enable a distinct type of meaningful functioning to which we respond in life (e.g. lingual laws of syntax, semantics, etc. enable meaningful communication). The laws of earlier aspects (quantitative, spatial, kinematic and physical) are determinative, while in the later aspects (biotic,

sensory, analytic, formative, lingual, social, economic, aesthetic, juridical, ethical and credal) they are normative, allowing freedom. Human activity - including engaging with software via a user interface - involves all aspects.

Guidelines for designing UIs can now be seen as an expression of this diversity of aspectual Meaning. To be of high quality and facilitate freedom, they should be rooted in the aspects and take all into account. This need not mean devising completely new guidelines or methodologies. Basden & Wood-Harper (2002) have shown how Dooyeweerd's philosophy can underpin and enrich soft systems methodology. Here we examine how it might underpin and enrich principles of proximal user interface, discussed by Basden, Brown, Tetlow & Hibberd (1996).

2 Principle of Proximal User Interfaces

"The real problem with the interface," said Norman (1990), "is that it is an interface. Interfaces get in the way. I don't want to focus my energies on an interface. I want to focus on the job." A proximal user interface (PUI) is one that is so 'natural' that it does not "get in the way". It embodies the norm that the software should be able to become, as it were, part of the user or, as Polanyi (1967) stated, 'proximal'. By contrast, conventional UIs are 'distal', relating to the user via a 'dialog' of commands, messages, clicks, menus, etc. To achieve proximality, the UI should not consume the user's thinking and attention nor interrupt the flow of thinking within the task (Basden & Hibberd, 1996).

The principles of PUI recognise that the real task for which the tool is being employed might be more fluid, flexible, dynamic and sophisticated than its formal specification (given in design of system, or in training of the user and work definition) might suggest. Of the nine principles, two concern the overall relationship between user, task and tool, and bring together the notions of direct manipulation and affordance, three concern what the user experiences via the UI, and the remaining four with how the user can act. We focus on visual interfaces, but the principles apply equally to aural and haptic devices. We illustrate how the principles apply to three different types of software: word processors (WPs) (where Dooyeweerd's lingual aspect is important), computer games of the Doom type, where the user moves about, finding objects, being ambushed and fighting enemies (spatial and aesthetic aspects), and mind-mapping toolkits such as Istar (Basden & Brown, 1996) that allow the user to build a knowledge base as a box and arrows diagram (the analytical aspect).

2.1 Principle of Direct Semantics

The principle of direct semantics states that there should be a direct mapping from lexical phenomena in the interface to the semantics of the user's task. For example, in computer games the enemy is a recognisable shape on screen and sound from speakers, and pressing the joystick button is directly linked to fighting it. In Istar, each item is a box and each relationship, an arrow, and pressing the mouse button draws a new box (creates a new item). This principle is not unlike later versions of direct manipulation (Shneiderman, 1992), but it concerns the user's view (e.g. shapes seen, sounds heard) as well as user action (e.g. mouse gestures). In conventional, distal, UIs, the mapping is indirect, via syntax of a 'dialog' between user and machine. Interpreting and managing syntactic paraphernalia like toolbars, menus, dialog panels, commands, etc., incurs a heavy penalty in cognitive load. Though excellent for novice users, they "get in the way" during everyday use, so they are to be avoided.

2.2 Principle of Appropriateness

The principle of appropriateness states that mapping between lexical phenomena and semantics of the application should be appropriate, or 'natural'. This implies three things. What is 'simple' in the application domain should be expressed by simple lexical gestures and shapes. Each lexical phenomenon should 'afford' the semantic meaning it expresses (Greeno, 1994). And the diversity encountered in the situation of use should be respected. If diversity is vested in distinct aspects (Dooyeweerd, 1955) the UI must reflect this by a different lexical-semantic mapping for each. For example, a line expresses a relationship in Istar, underlining in WPs and, say, the amount of damage sustained in the game. Dragging the mouse draws a new relationship in Istar or selects a region of text in WPs.

2.3 Principle of Large Easels

The 'easel' is what the user sees (or hears) at the user interface. In WPs, it is textual, not graphical, in form. This art-related term 'easel' is chosen to reflect two notions: holistic access and active engagement.

Just as the artist is aware of the whole even when focusing on a tiny part, so should the user be. The UI should encourage, not hinder, such holistic awareness, and let the user access any part with minimum cognitive effort. This implies a large easel within immediate reach. In Istar, this is achieved by a large, smooth-scrolling screen. In Doom-like games the main screen shows a limited view, but this easel is 'enlarged' to a more holistic awareness by provision of a map and by sounds. In WPs holistic access is via markers in the text. But windowing systems can tend to parcel up the detail separate from the whole, which reduces proximality.

Just as the artist not only views and examines the easel, but is continually making modifications, so the easel of PUI is not just a representation but a stimulation of thinking, and not just a stimulation but an invitation to make changes at the very instant of the stimulated thought. The user is 'situated' in the easel. So, in the design of the PUI, there must be no barrier between viewing and action. The page of a WP is a good example of this: as you read you type, without having to think about it.

2.4 Principle of Visual Cues

Visual cues (or auditory or tactile) are those indicators of relevant meaning that occur on the easel. Formal cues (e.g. colour codes) are mediated by the software. But informal cues are often more important - such as the pattern of lines around a box in Istar, the size of a paragraph and the shape of its ragged right hand edge in text. As the user draws, types or plays, such cues are left both on the easel and in the user's memory, forming a direct link lexical phenomena with the semantics in the user's mind. Visual cues should be encouraged and respected but desire for tidiness often works against them. Automatic layout and tidying facilities, common in visual programming software, destroy them. Icons all the same size in neat array, as on Windows, are less easily located than those with gross visual differences (Byrne, 1993). However, in computer games, disguise is important so cues should be such as to mislead.

2.5 The Clutter Principle

Clutter is lexical phenomena that distract from, rather than support, the meaning in the application. The clutter principle recognises that clutter will always occur when the user is engaged with ill-structured knowledge of non-trivial size, and thus demands proximal mechanisms to prevent and manage it. In box and arrows diagrams, congestion is one type of clutter and in Istar it is first minimized by a visually simple genre and then managed by making it very easy for the user to move things around and make space (see Basden, et. al. 1996). In WPs, clutter is of other sorts, such as unformatted paragraphs, spurious characters and bits of <html> that litter our emails, and semantic clutter, where the content of what is written is not well structured. These are managed by automatic formatting and spelling, grammar and style checkers. In computer games, clutter might be employed to mislead.

2.6 Principle of the Wide Input Channel

When we consider user action, we must choose lexical gestures for each semantic operation, for example, to create a new item (Istar), move along a corridor (game), start a new paragraph (WPs). In each type of application, many different operations will be needed because of the diversity of meaning in real life. But a two-button mouse allows only three operations - a very narrow lexical input channel. This principle says that we must give careful consideration to how we widen the lexical input channel. In street-fighter games complex gestures are used. In WPs, a multi-key keyboard is used. In Istar the free (non-mouse) hand presses keys to qualify the operation and different parts of the visual object give different operation.

2.7 Principle of Graded Effort

But remembering complex gestures, qualifying keys or the significance of different parts can result in extra cognitive effort or interrupt flow of thought when compared with the simple mouse operations of move, click or drag. The principle of graded effort recognises this, and says that we must carefully design each operation so that those operations that 'deserve' less effort will have it. Usually the most frequent operations should incur least effort, and thus mapped to the simplest click or drag. (Note that, since in each type of application different types of operation will be more common, this principle goes against established norms of implementing a standard 'look and feel' across all types of application.)

2.8 The Danger Principle

The danger principle modifies the previous principle by saying that operations that are more dangerous 'deserve' greater cognitive effort, and so should be designed to be more distal. But danger is of different forms. In Istar deletion is dangerous, so is made distal with an 'Are you sure?' dialog box (see Basden & Brown, 1996). But a game that kept asking 'Are you sure?' before an enemy was killed would fall into disuse! In games, danger has a different meaning.

2.9 The Principle of Tentative Action

Tentative action is user action that might not have a significant semantic result but is an important part of the user's thinking processes. Examples include 'doodling' while the user thinks something through, or the setting down of fleeting thoughts before they are forgotten. The principle of tentative action says that good, proximal support should be provided for such actions in the

software. In Istar, fleeting thoughts are recorded simply by creating a new box anywhere near. In WPs, they can be written down. Doodling was observed in Istar (see Basden, et. al., 1996): the user picked up a box with the mouse and wiggled it about. Doodling is not usually supported in WPs, but maybe should be. Both types are seldom necessary in games.

3 Guidelines for Freedom in PUI

We can see that the principles of proximal user interface may be applied to very different types of software, taking different forms in each. We have noted how some of them run counter to attempts to standardize look and feel, as they support a diversity of lexical-semantic mappings. But, as discussed in Basden (2000), this approach might, in the end, lead to a richer, more natural type of standard. The main thread running through this has been seeing freedom as the ability to respond appropriately to diversity, a notion derived from the philosophy of Dooyeweerd (1955). Reviewing the discussion above, we can see that much of the flexibility with which the principles are applied is influenced by Dooyeweerd's notion of aspects. Most pieces of software are aimed at aiding human tasks in which certain aspects predominate. It is these aspects that should guide the design of the UI, if we wish it to be proximal. We suggest that high quality guidelines, that will enable rather than hinder freedom, may be constructed by a knowledge of the relevant aspects: the analytical for mind-mappers, the lingual for word processors, and the spatial for games.

References

Basden, A., (2000). Guidelines for a 'Proximal' User Interface. In J. Vanderdonckt, C. Faraenc (Eds.) *Tools for Working with Guidelines* (p.339-56). Springer-Verlag.

Basden, A., Brown, A.J., (1996). Istar - a tool for creative design of knowledge bases. *Expert Systems*, 13, (4), 259-276.

Basden, A., Brown, A.J., Tetlow, S.D.A., Hibberd, P.R., (1996). Design of a user interface for a knowledge refinement tool. *Int. J. Human Computer Studies*, 45, 157-83.

Basden, A., Hibberd, P.R., (1996). User interface issues raised by knowledge refinement. *Int. J. Human Computer Studies*, 45, 135-55.

Basden, A., Wood-Harper A.T., (2002). A philosophical enrichment of CATWOE. *ANZSYS'2002*, Australia and New Zealand Systems Conference, 10-12 Dec 2002, University of Sunshine Coast.

Byrne, M.D., (1993). Using icons to find documents: simplicity is critical. *Proc. InterCHI '93*, April 1993, 446-453, ACM, USA.

Dooyeweerd, H., (1955). A New Critique of Theoretical Thought, Vol. HV., Jordan Station, Ontario: Paideia Press.

Greeno, J., (1994). Gibson's Affordances. *Psychological Review*, 101, 336-42.

Lyytinen, K.J., Klein, H.J., (1985). The critical theory of Jurgen Habermas as a basis for a theory of information systems. In E. Mumford, R.A. Hirschheim, G. Fitzgerald, A.T. Wood-Harper (Ed.), *Research methods in information systems* (p.219-31), North Holland.

Polanyi, M., (1967). The Tacit Dimension. London: Routledge and Kegan Paul.

Shneiderman, B., (1992). Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison Wesley.

Wilson, F.A., (1997). The truth is out there: the search for emancipatory principles in information systems design. *Information Technology and People*, 10, (3), 187-204.

It's all in a days work of a software engineer

Inger Boivie[1], *Jan Gulliksen*[1] *and Bengt Göransson*[1,2]

[1]Dept. for IT/HCI, Uppsala University, PO Box 337, SE-751 05 Uppsala, Sweden
[2]Enea Redina AB, Smedsgränd 9, SE-753 20, Uppsala, Sweden
Jan.Gulliksen@hci.uu.se

Abstract

This paper reports on two studies that have been performed in order to explore if and how software engineers make design decisions that affect the usability of the software. Our conclusion is that the main goal for software engineers is to produce program code. They consider usability being the responsibility of others, for instance, the users or usability experts. They do not feel that their code contributes to the overall usability of the software. Nevertheless, software developers make numerous decisions that affect usability as part of their work with analysis, design and programming. The conclusion one must draw from this is that there is a need for a strategic change of attitude towards usability among software engineers.

1 Introduction

Despite the attention usability[1] has received in the last number of years, poor usability is still a major problem for users. According to Clegg, et al. (Clegg, Axtell, Damodaran, Farbey, Hull, Lloyd-Jones, Nicholls, Sell and Tomlinson, 1997) usability issues are on the agenda in most IT development projects. The respondents in their study estimate that as many as 60-70 % of the projects addressed usability matters "successfully". They go on to say, however, that there is criticism that the area is still not sufficiently understood, and that the view of usability is rather mechanistic. This indicates that we need to know more about how usability is addressed in software development projects. Not least, we need to know how software engineers think about and work with usability.

We have conducted a couple of studies of how software engineers work, with particular focus on if and how they address usability issues. If we can understand more about how the software engineers work, we may be able to identify new tools for addressing usability in software development or improve existing ones.

2 Method

In order to understand what type of support software engineers need to develop usable systems we must study and analyse their work, i.e. their work practices, and how they think about their work. It is difficult to analyse work practices since most workers, including software engineers, are not fully aware of what they do. Their work practices are partly based on tacit knowledge and cannot

[1] We use the ISO 9241-11 (1998) definition where usability is defined as follows:
"The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use."

be described in all detail. Work studies must therefore be based on observations of the workers, combined with interviews.

The first study consisted of a series of *observation interviews* (Åborg, Sandblad, Gulliksen & Lif, 2003) with 7 software developers (4 men and 3 women). The software developers worked in an in-house development organisation with designing and developing user interfaces. The respondents were software engineers, programmers, project managers and process owners at the IT department of a large government organisation. Most of them worked in Stockholm but a few worked in sub-departments in other towns around Sweden.

This interview study was part of a project developing an on-line interactive corporate style guide (Olsson and Gulliksen, 1999). The main purpose of the study was to capture user requirements and the software developers were interviewed as potential users of the style guide. The project worked in accordance with the human-centred approach described in ISO 13407 (ISO, 1999). ISO 13407 prescribes activities for acquiring a good understanding of the users and their context of use, in this case the software developers and the software development process.

The second study consisted of a series of open-ended interviews with eight people (six women and two men) involved in software development projects in two in-house development organisations (one of which was the same as in the first study). The respondents had backgrounds in and worked with usability, user interface (UI) design and/or software development. The purpose of this study was to investigate how design decisions, affecting usability, are made. Who makes them, when and on what grounds?

3 Results

The two studies describe the software development process on two different levels. In the first study, we identified how software engineers work on a day-to-day basis, solving particular problems. Somebody, for instance, the project manager, typically hands down the design problem to them. Each design problem can usually be solved within two days. The result of the task is a piece of software that is delivered for integration with the rest of the system.

The observation interviews indicate that the developers could not fully understand the result of their programming task. The quality of their work was determined by the robustness and stability of the program code, rather than usability. The developers did not feel that they designed user interaction. The design simply emerged as if by magic, as a result of the integration of the various pieces of code produced by the developers. Nobody admitted to having designed a particular part of the user interaction. The below figure illustrates the workflow for the software developers.

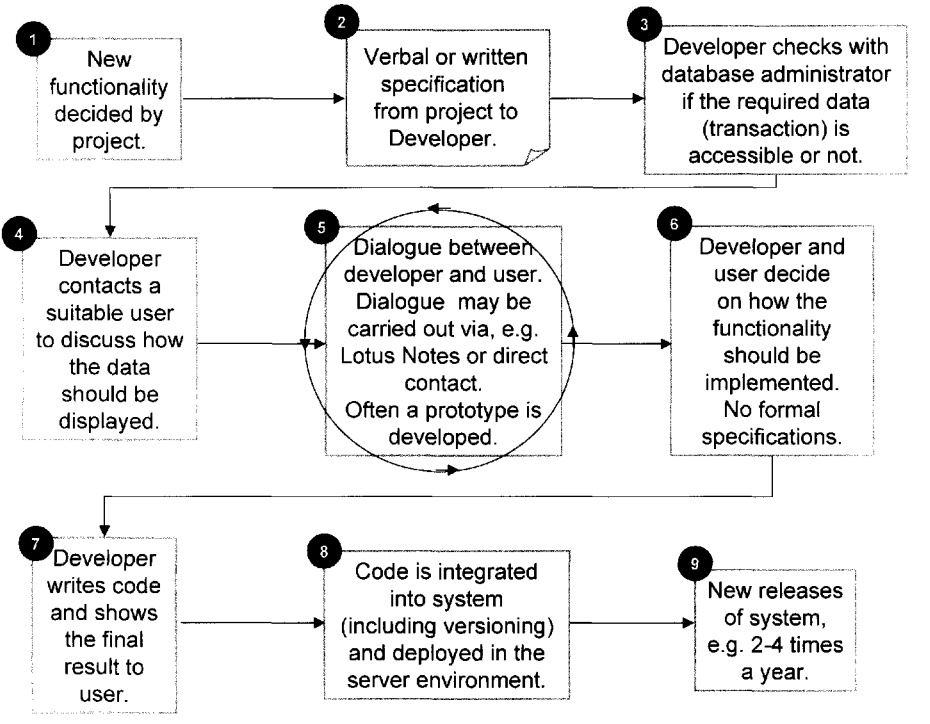

Figure 1: Model of how developers work. The model is a general model, created after studying several developers in one in-house development organisation.

In the second study, the respondents described the software development process. The typical project starts with some kind of requirement specification or description from the client. Requirements are captured by means of use case modelling and expressed mainly by use case models Users and/or domain experts[2] participate in the requirements capture process. The user interface modelling is done in parallel with the use case modelling, with the same group of users and domain experts. The use case models are then refined to analysis models and design models. The user interface developers/use case specifiers develop a solution, which is reviewed by the users and domain experts, and modified accordingly. The design solutions are often evaluated or reviewed by users a few times, before they are implemented. The number of iterations/reviews is determined by the date the implementation must start.

The focus of the interviews in the second study was design decisions. The respondents were asked what kind of decisions they are allowed to make, and whether design decisions are made by them or by other people.

When asked who makes the design decisions, none of the respondents said "I do". Instead, they pointed to the user representatives and the client representatives. According to several of the respondents the usability of the resulting system depends to a large extent on the user representatives. Quite a few of the design decisions are made in workshops (modeling sessions)

[2] Domain experts are representatives from the user organisations with expertise in the domain, e.g. laws and regulations affecting the new system. These people will not necessarily be active users of the system.

together with the user representatives. There are also matters over which the development teams do not have control, despite the fact that these matters are likely to affect the perceived usability of the system.

The use cases are the most important information carrier in the projects. They are the basis of the interaction design and thus limit the design "space". They specify, for instance, the workflow, what information should be displayed when, how the information should be grouped, etc. The respondents in both organisations reported difficulties with the use case modelling. One respondent said that the people in her project had misunderstood the concept, resulting in far too many, far too detailed system operations instead of real use cases. *"I still don't know what a use case is."*

4 Discussion

The two studies gave us a rich picture of the problems involved in addressing usability issues in the software development process

The development process often fails to provide the necessary support for addressing usability
The two descriptions of how software developers work differ on some point, mainly regarding the extent of formalisation. Figure 1 shows a rather informal process, where contacts with "a suitable" user are made by the individual software developer whenever he/she feels the need. In the process described in the second study, users participated on a more formal basis, in modelling sessions and by reviewing specifications, etc. The differences may, at least to some extent, ascribed to the different data collection methods used in the studies. The first study is based on observations whereas the other is based on interviews. When interviewed, the developers recount what they think they do at work, which is probably fairly close to the official version of how they should work, i.e. the software development model they are supposed to comply with. In an observation, you can see what they really do at work. These two views of work correspond to an "explicit" view and a "tacit" view as described by Kuhn (1996).

Software development work is often thought of as engineering-oriented, rational and described by means of workflow models consisting of well-defined steps. But our results indicate that certain parts of it are based on tacit knowledge, which is not easily captured and expressed in models. Development work also seems less formal, less ordered and less rational than we would expect from the "explicit" view of it.

The two pictures of software development do not have to be contradictory. Some of the respondents in the second study also contacted users on an informal basis even though they also described formal meetings and contacts. Thus, there is a need for quick, informal contacts with users in software development. This indicates that the models used to capture the requirements and other inputs are insufficient. They do not provide enough information for the software developers to get on with their work. One interesting question is, what information is missing.

Decisions affecting usability just "happen" as part of the process instead of being "made"
The software developers, and basically everybody else in the project, make usability decisions without being aware of it. These decisions encompass all levels from what is to be included in a certain release (i.e. what the user will be allowed to do in the system, in terms of tasks or functionality) to the use case modelling, to detail decisions about input/output controls.

The respondents claimed that the user representatives and domain experts in the projects were responsible for most of the interaction design decisions. Nevertheless, we would like to argue that the developers make a lot of the decisions. This phenomenon may be explained by the fact that in many cases design decisions just "happen" as part of a process, where the design slowly grows and develops. Design decisions are explicitly made in those cases where a design issue has been discussed, for instance, when not complying with the style guide or when there has been disagreements.

Frustration with user involvement
The respondents in both studies expressed some frustration with how users were involved, regardless of whether it was on an informal or formal basis. There was a great reluctance towards making contacts with the users among most of the developers. They did not consider it their responsibility to involve users. Nevertheless, they did contact users when they felt they had to, as described above. The developers also expressed frustration with the disproportionate influence some users get (for instance, the formally appointed user representatives).

In order to facilitate a focus on usability and the users' real needs in software development, we need to:

- Shift the attitudes towards a more user-centred design and development process (UCSD), and educate or train all software developers in UCSD.
- Make usability activities become Standard Operating Procedures within the organisation.
- Make management aware of their obligation to put usability on the agenda in development work
- Provide success stories that can serve as a examples for successful deployment of UCSD
- Make commercial systems development processes support and prescribe usability activities.
- Resolve the conflict between usability and deadlines.

References

Clegg, C. Axtell, C. Damodaran, L. Farbey, B. Hull, R. Lloyd-Jones, R. Nicholls, J. Sell, R. Tomlinson, C. (1997). Information technology: a study of performance and the role of human and organizational factors. *Ergonomics*, Vol. 40. No 9. pp. 851-87

Kuhn, S. (1996). Design for people at work. In Winograd, T. *Bringing design to software.* ACM press. Addison Wesley. New York. pp. 273-289.

ISO 9241-11 (1998), Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs). Part 11: Guidance on Usability. International Organization for Standardization, Geneva.

ISO 13407 (1999), Human Centered Design Processes for Interactive Systems. International Organization for Standardization, Geneva.

Olsson, E. & Gulliksen, J. (1999) A corporate style guide that includes domain knowledge. In G. Salvendy, M.J. Smith, & M. Oshima (eds.) International Journal of Human-Computer Interaction. Vol. 11, No. 4, Ablex Publishing Corporation, Norwood, New Jersey.

Åborg, C., Sandblad, B., Gulliksen, J. & Lif, M. (in press) Integrating work environment considerations into usability evaluation methods – the ADA approach. Accepted for publication in Interacting with Computers.

Semiotic Conference: Work Signs and Participatory Design

Rodrigo Bonacin

Institute of Computing, State University of Campinas, Caixa Postal 6176
13083 970 Campinas, SP - Brazil
+55 19 32958178
ra000470@ic.unicamp.br

M. Cecilia C. Baranauskas

Institute of Computing, State University of Campinas, Caixa Postal 6176
13083 970 Campinas, SP - Brazil
+55 19 37885870
cecilia@ic.unicamp.br

Abstract

In this paper we propose a participatory design technique named Semiotic Conference as a way to explore process-oriented issues related to the social context of the workplace and its connections with aspects of the product being designed. This technique was proposed to enable a design process in which workers (final users) and designers construct a new organizational context embedding the system in it. The software is a product of their social interaction and a new social context appears as a consequence of the software design and use. Results of applying the proposed technique to a real context are discussed.

1 Introduction

Literature in Software Engineering and Information Systems (Floyd, 1987; Muller et al 1998; Mahemoff & Johnston, 1998) has shown two basic perspectives to software design: the product-oriented, which focuses on the software artifact, and the process-oriented one, which focuses on the human work processes that the computer artifact is intended to support. In agreement with Floyd (1987), in this paper we argue that these approaches could be complementary to each other when considered in the proper balance. According to Floyd, the process-oriented perspective views the software in connection with human learning, work and communication, taking place in an evolving world of changing needs.

The close contact between designers and users, and the direct participation of the users during the system design are pointed out as way to promote the process-oriented design. In this paper we propose a Participatory Design (PD) technique named *Semiotic Conference* as a way to explore process issues from the social context of the human workplace and their connections with aspects of the product being designed. The *Semiotic Conference* was proposed in order to promote the participation and signification aspects (Baranauskas et al, 2002) during the design process.

This new PD technique enables a design process in which workers (final users) and designers construct a new organizational context embedding the system in it. The software is a product of their social interaction and a new social context appears as a consequence of the software design and use. This technique was motivated by the concept of "mutual learning" in design defined by Kyng (1991), where the application domain practitioners learn about the new possibilities brought about by the technology while designers learn about the application domain. The mutual learning is promoted through the participation of the users during the design using a semiotic model that

addresses at the same time the application being designed and the intended social context that the application is supposed to support.

The paper is organized in the following way: Section 2 presents the theoretical background based on the Participatory Design approach and methods from Organizational Semiotics, Section 3 describe the proposed technique, Section 4 shows results of using the technique in the design of a CSCW system interface, and discuss the drawbacks and direct benefits of using it, and Section 5 concludes.

2 Participatory Design and Organizational Semiotics

The *Semiotic Conference* is based on two main theoretical backgrounds: the Participatory Design, which promotes the participation of the user during the design and the Organizational Semiotics that enables to model the signs system at the social level. The Participatory Design approach was developed initially in the Scandinavia and it employs a variety of techniques to carry out design with the user, rather than for the user (Muller et al 1997). The Participatory Design approach stresses the importance of democracy in the workplace to improve the work methods, the efficiency in the design processes (with the users backgrounds and feedback), to improve the systems quality, and "to carry on" formative activities.

In order to develop a common view of both the technology and the organization, that is necessary for users and designers to explore new organizational structures, we combined the use of PD with Organizational Semiotics (OS) methods in a new PD technique. OS is defined as "the study of organization using the concepts and methods of Semiotics" ("OSW," 1995). The study of the organization is based on the observation that people affect all organized behaviors through the communication of signs, individually and in groups.

A set of methods proposed by Stamper (1973) enables to study the use of signs in organizations and their social effects. The two main methods of the OS used in the *Semiotic Conference* are the Semantic Analysis and the Norm Analysis. Semantic Analysis (SA) focuses on the agents and their pattern of behaviors to describe the organization. Some basic concepts of SA adopted in this paper based in Liu (2000) are: (a)"The world" is socially constructed by the actions of agents, on the basis of what is offered by the physical world itself, (b)"Affordance", the concept introduced by Gibson (1979) can be used to express the behavior of an organism made available by some combined structure of the organism and its environment, (c)"Agent" can be defined as something that has responsible behaviour, an agent can be an individual person, a cultural group, a language community, a society, etc., and (d)"An ontological dependency" is formed when an affordance is possible only if certain other affordances are available.

A norm analysis is usually carried out on the basis of the result of the semantic analysis (Liu 2000). At the pragmatic level the norm analysis describe the relationships between an intentional use of signs and the resulting behaviour of responsible agents in a social context. At the social level it describes beliefs, expectations, commitments, contract, law, culture, as well as business.

3 The *Semiotic Conference*

In this paper we describe the proposed technique in terms of the attributes proposed by Muller (1998) for Participatory Design techniques: the object model, the process model, the participation model, expected results and position of the technique in the whole product life cycle.

Object Model: Copies of Organizational Semiotic Models: ontology charts (semantic analysis) and norm analysis (pragmatic and social level analysis), prototype screen shots, and pens are the material used for running the participatory practice. The ontology charts are the graphic representation of relationships among concepts of the semantic analysis. The ontology charts represent agents (circles), affordances (rectangles), ontological dependencies (lines from left to right), role-name (parentheses) and whole-part (dot). The norm analysis are represented by Deontic logic expressions with the following basic form: <Norm>::= If <Condition> then <D> <Agent> <Action> Where <D> is a deontic operator that specifies that the action is obligatory, permitted or prohibited. The norms are not necessarily obeyed by all agents in all circumstances; they are social convention (laws, roles and informal conventions) that should be obeyed. A more detailed description of ontology charts and norm modelling can be found in Liu (2000).

Process Model: In the format of a Conference the designer shows concepts of the workplace modeled in ontology charts that include the affordances made available or/and that would made available by the system. This model contains the concepts compiled by the designers during the initial definition (the diagram must contain only the terms used by the users) of the system obtained from application of other PD techniques or from the last model of the former prototype. Copies of the models are distributed to all practitioners and if necessary the designer clarifies the notation and concepts described by the diagram. For each concept quoted in the model that any person of the group judge important, the practitioners discuss the semantic dependencies with other concepts and the social norms associated. Members of the group, as a result of discussion, propose changes in the semiotic models. Alternative solutions are raised on prototype screen shots to reflect the conceptual changes in the model. The changes in the prototype can be directly drawn in the prototype screen shots or, if necessary, other PD techniques can be used to define them.

Participation Model: Members from different levels and functions in the organization together with the design team participate in the Conference. Facilitators mediate the interaction among the group.

Results: Semantic and Norm models constructed during the conference with people from the organizational context are produced. The establishment of social norms that will be applied at the workplace considered with the prospective application is also produced as a result of the technique. The user interface is also produced through the "mutual learning" promoted by the technique.

Position in the product life cycle: The technique applies iteratively to several phases of the design life cycle: Problem Identification & Clarification, Requirements & Analysis, High-Level Design, Evaluation and Re-Design (after the creation of the first prototype).

4 Results of Using the Technique

The proposed technique was explored during the design of Pokayoke: a system prototype constructed in the context of a partnership between our Institution and an automotive industry, for supporting corrective and preventive actions related to problem solving in the factory (Bonacin & Baranauskas, 2003). This technique was applied to five iterations of the prototype cycles during the Pokayoke design: at the starting of each iteration, in order to review the concepts of the former prototype and to redirect the design of the next prototype, and at other moments when necessary for reviewing concepts. Figure 1 shows an example of using the *Semiotic Conference* during the Pokayoke design. It refers to concepts in one of the steps of problem solving process (Step IV). Figure 1a (left) presents part of the ontology chart for the second iteration of the prototype. It shows the drawing made by a practitioner suggesting changes in the semantic model (He proposed

to include Brainstorming in this step). Figure 1b (right) shows the ontology chart of the third prototype including the changes proposed by the practitioner.

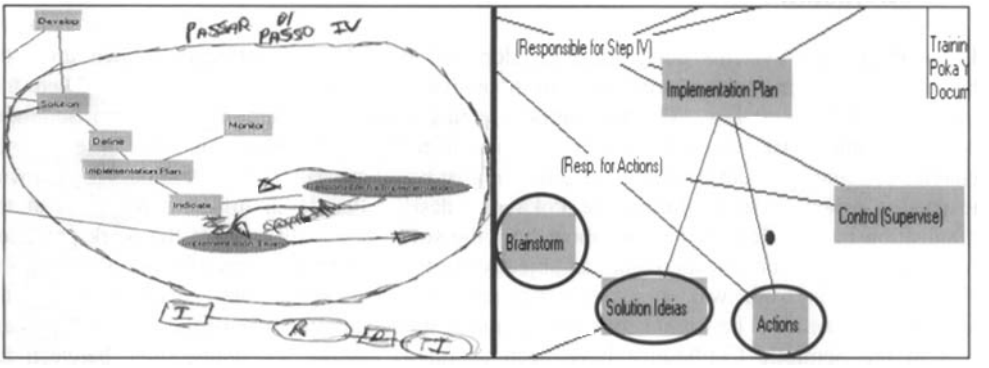

Figure 1(a and b): Changes in Ontology Charts

Figure 2 shows changes in the user interface caused by the alterations in the ontology charts presented in Figure 1. The affordances marked with circles in Figure 1 are associated to the interface constructions in Figure 2. The ontological dependencies of the objects are represented by the conditions of enabling or disabling functions in the software system; for example, it is necessary to exist a Multifunctional Team to enable the Brainstorming functionality. A norm interpretation package was developed to interpret formal norms (rules of the organization) that specify the responsibility, duties and permissions in the problem solving method. These norms specify the permission to access some functions of Pokayoke, for example: any user is allowed to include an idea in a brainstorming session but s/he is not allowed to specify actions to correct the problem. These norms also specify who could be responsible for which task at different phases of the problem solving processes. Changes in the norms specifications proposed during the *Semiotic Conference* are reflected in attributes configured by the norm interpretation module.

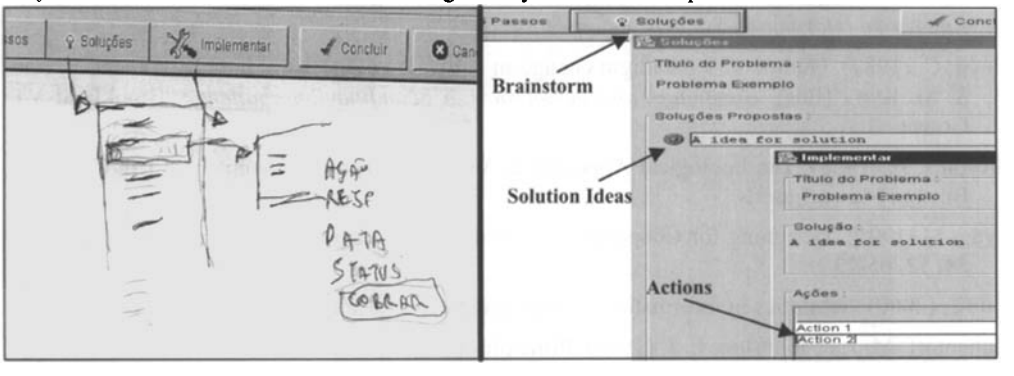

Figure 2(a and b): Changes in the system interface

The fifth prototype of Pokayoke has substituted the paper-based form of the "five steps" process. A group of seven workers of different levels and areas at the organization have participated in the *Semiotic Conferences*. These workers were in charge of training the other users of the system showing evidence of the mutual learning occurred. The mutual learning allied with the feeling of authorship was fundamental to the system acceptance. The workers have expressed this feeling many times during the system design and use, such as: *"... we have defined this in this way to avoid..."*. The main drawback of the proposed technique noticed during the Pokayoke design was the necessity of explaining the semantic and norm analysis concepts at the first applications.

5 Conclusion

Methodologies for systems design and development have been traditionally drawn upon the objectivist paradigm, which considers an objective reality to be discovered, modeled and represented in the software. On the contrary to this assumption, the Organizational Semiotics adopt the subjectivist paradigm, which understands reality as a social construct based on the behavior of agents participating on it. The subjectivist paradigm has shown usefulness to deal with the process-oriented issues during the Pokayoke design. People in different positions at the organization have different opinions about how the software should support the work practices. Therefore the approach considered not an objective truth about the best way of supporting problem solving, but this truth was a social construct based on negotiations occurred during the participatory practice. The formalism of the semiotic models and the use of the prototype screen-shots in the Semiotic Conference have been useful to explore the connections between the Process-Oriented and the Product-Oriented issues in the software design.

Acknowledge

This work was partially supported by grants from Brazilian Research Council (Capes 2214/02-4, Cnpq 301656/84-3, Fapesp 2000/05460-0). The authors also thank Delphi Automotive Systems in Jaguariúna, Brazil, and Nied-Unicamp for collaboration and partnership in the Pokayoke project.

References

Baranauskas, M. C. C., Bonacin, R. & Liu K., 2002. Participation and Signification: Towards Cooperative System Design. in *Proceedings of V Symposium on Human Factors in Computer Systems*, 3-14.

Bonacin, R. & Baranauskas, M. C. C. (2003) Designing Towards Supporting and Improving Co-Operative Organisational Work Practices, in *Proceedings 5th International Conference on Enterprise Information Systems*, to appear.

Floyd, C. (1987). Outline of a paradigm change in software engineering. In G. Bjerkners. P. Ehn. & M. Kyng (Eds), *Computers and democracy: A Scandinavian challenge*. Brookfield VT: Gower.

Gibson,J. J., (1968). The Ecological Approach to Visual Perception. Houghton Miffin Company. Boston, Massachusetts.

Kyng, M.(1991). Designing for Cooperatin: Cooperating in Design. *Communications of the ACM* 34, 12, 65-73.

Liu K., (2000). Semiotics in information systems engineering, Cambridge University Press.

Mahemoff, M. J. & Johnston, L. J. (1998). Principles for a Usability-Oriented Pattern Language In Calder, P. & Thomas, B. (Eds.), *OZCHI '98 Proceedings*, (pp.132-139). Los Alamitos, CA.

Muller, M. J., Haslwanter, J. H., & Dayton, T. (1997) Participatory Pratices in the Software Lifecycle. In M. Helander, T. K. Landauer, P. Prabhu (eds.), *Handbook of Human-Computer Interaction*, (pp. 255-297), Elsevier Science, 2 ed.

Muller, J. M., Matheson, L., Page, C. & Gallup, R. (1998) Participatory Heuristic Evaluation. *Interactions*. september + october 1998. V. 5, Issue 5, ACM Press, 13-18.

OSW, (1995). *The circulation document in the Organizational Semiotic Workshop*, Enschede.

Stamper, R. K., (1973). Information in Business and Administrative Systems. John Wiley & Sons.

Creative Design of Interactive Products and Use of Usability Guidelines – a Contradiction?

Michael Burmester

University of Applied Research – Hochschule der Medien
Wolframstr. 32, D-70191 Stuttgart
burmester@hdm-stuttgart.de

Joachim Machate

User Interface Design GmbH
Teinacher Str. 38,
D-71634 Ludwigsburg
joachim.machate@uidesign.de

Abstract

Often the use of guidelines for the design of interactive products is criticized for several reasons. A major point of criticism is that guidelines are not flexible enough, so that the design can be adapted to the context of use. Furthermore, there is an anxiety that guidelines hinder creative design solutions and innovations. In the following paper, the strategic use and definition of guidelines associated with a user centred design process (UCDP) is proposed.

1 Guidelines

Guidelines have been used since the beginning of human-computer interaction (HCI). The main goal of guidelines is to increase the usability of interactive products when designing them. Guidelines are a way to put theories, empirical findings or good practice in to a form which is intended to be useful for and adapted to the process of designing interactive products.

Two main categories of guidelines in HCI can be defined: design guidelines and process guidelines (Stewart & Travis, 2002). Design guidelines are providing support for design decisions and process guidelines to plan and structure design processes.

Design guidelines are differentiated by Stewart and Travis (2002) between guidelines, standards and style guides. According to Stewart and Travis, guidelines can be seen as recommendations of good practice relying on the credibility of their authors. Standards are "formal documents published by standard making bodies that are developed through some form of consensus and formal voting process" (Stewart & Travis, 2002, p. 992). Finally, style guides are "sets of recommendations from software providers or agreed within development organizations to increase consistency of design and to promote good practice within a design process of some kind" (Stewart & Travis, 2002, p. 992). This categorisation is based on the question, who has generated these guidelines.

Another possibility to categorize guidelines is along the degree of precision in guiding the design, and the degree of freedom in interpretation for taking design decisions. We have identified five categories: principles, rules, interaction patterns, standard interactions and user interface building blocks.

(1) Principles or heuristics (Nielsen, 1993) describe basic usability aspects which are important when designing user interfaces. In order to apply principles to design, the designer has to understand the rationale behind the principles. For all design decisions, the meaning of a principle for a specific design situation is a matter of interpretation. Examples for principles are the dialogue principles for visual display terminals described in the ISO 9241 part 10, the ten usability heuristics of Jakob Nielsen (1994).Ben Shneiderman's (1998, pp. 74-75) eight "golden rules" of dialogue design, or Alan Cooper's principles for polite software (Cooper, 1999, p. 162). All of these are principles in the sense of our definition.

(2) Rules focus on specific design decisions, e.g. if upper and lower case should be treated equivalent in search dialogues or not (Smith & Mosier, 1986, cit. Steward & Travis, 2002, p. 995). It is up to the designer to decide whether a particular rule is relevant and must be applied for the design decision under discussion. Interpretation is still necessary.

(3) Interaction patterns (Tidwell, 1999; van Welie, van der Veer & Eliëns, 2000) describe a complete context of an interaction and do not focus on a single design decision. Usually, interaction patterns are specified in a specific language. van Welie, van der Veer & Eliëns (2000) are describing for each pattern the name of the pattern, the usability principle behind it, the context in which the pattern should be applied, the force influencing the user, the design solutions within the pattern, examples where the pattern has been applied already, the usability impact, rationale behind the pattern and known uses (e.g. products using the described interaction pattern). Example interaction patterns are editor dialogues, browsers, warning message dialogues, wizard dialogues etc. (for more see Tidwell, 1999 or van Welie, 2002). The designer has still freedom in interpretation and in finding own solutions, but support is provided for a complete interaction task.

(4) Standard interactions are fully defined with regard to the way which steps need to be taken in order to accomplish a particular input task, e.g., single selection from a list There is no possibility left for interpretation of the dynamic input and output structure. These structures are formally described formalisms (e.g. in flow chart).Although the way of interaction is fixed, a user interface designer still has the freedom to define the visual presentation and style. An example for a description of standard interactions are the FACE guidelines for the design of electronic home devices (Burmester, 1997).

(5) User interface building blocks are representing interactions which are completely defined with regard to their interaction and sensual, e.g. visual and acoustic, presentation (Görner, Burmester & Kaja, 1997). No freedom of interpretation is left to the designer. Building blocks are sometimes also referred to as idioms (e.g. Sun Microsystems, 2001, pp. 65). Only selecting the right building block for a specific user task and setting the right options of the building is required. Building blocks are described in style guides (e.g. Microsoft Windows XP, 2002).

2 Problems with Guidelines

It is quite common that problems are reported in properly applying guidelines for a specific design situation(e.g. Görner, Burmester & Kaja, 1997). One reason for that is that there is an inflation of guidelines, which makes a designers' live quite hard: Smith & Mosier (1986, 944 HCI guidelines)[1], Brown (1988, 302 HCI guidelines)[1], Mayhew (1992, 288 HCI guidelines)[1], Burmester (1997, ~400 CE guidelines), ISO 9241 12 – 17 (1996-1999 ~500 HCI guidelines).

[1] cit. Nielsen (1993)

Nielsen Norman Gr. (2002, ~600 Web guidelines), etc. Furthermore, guidelines are difficult to understand, difficult to interpret, often too simplistic or too abstract, can be difficult to select, can be conflicting, often have authority issues concerning their validity (see Spool, 2002) or are made for a specific contexts (e.g. specific input output technology, tasks, user groups) which are not obvious (van Welie, van der Veer & Eliëns, 2000).

3 Guidelines as part of the user centred design process

3.1 Guidelines as input for user centred design

ISO 9241-11 formally defines usability as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". In other words, usability describes the fit between products, users, tasks and environment. In order to achieve a good quality of use, or usability, there is a quite common understanding in the international community of usability experts how this should be achieved. It is the user centred design process (UCDP). Several more or less detailed process models are available (e.g. ISO 13407, 1999; Mayhew 1999; IBM, 2002; Rosson & Carroll, 2002). Across the different process models, four central phases of a UCDP can be identified: a) analysis of context of use, b) designing, c) prototyping, and d) evaluation of the user interface.

All process models focus on the three main aspects of user interface design defined by Gould and Lewis (1985): early focus on users and tasks, empirical measurement and iterative design. The iterative process runs until a defined criteria derived from user requirements is achieved (for further details see ISO 13407, 1999).

Guidelines of the different levels described above are an important input for a user centred design process. Derived from project experience and some research literature (e.g. Rosenzweig, 1996) we see the involvement of guidelines in the following UCDP phases.

(1) Context of use analysis: When having finished the context of use analysis the relevant principles for the quality of use for a product can be selected and prioritized. Often it turns out that not all available principles have the same importance. Basic principles like "suitability for the task" (ISO 9241-10) are of central importance in most cases, but e.g. "suitability for individualisation" (ISO 9241-10) has not always top priority. (2) Design phase: The high prioritised principles should guide the process of finding design solutions. The first step in the design phase is to design the main views of dialogues according to the actions of the important and frequently performed tasks of the identified user roles (e.g. Rosson & Carroll, 2002). Here rules and interaction patterns are important input for designing information views. The second step in the design phase is to design single interactions. Here rules, interaction pattern, standard interactions and user interface building blocks can be used as input for design solutions and supporting design ideas. (3) Prototyping phase: Some prototyping tools provide user interface building blocks, e.g. as libraries, which can be used for developing a prototype of the user interface. Style guides help to ensure consistency with the underlying platform, and idioms provide building blocks which are already available for a particular look and feel. (4) Evaluation phase: In UCDP, usability testing is a crucial method to detect usability problems (Nielsen, 1995). Before running a usability test, quite often expert based evaluation methods, e.g. heuristic evaluation (Nielsen, 1993) are applied. For heuristic evaluation the selected and prioritized principles set in the context of use analysis phase can be used to guide the evaluation. The outcome of the evaluation phase is also be used to decide whether a further design iteration is

necessary. For this decision metrics are needed. The selection of such metrics (see ISO 9241-11) can be guided by the top prioritized principles, e.g. if learnability is important, it might be useful to measure the time and repetitions until a specific task has been carried out correctly.

3.2 Creativity, the user centred design process and guidelines

An important criticism concerning the use of guidelines is that innovations and creative ideas are hindered by guidelines. On the one hand it is clear that guidelines have the goal to decrease the number of possible design solutions in order to sort out solutions which are not usable or inconsistent. On the other hand, new design solutions which are more appropriate for the context of use might not be found. The idea behind that is that guidelines are treated as laws. In UCDP it is more appropriate to take guidelines as an input for the design process in order to help the selection of usable design alternatives. Design decisions can be seen as design hypotheses which have to proved during the evaluation phase. This holds also for guidelines. If certain guidelines, e.g. interaction patterns or even user interface building blocks have been used to construct a dialogue, it is also the guidelines which must be tested in the evaluation phase. In many cases it will turn out that the solution is good. Then it can be kept. In other cases. usability problems will be detected although the design was based on sound guidelines. In these cases the design and also the related guidelines have to be revised. Product design can also be seen as theory development (Carroll, Singley & Rosson, 1992). In this view the product is a theory on how the user can be optimally supported in a certain context of use. By continuous testing this theory is refined step by step until it covers the main requirements of the context of use.

3.3 Guidelines as output of user centred design

If a UCDP brings about design solutions which contribute to an improved quality of use, the design solutions can be formulated as new guidelines. Now, the underlying principles should be described and interaction patterns, standard interactions or user interface building blocks should be defined. These guidelines have proved their compatibility with the context of use in question. Now, guidelines can ensure that the design solutions developed in a UCDP are consistently applied during implementation. Style guides ensure consistent dialogues within a product or a product family or platform. Rosenzweig (1996) pointed out that it is important to explain the rationale of the guidelines to the development team and give them the chance to provide feedback from their point of view. Our experience is that development teams must have the possibility to discuss with the guideline authors upcoming questions and remarks (see also Gale, 1996).

4 Conclusion

We showed that treating guideline documents as living documents in the sense of Rosenzweig and providing possibilities for continuously testing the guidelines and making revisions and improvements will lead to a sustained increase of the acceptance of guidelines. By using and generating guidelines in the framework of user centred design creative design of interactive products and use of usability guidelines is NOT a contradiction.

References

Burmester, M. (Ed.). (1997). Guidelines and Rules for Design of User Interfaces for Electronic Home Devices. Stuttgart: IRB.

Carroll, J.M., Singley, M.K. & Rosson, M.B. (1992). Integrating theory development with design evaluation. Behaviour & Information Technology, 11[5], 247-255.

Cooper, A. (1999). The Inmates Are running the Asylum. Indianapolis, Indiana: SAMS Publ.

Gale, S. (1996). A Collaborative Approach to Developing Style Guides. In Proc. of the ACM Conference on Human Factors in Computing Systems (pp. 361-366). New York: ACM Press.

Görner, C., Burmester, M. & Kaja, M. (1997). Dialogbausteine: Ein Konzept zur Verbesserung der Konformität von Benutzungsschnittstelle mit internationalen Standards. In Tagungsband Software-Ergonomie 97 (pp. 157-166). Stuttgart: Teubner.

Gould, J.D. & Lewis, C.H. (1985). Designing for usability: key principles and what designers think. Communications of the ACM, 28[3], 300-311.

IBM (2002). IBM Ease of Use: What is User-Centered Design? Retrieved Feb., 5 2003, from www-3.ibm.com/ibm/easy/eou_ext.nsf/EasyPrint/2.

ISO 9241 (1996-1999). Ergonomic requirements for office work with visual display terminals (VDTs) Part 10, 12-17. Berlin: Beuth-Verlag.

ISO 13407 (1999). Human-centred design processes for interactive systems. International Organization for Standardization. Berlin: Beuth-Verlag.

Mayhew, D.L. (1999). The usability engineering lifecycle. A practitioner's handbook for user interface design. San Francisco, CA: Morgan Kaufmann.

Microsoft (2001). Windows XP - Guidelines for Applications. Retrieved February 5, 2003, from www.microsoft.com/hwdev/windowsxp/downloads/

Nielsen, J. (1993). Usability Engineering. Boston, San Diego: Academic Press.

Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. Proc. of the ACM. CHI'94 Conference (pp. 152-158). New York: ACM Press.

Nielsen, J. (1995). Getting usability used. In Proc. of Human Computer Interaction- INTERACT '95 (pp. 3-13). London: Chapman & Hall.

Nielsen – Norman – Group (2003). Retrieved Feb., 5 2003, from www.nngroup.com/reports

Rosenzweig, E. (1996). A Common Look and Feel or a Fantasy? Interactions, 5, 21-26.

Rosson, M.B. & Carroll, J.M. (2002). Usability Engineering – Scenario-based development of human-computer interaction. San Francisco: Morgan Kaufmann.

Shneiderman, B. (1998). Designing the User Interface: Strategies for Effective Human-Computer Interaction (3rd edn.). Reading Mass: Addison-Wesley.

Spool, J.M. (2002). Evolution Trumps Usability Guidelines. Retrieved Nov., 23 2002, from www.uie.com/Articles/evolution_trumps_usability.htm

Steward, T. & Travis, D. (2002). Guidelines, Standards and Style Guides. In J.A. Jacko & A. Sears (Eds.), The Human-Computer Interaction Handbook (pp. 991-1005). Mahwah: L.E.A.

Sun Microsystems (2001). Java Look and Feel Design Guidelines: Advanced Topics. Boston, Mass.: Addison Wesley.

Tidwell, J. (1999). Common Ground - A Pattern Language for Human-Computer Interface Design. Retrieved Feb., 5 2003, from www.mit.edu/~jtidwell/common_ground_onefile.html.

van Welie (2002). Interaction Design Patterns. Retrieved Feb., 5 2003, from www.welie.com/patterns/index.html

van Welie, M., van der Veer, G.C. & Eliëns, A. (2000). Patterns as Tools for User Interface Design: In: Int. Workshop on Tools for Working with Guidelines (pp. 313-324). France. Retrieved Feb., 5 2003, from www.cs.vu.nl/~martijn/gta/docs/TWG2000.pdf

Study Of Spatial Biological Systems Using a Graphical User Interface

Nigel J. Burroughs, George D. Tsibidis
Mathematics Institute,
University of Warwick, Coventry,
CV47AL, UK
njb,tsibidis@maths.warwick.ac.uk

William Gaze, Liz Wellington
Department of Biology,
University of Warwick, Coventry,
CV47AL, UK
w.gaze,e.m.h.wellington@warwick.ac.uk

Abstract

In this paper, we describe a Graphical User Interface (GUI) designed to manage large quantities of image data of a biological system. After setting the design requirements for the system, we developed an ecology quantification GUI that assists biologists in analysing data. We focus on the main features of the interface and we present the results and an evaluation of the system. Finally, we provide some directions for some future work.

1. Introduction

Biologists experience difficulties in managing and analysing large amounts of data. To tackle this problem, we designed a GUI, which incorporates tools for a systematic analysis of image data. The objective of the interface is to enable biologists to manage and analyse a large number of images by aiding classification of, and viewing of object groups that are present in image set. Its design is based on a consideration of system requirements and an assessment was conducted by allowing experts to evaluate the system. This paper focuses on three main aspects of the development of the GUI:

- The requirements of the system design.
- The design and development of the interface.
- An evaluation of the system focusing on the quality of the results.

2. System requirements

The design of the system should be based on the following requirements: The interface should

- offer a rapid data overview
- require minimum intervention by the user for object classification,
- allow fast error identification,
- be dynamic and adaptive to changes,
- be reliable and robust.

3. Application and User Interface Design

A biology experiment was carried out in the Department of Biology at the University of Warwick whose objective was the study of the ecology of a biological system (*Acanthamoeba polyphaga*, a ubiquitous unicellar amoeba, and *Salmonella typhimurium*, an enteric bacterial pathogen on non nutrient on agar). A large number of images were obtained (a series of at least 3,000 spatial images, each image about 2MB, 1022 x1024) and within each image a number of biological objects were present (see Figure 1). The aim of the analysis is to extract and classify all objects in the data set, distinguish dead from live objects, group objects into classification categories and finally carry out a detailed quantitative and qualitative spatial analysis of the results.

Basic Image analysis techniques (Weeks, 1996) were used and modified to locate and count all objects. The refractive halo was used to locate all objects in each image by intensity thresholding, giving morphological information about the objects such as perimeter, centre of mass and shape details. To distinguish dead from live objects, a delayed image (4secs) comparison (image subtraction) was performed.

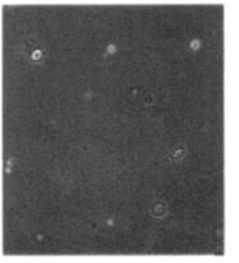

Figure 1: Typical image containing amoeba and cysts, (perimeter and centre of mass shown) and bacteria, e.g. filaments.

However, four problems are encountered with the automated procedure:

- Classification error,
- Incorrect perimeter,
- Missing objects,
- Multiple objects (refractive halo objects comprising more that one biological object)

To overcome these problems, we built a GUI (see Figure 2) using Matlab 6, which is one of a pair of graphical interfaces we have developed, an amoeba classification GUI (this paper) and a bacterial quantification GUI (not shown). The amoeba classification GUI allows the user to intervene and correct interactively classification and morphological errors.

There exist two types of classification for the objects of the experiment, one is based on the type of the object (amoeba, cyst, etc) while the second is based on the state of the object (live or dead). The GUI can help the user to avoid a time consuming manual classification for every individual object by running an automated classification procedure. The basis of these algorithms is illustrated in the two plots in Figure 3; cysts and dead amoebae are clustered in the lower section of the respective graphs and can thereby be classified based on approximate clustering. As a result, the interface offers the facility to create clustering patterns that allow a faster classification and decision making process. This is very important in minimising the effort of the user to achieve object classification throughout the entire data set.

The GUI consists of viewing facilities and modification and analysis tools. It operates as a management tool allowing the user to view summary information for individual objects or whole categories of objects, and also conduct a detailed analysis. The modification tools enable the user to either redraw the perimeter of an object if it is incorrect, introduce new objects if they were missed by the automated image analysis and to modify an object's classification at any time. All these manual changes are recorded and an overall analysis is adapted to the changes making the GUI a dynamic interface.

The analysis tools are used to facilitate a fast quantitative and qualitative analysis for objects belonging to a particular category by enabling the user to view graphical representations of the results in terms of plots (see Figure 3).

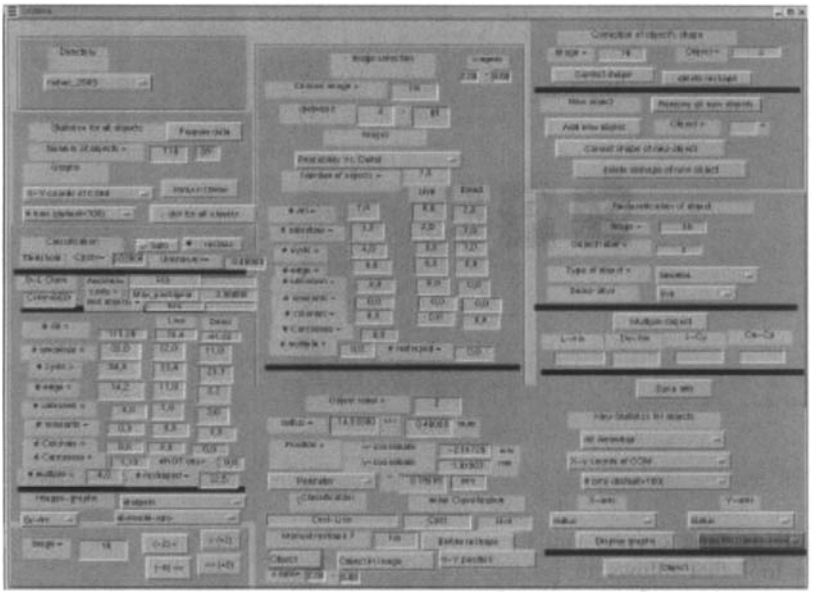

Figure 2: The Graphical User Interface

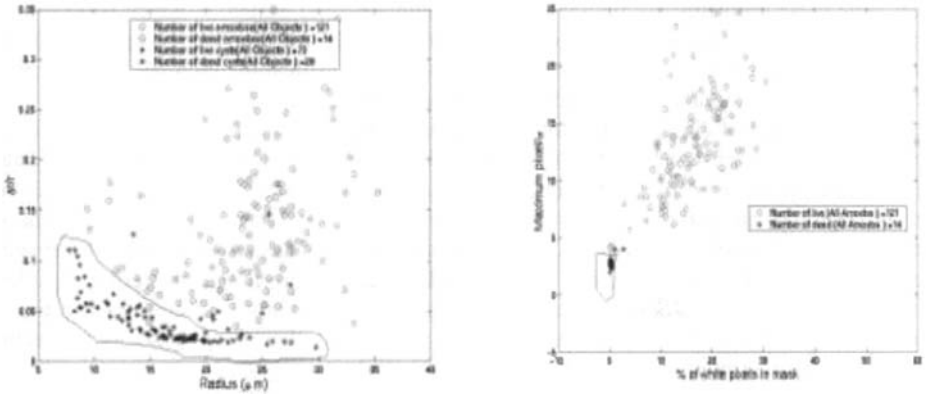

Figure 3: Amoeba-cyst and Dead-live classification. *Left panel*: shape analysis, using relative, standard deviation of object by radius. *Right panel*: maximum pixel intensity, in object (relative to an estimated intensity standard deviation σ) by fraction of live, pixels in object (live being significantly above noise, 95%).

The viewing facilities aids the user to scan quickly a library of objects belonging to each category (see Figure 4), compare them and decide whether a object classification change is required. The decision of the user can be assisted by the fact he/she is able to compare all objects that are in the same category and determine an optimal object classification criterion. If the classification of an object has been modified, it is indicated to inform the user of the change. In the dead or live object category, a library of images resulting from the delayed image comparison for all objects is produced and it provides a clear and fast view of the dead-live classification.

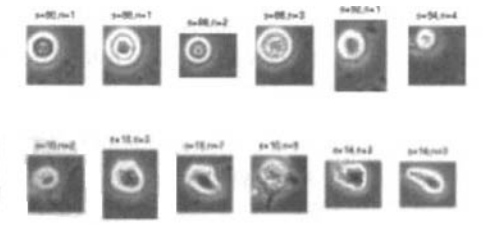

Figure 4: Library of images (upper row: cysts, lower row: amoeba)

The user can also employ the viewing tools to obtain data, measurements, confidence statistics and images for individual objects. In Figure 5, the first row shows the image of an extracted object (left box) and the same figure with the perimeter of the object drawn (right box) while the second row provides a summary of measurement and results (left box) and the delayed image subtraction for live or dead analysis (right box).

Another useful feature of the viewing facility is that it allows to view the full image sequence displaying all existing objects and their perimeters and centres of mass. The advantage of supplying a tool that allows to go through the whole set of images and see this type of characteristics provides a fast process of determining which objects will be used in the analysis and which objects must be included and redrawn manually.

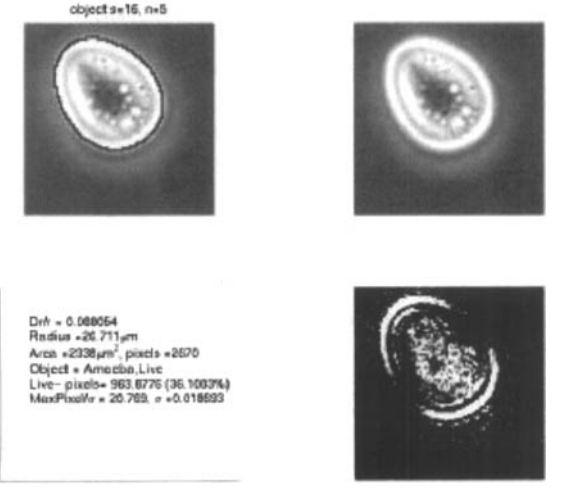

Figure 5: Images and measurements for an individual object

Both analysis and viewing facilities are very helpful tools because they simplify data management and reduce the number of objects requiring manual reclassification. This is a very important issue in terms of speeding up the analysis of a huge data set.

Finally, the efficiency of the interface can be tested by comparing the results derived from the automated and manual classifications of the objects and measuring the level of accuracy and confidence level of the automated procedure.

4. Evaluation

The interface was run on a Sun (Unix) platform with 1Gb of memory. It is mobile in the sense that it can run on Windows or Mac platforms provided there is substantial memory and the computer is equipped with a drive that can accommodate a large number of data.

The main objective of the evaluation procedure was to assess the set of requirements against the decision of experts. Two biologists from the Department of Biology at the University of Warwick helped the evaluation of the system. The focus was centred on the efficiency of the GUI and the simplicity it provided in the analysis of the data:

- The accuracy in the object classification was very high; More than 85% of the objects in the dataset did not require manual classification.
- The number of the objects that required their shape to be redrawn or had to be introduced manually was very small (≈5% of the total number of objects).
- The pattern recognition for the objects through the ROI clustering led to a faster object classification.
- The GUI is user friendly.

These demonstrate that the GUI leads to an efficient fast classification, minimises the need for the user intervention and simplifies the analysis.

5. Conclusions-Future Work

This GUI helped to analyse data collected in the Department of Biology at the University of Warwick. The aim of this project was primarily the simplification of object analysis by providing the means to assist a researcher in analysing a biology experiment in a fast and efficient way. The fundamental findings of our inquiry are that the GUI:

- operates as an information management tool,
- helps to group data and produce graphical representations that simplify the analysis,
- enables pattern recognition for object identification,
- minimises user intervention in object classification,
- is dynamic and adaptive to changes,
- is mobile provided the hardware requirements are satisfied

The present interface is one of a pair of GUIs (the other focuses on bacteria extraction and bacteria coverage in a huge set of images) that have been implemented with excellent results. An improvement would be to build an on-line version of the interface. A further study is ongoing that analyses sequences of images (videos) by studying trajectories of biological objects.

6. References

Weeks A.R. Jr., (1996). Fundamentals of Electronic Image Processing. IEEE press.

The use of Metaphors for Interaction between Children and Children's sites

Alessandra Carusi

PUC – Catholic University
of Rio de Janeiro
Rua Henrique Stamile Coutinho,
470/
201 – Rio de Janeiro – RJ – Brasil
alecarusi@alternex.com.br

Vera Nojima

PUC – Catholic University
of Rio de Janeiro
Rua Marquês de São Vicente, 225
Rio de Janeiro – RJ – Brasil
nojima@dsg.puc-rio.br

Abstract

This project puts in evidence the importance of observing principles and presupposes related to communicational issues, mainly in what concerns the production of meaning in which web design projects are involved. Hence, defining message is the starting point, until its minimal unit – the sign. Thus, this study focus on the construction of meaning, which is accomplished through the use of metaphors in graphic interfaces.

1 Introduction

The Internet is gradually becoming more and more present in the routine of those children who have a computer at hand in their environment. It can be considered that the construction of meaning of graphic representations, which comprises the interfaces of these sites, is the issue which is responsible for the achievement of these communicative goals. These children take decisions once they realize the design elements, emphasizing the designer's need of knowing their tendencies of perception while producing web sites for them.

This assembly of children is growing up surrounded by digital media, spending their upbringing years in a context and environment quite different from those of their parents. They learn, communicate and play with video games, CD-ROMs and computers connected to the Internet. Novelties are plenty. Knowledge isnot only acquired through reading and doing written exercis es or by the presentation of the information by their teachers. It is believed that the computer intervenes in these children's routine and intelligence development.

The Internet operates as a means for children to find the latest news and have fun in children'sweb sites, which are presented by their parents, friends or television. TV networks see the Internet as a vehiclefor presenting, advertising and interacting with its public. Therefore, children's sites linked to television channels are increasing in number and working as a complement to TV programs and as a publicityentreaty. However, the rise of available sites for children on the Web does not mean that these sitesare effectively stimulating their visits or interactions with the messages displayed

on their interfaces. This fact will only take place under the condition of the graphic interfaces presenting a language, which is understood by its public. The designer does not always know the expectations and repertoire of his or her children's public.

2 The Use of Metaphors in Interfaces

Graphic interfaces facilitate the way human beings interact with computers, making the communicative process feasible between them. The graphic interface was the feature, which contributed to make the usage of computers something simpler, acting like a translator of a signals and symbols language, proper to its functioning. In people's homes and in daily life there was a mounting of the presence of textual and pictorial interface computers, which helped user and machine interact in a semantic relationship, characterized by meaning and expression.

According to Peirce (1977), all thinking has subjective and emotional qualities, which are attributed to its sign, in an absolute or relative way, or through a convention. The sign comes up to the human mind through words, concepts, images, sounds and associations. Johnson (2001) declared that a man's visual memory of pictorial contents is far more lasting than verbal one. Therefore it is easier to forget a name than a face or remember the place where a specific text segment was located on a site, even if what was written there was forgotten. When it comes to graphic interfaces the value of images is considerable. The user associates the images on the screen to objects and actions of his or her daily life, and thus, considers pleasant the environment of the site.

The representations of graphic interfaces frequently take the form of metaphors. According to Fiorin & Savioli (1998), the metaphor is characterized by changing the meaning of a word by attaching a second meaning which shares a relation of similarity between thefirst and the second one. That is, when theyshare semantic characteristics. A command line constituted of 'zeros'and 'ones' is substituted by a metaphorof a virtual folder in a virtual desktop. This manner of representing the message expanded the capacity of using a computer amongst people who do not have the knowledge ofthe antecedent language expressed by 'command lines'.Johnson (2001) comments that visual metaphors, presented for the first time in the sixties, probably had more to do with the digital revolution than any other advance ever recorded within the software field.

3 The Relevance of the Representation Forms of Information

The metaphors on the interfaces are elements which, whether associated with others or not, lead to a meaning which is constituted by the associations of other meanings. Its interpretations are influenced by their public'sculture and context. A child, as a receptor of information, while accessing a site, not only decides if she orhe should click an icon, but should also decide which one to click. Understanding text combinations and images in their different colors and shapes, motioned or not, and transform them in meaning is a process which demands the understanding of each of these elements as well as the relations amongst them. To interact in a site, this child needs to feel encouraged. Nevertheless,motivation will only turns out if there is enough comprehension of the content of the commu nication object that is being presented.

Fiorin & Savioli (1998) gave the definition of a communication object: it is any product which has the aim of communicating something to someone, once it must establish a relation between its parts. What will determine a unique sense to the product will be the many relations between these parts and not the mere adding of them. When analyzing a text it is possible to be noticed that it is a

context in which smaller units, as sentences, are part of and function as context to other words. Each word has a particular meaning, denotation, or a second meaning, connotation, which can vary according to the context in which it isinserted.

> 'A hat in a peasant's head is just a sun protecting outfit; over a lady's head in a ceremony, it is an adornment; in front of a clergymanit is a symbol of power; in a beggar's supplicating hand it means a demand for help. To sum up: meaning is defined by relations'. (Fiorin & Savioli, 1998)

Therefore, context influences the reading of any product, pictorial or verbal. It is not possible to base oneself on isolated fragments, because the meaning of the parts is determined by the whole to which they belong. For this reason, graphic elements such as texts, images, colors, and contrasts of a communication object are fundamental to constitute a exclusive element with an intelligible voice. This will be expressed by language.

Andrade & Medeiros (1997) state that the language used for interpersonal communication is also helpful for man to structure his inner world. Consequently, language is used to think and communicate thoughts and emotions. Communication needs to use a common language between the transmitters and receptorsof the message, being it a language or a dialect, spoken or written, gestures, taps, colors, light or sound signals, etc. Communicating is not only transmitting information. An effective communication process occurs when the receptor, after having understood, makes a move or changes his or her behavior. The importance of the message isequal as its power of persuasion, which is the act of leading another person to accept what isbeing stated. Consequently, communicating is not only the knowledge of how to say something but, more than that, make others believe it and do what the transmitter has in mind. The acceptance depends on a series of factors such as emotions, feelings, values, ideology, vision of the world, political convictions etc. These values should be organized in a language known by the speakers of the message. There is no such thing as neutral communication. Somehow, all communication aims to convince the receptor of something.

Andrade & Medeiros (1977) claim that language, knowledge and communication are mixed, being on that account any knowledge is considered incomplete if it is not capable of being communicated through language.

A message may be constituted by verbal and non-verbal language, or by one of them separately. Fiorin & Savioli (1998) remark that there are many forms of language. Men, besides articulated verbal language, also display of other language systems, some of each are non-verbal ones. According to these authors, language is structured in two levels: content, which comprehends the meanings conveyed, and expression, which is constituted by the vehicle of these meanings. In what concerns expression, two voices are implicated: one which affirms and another which refutes former information. This is how contrast is fully and clearly expressed. This opposition of the senses provides unityto the elements of the message, because it establishes a relation among each element within the context. To understand the organization of these relations is what will determine the apprehension of concrete meaning within an specific context. An isolated figure does not have only one meaning. It can imply many ideas.

The smallest unit in a language system is the sign. According to Nöth (1995), the sign does not have the role of a class of objects but the function of an object in the communication process. The interpretation of a signis a dynamic process that happens within the receptor's mind. The apprehension process of a sign is called semiosis. Eco (1973) used to comment that the sign is

used to transmit an information, to indicate to someone, something known by another person, who by his or her turn, wants to share this information with other people.Fiorin & Savioli (1998) believe that this relation is constituted by three elements: the objector referent, the sign or representant, and the interpreter. The relation, which takes place between signand referent, determines the existence of three types of signs: whether the relation is arbitrary, there is a symbol (word). If there is contiguity, proximity between sign and object (smoke – fire; footsteps – people; yell – pain; wet ground – rain), there is an index. If the relation is based on similarity, there is an icon (map, photograph, drawings, comic strip, caricature, metaphor).

Being the sign the unit of a message, it can be concluded that its significant, its information and its meaning are based on the relations amid the signs which are part ofthis message. The organization of the relations between these signs is formed by coherent nets, which are comprised of the theme of the message or product. Fiorin & Savioli (1998) allege that coherence is a tight union amongst many parts. Furthermore, it is the relation between ideas which are in harmony, causing an absence of contradiction. Coherence distinguishesa product amidst an assemble of things, creating a unit of meanings. In a graphic interface, for example,the elements enchain themselves in a special way to express a particular theme. Thus, they must be compatible ones with others, on the contrary the user will not distinguish which theme is being conveyed. A very same theme may be treated in many different ways, but its coherence is fundamental for it to be plausible. A graphic interface for kids may lose its verisimilitude if it contains a design constituted of a gray scale of colorstogether withfew iconic images. If this occurs, the power of persuasion becomes weaker because the user will not believe in the message, consequently, he or she will not make the expected moves.

4 Semiotic applied to the Language of Sites

Santaella (2002) makes a commentary that the classification of signs is the division of Peirce's most important semiotictheory 'When it isintendedto analyze manifest languages in a semiotic way, it can be observedthat semiotics furnishes us with the definitions and general classifications of all types of codes, languages, signals andsigns of any species and the main aspects which embrace them, for instance: signification, representation, setting objectives and interpretation'. Therefore, semiotic offersbacking for the relations established in a semiosis, allowing the foreshadowing of meaning and the applicability of what was elaborated.

Semiotic provides information and knowledge, to put in other words, semiotic theoretical base allows the elaboration of a webdesign project based on the solutions of its communicational issues and meaning, in the relation between product – user. Any webdesign project contains elements which are signs and which convey meaning to the public to whom this project is somehow addressed to. Through the study of signs, which interact in the project, the process of production of meaning can be amended and evaluated if it will meet its public'needs.

5 Conclusion

A webdesign project organization should not be based on formal resources as color sets, shapes and words, with the mere intention of adorning it. The most important circumstance is to ascertain the best manner of displaying signs because doing it exclusively this way, it will be possible to generate coherent and cohesive messages.

While creating a communication object composed of many metaphors, due to the fact that it occurs in many sites for children, the designer and his or her target public must share multiple information.In the intersection ofthis knowledge, the metaphor constitutes and provides, for the children's public, the meaning idealized by its creators. This metaphoric content within graphic interfaces will be likely to unchain feelingsandemotions which are necessary to arouse the child to take many moves while accessing a site addressed to children.

An analysis of how children identify structured images as language could reveal the importance of doing a specific study about howinformation will be presented in projects. One should be cautions when publicity for children is being conveyed and it should avoided, for instance, pages with too much information, incorrect use of typography or colors,metaphors which are out of context or are repetitive etc. The way of presenting graphic elements is fundamental so they do not become mere points of exploitation. Consequently,it is possible to conclude that , facing all new possibilities offered by the latest technologies of communication, the most adequate solution would be the observation of children's behavior and their interaction with sites which already exist. It would also be useful to observe children's interaction with projects which are being developed. Once this observation takes place, it would be possible to make propositions for further projects.

Adults often forget that children are children. The designer can manage the elements present on graphic interfaces to facilitate a child's perceptive process, terminating with obstacles that make reading and understanding difficult. If the designer, during the project, is aware of the mechanisms that construct the meaning of these elements, the child will understand their content in a better manner.

Thereafter, the designer's message to the computer user by the intermediacy of the graphic interface of a site becomes an effective communication process, where the targets outlined during the design project are achieved.

References

Andrade, Maria Margarida de & Medeiros, João Bosco (1997). *Curso de lingua portuguesa: para a área de humanas: enfoque no uso da linguagem jornalística, literária, publicitária*. São Paulo: Atlas.

Eco, Umberto (1973). *O Signo*. Lisboa: Editorial Presença.

Fiorin, José Luiz & Savioli, Francisco Platão (1998). *Lições de Texto: Leitura e Redação*. São Paulo: Ática.

Johnson, Steven (2001). *Cultura da interface: como o computador transforma nossa maneira de criar e comunicar*. Rio de Janeiro: Jorge Zahar.

Noth, Winfried (1995). *Panorama da Semiótica*. São Paulo: Annablume.

Peirce, Charles S. (1997). Tradução : COELHO NETO, José Teixeira. *Semiótica*. São Paulo: Perpectiva.

Sataella, Lúcia (2002). *Semiótica Aplicada*. São Paulo: Pioneira Thomson Learning.

Tiski-Franckowiak, Irene T. (1997). *Homem, comunicação e cor*. São Paulo: Ícone, 1997.

Can novice designers apply usability criteria and recommendations to make web sites easier to use?

Aline Chevalier

Research Center in Psychology of Cognition, Language and Emotion
University of Provence
29, avenue Robert Schuman
13621 Aix en Provence, France
alinech@up.univ-aix.fr

Melody Y. Ivory

The Information School
University of Washington
Seattle, WA 98195-2840 USA
myivory@u.washington.edu

Abstract

We present two studies that we conducted to determine the effect of a short training program in applying usability criteria and recommendations on the evaluation and improvement of a web site, which contained usability problems. This training program presented web site usability criteria and recommendations. We conducted the two studies with novice web site designers, who had or had not participated in the short training program, to compare their performances. Study results showed that the trained novices identified 30.7% more usability problems than the untrained novices. Moreover, the trained novices managed to make several usability-related modifications to the web site to make it easier to use.
These first results have to be verified by other studies, but they are encouraging. They suggest that web site designers, who have no university training in ergonomics, can use usability criteria. Furthermore, results suggest that usability criteria should be integrated into web site design training.

1 Introduction

In an experimental study, Scapin and Bastien (1997) asked 17 cognitive ergonomics and human factors students to evaluate a music application that had usability problems. Students conducted evaluations in one of three conditions:

- Condition 1: Six students used a list of usability criteria for the design of hypermedia interfaces (defined by Bastien, Scapin, & Leulier, 1997).
- Condition 2: Five students used the ISO/DIS 9241-10 standard (International Organization for Standardization, 1994).
- Condition 3: Six students used no guidelines (control group).

Results showed that students identified 51% more usability problems in the condition with the usability criteria than in the conditions without guidelines or with the ISO/DIS 9241-10 standard. Nevertheless, we must underline that in this study, participants were familiar with ergonomic and usability aspects (they were in a cognitive ergonomics university program). We argue that it remains to be determined whether people, who do not have a background in ergonomics or human factors, can similarly apply usability criteria.

2 General presentation of the two studies

2.1 Objective

Bastien *et al.* (1998) adapted the usability criteria for the design of hypermedia interfaces (used in the study described above) to the design of web sites. We wanted to determine whether web site designers could apply the adapted usability criteria to improve web sites; thus, we conducted two studies with novice designers. We chose this domain, because in most cases, web site designers have not had any training in human factors or in usability, yet these aspects are very important in their design activities. Indeed, designers must address the needs of the client (person who requests the site) and of the users (Chevalier, 2002; Chevalier & Ivory, 2003) while designing web sites.

These two studies aim at determining the effect of a short training session on a designer's ability to use usability criteria to evaluate and improve a web site that had a set of usability problems.

2.2 Web site designers

Numerous HTML authoring tools (e.g. Netscape Composer®, Macromedia Dreamweaver®) are available and their use, after a short training time, is not very complex. Therefore, many short training programs are available for people who want learn how to create web sites. So, more and more persons can be considered as web site designers. On this basis, the two studies were led with novice designers, who had just followed a short training program in using the Macromedia Dreamweaver® authoring tool. These designers had not yet dealt with real web site clients, and they had created only one site during their training program.

We chose to focus on novice designers for two main reasons:

- A request from the University of Mediterranean (Marseille-France) for training students in how to consider and apply usability criteria during web site design was suggested to us.
- Novice designers, who had never dealt with clients, would not have developed any procedure or skill linked with clients' needs (see Chevalier & Ivory, 2003).

2.3 Web site to evaluate

For the two studies, we created a web site in HTML for a music store, which commercialised music products (CD, concert tickets, etc.). The web pages were hyperlinked such that we could navigate on the site. The site comprised 18 pages and had a total of 337 usability problems (34 of them were unique); example problems included: the absence of page titles, the lack of color changes to indicate visited links, the use of too much text, and so on. We chose the nature and the number of usability problems based on a previous study, wherein we analysed similar HTML web sites produced by web site designers (see Chevalier & Ivory, 2003).

2.4 Hypothesis

One specificity of web site design is that designers are also web users, which is not the case for most design situations (e.g., the design of aerospace products). Nevertheless, designers can become, with experience, expert web users. This statute will allow them to navigate easily on the Web. So, they may not be able to recognize usability problems, which can be considered as such by non-expert web site users. On the contrary, the training program in applying usability criteria

and recommendations will support novice designers to activate user knowledge as opposed to client knowledge (by guiding their evaluation activity). Therefore, they should identify more usability problems in the web site than untrained novices. On this basis, we hypothesized that trained designers will manage to apply user knowledge to improve certain usability problems that exist in the study web site. To test this hypothesis, we led two studies with novice designers.

3 First study: Evaluation without receiving training in the application of usability criteria

3.1 Procedure and objective

Seven novice designers evaluated the web site without receiving training in the application of usability criteria. They had completed the same training in using the Macromedia Dreamweaver® authoring tool. All the designers had to evaluate the study web site while using the same Personal Computer (PC). They had to inform the experimenter of the positive or negative aspects of the web site (i.e., "to think aloud;" see Ericsson & Simon, 1993); There was no time constraint for this task. We videotaped their verbal protocols and their use of the site. More precisely, this study aimed at determining:

- The number of web pages that designers did not visit in order to identify their navigational behaviour.
- The number of usability problems that designers could identify in the web site

3.2 Results

We recorded designers' verbal protocols and then transcribed them for analysis by two different persons. Study results showed that novice designers did not visit in mean 8.2 of the 18 web pages (see Table 1). Moreover, they identified very few usability problems introduced in the web site to evaluate (see Table 1). One natural question that arises from these study results is whether or not designers, who completed a training program in the application of usability criteria, would manage to identify and to improve more usability problems than the untrained novice designers did.

Table 1: Number of identified usability problems and unvisited web pages for untrained novice designers (in mean and percentage)

	Identified usability problems	**Unvisited web pages**
Mean	16.2 of 337	8.2 of 18
Percentage	4.8%	45.6%

4 Second study: Evaluation with receiving training in the application of usability criteria

4.1 Procedure and objective

Six novice designers evaluated and modified the web site. These designers had recently completed the training program on developing web sites with the Macromedia Dreamweaver® authoring tool.

In our first study, seven novice designers evaluated the web site without receiving training in the application of usability criteria. In this second study, the novice designers completed a short training session in the application of usability criteria. During the six-hour training session, the instructor used several existing web sites to demonstrate how to identify usability problems as well as how to modify sites to mitigate problems (i.e., how to respect usability criteria). After the training session, the designers evaluated the web site. To further determine if designers could apply the usability criteria, we asked them to rectify usability problems on one web page in the site. We instructed designers to indicate positive or negative aspects of the site and to make the home page easier to use; Thus, the corrected page had to have fewer usability problems than the original page.

4.2 Results

Verbal protocols were analyzed by two different persons. Study results showed that the designers visited all the web pages. They identified, in mean, 35.7% of the usability problems introduced in the site. They also managed to address usability problems of the home page by implementing, in mean, 6.3 modifications. The home page had thirteen usability problems, and the designers managed to correct 6.3 (48.6%) of them. (see Table 2). For example, if we compare the original home page (Figure 1) with the one modified by a novice designer (Figure 2), we can see that the designer modified certain usability problems like reducing the page's length or modifying the fonts used However, other problems still exist in the modified design (e.g., the page's background color or the presence of animations may impede user's navigation). Therefore, we can conclude that the designers, who had no background in ergonomics and human factors, were able to successfully apply the usability criteria, because the modified home pages had fewer usability problems than the original pages.

Table 2: Number of identified usability problems, usability problems in the modified home page, and unvisited web pages for trained novice designers (in mean and percentage)

	Identified usability problems	**Modified usability problems**	**Unvisited web pages**
Mean	119.5 of 337	6.3 of 13	0 of 18
Percentage	35.7%	48.5%	0%

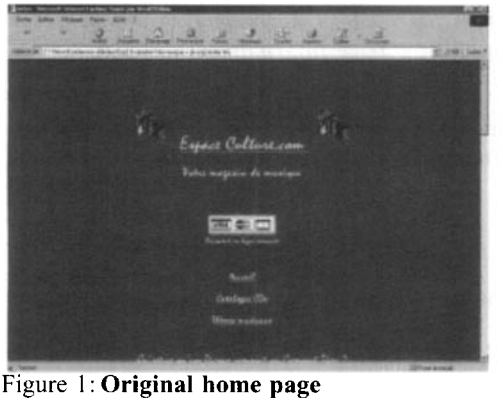

Figure 1: **Original home page**

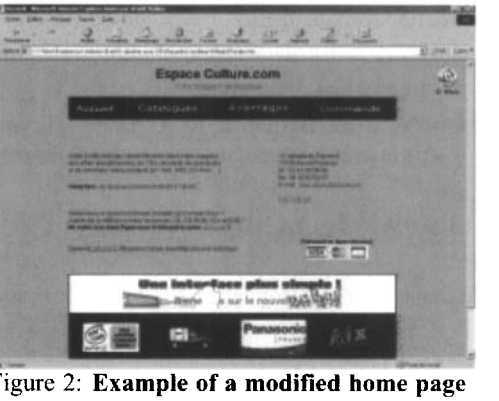

Figure 2: **Example of a modified home page**

5 Discussion and future work

Although these preliminary results need to be verified by additional studies, they are encouraging. Designers who completed a short training in applying usability criteria and recommendations were able to identify more usability problems in the study site than untrained designers did (19.5% *vs* 16.2%; $F(1,11)=95.399$; $p<.0001$). Moreover, trained designers visited all pages, whereas untrained designers did not. Untrained designers were hindered by the usability problems on some pages, which kept them from visiting all pages. For instance, they did not see a hyperlink to visit another page, because this hyperlink was at the bottom of a web page. Moreover, the trained designers managed to modify the home page, but we do not know whether these modifications actually improved usability. Future work entails asking users to evaluate the modified home pages.

These results suggest that novice web site designers, who have no background in human factors, can apply usability criteria with some training. Therefore, usability criteria need to be integrated into web site development training. In addition, usability criteria need to be adapted such that designers, who do not have a background in human factors, can apply them.

Toward this end, several questions could be explored. First, these two studies have to be replicated with professional web designers to determine whether they produce the same results. According to the obtained results, a new study could be led. It would aim at determining the influence of usability criteria, not in evaluation or modification situations, but in a design situation with both professional and novice web designers.

References

Bastien, J. M. C., Leulier, C., Scapin, D. L. (1998). *L'ergonomie des sites web. Créer et maintenir un service web.* Cours INRIA, ADBS, Paris, 111-173.

Chevalier, A. (2002). *Le rôle du contexte et du niveau d'expertise des concepteurs de sites web sur la prise en compte de contraintes.* Ph. D. thesis, University of Provence (France), Cognitive Psychology Department.

Chevalier, A., & Ivory, M. Y. (2003). Web Site Designs: Influence of Designer's Experience and Design Constraints. *International Journal of Human-Computer Studies, 58,* 57-87.

Ericsson, K. A., & Simon, H. A. (1993). *Protocol analysis: Verbal reports as data* (Revised edition). Cambridge (MA): MIT Press.

International Organization for Standardization (1994). *ISO 9241-10. Ergonomic requirements for office work with visual display terminals (VDTs) -- Part 10: Dialogue principles.* Draft International Standard.

Scapin, D. L., & Bastien, J. M. C. (1997). Ergonomic criteria for evaluating the ergonomic quality of interactive systems. *Behavior & Information Technology, 16,* 220-231.

Acknowledgements

We thank Nathalie Bonnardel for her advices and the web site designers for their participation.

Learning from Museum Visits: Shaping Design Sensitivities

Luigina Ciolfi & Liam J. Bannon

Interaction Design Centre
Department of Computer Science & Information Systems
University of Limerick, Ireland
Luigina.ciolfi@ul.ie, liam.bannon@ul.ie

Abstract

This paper describes an observational study of visitors interacting with artefacts in a museum, and attempts to draw from these studies a number of design considerations. Gaining a thorough understanding of the context, and the way visitors move through the exhibitions and interact around the objects on display, is crucial in designing effective museum installations. In our research, we are designing novel installations that will engage visitors in a natural and unobtrusive way. Our designs – which are ongoing at the moment – are informed by our field study work at the museum site. Our purpose here is to show how our observations can indeed be made relevant to design concerns, a topic that is fundamental to the development of successful pervasive technology.

1 Introduction

Despite the increasing prevalence of computer-based museum installations, visitor experiences with computer technology remain mixed, at best. Many of the most insipid and uninspiring multimedia kiosk-type installations that proliferated in the mid-nineties have thankfully been removed. These kiosk installations tended to separate the person from the actual artefacts, and called attention to the computer interface itself as the object of interest, rather than the actual artefacts. Far from augmenting the artefacts, they served to distance visitors from the items in the collection. People are not happy about moving away from the exhibitions to consult a kiosk, they would prefer to focus on the displayed objects. These kiosk technologies also provide essentially an individual experience, whereas visitors usually go to museums in small groups. Nowadays, there is a greater realization within the computing community that the technology needs to be in the service of the exhibition, rather than stand out from it. The increasing interest in such areas as ubiquitous or pervasive computing, where computational power is taken "out of the box", distributed and attached to material objects, provides interesting new possibilities for how one might augment museum exhibits. "Traditional" desktop computers are not an effective form of technology for supporting visitor behaviour within such a rich, interesting and complex environment as a museum. Novel mobile, "ultralight" appliances and "Disappearing Computer" technologies could be more effectively employed in order to design better technological installations in the museum, both for supporting the visit and for creating interesting visitor experiences or educational activities.

This paper provides a number of insights derived from observing visitors, and interviewing them, at a specific site - the Hunt Museum in Limerick, Ireland. The purpose of these studies is to provide some background material that will inform and inspire our creativity in designing new augmented museum exhibits. However, within the design community, while there is interest in ethnographic and other forms of field studies, there is increasing concern as how one can take the results of such studies and make them relevant for design. Often, social scientists focus on issues that do not seem of relevance to design, and they do not have the training or expertise to be able to move from observation to the specification of requirements. Our own expertise crosses the human and social sciences, but also extends to concept and software design, a somewhat unusual combination. Thus we are especially interested in ensuring our observational studies are of relevance to the design process. In an effort to show how such observations can indeed be shown to be relevant for design, we, along with colleagues, have been involved in exercises that attempt to shape the design process so as to be sensitive to various issues, as determined from the observational studies. In a word, we are looking for "design sensitivities", rather than requirements *per se*.

Our work has been conducted within the EU-Disappearing Computer SHAPE[1] project (Situating Hybrid Assemblies in Public Environments). The SHAPE project is not strictly concerned with introducing technology within museums and galleries, rather the project focus is on creating hybrid public environments that allow visitors to actively interact with features of the physical, and of the digital, space. However, museums and exploratoria are the chosen context to inspire and support the development of such installations. The SHAPE team at the IDC, University of Limerick, jointly with the Hunt Museum, are currently developing a full design scenario informed and inspired by the analysis of a corpus of field studies aimed at understanding human behaviour within the museum environment, and, specifically, the way visitors approach and make sense of particular exhibits and specific objects. In this paper, we attempt to match specific vignetttes culled from our video studies, and pair them to design sensitivities, to show how particular user behaviours around existing objects and exhibits might sensitize us to certain issues, which we may wish to explore further in our computer-augmented installation.

2 Cabinets of Curiosities

The Hunt Museum has three "cabinets of curiosities" exhibits: they are wooden closets where objects are arranged both on the upper shelves and in the drawers that the visitors can open if they wish. The objects are protected by glass on the top of each drawer. The nature of the objects contained in the cabinets is varied: ivory pieces, tapestry, drawings, coins, etc (see Figure 1).

We conducted numerous observations on the behaviour of people in the "Study Collection" room, where one of the most important of the cabinets is located. We were interested in understanding the way visitors relate to the exhibit and their interaction with the drawers. These cabinets are particularly appreciated by the visitors. An evident reason is the potential of drawers, chests and boxes to stimulate curiosity and exploration. Containers suggest the presence of secrets (Elsner & Cardinal, 1994), or of objects that have some kind of symbolic relevance for being sheltered from the eyes of the public (Bachelard, 1958). "Curiosity is a major factor in determining whether environments are appealing, and indeed curiosity triggers interaction towards its object." (Falk & Dierking, 2000, p. 115). The interactions around the drawers reveal interesting visitor activities, often involving collaborative understanding and appreciation of the objects.

[1] Members of the SHAPE Consortium are: the Royal Institute of Technology-KTH (Sweden; Coordinating Partner), King's College London (UK), the University of Nottingham (UK) and the University of Limerick (Ireland).

Fig.1. A cabinet of curiosities in the Hunt Museum

3 From Field Studies to Design Sensitivities

In what follows, we list in more detail a few of the interesting observations and our interpretation of their relevance to a potential design process.

Observation 1. People tend to interact with the drawers in one of two ways: they open either a few drawers or all of them in sequence. Those who cannot resist opening all the drawers usually spend a longer time on each, and tend to comment more on the objects with their companions.

Design concern 1. *To keep the user's interest and engagement high, we must envision ways to support different 'layers of activity': this could provide participants with the ability to engage in a progressive sequence of actions (both alone and with others)in order to provide successive surprises and discoveries. For example, more, and varied information on assemblies of objects and their mutual relationships is provided to those who want to explore all the parts of the exhibit.*

Observation 2. In the case of couples visiting the exhibit, each person usually takes control of one side of the cabinet, resulting in two drawers being open at the same time. This pattern of interaction facilitates the exchange of opinions and comments among the two visitors, usually comparing the contents of the respective drawers. Gestures accompany verbal comments on objects' similarities, possible relations, visual features, etc.

Design concern 2. *The augmented cabinet of curiosities should provide the possibility of parallel discovery and of making comparisons in order to support collaborative understanding of objects: the interactivity we support should not be limited to the one between individual and exhibit, but we should consider the different degrees and combinations of verbal gestural interaction amongst individuals. The installation should also provide some kind of added value associated with collaborative interaction around the drawers.*

Observation 3. The glass surface on top of each drawer is often used for sketching or taking notes: children especially spend time in taking notes and drawing sketches of the objects contained in the drawer, using the drawer itself as support. In many cases, we observed children visiting the Study Collection with their parents, and literally taking the lead in discovering the content of the drawers and showing to and commenting on it with their parents (see Figure 2).

Figure 2. A child is leading his parents through the exploration of the drawers

Design concern 3. *Children should be allowed to take part in the activity and be able to take notes or sketches around the augmented cabinet of curiosities. The installation should also give children the possibility to lead the process of discovery and to show it to their companions.*

Observation 4. Some people think that it is not possible to touch or open the drawers and they might open one only after seeing somebody else doing it, and in this way being reassured they are actually allowed to interact with the exhibit. Even if one of the drawers is left slightly open to suggest its real use, these visitors did not seem to investigate further, assuming the open drawer was the result of some work being done by museum personnel. When visitors proceed to open the drawers and discover what they contain, all of them express their surprise.

Design Concern 4. *The technologically augmented cabinet should provide clues, triggers and adequate affordances to make visible which actions the visitors are allowed to perform on each component of the installation. We must consider possible ways of encouraging interaction with, and around, the exhibition, and specifically collaborative interaction.*

Observation 5. Unlike couples, members of larger groups of visitors cannot all simultaneously open the drawers: usually, two people act as "openers" on the two sides of the cabinet, but all the people in the group comment on the objects and tend to draw each other's attention to what has been discovered.

Design Concern 5. *The augmented cabinet should support the group visit experience with appropriate feedback that all the members of the group can appreciate. The possibility for the visitors to talk to each other must also be insured, as discussing the objects together is an essential part of the group experience around the cabinet. This means that devices as head-mounted displays or headphones are not appropriate for such an installation.*

4 Developing an augmented cabinet of curiosities

Following the data analysis sessions and an initial phase of reflection on our collected material, we are developing an augmented reality Study Collection room. The existing cabinet has some important features that make it a successful and inspiring installation: it is robust, the visibility is good, it allows for collaborative interactions and supports small groups. It is also easily accessible by children. The cabinet adds value to the experience of museum visit as it includes an element of active discovery and plays on a natural curiosity towards drawers and cabinets. Each object is interesting by itself, but multiple objects are connected by similarities or unexpected connections. What can pervasive technologies do to achieve such features in the technologically enhanced exhibits? At a minimum, we need to seamlessly integrate aspects of the exhibit with a visually pleasing structure, react to collaborative behaviour, and provide ambient feedback. We need to create a space that integrates with the rest of the museum. This is our design brief. Currently several researchers at the Interaction Design Centre, are engaged in concept design, inspired by the material presented above. We have found that the video material has helped us in creating outline scenarios, and storyboards. In the process of defining what kind of added value a technology-enhanced cabinet of curiosities could provide to people, we noticed that the only element currently missing in the process of interaction with the cabinet and its objects is the "physical" one: what if the visitors could handle the objects and explore them directly? Experiencing the exhibit has a physical aspect as well as a reflective one. As museum researchers have noted: "Exhibits ... allow people to see, touch, taste, feel, and hear real things from the real world. ... Visitors devote most of their time to looking, touching, smelling, and listening, not to reading. Visitors tend to be very attentive to objects, and only occasionally attentive to labels." (Falk & Dierking, 1995, p. 77). As emerged in our series of observations, interaction among visitors, collaborative content discussion and discovery all happen *around* the objects. The current cabinets in the Museum do not provide a way to experience features such as the look and feel of the surface of an object, the light and shadow it presents, the shape, texture, weight, etc. Our scenarios and prototypes are exploring

narratives and progressive unfolding of events, attempting to focus attention on aspects of the artefacts themselves and their context of discovery and use. Space does not permit us to detail our designs here, but we will show some of our prototypes during the Conference presentation.

5 Conclusion

The field studies conducted at the Hunt Museum have provided a useful corpus of material concerning the nature of exhibits, and visitor interaction around these exhibits, that is still under analysis. The elaboration of the "cabinet of curiosities" represents the first prototype we are developing, where we wish to augment these physical cabinets in a variety of ways with new technologies. Outline scenarios include cabinets where people can actually handle replicas of the real objects and experience some of the characteristics of these real objects. Our interest in this scenario has been driven by analyses of visitor engagement with exhibits and their interactions during specific "handling sessions" supervised by museum personnel that we have analysed (see Ciolfi & Bannon, 2002.)

This scenario, in which the objects themselves are envisioned as interfaces through which visitors can make sense of the object, of their history and their multiple relationships and features, poses a challenge for designers. Several projects are underway regarding the design of graspable interfaces and physical icons instead of GUIs (e.g. Ishii & Ullmer, 1997), but usually the object itself is not the locus of information, nor the focus of attention. Rather, objects are essentially tools for interacting with a computer system, and they are intended to act as a physical representation of surface interface elements such as icons and pointers. Thus our approach would be distinct, as we are interested in objects as both material and symbolic devices in their own right, with a history, context of use, etc, both mediating, and being the object of, interaction. Work in the coming months will involve further exploration of a variety of technical devices and platforms that may assist us in achieving some of our objectives for the prototypes, such as RFID tags, use of accelerometers and potentiometers for sensors, projection surfaces, webcam tracking etc., and we will show the results of our design and development work in the Conference presentation.

References

Bachelard, G. (1958). The Poetics of Space, Boston: Beacon Press.

Ciolfi, L. & L.J. Bannon (2002). Designing Interactive Museum Exhibits : Enhancing visitor curiosity through augmented artefacts. Proceedings of ECCE11, European Conference on Cognitive Ergonomics, Catania (Italy) September 2002, pp. 311-317.

Elsner, J. & Cardinal, R. (eds.) (1994). The Cultures of Collecting, Cambridge, MA: Harvard University Press.

Falk, J.H. and Dierking, L.D. (1995). The Museum Experience, Washington, D.C.: Whalesback.

Ishii, H. & Ullmer, B. (1997). "Tangible Bits: Towards seamless Interfaces between People, Bits and Atoms", *Proceedings of CHI '97*, New York: ACM Press.

Scenarios in the model-based process for design and evolution of cooperative applications

Bertrand David, René Chalon, Olivier Delotte, Gérald Vaisman

Laboratoire ICTT, Ecole Centrale de Lyon
36, avenue Guy de Collongue, 69134 ECULLY, France
{Bertrand.David, Rene.Chalon, Olivier.Delotte, Gerald.Vaisman}@ec-lyon.fr

Abstract

In this paper we describe the role of the scenario-based approach which we integrate in a more global perspective of a well-organized process based on a collection of models which are used in design and evolution to transform scenarios in an operational cooperative application.

1 Introduction

CSCW (computer supported cooperative work) [Ellis-94, Andriessen-03] research proposes a new type of software, called groupware, which is an interactive multi-participant application allowing participants to carry out a "joint" task working from their own workstations. It is now a question of managing not only the man-machine interface but also the man-man interface mediated by the machine. The relationship between the participants can be considered from various points of view. Ellis et al. [Ellis-94] proposed a matrix which classifies the nature of cooperation in regard to time - synchronous or asynchronous, and to distance - local or remote aspects of cooperation. This classification was extended later, introducing awareness of cooperation, foreseeability or unpredictability of collaboration and location. The possibility of bringing together geographically distant people is an important contribution of groupware. The first aim of groupware is thus to propose a support for the abolition of space and time distances. Moreover, knowledge and management of the interventions of the multiple participants appear necessary. In fact, the participants constitute a work group that has to be organized with respect to working conditions, time and location. The organization can lead to the definition of different roles, sub-groups and phases of project work. The success of cooperative work can be measured by the way in which the groupware is able to create and support good group dynamics, which contributes to the disappearance of the virtuality of participants' presence. The project must be able to proceed as naturally as in collocation and without IT support. It must even take advantage from an organization of more effective work based on the new possibilities offered by information technologies (IT). The technological devices used should not interfere with the work or the group dynamics needed for project accomplishment. When designing cooperative systems, it is thus necessary to be aware that the usability aspect, the aim of which is to validate the environment suggested, is at least as significant as the engineering aspect. The evolution of users' practices during the project life-cycle must be taken into account in order to provide an effective and adaptable environment.

In-depth analysis of cooperation reveals several dimensions which must be examined, as initially proposed by Ellis [Ellis-94] with the Clover model, i.e. a support of production, conversation communication and coordination between participants.

2 Scenario-based Design and CAB Model

Carroll's view of Scenarios-Based Design [Carroll-00] is an interesting starting point for collaborative system design and evolution. However, it seems important to go further. We propose to approach this scenario-based approach in a more organized way. In particular, in respect with the Clover model, it is important to locate each scenario in production, coordination or conversation space or in their intersections. More globally, it seems important to synthesize these scenarios in a model integrating collaborative application behaviors. We call this model a CAB model (Collaborative Application Behavior Model).

"Scenarios are stories", as Carroll wrote [Carroll-00], which, to be useful, can be expressed more or less freely or formally. We consider that it is important to integrate them as soon as possible in CAB perspective i.e. to ask the scenario writers to express explicitly the position of the scenario in relation with the CAB Model. The main goal of the CAB model is to describe explicitly the structure of actors, artifacts, contexts and tasks that characterize the behavior of the cooperative application in three Clover spaces (co-production, coordination, conversation). Each scenario expressing a task, might indicate its position in relation with these actors, processes, artifacts and contexts. In this way it is possible to elaborate progressively the CAB model for a given application. The CAB model for a specific collaborative application, contains concrete actors, artifacts, tasks and contexts which the cooperative application will take into account. Tasks expressed in different scenarios are studied in order to organize them. The goal is to eliminate redundancies and to elaborate a task tree and a task process. The task tree can be expressed in ConcurTaskTree formalism proposed by Paterno [Paterno-00]. The process view is a workflow view with temporal and logical dependencies between tasks. The context view is an expression of different contexts (logical or physical) related to environment and devices constraints, if any. Once we have collected several scenarios, we can start to elaborate the CAB model. This model will be used in the elaboration stage (development or prototyping).

3 Software infrastructure

With respect with software engineering considerations, the cooperative application cannot be carried out from scratch. It is necessary to identify different levels of development which are more or less dependent on the application. Usually, three functional layers are recognized.

The top layer corresponds to the collaborative application level. It contains all the cooperative software employed by the users. This level is definitely user-oriented, which means that it manages interaction control and proposes interfaces for notification and access controls. This model is called CUO-M (Collaboration User-Oriented Model). It uses multi-user services provided by a second layer called the groupware infrastructure, called CSA-M (Collaborative System Architecture Model). It is a generic layer between applications and the distributed system. This layer contains the common elements of group activities and acts as an operating system dedicated to groups. It supports collaborative work by managing sessions, users, it groups and provides generic cooperative tools (e.g. telepointer) and is responsible for concurrency control. It also implements notification protocols and provides access control mechanisms. The last layer is essentially in charge of message multicast and consistency control. We call it DSI-M (Distributed System Infrastructure Model). Usually, it is a computer-oriented layer which provides transparent mechanisms for communication and synchronization of distributed components which misfit with CSCW aims but which are very useful.

The degree of generality (and genericity) is not the same for these tree layers and models. The lowest layer (DSI-M) is for the most part independent from the collaborative applications. The middle level (CSA-M) can be dependent on a category of applications, but is stable for each category and each application during its life-cycle. The highest level (CUO-M) is, by construction, dependent on the application, because it is constructed (or specialized) with respect to the CAB-M.

4 AMF-C as CUO Model

An appropriate CUO model should fulfill three main objectives. Firstly, it organizes the software structure to improve implementation, portability and maintenance. Secondly, it helps identify the functional components, which is essential during the analysis and design process. Its third role is to facilitate the understanding of a complex system, not only for designers, but also for end-users.

AMF-C (the French acronym for Collaborative Multi-Faceted Agent) [Tarpin-Bernard-97] is our proposal for the CUO model for collaborative software which fulfills all these objectives. AMF-C is a generic and flexible model that can be used with design and implementation tools. It includes a graphical formalism that expresses the structures of software, and a run-time model that allows dynamic control of interactions.

The current trend in software engineering is to identify design patterns [Gamma-95], which help developers to share architectural knowledge, help people to reuse architectural style, and help new developers to avoid traps and pitfalls traditionally learned only as a result of costly experience.

AMF proposes a multi-faceted approach, in which each facet can be a pattern. Each new identified behavior which seems to be reusable, can be formalized as a new facet, i.e. a new pattern. AMF also proposes a very powerful graphical formalism which helps understand complex systems. This formalism is used as a design tool by editors and builders. It represents agents and facets with overlapped boxes, communication ports with rectangles which contain the associated services, and control administrators with symbols which express their behavior.

5 Evolution during the life-cycle

During application life-cycle, users progressively change their perception of the system and their use. The cooperative application could, during its use, take into account requirements concerning evolution expressed explicitly or implicitly by its users. This is particularly the case in the context of Capillary CSCW (cooperative work using handheld devices) [David-03, Brown-02] in which behavior evolution is more important related to context condition (connected or disconnected work) and the device used. To take into account this evolution in an explicit way, new scenarios can be presented which upgrade or extend initial the COB model. The Cooperative Application Behavior model evolves smoothly and it is important to be able to take this evolution into account. This evolution can vary in importance and its impact can be taken into account in different ways:

- If the evolution is within the scope of the system, i.e. these adjustments have been imagined and they are at the disposal of the user in the "configuration panel" (use of different interaction modalities, modification of awareness, choice of different WYSIWIS relaxation, plasticity of user interface, etc.)
- If the evolution is more important and leads to modification of the CAB model, two different solutions are possible:

- Change of interaction pattern with the same algorithmic behavior: this evolution is relatively easy and can be performed by the end-user (i.e. by visual programming using AMF-C graphic formalism),
- Change is more important also with algorithmic behavior evolution: in this case a developer intervention seems inevitable.

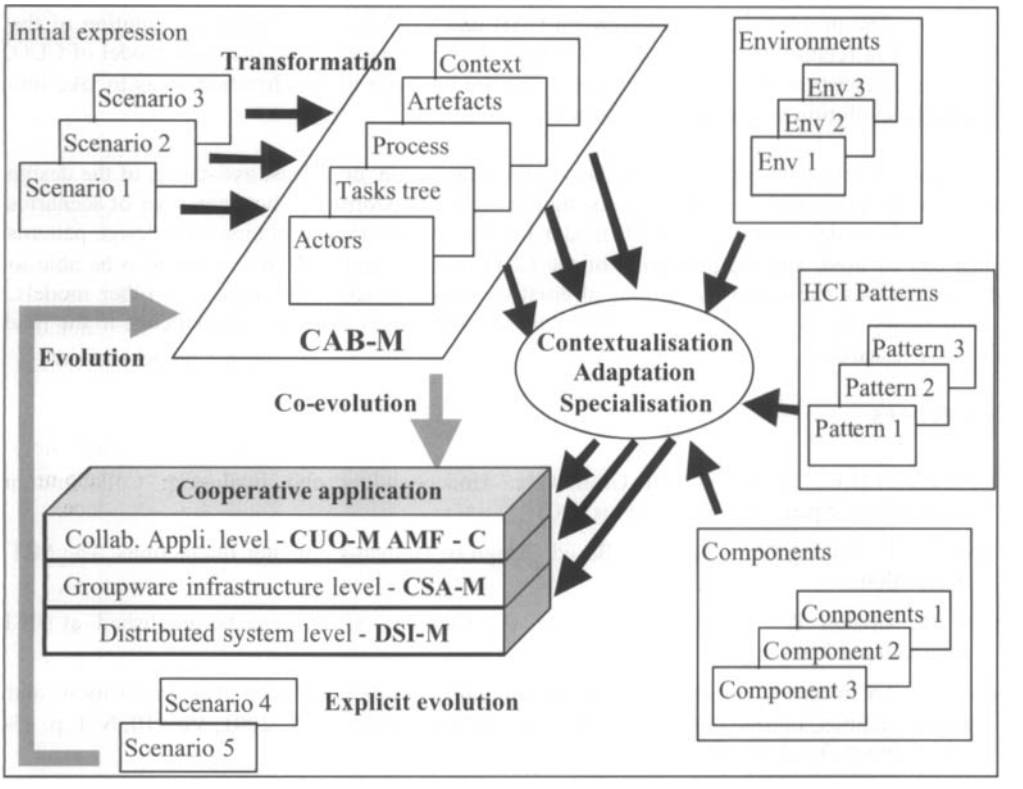

Figure 1: model-based process for design and evolution.

To take into account this re-configurability, adaptability and flexibility, we need new scenarios, which can also replace or modify existing ones. Their processing leads to modification of the CAB model and has an impact on the CUO model. The evolution can either be implemented on the existing IT support or this support can evolve too. We expect that the latter evolution, which concerns the CSA and DSI models, will seem to be out of scope of the evolutions which can be taken into account dynamically. To be able to modify dynamically CUO, we need to have at our disposal a meta-model of the CUO model to create dynamically new AMF-C agents in the relation with the re-configuration of the COB model. This co-evolution can be implemented either by the end-user himself or at least expressed by him, on his directives or under his control, by the developer.

6 Conclusions

We propose a development process which is based on a scenario-based approach and a collection of well-organized models. The synthesis of collected scenarios leads to the definition of a CAB model. These scenarios contribute to the elaboration of task and process models which are the

components of the CAB model. It synthesizes different scenarios and constructs a comprehensive model of the cooperative application with its users, processes, tasks, data and contexts.

Initial design is in this way properly assisted and the evolution during the application life-cycle needs new concepts. In actual fact, the CAB model seems to be appropriate for the evolutions. Lower level software models (CSA and DSI) are generic enough to be able to support these evolutions. The problem is mainly between CAB and CUO models. Dynamic evolution of the CUO model in relation to changes in the CAB model requires access to the meta-model of CUO, to be able to construct the new CUO model. We are working in this direction so as to take into account the evolution in a comprehensive way.

The second direction of our research relates to the automation, or at least assistance, of the design and evolution processes. We aim at assisting the initial transformation between a set of scenarios and the CAB model, between the CAB model and the CUO model in relation with design patterns which can be used, and the projection of the CUO on CSA and DSI. We would also be able to assist evolutions expressed by new or alternative scenarios and their impact on other models. Consideration of this joint evolution (known as co-evolution) is a significant challenge in the field of CSCW research.

References

Andriessen J.H.E., Working with Groupware: Understanding and Evaluating Collaboration Technology, Springer, CSCW Series 2003

Carroll J. M., Making Use : Scenario-Based Design of Human-Computer Interactions, The MIT Press 2000

David B., Chalon R., Vaisman G., Delotte O., Capillary CSCW, to be published at HCI International 2003

Dewan P., An Integrated Approach to Designing and Evaluating Collaborative Applications and Infrastructures, Journal os Computer-Supported Cooperative Work, 2001, Vol: 10, N°1, p. 75-111, Kluwer Academic Publishers

Ellis C.A., Wainer J., A Conceptual Model of Groupware, ACM CSCW'94 Conference, p. 79-88, ACM Press

Gamma E., Helm R., Johnson R., Vlissides J. (1995), Design Patterns : Elements of Reusable Object-Oriented Software, Addison Wesley, Reading, MA.

Paterno F., Model-Based Design and Evaluation of Interactive Applications, Applied Computing Series, Springer, 2000

Seffah, A.and Forbrig, P. Multiple User Interfaces: Towards a Task-Driven and Patterns-Oriented Design Model, Proceedings of DSVIS 2002 (Rostock, Germany, June 2002) p. 160-174, to appear in Lecture Notes in Computer Science.

Tarpin-Bernard F., David B.T., AMF a new design pattern for complex interactive software ?. International HCI'97, Elsevier, ISBN: 0 444 82183 X. San Francisco. Vol. 21B. p. 351-354. August 1997.

Tarpin-Bernard F., David B.T., Primet P., Frameworks and patterns for synchronous groupware : AMF-C approach. IFIP Working Conference on Engineering for HCI : EHCI'98. Kluwer Academic Publishers. Greece. p. 225-241. September 1998

Configuring the Design Process – Applying the JIET Design Process Framework

Helmut Degen

Vodafone Holding GmbH
Global Products & Services
Berger Allee 25
D-40213 Duesseldorf,
Germany
Emails:
helmut.degen@vodafone.com
hdegen@acm.org

Sonja Pedell

The University of Melbourne
Department of Information
Systems
111, Barry Street
3010 Victoria, Australia
Emails:
spedell@studentmail.dis.unim
elb.edu.au, pedell@acm.org

Stefan Schoen

Siemens AG
Corporate Technology
Otto-Hahn-Ring 6
D-81730 Munich, Germany

Email:
stefan.schoen@siemens.com

Abstract

To be able to offer appropriate and efficient design processes, the JIET design process framework developed for web based e-business applications is presented. By means of a configuration model, the appropriate design process can be tailored respective to the conditions of industrial Tippfehler projects. The configuration model consist of four parameters to identify a suggest process: availability of users (or not), existing user interfaces (or not), available resources, and intended quality standard. The JIET design process framework and one part of the configuration model is outlined.

1 Introduction

Several publications deal with design processes, also known as usability engineering processes (e.g. 2002; Constantine & Lockwood, 1999; Beyer & Holtzblatt, 1998). These design processes seem to be carried out from one starting point in a very similar way. However, in some industrial projects user interfaces already exist, while in others they do not. Sometimes users can be involved, sometimes not. Also the goal, i.e. the intended quality of user interfaces, may differ from project to project. One of the key questions in industrial projects is: Which parts of the design process should be carried out, if you have only 70 %, 50 %, or even 30 % of necessary resources available?

The published design processes do not consider different starting conditions explicitly and directly applicable for tailoring user interface projects in industry with a short time to market. Beyer & Holtzblatt offer an heuristic approach to tailor the process, but this is focussed on the contextual inquiry method only (Wood, 2002). Constantine (2000, 2002) offers some methods around his model based approach, but it is only implicitly described and therefore not directly applicable.

Therefore a user interface design process that takes these different conditions into account, and that can be customized to the needs of individual projects, is needed (Vredenburg, Isensee & Righi 2002). How does a design process look like when different steps have to be left out because of restricted time or resources, or where steps have already been carried out? The idea is to develop a design process framework together with a configuration model. With this configuration model, an appropriate design process can be tailored according to existing project conditions.

This paper is structured in four further sections: In the following section related work is sketched. Then the development of the JIET Design process is described. The third section describes the

JIET design process framework, and why it is a usable design process. The last section summarizes and shows future work.

For better understanding, we would like to introduce some basic terms. There is a *provider* of an e-business solution, e.g. Siemens. The *client* is the sponsor of the user interface designers and often responsible for developing the application, e.g. Boston Consulting Group. The *users* are the persons who use the e-business application. User interface designers are people with two core competencies: The *interaction designer* is responsible for the interaction architecture as well as usability work. The *visual designer* is responsible for finding and designing the visual language of the user interface.

The presented design process framework was elaborated during industrial practice in an inductive way over the last 2.5 years. We followed the following steps: 1. New e-business methods modules were developed or existing modules were adapted to the specific requirements of e-business projects. We optimized them during many e-business projects. 2. The structure of the new design process framework was reorganized. In order to find a structured framework, a literature research was conducted. The most relevant result of the literature research was the process model of ISO 13407. We used this model as a starting point and enhanced it according to the requirements of e-business design processes. 3. This design process framework was applied in every new e-business project followed by a systematic evaluation and improvement. 4. Within a 2-day process workshop, eight usability experts from the Competence Center "User Interface Design" (Siemens AG, Munich) reviewed and improved the process framework.

2 JIET Design Process Framework

2.1 Overview

We call this new design process framework for e-business applications "JIET Design Process Framework" (herein called JIET = Just in ETime). Goal of the framework is to offer a set of method modules in order to be able to deliver user interfaces with the highest possible quality according to project conditions and goals. The JIET framework consists of nine phases that are shortly described (see Table 1).

Project setup (phase 0) is the start of the user interface design project. The user interface designers obtain an understanding of the project, the client, the audience, the application and the expected kind of support regarding the intention of involving user interface experts. The starting conditions of the project will be identified, and the decision process will be tailored. After obtaining the signed contract, the audience identification (phase 1) is the next phase. The purpose of the audience identification phase (1) is to understand the business of the provider and to identify the user roles. Within the use context analysis (phase 2), the use context of the target audience will be analyzed. For each user role we identify job-related goals, tasks, core tasks, scenarios, processes, tools, and content objects. The purposes of the requirements gathering phase (3) is to gather use scenarios and requirements for each scenario and user role. The purpose of the user interface design phase (4) is to create the user interface architecture. Within the evaluation phase (5), the designed user interfaces are evaluated. The results of the evaluation phase are considered to improve the user interface architecture within the redesign phase (6). All existing results of the design phase (4) will be improved. Within the design specification phase (7) the design specification is compiled based on the results of the last design phase. In the last phase (8 Process Improvement), the applied design process is analyzed, and improvement potentials of applied method modules are identified, based on the results of the usability evaluation (phase 6).

Table 1: JIET Design Process Framework

Phase	Available method modules
0 Project Setup*	0.1 Project Characteristics (Client)** 0.2 Project Planning** 0.3 Proposal**
1 Audience Identification*	1.1 Enterprise Profile (Provider)** 1.2 Business Process Profile** 1.3 Business Customer Profile (Provider) 1.4 Private Customer Profile (Provider) 1.5 Supplier Profile .(Provider) 1.6 Competitor Profile (Provider)
2 Use Context Analysis*	2.1 Enterprise Profile (User)** 2.2 Business Process Profile (refined)** 2.3 Job Profile (User)** 2.4 Activity Profile (User)** 2.5 User Profile (User) 2.6 Competitor Profile (User) 2.7 Process Profile (User) 2.8 Content Profile (User) 2.9 Tool Profile (User) 2.10 User Role Profile (User)* 2.11 Usability Goals
3 Requirements Gathering*	3.1 NOGAP** 3.2 Card Sorting
4 User Interface Design*	4.1 Interaction Architecture** 4.2 Content Architecture 4.3 User Interface Architecture** 4.4 Prototype
5 Evaluation	5.1 Usability Check 5.2 Optimization Profile** 5.3 Usability Inspection 5.4 Usability Test (summative) 5.5 Usability Test (formative)
6 Redesign	6.1 Optimized Interaction Architecture** 6.2 Optimized Content Architecture 6.3 Optimized User Interface Architecture** 6.4 Optimized Prototype
7 Design Specification	7.1 Design Specification**
8 Process Improvement	8.1 Process Control**

Legend

* This is a duty phase in each user interface project and should be applied in any case

** This method module is a duty module within its phase and should be applied in any case

2.2 Configuration Model

Each project has several starting conditions. In order to be able to offer the most efficient design process for certain project conditions an adapted design process is necessary. For this purpose, a configuration model is required. One particular task in creating a configuration model means to identify all of these starting conditions that help to select an appropriate design process. These relevant starting conditions are herein called *starting parameters*. All in all, four parameters could be identified that enable us to select the appropriate design process. Two of the four parameters are basis parameters. One parameter stands for the status of user interfaces (do they already exist

or not), the other for the availability of users during the process (they are involved or not). With these two starting parameters, we have four different configuration variants, shown in Table 2.

Table 2: Configuration Variants, based on two basis parameters

	User Interfaces do not exist	User Interfaces exist
Users are not available	Configuration Variant 1	Configuration Variant 3
Users are available	Configuration Variant 2	Configuration Variant 4

Table 3: Configuration Variant 3

Phases	Basis Process 1: Usability Assessment Process	Small	Medium
Project Setup	0.1 Project Characteristics (Client)	ID,C	ID,C
	5.1 Usability Check	ID	ID,VD
	0.2 Project Planning	ID	ID,VD
	0.3 Proposal	ID	ID,VD
Audience Identification (Provider)	1.1 Enterprise Profile (Provider)	ID,P	ID,P
	1.2 Business Process Profile	ID,P	ID,P
	1.3 Business Customer Profile (Provider)	ID,P*	ID,P*
	1.4 Private Customer Profile (Provider)	ID,P*	ID,P*
	1.5 Supplier Profile (Provider)	ID,P*	ID,P*
Use Context Analysis (User)	2.1 Enterprise Profile (User)	ID,P	ID,VD,P
	2.2 Business Process Profile (refined)	ID,P	ID,VD,P
	2.3 Job Profile (User)	ID,P	ID,VD,P
	2.4 Activity Profile (User)	ID,P	ID,VD,P
	2.10 User Role Profile (User)	ID,P	ID,VD,P
Evaluation	5.2 Optimization Profile	ID,P	ID,VD,P
	5.3 Usability Inspection		D,VD,P**
	Basis Process 2: User-Centered Design Process		
Audience Identification (Provider)		ID,P****	ID,P****
Use Context Analysis (User)		ID,P****	ID,VD,P****
Requirements Gathering		ID,P****	ID,VD,P****
User Interface Design	4.1 Interaction Architecture	ID	ID,VD
	4.2 Content Architecture		ID,VD
	4.3 User Interface Architecture	ID	ID,VD
Evaluation	5.3 Usability Inspection		ID,VD,P
Redesign	6.1 Optimized Interaction Architecture		ID,VD
	6.2 Optimized Content Architecture		ID,VD
	6.3 Optimized User Interface Architecture		ID,VD
Design Specification	7.1 Design Specification		ID,VD
Required Resources		Low	Medium
Achievable Quality		Medium -	Medium o

Legend

* Will be carried out, if this profile is relevant for the project.

** Will be carried out, if the usability check (phase 0) established that the user interface quality is sufficient for a usability test.

*** Will be conducted, if a separate order is requested for the second basis process (user centered design);

**** Will be carried out, if additional information is required.

ID: Interaction designer involved; VD: Visual designer involved; C: Client involved; P: Provider involved; U: User involved

For each configuration variant, the configuration model offers different alternatives. Each of the further alternatives is identified by the third and fourth parameter: Resources and Quality. To describe how to apply this configuration model, we selected the configuration variant 3 (user interfaces already exist, but users are not available). A usability test is required to achieve a high quality standard. Since users are not available we cannot conduct such tests and therefore the achievable quality can be only "medium" for variant 3. Therefore, we offer two alternatives (see Table 3), a "Small" and a "Medium" alternative.

The "small" version begins with the project setup (phase 0). To ensure that the existing user interfaces have a necessary quality standard, a short usability check has to be carried out. If the quality is too bad we cancel the rest of the usability assessment and suggest to design new user interfaces. If the usability quality achieves a certain quality standard, we carry out the usability assessment. After setting up the project the target audience (phase 1) will be identified. We carry out a use context analysis (phase 2). These results are used for the evaluation (phase 5). Within the "small" variation the evaluation is an open interview (5.2 Optimization Profile). In the "medium" variation a usability inspection is carried out (5.3 Usability Inspection). In both cases the interview or inspection partners are representatives of the provider.

The second basis process uses the results of the usability assessment. The purpose is to improve the user interfaces. The first four phases (Project Setup, Audience Identification, Use Context Analysis, Requirements Gathering), or parts of it, are only repeated if necessary information is missing. Within the "small" variation, the user interface design consists of the interaction and the user interface architecture. Within the "medium" variation, the content architecture will be carried out in addition. Since we do not evaluate the user interfaces by means of a usability test, we do not need a (interactive) prototype. Only the "medium" version consists of another evaluation (phase 5) and redesign (phase 6) phase. If required a design specification (phase 7) will be compiled.

Apart from these different method modules, competence of the people is a relevant distinction between the "small" and "medium" version . In the "small" version, only an interaction designer is involved, because of the intended quality and available resources, whereas in the "medium" version, a visual designer is involved in addition.

3 Conclusions and Future Work

Compared to ISO 13407 user-centered design process, two important phases are added: The "Audience Identification" (Phase 1) because in e-business projects the target audience is not obvious. Another new phase is the "Process Improvement" (Phase 8), which is applied to evaluate and improve the applied design process.

During industrial practice, the JIET design process framework should be improved. We assume the framework itself can be applied to other domains outside e-business.

4 References

Beyer, H. & Holtzblatt, K. (1998): Contextual Design. San Francisco, USA: Morgan Kaufmann.

Constantine, L. L. & Lockwood, L. D. (1999): Software for Use: Reading, USA: Addison-Wesley, ACM Press.

Constantine, L. (2000): Cutting Corners: Shortcuts in Model-Driven Web Design. Retrieved February 10, 2003, from http://foruse.com/articles/shortcuts.htm.

Constantine, L. (2002): Process Agility and Software Usability: Toward Lightweight Usage-Centered Design. Retrieved February 10, 2003, from http://foruse.com/articles/agiledesign.htm.

Vredenburg, K.; Isensee, S. & Righi, C. (2002): User-Centred Design. Upper Saddle River, USA: Prentice Hall.

Wood, S. (2002): Targeted Contextual Design — Scope Your Project. Document sent by email at 12th of November 2002.

Finding Decisions Through Artefacts

Alan Dix, Devina Ramduny, Paul Rayson, Victor Onditi,
Ian Sommerville and Adrian Mackenzie

Lancaster University, Computing Department
Lancaster, LA1 4YR, UK

alan@hcibook.com, devina@comp.lancs.ac.uk, paul@comp.lancs.ac.uk,
v.onditi@lancaster.ac.uk, is@comp.lancs.ac.uk, a.mackenzie@lancaster.ac.uk

http://www.hcibook.com/alan/papers/HCII2003-artefacts/

Abstract

This paper addresses the use of artefacts as a resource for analysis. Artefacts are particularly useful in situations where direct observation is ineffective, in particular for infrequent activities. We discuss two classes of techniques: focusing on the 'artefact as designed' as a means of recovering designers' explicit and implicit knowledge and 'artefacts as used' as a means of uncovering the trail left by currently inactive processes. These techniques have been applied using a meeting capture system and meeting minutes as the artefact resources. This is part of a wider study to understand the nature of decisions and so reduce rework.

1 Introduction

The ethnographic literature is full of the importance of artefacts as the means with which individuals represent, mediate and negotiate work in collaborative settings (Hughes, O'Brien, Rouncefield, Sommerville & Rodden 1995). This is also recognised in approaches such as distributed cognition (Hutchins, 1990) and situated action (Suchman, 1987) as well as some more traditional cognitive models (Howes & Payne, 1990). In our previous work, we have studied the way in which artefacts in their setting act as *triggers* for action (Dix, Ramduny & Wilkinson, 1998) (Dix, Ramduny & Wilkinson, 2003) and *placeholders* for formal and informal processes (Dix, Wilkinson & Ramduny, 1998). In related work, we emphasised the centrality of artefacts as the focus of work and as the locus of communication *through the artefact* or feedthrough (Dix, 1994).

Because of this we have proposed various forms of artefact-centred analysis to run alongside more direct methods of observation (Dix, Ramduny, Rayson & Sommerville, 2001). We consider both:

- the artefact *as designed* – looking at the ways in which the explicit and implicit knowledge of the designer are exposed in artefacts
- the artefact *as used* – looking at the way on which people have appropriated, annotated and located artefacts in their work environment

Artefact centred sources are particularly useful where an activity occurs or is only active infrequently so that direct observation may fail to record any instance or part of the activity at all. In the Tracker[1] project we are seeking to understand the nature of decisions in teams and

[1] http://www.comp.lancs.ac.uk/computing/research/cseg/projects/tracker/

organisations (Rayson et al., 2003); in particular the way past decisions are acted on, referred to, forgotten about and otherwise function as part of long term organisational activity. In one of the ethnographic studies we carried out, we found that 'real' decisions were made between meetings or implicitly assumed, but rarely explicitly voiced during official meetings. Furthermore, interactions between decisions take place over periods of month or years. In other words, they have exactly the properties that make artefact-centred approaches attractive.

2 The artefacts as a resource

Like the fossil left where the soft parts of the body have decomposed, artefacts act as a residual record of work done and work in progress. In and of themselves they form a resource for analysis. Furthermore, just like the palaeontologist looking at fossils, there are a variety of circumstances in work domains where the 'soft tissue' of lived work, the ephemeral actions and words, are difficult or impossible to collect and so the matrix of artefacts that remains needs to be interpreted.

This may be because the actions have already taken place and so the physical remains are our only resource. In the Tracker project we have access to a corpus of meeting minutes. The meetings have long past; we cannot go back and observe what happened; at best we can interview some of the participants; but the formal minutes remain – fossils of the moment. We will return to these formal minutes later.

Perhaps more fundamentally there are some classes of human activity that direct observation cannot, or cannot easily, capture. Where a class of activity is frequent and short lived we can expect that periods of direct observation, such as ethnographic studies, will completely capture some instances of the activity from end to end. Where activities are longer lived, direct observation can at best hope to capture aspects of the activity at different points and so piece together the complete story from parts. But where a class of activity happens infrequently or is only active infrequently, direct observation can only record the activity if it happens to overlap with one of the infrequent episodes.

However, these activities, even when inactive must in some way still have a representation within the organisational ecology: in people's memories *and* in physical or electronic artefacts. The 'and' in the previous sentence is not just in the sense that both will be present, but in the more holistic recognition that the interpretation of artefacts is itself invested within the human understanding of the context. Artefacts tell a story to the extent that they invoke stories. To some extent as analysts we may understand the contexts well enough to 'read' artefacts, in others the artefacts can form the prompts to evoke memories during formal and informal interviews.

3 The artefact as designed

Long lasting artefacts: tools, procedures, documentation, buildings, organisational structures, have all by explicit action been 'designed'. As we know these designs can often fail and so are not paradigmatic. However, they are a powerful resource embodying the knowledge, skills and assumptions of the original designer. We call this *archaeologically-inspired artefact analysis*. An archaeologist will look at the artefacts produced by long-dead civilisations and by considering the design infer the patterns of use, work and social activity that surrounds those artefacts. This process is problematic as we may draw tenuous conclusions from meagre evidence, but is in fact more robust as a contemporary technique as we are in a better position to understand the target context and may also be in a position to use this as a resource in participative critique.

In the early stages of the Tracker project, we reviewed a number of meeting support systems. We analysed in greatest detail TeamSpace[2] (Richter et al., 2001), which is related to the very successful Classroom2000 (eClass) system (Abowd, 1999). In the version of TeamSpace we tested, we found various classes of context assumptions. Some are explicitly embedded in the software; for example, TeamSpace requires meetings to be scheduled. Some are explicit in the documentation but not enforced; for example, the suggestion that a facilitator is necessary. Some are implicit in the software; for example, if you stop and then restart a meeting, the audio recording for part of the meeting is lost, implicitly assuming meetings do not break and reconvene.

4 The artefact as used

In previous work we have focused especially on the fact that artefacts encode the state and trigger action not just by their explicit content or significance, but also by their disposition in the environment. A piece of paper at a particular location on the desk may mean "file me"; in another location, perhaps in a straight pile means "in progress"; and on the same pile, but higgledy-piggledy at an odd angle means "to be read". By taking an office at the end or beginning of a day we can use these artefacts to tell the story of the activities that are, in a temporal sense, passing through the office at that moment. Most significantly this includes activities which are not currently captured in the 'official' systems or whose state is indeterminate or intermediate between 'official' stages. We call this *transect analysis* as it is similar to the field biologist's use of a transect through an ecosystem such as a shoreline.

Unfortunately meetings are an extreme case of 'clean desk policy'. The documents and artefacts are removed from the room with the participants – the only remnant of the meeting is the explicit records and the changed memories and attitudes of the participants.

The one obvious artefact that is left behind by a meeting is the formal minutes. These are problematic as they are not a record of *what happened* at the meeting, but rather a *sanitised account* prepared for a purpose, by an individual. Although problematic the minutes are significant as they are the foci by which the participants agree (or are forced to agree) to a fiction that in some way legitimises future actions. In the extreme, in certain legal situations, minutes of meetings are created which never occurred – quite literally legitimising the desired end state by an agreed legal fiction of the process.

To some extent the artificial nature of the formal minutes reflects the artificial nature (in the sense of artifice) of collaborative activity. Ethnomethodology makes a strong focus on the accountability of individuals – that they can make stories (accounts) about their actions that legitimise them socially. We have to read formal minutes carefully, more like an *historical document*, written by someone, for a purpose, but nonetheless exposing aspects of the real process.

As noted our focus is on decisions and this has proved even more problematic. In ethnography of actual meetings one of the marked results was the fact that decisions did not 'happen' in the meeting. This is not to say that formal minutes would not record decisions (or their consequences), but that there are not clear points of decision-making. Instead decisions have either clearly been made prior to the meeting and are merely brought into the meeting to validate them, or alternatively decisions are 'made' implicitly by simply discussing an issue that the minute taker reads later as a particular outcome.

[2] http://www.research.ibm.com/teamspace/

This problematic nature is also evident in the minutes themselves. Formal minutes do not explicitly record 'decisions' but instead either note agreed statements or 'actions', usually relating to formally numbered items in the meeting. Whereas formal actions are explicitly marked there is no such explicit marking for decisions (or related topics such as options, issues etc.). Instead an extensive manual analysis was required to identify salient features.

When the analysis started we had some discussion about the level of structure required in the analysis. The minutes we studied themselves had a fairly consistent formal structure: date, participant list, numbered items, comments and listed actions against each item. Also there are a number of ontologies of decision making from the design rationale and decisions support literature (e.g. IBIS (Conklin & Begeman, 1988), QOC (MacLean, Young, Bellotti & Moran, 1991), DRL (Lee & Kai, 1991). Based on these a database structure was created to record decisions, actions, issues and relations between them. So, for example, a decision would have associated actions, actions would have responsible persons and optionally a deadline.

As the analysis proceeded, it became increasingly clear that the reality of the 'formal' minutes was, perhaps not surprisingly, far less structured and far more ad hoc than our predefined structure. Even the explicit 'actions' sometimes turn out to be more comments or statements of intent and some actions are not marked as such. Decisions are far more complicated as they are sometimes explicit in the text and sometimes inferred from context (e.g. an action presupposes a decision to take action).

In the end the rigid structure has been dropped, except for the record of the explicit structure of the minutes themselves, and the analysis uses a simple recording (in a database to make it amenable to search and analysis) of 'things' and relations between them. With this more flexible structure, the analyst is no longer restricted to a fixed repertoire of concepts. While the analyst is reading the minutes, if she feels that any issue ought to be recorded she can now easily do so by adding a 'thing' with as many named attributes as desired. Only a short title/description, link to the raw transcript and 'type' field are required. The last of these is to enable the recording of terms such as 'decision', 'action' and the like, but not constrained to a predetermined vocabulary. The aim is to see an ecologically valid ontology *emerge* from the ongoing analysis.

5 Conclusion

We have seen using the example of TeamSpace, how it is possible to use the artefact as designed as the analytic resource. Similarly, our analysis of minutes is an example of using artefacts as used. In both cases, the permanent record embodied in artefacts has given us an understanding of the nature of decisions which otherwise prove elusive to direct observation.

6 Acknowledgement

The Tracker project is supported under the EPSRC Systems Integration programme in the UK, project number GR/R12183/01.

7 References

Abowd, G. (1999). Classroom 2000: An Experiment with the Instrumentation of a Living Educational Environment. *IBM Systems Journal, Special issue on Pervasive Computing*, 38(4) 508-530.

Conklin, J., & Begeman, M. L. (1988). gIBIS: A hypertext tool for exploratory policy discussion. In *Proceedings of the Second Conferences on Computer-Supported Co-operative Work*. New York, USA. ACM Press, 140-152.

Dix, A. (1994). Computer-supported cooperative work - a framework. In D. Rosenburg and C. Hutchison (Eds.) *Design Issues in CSCW* (pp 23-37). Springer Verlag.

Dix, A., Ramduny D., & Wilkinson, J. (1998). Interaction in the Large. *Interacting with Computers*, 11(1), 9-32.

Dix, A., Wilkinson, J., & Ramduny, D. (1998). Redefining Organisational Memory – artefacts, and the distribution and coordination of work. In *Understanding work and designing artefacts* (York, 21st Sept., 1998). http://www.hiraeth.com/alan/papers/artefacts98/

Dix A., Ramduny, D., Rayson, P., & Sommerville, I. (2001). Artefact-centred analysis - transect and archaeological approaches. In *Team-Ethno Online, Issue 1 - Field(work) of Dreams*. Lancaster University, UK. http://www.teamethno-online.org/Issue1/Dix.html

Dix, A., Ramduny-Ellis, & D., Wilkinson, J. (2003). Trigger Analysis - understanding broken tasks. In D. Diaper & N. Stanton (Eds.) *The Handbook of Task Analysis for Human-Computer Interaction*. Lawrence Erlbaum Associates.

Hughes, J., O'Brien, J., Rouncefield, M., Sommerville, I., & Rodden, T. (1995). Presenting ethnography in the requirements process. In *Proceedings of IEEE Conf. on Requirements Engineering, RE'95*. IEEE Press, 27–34.

Howes, A., & Payne, S. (1990). Display-based competence: towards user models for menu-driven interfaces. *International Journal of Man-Machine Studies*, 33, 637-655.

Hutchins, E. (1990). The Technology of team navigation. In Gallagher, J., Kraut, R. and Egido, C., (Eds.) *Intellectual teamwork: social and technical bases of collaborative work*. Lawrence Erlbaum.

Lee, J., & Lai, K.-Y. (1991). What's in Design Rationale? *Human-Computer Interaction*, 6(3&4), 251-280.

MacLean, A., Young, R., Bellotti, V., & Moran, T. (1991). Questions, options and criteria: Elements of design space analysis. *Human-Computer Interaction*, 6(3 & 4), 201-250.

Suchman, L. (1987). Plans and Situated Actions: The problem of human–machine communication. Cambridge University Press.

Rayson, P., Sharp, B., Alderson, A., Cartmell, J., Chibelushi, C., Clarke, R., Dix, A., Onditi, V., Quek, A., Ramduny, D., Salter, A., Shah, H., Sommerville, I., & Windridge, P. (2003). Tracker: a framework to support reducing rework through decision management. To appear in *5th International Conference on Enterprise Information Systems ICEIS2003*, Angers, France, April 23-26, 2003.

Richter, H., Abowd, G., Geyer, W., Fuchs, L., Daijavad, & S., Poltrock, S. (2001) "Integrating Meeting Capture within a Collaborative Team Environment". In *Proceedings of the International Conference on Ubiquitous Computing, Ubicomp 2001*, Atlanta, GA, September, Springer, 123-138.

The Constrained Ink Metaphor

Björn Eiderbäck
CID/KTH
SE-100 44 Stockholm
Sweden
bjorne@nada.kth.se

Sinna Lindquist
CID/KTH
SE-100 44 Stockholm
Sweden
sinna@nada.kth.se

Bosse Westerlund
CID/KTH
SE-100 44 Stockholm
Sweden
bosse@nada.kth.se

Abstract

In this paper we describe a novel metaphor for developing interactive computer applications, *the constrained ink metaphor*. Crucial to the development of the constrained ink was an aim to find simple and natural means for defining and implementing interaction among persons. We will describe how we was lead to considering this metaphor, some basic inks following the metaphor, and finally some typical applications and their impact on the development of the metaphor.

1 Background, Goals and Influences

1.1 The interLiving Project

The interLiving project aims to study and develop, together with families, technologies that facilitate generations of family members living together with the objectives: to understand the needs of diverse families; to develop innovative artefacts that support the needs of co-located and distributed families; to understand the impact such technologies can have on families (Beaudouin-Lafon, 2002, Hutchinson, 2003).

1.2 Goals and Research Questions

A key objective of the interLiving project is to experiment with different design methodologies. We would like to develop better ways of letting the family members directly influence and shape the design of communication technologies we develop together with them.
The *premiere goal* with the particular work described in this paper is: to develop an infrastructure and metaphor that will enable us to build applications were we leave as much us possible open to the co-development with families, even late in the development process. *Secondary goals* are: i) that it should be easy and natural to develop all our intended applications by means of this infrastructure and metaphor; ii) that the metaphor should encourage development of applications that are fun to use (and develop!)
Research Questions are:

- Is it possible to create an infrastructure and metaphor of the type we strive for in the goals?
- For which types of applications is the metaphor well suited and for which types is it not naturally applicable?

1.3 Technology Probes

As inspiration and triggering techniques we have used technology probes. A 'technology probe' combines the social science goal of collecting data about the use of the technology in a real-world setting, the engineering goal of field-testing the technology and the design goal of inspiring users (and designers) to think of new kinds of technology (Beaudouin-Lafon, 2002 Chapter 2). The

probe that influenced the development of the applications we currently are working on most is *The Message Probe*, a simple application that enables members of a distributed family to communicate with digital notes using a pen and tablet interface. Already at early stages of the development of applications inspired by these probes we realized that we required something that both was fun to use and easily adoptable to various and changing requirements. This in turn led us to the development of *The Constrained Ink Metaphor.*

1.4 Influencing Approaches

There are of course a lot of achievements in the history that has inspired, or at least influenced, our development. For instance Ivan Sutherlands pioneering work on Sketchpad (Sutherland, 1963), the NLS system in the SRI project (Engelbart, 1975), the very direct manipulated A Reality Toolkit (ARK) (Smith, 1987), editors for drawing and animation like Macro Mind Director, Calendaring facilities (Beaudouin-Lafon, 2002), and the more recent KidPad (Benford, 2000). We have also been inspired by work done in CSCW and design patterns (Eiderbäck 2001).

2 Ink of Various Kind and their Usage

2.1 The Metaphor: What, Why, and How

The constrained ink metaphor is a novel metaphor for developing interactive computer applications. The idea of it sprung from an attempt to develop a common base for a message central and a distributed shared drawing editor, intended for communication between family members possible living in different households. In the former case we focus on the same place different time aspects were we want to provide for submitting shared notes visible within certain time frames. In the latter case we focus on same time different place aspects were we for instance want to provide for co-operative drawing, communication and address awareness aspects. Our intention is to enabling communication of both important facts and more informal chatting in a way youngsters, adults, and elder members of the family, computer literate or not, could find useful and "fun"! We discussed the concept together with the families and agreed that it seemed to be promising, useful and fun.

2.2 Ink

Central to the Constrained Ink Metaphor, as its name suggests, is the Ink!

2.2.1 Natural Inks

There are a lot of different types of ink that could be considered natural in the sense that they more or less have their counterparts in the real world. For instance, we have the invisible ink that even a small children most likely have experiences from using a special purposes pen with ink that only appears after one heat the paper it is written on. Another natural ink is the aging ink; actually this is the way all inks work, where the ink slowly disappears from the material it is written on. However in our computerized versions we have speeded up and made the aging more controllable.

The Coloured Ink

As a basis we use ordinary coloured ink, i.e. all inks have a defined colour or texture. On top of this basic ink all the other inks was developed, by applying various constraining schemas that made them behave and response to external events.

The Invisible Ink

The Invisible Ink is the most natural of all the constrained inks.

Context
The user wants to write a note that should be presented at a specified time in the future. Thereafter the note should stay until someone actively removes it.

Problem
How could we provide model providing a means to construct entities that should appear at a specific time in the future? How could we develop a model that fits into and is suitable for all the various applications we are developing within the project?

Forces
The model should be natural to use. The usage of the model should not constrain the process or the interaction. The model should be natural for handling constrained entities of various kinds as graphical one, e.g. lines and ovals, and non graphical ones, e.g. email and speech. It must be feasible to implement the model in software.

Solution
Make a computerized version of an invisible ink. For convenience for programmers incorporate the ink model into the system's ordinary model of drawing with various colours and textures, i.e. it should be possible to use the ink for colouring objects even in "non-ink aware" applications. Therefore separate parts for handling the interaction with the ink from ones handling its behaviour and ones handling its visible appearance. In this way one could easily change or adopt new behaviour to ink and at a very fined grained level control its constraints.

The Aging Ink

The Aging Ink is ink that disappears after a pre-defined time. It works as ordinary ink, but we have speeded up the decaying process and also made it more abrupt.

Context
The user wants to write a note that is valid from the time the note is written until a certain time in the future.

Problem and Forces
The problem forces are the same as for the Invisible Ink but now the entities should disappear after a while instead.

Solution
The solution follows the same lines as the one for the Invisible Ink. With the separation of controlling and behaviour from appearance we only has to replace the constraint controller for one that makes the ink disappear after a certain time, instead of appear as for the former ink.

2.2.2 Generally Constrained Inks

After discussing applications, and reflecting on our earlier prototypes among ourselves but also with our families we considered the ink metaphor in more dept. We realised that the natural inks would not solve all the problems that we intended. We require to entities responding to general events, as someone pushing a button or joining a family's network. Therefore we decided to expand the metaphor further to see if it could be useful even in ways that not have their direct counterparts in ink from the natural world.

2.2.3 Asymmetric Inks

We also want to be able to show things differently, or at different times, at diverse platforms. Sometimes everything should be visible to all users in the same way at other times some parts are not visible to all users or just presented differently to some of them. Entities could even be handled on dissimilar platforms and by different media by various users, i.e. on use speech at a

PDA whereas another user has a graphical platform with a text interface. Therefore we try to investigate the impacts these situations has on the ink and try to develop ink that also are suitable for them.

2.2.4 Inks Intended for Sharing

In some senses we could use the previously described inks for sharing. We have inks visible at all platforms, inks that appear differently for diverse users, etc. However, only relying on these inks makes sharing of artefacts required in a more general sense very clumsy. To address this we have played with inks that could define certain (filled) areas where all other inks painted on the area should be visible by a shared and connected community. Thereby we could easily, within the limits of the constrained ink metaphor, even provide for shared desktops and other means of co-operative work. Therefore we also investigate how this type of ink is usable and fits into the metaphor.

3 Some Typical (Ink Based) Applications

3.1 Type of Application

The applications we currently are working on affect the type of ink required in different ways. In this section we very briefly exemplify of the various types of applications we consider. These considerations are a basis for our further development and exploration of the constrained ink metaphor. Some typical kinds of applications are:

- *Synchronous vs. Non-Synchronous Applications.* There is an obvious difference between synchronous and non-synchronous applications. In the former case communication takes effect momentarily whereas the latter case is more indirect, probably taken its way via some server, storage medium, or alike.
- *Shared vs. Non-Shared Applications.* Another situation we must consider is if the application should be shared, i.e. everyone manipulates a shared set of entities, or non-shared where different users could manipulate their own restrictive set of the entities.
- *(Just) Graphical vs. Multimedia.* Typical shared applications of today also provide for other media than graphics. Examples are telephony over IP, and videoconferences.
- *Sinking Ships.* An archetypical application where different users at certain times sees different parts of the entities or even presented in different ways is the famous game Sinking Ships.

3.2 Applications

Currently we are focusing on two different applications. The InkPad and the Door. We also explore some types of interaction, not central in the other two, in a Pie Diagram framework. In the sense of exploring the constrained ink metaphor the InkPad is the most central and new kinds of ink and constraints are first tested within this application.

3.2.1 A Shared (Drawing) Editor, InkPad

The InkPad is a tool with the main aim to enabling free and non-formal communication among family members of all ages. To support free communication we try to make InkPad an as relaxed environment as possible. The focus on this prototype is on enabling communication of both important facts and more informal chatting in a way both youngsters, adults, and elder members of the family, computer literate or not, could find useful and "fun".

The user could choose ink from any of the previously described types of ink. In this way the user could achieve effects as writing messages and notes that will appear or disappear at specific times. We have also considered other media, such as audio, video, and speech.

3.2.2 Message Central, the Door

We also develop a message central nick named *The Door*, from the first intended placement in the household. The Door prototype is an effort to improve the communication and scheduling of activities among family members. At the start we concentrate on communication between members living in the same household. In this case we use the ink metaphor for controlling and delivering messages.

3.2.3 Pie Diagrams

Pie Diagrams are just like ordinary pop up menus but circular. In particular we investigate how invisible ink could be used to supporting expert users that now the relative location of certain submenus and the items they want to chose. The ink is constrained to only paint a certain sub-pie if the user "fires a certain event", by for instance stopping the movement more than a pre-defined time limit. In such a case the ink reacts by switching from transparent colour to non-transparent ones and thereby makes the pie visible.

4 Conclusions and Future Work

In this paper we have described the *Constrained Ink Metaphor* by describing various forms of (constrained) inks and their usage. We demonstrated that the metaphor is both natural and useful for developing a various set of interactive and distributed applications.
From now on we will investigate the metaphor further by using it to full extent while continuing the development of a various set of applications within the interLiving project.

5 References

Beaudouin-Lafon M., Bederson B, Conversy S., Druin A., Eiderbäck B., Evans H., Hansen H., Harvard Å., Hutchinson H., Lindquist S., Mackay W., Plaisant C., Roussel N., Sundblad Y., Westerlund B (2002). *Co-design and new technologies with family users*, Deliverable 1.2 & 2.2, 2002-09-23.

Benford, S., Bederson, B., Akesson., K., Bayon, V., Druin, D., Hansson, P., Hourcade, J., Ingram, R., Neale, H., O'Malley, C., Simsarian, K., Stanton, D., Sundblad, Y., and Taxen, G (2000). *Designing Storytelling Technologies to Encourage Collaboration Between Young Children.* Proceedings of CHI, pp. 556-563.

Eiderbäck B. (2001) *Object Oriented Frameworks with Design Patterns for Building Distributed Information Sharing*. PhD thesis, ISBN 91-7265-240-3, http://www.nada.kth.se/~bjorne/th/.

Engelbart, D.C. (1975), *NLS tele-conferencing features: the journal and shared-screen telephoning*, in Proceeding Fall COMPCON, Sept. 1975.

Hutchinson, Mackay, Westerlund, Bederson, Druin, Plaisant, Beaudouin-Lafon, Conversy, Evans, Hansen, Roussel, Eiderbäck, Lindquist, Sundblad. (2003). *Technology Probes: Inspiring Design for and with Families. Proceedings of ACM CHI 2003.*

Smith, R (1987). *Experiences with the alternate reality kit: an example of the tension between literalism and magic*, Proceedings of CHI + GI.

Sutherland, I (1963). *SketchPad: A man-machine graphical communication system*, AFIPS Spring Joint Computer Conference, No 23.

Meta—Design: Beyond User-Centered and Participatory Design

Gerhard Fischer

University of Colorado, Center for LifeLong Learning and Design (L3D)
Department of Computer Science, 430 UCB
Boulder, CO 80309-0430 – USA
gerhard@cs.colorado.edu

Abstract

Meta-design characterizes objectives, techniques, and processes for creating new media and environments that allow the owners of problems to act as *designers*. A fundamental objective of meta-design is to create socio-technical environments that empower users to engage in informed participation rather than being restricted to the use of existing systems. The seeding, evolutionary growth, reseeding model is a process model that supports meta-design. We have explored and assessed meta-design approaches in the development of innovative computational environments and in our teaching and learning activities.

1 Introduction

Our research interest is in designing the social and technical infrastructures in which new forms of collaborative design can take place. For most of the design domains that we have studied over many years (ranging from urban design to graphics and software design) [Arias et al., 2000], the knowledge to understand, frame, and solve problems is not given, but is constructed and evolved during the problem-solving process.

2 Design Time and Use Time

In all design processes, two basic stages can be differentiated: design time and use time. At *design time*, system developers (with or without user involvement) create environments and tools. In conventional design approaches they create complete systems. At *use time*, users or "stakeholders" use the system but their needs, objectives, and situational contexts can only be anticipated at design time, thus, creating a system that often requires modification to fit the user's needs. In order to accommodate unexpected issues at use time, systems need to be underdesigned at design time. *Underdesign* in this context does not mean less work and demands for the design team, but is fundamentally different from creating complete systems. The primary challenge of underdesign is in developing environments and not the solutions allowing the "owners of problems" at use time to create the solutions themselves. This can be done by providing a context and an interpretive background against which situated cases coming up later can be interpreted. Underdesign is a defining activity for meta-design by creating design spaces for others.

3 User-Centered Design and Participatory Design

User-centered design approaches [Norman & Draper, 1986] (whether done *for* users, *by* users, or *with* users) have focused primarily on activities and processes taking place at design time in the systems' original development and have given little emphasis and provided few mechanisms to support systems as *living* entities which can be evolved by their users. In user-centered design, designers generate solutions placing users mainly in a reactive role.

Participatory design approaches [Schuler & Namioka, 1993] seek to involve users more deeply in the process as co-designers by empowering them to propose and generate design alternatives themselves. Participatory design supports diverse ways of thinking, planning, and acting making work, technologies, and social institutions more responsive to human needs. It requires the social inclusion and active participation of the users. Participatory design has focused on system development at design time by bringing developers and users together to envision the contexts of use. But despite the best efforts at design time, systems need to be evolvable to fit new needs, account for changing tasks, and incorporate new technologies.

4 Meta-Design

Meta-design [Fischer & Scharff, 2000] extends the traditional notion of system design beyond the original development of a system to include an ongoing process in which stakeholders become *co-designers*—not only at design time, but throughout the whole existence of the system. A necessary, although not sufficient condition for meta-design is that software systems include advanced features permitting users in creating complex customizations and extensions. Rather than presenting users with closed systems, meta-design provides them with opportunities, tools, and social reward structures to extend the system to fit their needs. Meta-design shares some important objectives with user-centered and participatory design, but it *transcends* these objectives in several important dimensions and it has changed the processes by which systems and content is designed. Meta-design has shifted the control from designers to users and empowered users to create and contribute their own visions and objectives. Meta-design is a useful perspective for projects where 'designing the design process' is a first-class activity, meaning that creating the technical and social conditions for broad participation in design activities is as important as creating the artifact itself [Wright et al., 2002].

The Seeding, Evolutionary Growth, and Reseeding (SER) Process Model. The major conceptual framework which we have developed to support meta-design is the *seeding, evolutionary growth, and reseeding (SER) process model* [Fischer & Ostwald, 2002]. The SER model is a descriptive and prescriptive model for large evolving systems and information repositories postulating that systems that evolve over a sustained time span must continually alternate between periods of activity and unplanned evolutions and periods of deliberate (re)structuring and enhancement. The SER model encourages designers to conceptualize their activity as meta-design, thereby supporting users as designers in their own right, rather than restricting them to being passive consumers. Figure 1 provides a graphical illustration of the SER model.

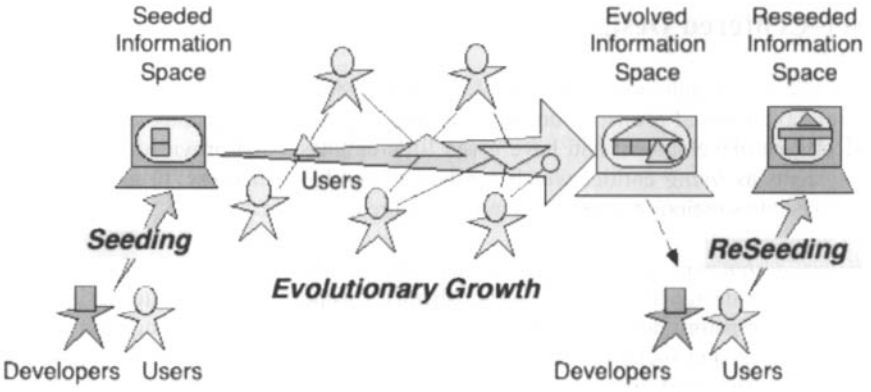

Figure 1: The Seeding, Evolutionary Growth, and Reseeding Process Model

Informed Participation and Unselfconscious Cultures of Design. *Informed participation* [Brown & Duguid, 2000] is a form of collaborative design in which participants from all walks of life—not just skilled computer professionals—transcend beyond the information given to incrementally acquire ownership in problems and to contribute actively to their solutions. It addresses the challenges associated with open-ended and multidisciplinary design problems. These problems involving a combination of social and technological issues, *do not have right answers*, and the knowledge to understand and resolve them changes rapidly. To successfully cope with informed participation requires social changes as well as new interactive systems that provide the opportunity and resources for social debate and discussion rather than merely delivering predigested information to users.

Being ill-defined, design problems cannot be delegated (e.g., from users to professionals), because they are not understood well enough that they can be described in sufficient detail. Partial solutions need to "talk back" [Schön, 1983] to the owners of the problems who have the necessary knowledge to incrementally refine them. Alexander [Alexander, 1964] has introduced the distinction between an unselfconscious culture of design and a selfconscious culture of design. In an *unselfconscious* culture of design, the failure or inadequacy of the form leads directly to an action to change or improve it. This closeness of contact between designer and product allows constant rearrangement of unsatisfactory details. By putting owners of problems in charge, the positive elements of an unselfconscious culture of design can be exploited in meta-design approaches by creating media that support people in working on their tasks, rather than requiring them to focus their intellectual resources on the medium itself.

5 Environments Supporting Meta-Design

The goal of making systems modifiable by users does not imply transferring the responsibility of good system design to the user. In general, "normal" users do not build tools of the quality a professional designer would since users are not concerned with the tool per se, but in doing their work. *Domain-oriented design environments* support meta-design by advancing human-computer interaction to *human problem-domain interaction*. Because systems are modelled at a conceptual level with which users are familiar, the interaction mechanisms take advantage of existing user knowledge and make the functionality of the system transparent and accessible so that the computational drudgery required of users can be substantially reduced. The *Envisionment and*

Discovery Collaboratory [Arias et al., 2000] is a second generation design environment focused on the support of *collaborative design* by integrating physical and computational components to encourage and facilitate informed participation by all stakeholders in the design process.

6 Application of Meta-Design Approaches

Social Creativity. Complex design problems require more knowledge than any single person can possess, and the knowledge relevant to a problem is often distributed among all stakeholders who have different perspectives and background knowledge, thus providing the foundation for *social creativity* [Arias et al., 2000]. Bringing together different points of view and trying to create a shared understanding among all stakeholders can lead to new insights, new ideas, and new artifacts. Social creativity can be supported by innovative computer systems that allow all stakeholders to contribute to framing and solving these problems collaboratively.

Open Source. *Open source development* [Raymond & Young, 2001] is an activity in which a community of software developers collaboratively constructs systems to help solve problems of shared interest and for mutual benefit. The ability to change source code is an enabling condition for collaborative construction of software by changing software from a fixed entity that is produced and controlled by a closed group of designers to an open effort that allows a community to design collaboratively based on personal desires following the framework provided by the seeding, evolutionary growth, and reseeding process model. Open source invites passive consumers to become active contributors [Fischer, 2002].

Learning Communities. *Courses-as-seeds* [dePaula et al., 2001] is an educational model that explores meta-design in the context of university courses by creating a culture of informed participation. Courses are conceptualized as seeds, rather than as finished products, and students are viewed as informed participants who play an active role in defining the problems they investigate. The output of each course contributes to an evolving information space that is collaboratively designed by all course participants, past and present.

Interactive Art. *Interactive art*, conceptualized as meta-design, focuses on collaboration and co-creation. The original design (representing a seed in our framework) establishes a context in which users can create content. Interactive art puts the tools rather than the object of design in the hands of users. It creates interactive systems that do not define content and processes, but the conditions for the process of interaction. Interactive art puts the emphasis on different objectives compared to traditional design approaches, including shifts from (1) guidelines and rules to exceptions and negotiations; (2) from content to context; (3) from objects to process, and (4) from certainty to contingency (these "cultural shifts" have been developed jointly with Elisa Giaccardi who has explored the concept of meta-design in interactive arts [Giaccardi, 2003]).

7 Conclusions

We have evaluated our approaches in different settings, with different task domains, and with different stakeholders. While meta-design is a promising approach to overcome the limitations of closed systems and to support social creativity, it creates many fundamental challenges: in the technical domain as well as in the social domain including the need for social capital, the willingness of users to engage in additional learning to become designers, and the additional efforts to integrate the work into the shared environment. Meta-design addresses one of the fundamental challenges of a knowledge society [Florida, 2002]: to invent and design a culture in

which all participants in a collaborative design process can express themselves and engage in personally meaningful activities.

8 Acknowledgements

The author thanks the members of the Center for LifeLong Learning & Design at the University of Colorado, who have made major contributions to the conceptual framework described in this paper. The research was supported by (1) the National Science Foundation, Grants (a) REC-0106976 "Social Creativity and Meta-Design in Lifelong Learning Communities", and (b) CCR-0204277 "A Social-Technical Approach to the Evolutionary Construction of Reusable Software Component Repositories"; (2) SRA Key Technology Laboratory, Inc., Tokyo, Japan; and (3) the Coleman Initiative, San Jose, CA.

9 References

Alexander, C. (1964) *The Synthesis of Form*, Harvard University Press, Cambridge, MA.

Arias, E. G., Eden, H., Fischer, G., Gorman, A., & Scharff, E. (2000) "Transcending the Individual Human Mind—Creating Shared Understanding through Collaborative Design," *ACM Transactions on Computer Human-Interaction*, 7(1), pp. 84-113.

Brown, J. S., & Duguid, P. (2000) *The Social Life of Information*, Harvard Business School Press, Boston, MA.

dePaula, R., Fischer, G., & Ostwald, J. (2001) "Courses as Seeds: Expectations and Realities," *Proceedings of the Second European Conference on Computer-Supported Collaborative Learning (Euro-CSCL' 2001)*, Maastricht, Netherlands, pp. 494-501.

Fischer, G. (2002) *Beyond 'Couch Potatoes': From Consumers to Designers and Active Contributors, in FirstMonday (Peer-Reviewed Journal on the Internet)*. Available at http://firstmonday.org/issues/issue7_12/fischer/.

Fischer, G., & Ostwald, J. (2002) "Seeding, Evolutionary Growth, and Reseeding: Enriching Participatory Design with Informed Participation," *Proceedings of the Participatory Design Conference (PDC'02)*, Malmö University, Sweden, pp. 135-143.

Fischer, G., & Scharff, E. (2000) "Meta-Design—Design for Designers," *3rd International Conference on Designing Interactive Systems (DIS 2000)*, New York, pp. 396-405.

Florida, R. (2002) *The Rise of the Creative Class and How It's Transforming Work, Leisure, Community and Everyday Life*, Basic Books, New York, NY.

Giaccardi, E. (2003) *Meta-Design*. Available at http://x.i-dat.org/~eg/research.htm.

Norman, D. A., & Draper, S. W. (Eds.) (1986) *User-Centered System Design, New Perspectives on Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ.

Raymond, E. S., & Young, B. (2001) *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly & Associates, Sebastopol, CA.

Schön, D. A. (1983) *The Reflective Practitioner: How Professionals Think in Action*. Basic Books, New York.

Schuler, D., & Namioka, A. (Eds.) (1993) *Participatory Design: Principles and Practices*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Wright, M., Marlino, M., & Sumner, T. (2002) *Meta-Design of a Community Digital Library, D-Lib Magazine, 8 (5)*. Available at http://www.dlib.org/dlib/may02/wright/05wright.html.

Usability Patterns in Software Architecture

Eelke Folmer and Jan Bosch

Department of Mathematics and Computing Science
University of Groningen, PO Box 800, 9700 AV the Netherlands
mail@eelke.com, Jan.Bosch@cs.rug.nl

Abstract

Over the years the software engineering community has increasingly realized the important role software architecture plays in fulfilling the quality requirements of a system. Practice shows that for current software systems, most usability issues are still only detected during testing and deployment. To improve the usability of a software system, usability patterns can be applied. However, too often software systems prove to be inflexible towards such modifications which lead to potentially prohibitively high costs for implementing them afterwards. The reason for this shortcoming is that the software architecture of a system restricts certain usability patterns from being implemented after implementation. Several of these usability patterns are "architecture sensitive", such modifications are costly to implement due through their structural impact on the system. Our research has identified several usability patterns that require architectural support. We argue the importance of the relation between usability and software architecture. Software engineers and usability engineers should be aware of the importance of this relation. The framework which illustrates this relation can be used as a source to inform architecture design for usability.

1 Introduction

In the last decades it has become clear that the most challenging task of software development is not just to provide the required functionality, but rather to fulfil specific properties of software such as performance, security or maintainability, which contribute to the *quality* of software (Folmer & Bosch, 2002). Usability is an essential part of software quality; issues such as whether a product is easy to learn to use, whether it is responsive to the user and whether the user can efficiently complete tasks using it may greatly affect a product's acceptance and success in the marketplace. Modern software systems are continually increasing in size and complexity. An explicit defined architecture can be used as a tool to manage this size and complexity. The quality attributes of a software system however, are to a large extent determined by a system's software architecture. Quality attributes such as performance or maintainability require explicit attention during development in order to achieve the required levels (Bosch & Bengtsson 2002). It is our conjecture that this statement also holds for usability. Some changes that affect usability, for example changes to the appearance of a system's user interface, may easily be made late in the development process without incurring great costs. These are changes that are localised to a small section of the source code. Changes that relate to the interactions that take place between the system and the user are likely to require a much greater degree of modification. Restructuring the system at a late stage will be extremely and possibly prohibitively, expensive. To improve on this situation, it would be beneficial for knowledge pertaining to usability to be captured in a form that can be used to inform architectural design, so that engineering for usability is possible early in the design process. The usability engineering community has collected and developed various design solutions such as usability patterns that can be applied to a system to improve usability. Where these prescribe sequences or styles of interaction between the system and the user, they are likely

to have architectural implications. For example, consider the case where the software allows a user to perform a particularly complex task, where a lot of users make mistakes. To address this usability issue a wizard pattern can be employed. This pattern guides the users through the complex task by decomposing the task into a set of manageable steps. However implementing such a pattern as the result of a design decision made late on proves to be very costly. There needs to be provision in the architecture for a wizard component, which can be connected to other relevant components, the one triggering the operation and the one receiving the data gathered by the wizard. The problem with this late detection of usability issues is that sometimes it is very difficult to apply certain usability patterns after the majority of a system has been implemented because these patterns are 'architecture sensitive'. The contribution of this paper is to make software engineers aware that certain 'design solutions' that may improve usability are extremely difficult to retro-fit into applications because these patterns require architectural support. Therefore being able to design architectures with support for usability is very important. The framework that we present in the next section can be used as an informative source during design.

2 Usability Framework

Through participation in the STATUS[1] project we have investigated the relationship between usability and software architecture. Before the relationship between usability and software architecture was investigated an accurate definition of usability was tried to obtain by surveying existing literature and practice. Initially a survey was undertaken to try and find a commonly accepted definition for usability in terms of a decomposition into usability attributes. It was soon discovered that the term usability attribute is quite ambiguous. People from industry and academia have quite different perceptions of what they consider to be a useful usability attribute. The number of "usability attributes" obtained in this way grew quite large therefore we needed a way to organise and group the different attributes. Next to the need for organising these different interpretations of usability attributes, a relation between usability and software architecture was tried to discover. The only 'obvious' relation between usability and architecture is that there are some usability patterns that have a positive effect on usability and that are architecture sensitive. However, it was soon discovered that it was extremely difficult to draw a direct relationship between usability attributes and software architecture. An attempt was made to decompose the set of usability attributes into more detailed elements such as: "the number of errors made during a specific task", which is an indication of reliability, or "time to learn a specific task" which is an indication of learnability, but this decomposition still did not lead to a convincing connecting relationship with architecture. The reason is that traditionally usability requirements have been specified such that these can be verified for an implemented system. However, such requirements are largely useless in a forward engineering process. For example, it could be stated that a goal for the system could be that it should be easy to learn, or that new users should require no more than 30 minutes instruction, however, a requirement at this level does not help guide the design process. Usability requirements need to take a more concrete form expressed in terms of the solution domain to influence architectural design. To address to these problems discussed a framework has been developed. Figure 1 shows the framework developed so far. It shows a collection of attributes, properties and patterns and shows how these are linked to give the relationship between usability and software architecture. This relation is illustrated by means of an example. Figure 1 shows the wizard pattern linked to the "guidance" usability property which in

[1] STATUS is an ESPRIT project (IST-2001-32298) financed by the European Commission in its Information Society Technologies Program. The partners are Information Highway Group (IHG), Universidad Politecnica de Madrid (UPM), University of Groningen (RUG), Imperial College of Science, Technology and Medicine (ICSTM), LOGICDIS S.A.

turn is linked to the "learnability" usability attribute. The wizard pattern guides the user through a complex task by decomposing the task into a set of manageable subtasks. To implement a wizard a provision is needed in the architecture for a wizard component, which can be connected to other relevant components: the one triggering the operation and the one receiving the data gathered by the wizard. The wizard is related to usability because it uses the primitive of guidance to "guide" the user through the task. Guidance on its turn has two "obvious" relations with usability. Guidance has a positive effect on learnability and a negative effect on efficiency. The concept of "guidance" is defined as a usability property; a usability property is a more concrete form of a usability requirement specified in terms of the solution domain. Patterns relate to one or more of these usability properties. Properties on their turn relate to one or more usability attributes. This relation is not necessarily a one to one mapping. The relationship can be positive as well as negative. To avoid the table becoming too cluttered, and the risk of possibly producing a fully connected graph, only the links thought to be strongest and positive are indicated in the table. The framework relates the problem to the solution domain; a usability attribute can be measured on a completed solution, whereas a usability property exists in the problem domain, and can be used as a requirement for system design. A usability pattern bridges the gap between problem and solution domains, providing us with a mechanism to fulfil a requirement, providing us with a solution for which the corresponding usability attribute can be measured. The next sections enumerate the concepts of usability attributes, properties and patterns which comprise our framework.

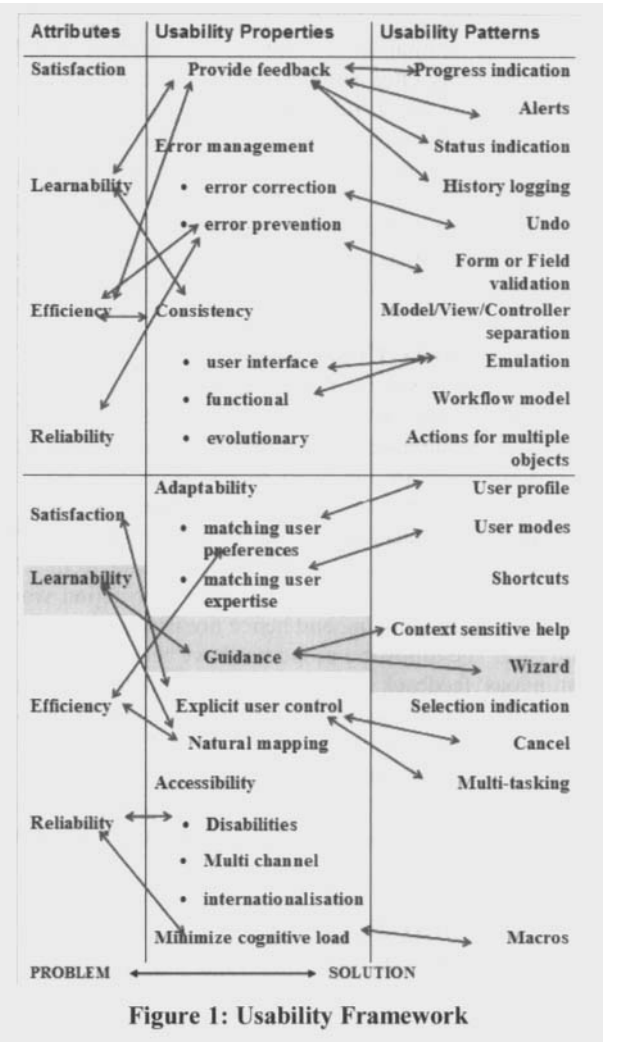

Figure 1: Usability Framework

3 Usability Attributes

A comprehensive survey of the literature (Folmer & Bosch, 2002) revealed that different researchers have different definitions for the term usability attribute, but the generally accepted meaning is that a usability attribute is a precise and measurable component of the abstract concept that is usability. After an extensive search of the work of various authors, the following set of usability attributes is identified for which software systems in our work are assessed. No

innovation was applied in this area, since abundant research has already focussed on finding and defining the optimal set of attributes that compose usability. Therefore, merely the set of attributes most commonly cited amongst authors in the usability field has been taken. The four attributes that are chosen are: *Learnability* - how quickly and easily users can begin to do productive work with a system that is new to them, combined with the ease of remembering the way a system must be operated. *Efficiency of use* - the number of tasks per unit time that the user can perform using the system. *Reliability in use* - this attribute refers to the error rate in using the system and the time it takes to recover from errors. *Satisfaction* - the subjective opinions that users form in using the system. These attributes can be measured directly by observing and interviewing users of the final system using techniques that are well established in the field of usability engineering.

4 Usability Properties

Essentially, our usability properties embody the heuristics and design principles that researchers in the usability field have found to have a direct influence on system usability. These properties can be used as requirements at the design stage, for instance by specifying: "the system must provide feedback". They are not strict requirements in a way that they are requirements that should be fulfilled at all costs. It is up to the software engineer to decide how and at which levels these properties are implemented by using usability patterns of which it is known they have an effect on this usability property. For instance providing feedback when printing in an application can be very usable, however if every possible user action would result in feedback from the system it would be quite annoying and hence not usable. Therefore these properties should be implemented with care. The following properties have been identified: *Providing feedback* - the system provides continuous feedback as to system operation to the user. *Error management* - includes error prevention and recovery. *Consistency* - consistency of both the user interface and functional operation of the system. *Guidance* - on-line guidance as to the operation of the system. *Minimise cognitive load* - system design should recognise human cognitive limitations, short-term memory etc. *Natural mapping* - includes predictability of operation, semiotic significance of symbols and ease of navigation. *Accessibility* - includes multi-mode access, internationalisation and support for disabled users.

5 Usability Patterns

One of the products of the research on this project into the relationship between software architecture and usability is the concept of a usability pattern. The term "usability pattern" is chosen to refer to a technique or mechanism that can be applied to the design of the architecture of a software system in order to address a need identified by a usability property at the requirements stage. Various usability pattern collections have been defined (Welie & Trætteberg 2000), (Tidwell 1998). Our collection is different from those because we only consider patterns which should be applied during the design of a system's software architecture, rather than during the detailed design stage. (Bass et al, 2001) have investigated the relationship between the usability and software architecture through the definition of a set of 26 scenarios. These scenarios are in some way equivalent to our properties and usability patterns. However there are some differences. They have used a bottom up approach from the scenarios whereas we have taken a top down approach from the definition of usability. Our approach has in our opion resulted in a more clearly documented and illustrated relationship between those usability issues addressed by the design principles and the software architecture design decisions required to fullfill usability requirements. Another difference is that our patterns have been obtained from an inductive process from different practical cases (e-commerce software developed by the industrial partners in this project) whereas their scenarios result from personal experience and literature surveys. Their work has been useful to support our statement that usability and software are related through usability

patterns. A full catalogue of patterns identified is presented on http://www.designforquality.com. There is not a one-to-one mapping between usability patterns and the usability properties that they affect. A pattern may be related to any number of properties, and each property may be improved (or impaired) by a number of different patterns. The choice of which pattern to apply may be made on the basis of cost and the trade off between different usability properties or between usability and other quality attributes such as security or performance. This list of patterns presented in Figure 1 is not intended to be exhaustive, and it is envisaged that future work on this project will lead to the expansion and reworking of the set of patterns presented here, including work to fill out the description of each pattern to include more of the sections which traditionally make up a pattern description, for instance what the pros and cons of using each pattern may be.

6 Summary and conclusions

Our research has argued the importance of the relation between usability and software architecture. A framework has been developed which illustrates this relation.The list of usability patterns and properties identified/defined in our framework is substantial but incomplete, new usability patterns or properties that are developed or discovered can be fitted in the existing framework. Future research should focus on verifying the architectural sensitiveness of the usability patterns that have been identified. For validation only e-commerce software provided by our industrial partners in this project has been considered. To achieve more accurate results our view should be expanded to other application domains. The usability properties can be used as requirements for design, it is up to the software architect to select patterns related to specific properties that need to be improved for a system. It is not claimed that a particular usability pattern will improve usability for any system because many other factors may be involved that determine the usability of a system.. The relationships in the framework indicate potential relationships. Further work is required to substantiate these relationships and to provide models and assessment procedures for the precise way that the relationships operate. This framework provides the basis for developing techniques for assessing software architectures for their support of usability. This technique allows for iteratively designing for usability on the architectural level.

References

Bass, Lenn; Kates, Jessie & John, Bonnie. E.(2002) Achieving Usability through software architecture, http://www.sei.cmu.edu/publications/documents/01.reports/01tr005.html

Bosch, J. (2000). Design and Use of Software Architectures: Adopting and Evolving a Product Line Approach, Pearson Education (Addison-Wesley and ACM Press).

Bosch, J. & Bengtsson, P. O. (2002), Assessing optimal software architecture maintainability. In proceedings of Fifth European Conference on Software Maintenance and Reengineering (CSMR'01), IEEE Computer Society Press, Los Alamitos, CA pp. 168-175

Folmer, E. & Bosch, J. (2003). Architecting for usability; a survey. Submitted for the Journal of systems and software. Accepted for the Journal of systems and software.

Tidwell, J. (1998). Interaction Design Patterns. Conference on Pattern Languages of Programming.

Welie, M. & Trætteberg, H., (2000). Interaction Patterns in User Interfaces. 7th Conference on Pattern Languages of Programming (PloP).

Bridging the Gap between Scenarios and Formal Models

Peter Forbrig and Anke Dittmar

University of Rostock, Department of Computer Science
Albert-Einstein-Str. 21, D-18051 Rostock, Germany
[pforbrig | ad]@informatik.uni-rostock.de

Abstract

A software design process has to consider both the user- and system-oriented aspects. On the one hand, scenarios are often used to illustrate the interface between a system and its application context. On the other hand, abstract design models with a preferably constructive character are requested for implementing software systems. This paper shows a combination of the scenario- and model-based approaches to fulfil the demands mentioned above.

1 Introduction

Scenario-based design (SBD) as well as model-based design (MBD) aim to support a user-centred design process. Both approaches emphasize that software developers have to pay attention to the application context to get usable systems. It is necessary to analyse the current working situation to know about the activities users have to perform, the objects they manipulate, tools they apply, and the way they cooperate. The analysis results in a set of goals showing which current practices have to be changed and which things are worth to maintain. Goals describe an intended state, called envisioned working situation. Thus, it is not enough to design a software system fulfilling these requirements only but the whole context of use has to be designed.
By taking this point of view one has to accept software design as a process which involves many stakeholders. Although MBD approaches catch different perspectives on the system design in different kinds of models, they don't really support a participatory design. The process is rather specification driven. That is, the main focus is on the construction of models with a more or less formalized structure and it is not guaranteed that all stakeholders are able to participate. SBD approaches claim to give every stakeholder an active role by "telling stories" (scenarios) which he can understand and which evoke his interest and empathy. Nevertheless, the system-centred side of the design process cannot be ignored. A precise constructive specification of the software system under development which can serve as a direct input to the implementation is necessary. Hence, SBD approaches have to be seen as reasonable supplement to a specification driven design [Car02]. The problem of linking scenarios and formal models effectively is still an active field of research. In this paper, it is argued that an effective combination of ideas coming from SBD and MBD is possible. It is shown how task models can mediate between scenarios and software specifications.

2 Scenarios

Prof. Thiel goes through the results of the examination the 213 students of the 1.term had to write to pass the course about programming techniques. The students had to work on 5 exercises. He sighs. 51 students got less than 20 points. That means they have to repeat the exam to be allowed to continue their studies. Prof. Thiel asks his assistant Peter Meyer who compiled the list of the total numbers of points to check the list once more. While Peter is checking the list, Anne Holz who had to mark the second exercise sent him, he discovers

that 6 students got no points. He wonders if Anne forgot to record some results. Furthermore, he decides to ask Prof. Thiel how to treat the 12 students who got 18 or 19 points. Perhaps it would be reasonable to review their exam papers.

A "scenario is a story about people and their activities" [RC02]. The above example gives an impression. It can be seen as a *problem scenario* describing aspects of a current practice. Problem scenarios are developed during a requirements analysis. Together with claims which give an evaluation of the current setting they serve as basis for the description of requirements on the system under design. It is further distinguished between activity, information, and interaction scenarios on design level.

Rosson and Carroll emphasize that scenarios are rather constructions based on general descriptions about actors, tasks, artifacts etc. than stories based on pure observation or imagination. Perhaps this is surprising. Surely, SBD aims to evoke the interest and participation of all stakeholders by "concrete" stories – but not without purpose. This participation should lead to a better system design. Hence, the stories have to be told in a way which people let think more consciously about the impact a system design would have on working practices. The stories should force people to think about alternative design ideas. Scenarios also comprise planning and evaluating activities of the actors.

A slightly different view on scenarios is used in the object-oriented software development. Here, a scenario describes a path through a use case. Thus, the focus is on the description of the interaction between the actor and the system. There is nothing told about mental activities of the users and the description of the application context is more restricted. Whereas Rosson and Carroll say "much of the richness of a scenario is in the things that are not said" and allow partial task descriptions and the involvement of more than one task in a scenario, scenarios as instances of use cases are more rigid. They are suitable for test cases in later phases of the system development.

3 Models in MBD

In MBD, one can distinguish between two main groups of sub-models. One group forms the specification of the software system itself. There exists at least an application model to describe the application core and a dialogue model to specify the UI. The other group constitutes the task model. Here, the same concepts are considered as in SBD but in an abstract and formalized way to make a transformation of task knowledge to formal system specifications (as finally requested in the software development) easier. Consequently, task models are suited to mediate between scenarios and system specifications. In this section, the sub-models of a task model are explained shortly. [FD03] gives a more detailed description of the interdependencies between task models and appropriate system specifications.

Task models consist of sub-models specifying goals, actions, business objects, and actors. A goal is represented by a network of sub-goals each describing an intended sub-state of the domain. The refinement of a goal has to result in an action structure. Actions are hierarchically decomposed into sub-actions until the level of basic actions. The objects of a business object model characterized by a name and a set of attribute-value-pairs serve to specify states of the domain. In contrast to most object-oriented modelling approaches, it is not distinguished between objects and classes but objects can be instances of other ones as will be explained in the next section.

There are inner and outer relationships between elements of the sub-models. For example, two sub-goals can be related by operators describing that both/at least one/at most one of them have to be achieved (and-/or-/xor-operator) etc.. Temporal constraints between the sub-actions of a super-action are specified by using temporal operators in a temporal equation with the super-action on the left hand side and the sub-actions on the right hand side. Attributes of objects reflect relationships between different objects.

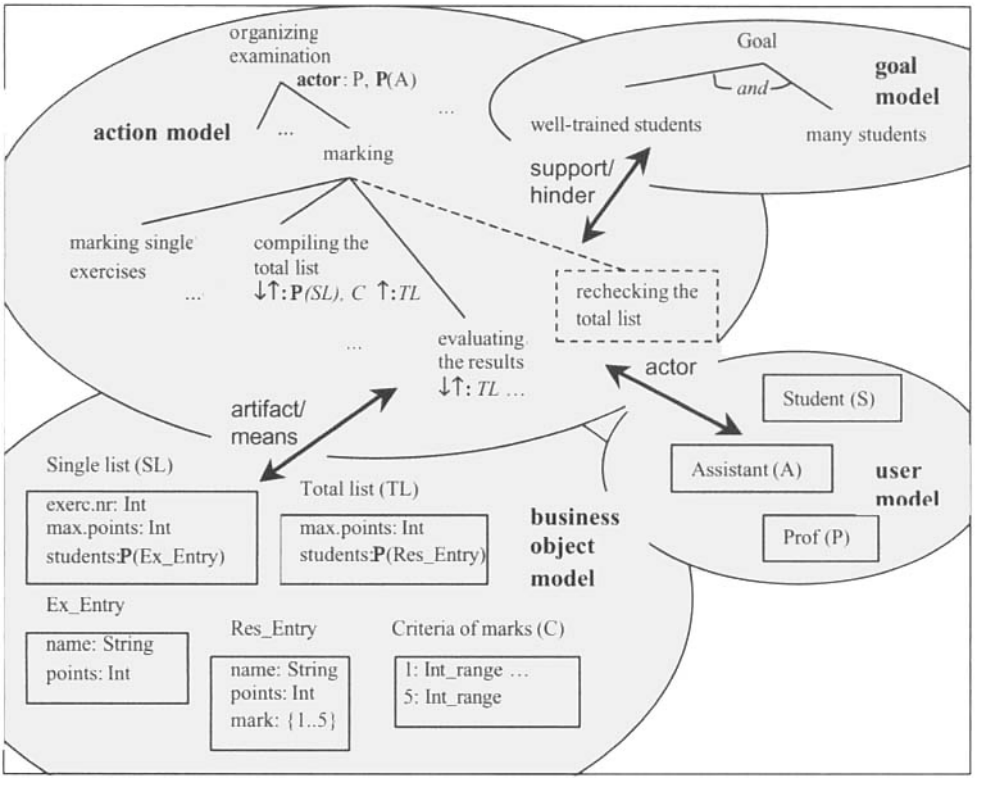

Figure 1: The sub-models of an example task model

A goal cannot be fulfilled without performing some appropriate actions changing the state of the domain. In order to emphasize the central role of actions only the relationships between the action model and the other ones are sketched out in the example of Fig.1 which partly explores the task of educating students. Every sub-action supports or hinders some sub-goals. Thus, the action hierarchy reflects the compromises made in planning the achievement of the goal.

Executing an action means to create, destroy, use, or change some business objects. In other words, an action needs a set of objects in a certain state (precondition) and supplies a set of objects in a certain state (post condition or effect). Actions at a higher level of the hierarchy and, finally, the goals represent approximations of or views on affected business objects and their states. For example, the effect of *compiling the total list* in Fig.1 is the object *Total list* (abbrev. *TL*, ↓/↑ mark pre-/post conditions). With respect to an action a business object can play the role of an artefact or a means of work. Means are applied to change the state of the artefact which is essential for the achievement of the goal. They are further divided into tools used and resources consumed in actions. The artefact of *compiling the total list* is *Total list*, a set of *Single lists* and the object *Criteria of marks* are used as tools.

A task can demand the participation of a whole group of people. In this paper, the division of labour is simply modelled by assigning actors to the nodes of an action hierarchy. Consequently, it is reasonable not to stop the action decomposition until the basic actions (the leaf nodes) are executed by single persons. Otherwise, questions concerning the division of labour are still open. In Fig.1, *Prof.* and (set of) *Assistant(s)* are the actors in *organizing examination*. Task models are described e.g. in [Dit02] more formal and detailed.

4 Task Models as Mediator between Scenarios and Formal System Specifications

4.1 Scenario Elements are Instances of Model Elements

An action model describes a set of sequences of basic actions which all lead to task completion. Each sequence is a kind of instance of the action model. An object O_1 is an instance of an object O_2 in the business object model (denoted by $O_1::O_2$) if for all attributes $a_i:v_i$ of O_2 there is an attribute $a_i:v'_i$ in O_1 and v'_i is an instance of v_i. Further, instance relations between actors are assumed.

Let T be a task model with the action model A, the business object model O and the user model R. A *complete scenario* is a sequence $\langle a_1(r_1,\{o^1_1,\ldots,o^1_{n1}\}),\ldots,a_m(r_m,\{o^m_1,\ldots,o^m_{nm}\}\rangle$ where a_i are basic actions in A and $\langle a_1(R_1,\{O^1_1,\ldots,O^1_{n1}\}),\ldots,a_m(R_m\{O^m_1,\ldots,O^m_{nm}\}\rangle$ is a possible sequence in A with

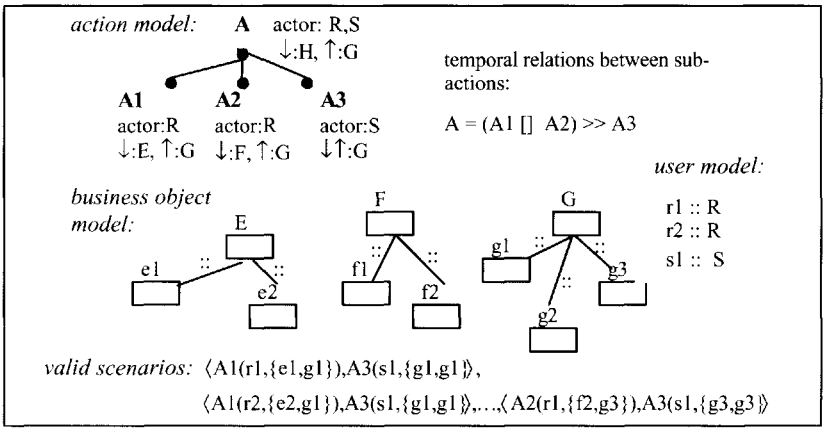

Figure 2: Complete scenarios derived from a task model

actors R_i from R and objects O_i from O assigned to a_i (for brevity pre-and post conditions, artefacts and means are not distinguished). Further, r_i resp. o^j_i have to be instances of R_i resp. O^j_i (i=1,..,m, j=1,2,...). The notion **P**(O) (e.g. **P**(SL) in Fig.1) describes a set of instances of O.

A task model describes a (mostly infinite) set of complete scenarios. Fig.2 visualizes this idea by a small abstract example. The used temporal operators are [] for alternative and >> for sequential sub-actions.

Comparing the scenario at the beginning of Sect.2 with the task model in Fig.1 it should be clear now that *Prof. Thiel* is an instance of *Prof.*, *Peter Meyer* and *Anne Holz* are instances of *Assistant*, *Prof. Thiel* is *evaluating the results* of an instance of *Total list* and so on.

4.2 Scenarios as Stories vs. Scenarios as Test Cases

This paper proposes a distinction between two kinds of scenarios.

Scenarios as stories have an incomplete character in that sense that they are constructed from different parts of the appropriate task model. This kind reflects rather the approach taken by Rosson and Carroll. Scenarios can involve different actors and actions, and can even refer to goals or pieces of action models to describe mental activities of actors. Scenarios as stories are mainly used to improve and illustrate (initial) task models. For example, the scenario in Sect.2 was

constructed on the basis of the task model in Fig.1 and revealed the sub-action *rechecking the total list* which was added afterwards to the task model (indicated with dotted lines in the figure). "In SBD new activities are always grounded in current activities" [RC02]. Scenarios as stories support the transition from current to envisioned models. Taken the scenario in Sect.2, for example, one can easily imagine that in the envisioned practice the *compiling of the total list* out of the single lists for each of the 5 exercises can be done automatically by the software system under development. It is also of advantage that the system can label the "borderline cases" (the entries of the students who miss 1 or 2 points to pass the examination). An envisioned task model has to reflect the application of the software system by the actors.

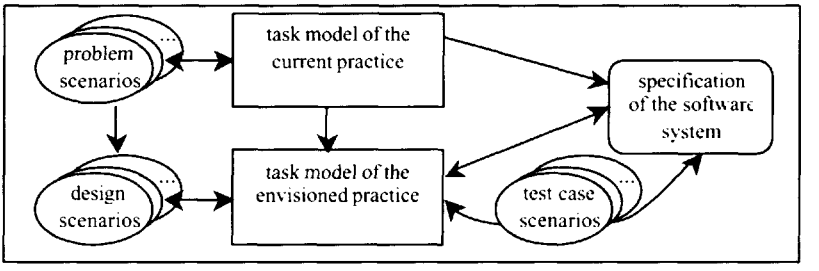

Figure 3: A combination of MBD and SBD

Scenarios as test cases are complete scenarios in the sense of Sect. 4.1. They can be generated from sub-trees of the action hierarchy as should be underlined by Fig.2. They are useful for validating a system specification with respect to a task model and vice versa because they describe the interaction between a user and the system precisely. Scenarios of this kind are rather comparable with scenarios as paths through use cases.
Fig.3 illustrates possible applications of scenarios to supplement a model-based design process.

5 Summary

A combination of SBD and MBD to improve the software design process was shown. Task models play a mediating role. They supply a general description which is useful for constructing scenarios on the one side but also for deriving formal system specifications on the other side. A distinction between scenarios as stories and scenarios as test cases was proposed and their different fields of application were explored. There are a lot of open questions. The problem of choosing a representative set of test case scenarios out of a possibly infinite set is only one of them. The interested reader is referred to [FD03] where tool support is described.

References

[Car02] J.M.Carroll: Scenarios and Design Cognition, Proc. of the Int. Conf. On Requirements Engineering RE2002, Essen, Germany.

[RC02] M.B.Rosson, J.M.Carroll: Usability Engineering – Scenario-Based Development of Human-Computer Interaction, Morgan Kaufmann Publishers, 2002.

[FD03] P.Forbrig, A.Dittmar: Interfacing Business Object and User Models with Action Models in this Proceedings of HCI 2003.

[Dit02] A.Dittmar: Ein formales Metamodell für den aufgabenbasierten Entwurf interaktiver Systeme. PhD-Thesis. Universität Rostock, 2002.

A Layered Approach for Designing Multiple User Interfaces from Task and Domain Models

Elizabeth Furtado, João José Vasco Furtado, Quentin Limbourg, Jean Vanderdonckt*, Wilker Bezerra Silva, Daniel William Tavares Rodrigues, and Leandro da Silva Taddeo*

Universidade de Fortaleza, NATI - Célula EAD - Fortaleza, BR-60455770 Brazil
{elizabet, vasco, wilker, danielw, taddeo}@unifor.br
*Université catholique de Louvain, BCHI – B-1348 Louvain-la-Neuve (Belgium)
{limbourg,vanderdonckt}@isys.ucl.ac.be

Abstract

More frequently, design of user interfaces covers design issues related to multiple contexts of use where multiple stereotypes of users may carry out multiple tasks, possibly on multiple domains of interest. Existing development methods do not necessarily support developing such user interfaces as they do not factor out common parts between similar cases while putting aside uncommon parts that are specific to some user stereotypes. To address this need, a new development method is presented based on three layers: (i) a conceptual layer where a domain expert defines an ontology of concepts, relationships, and attributes of the domain of discourse, including user modeling; (ii) a logical layer where a designer specifies multiple models based on the previously defined ontology and its allowed rules; and (iii) a physical layer where a developer develops multiple user interfaces from the previously specified models with design alternatives depending on characteristics maintained in the user models.

1 Introduction

Universal design (Savidis, Akoumianakis & Stephanidis, 2001) adheres to a vision where user interfaces (UIs) of interactive applications are developed for the widest population of users in different contexts of use by taking into account differences such as preferences, cognitive style, language, culture, habits, conventions, and system experience. Universal design of UIs poses some difficulties due to the consideration of these multiple parameters depending on the supported differences. In particular, the multiplicity of parameters dramatically increases the complexity of the design phase by adding many design options among which to decide. In addition, methods for developing UIs do not mesh well with this variety of parameters as they are not necessarily identified and manipulated in a structured way nor truly considered in the design process. The method structures the UI design in three levels of abstraction as represented in fig. 1 so as to delegate specific conditions and decisions the latest possible in the whole development life cycle:

1. The *conceptual level* enables a domain expert to define ontology of concepts, relationships, and attributes involved in the production of multiple UIs.
2. The *logical level* allows designers to capture requirements of a specific UI design case by instantiating concepts, relationships, and attributes with a graphical editor. Each set of instantiations results in a set of models for each considered design case (n designs in fig. 1).
3. The *physical level* helps developers in deriving multiple UIs from each set of models thanks to a model-based UI generator: in fig. 1, m possible UIs are obtained for UI design #1, p for UI design #2,..., r for UI design #n. The generation is then exported to imported in a traditional development environment for any manual edition (here, MS Visual Basic).

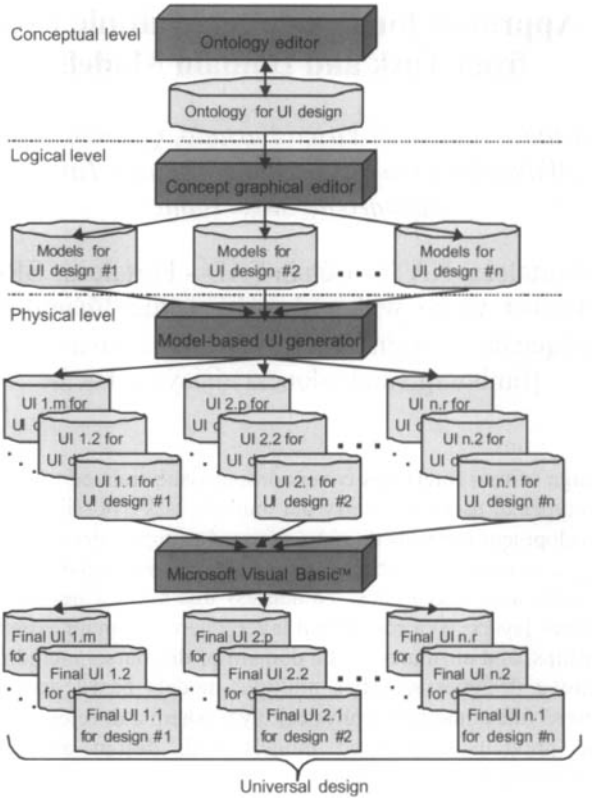

Figure 1: The different levels of the proposed method for universal design of user interfaces

2 Ontology Editor

An ontology (Guarino, 1995) explicitly defines any set of concepts, relationships, and attributes that need to be manipulated in a particular situation, including universal design. The ontology notion comes from the Artificial Intelligence context where it is identified as the set of formal terms with one represents knowledge, since the representation completely determines what "exists" for the system. We define a *context of use* as the global environment in which a user population, perhaps with different profiles, skills, and preferences, are carrying out a series of interactive tasks on one or multiple semantic domains. In universal design, it is expected to benefit from the advantage of considering any type of the above information to produce multiple UIs depending on the varying conditions. These pieces of information of a context of use can be captured in different models (Puerta, 1997). In order to input model properly, an ontology of these models is needed so as to preserve not only their syntactical aspects, but also their semantics. For this purpose, an ontology defines a specification language with which any model can be specified. The concepts and relations of interest at this layer are here introduced as meta-concepts and meta-relations belonging to the meta-modeling stage. An ontology defines a kind of reference meta-model. Of course, several meta-models can be defined, but only one reference is used here. Moreover, the ontology may allow the transformation of concepts expressed according to one meta-model to another one. Fig. 2 exemplifies how these fundamental concepts can be defined in an ontology editor that will serve ultimately to constrain any model that will be further defined and instantiated. The core entity of fig. 2 is the concept, characterized by one or many attributes.

each having here a data type (e.g., string, real, integer, Boolean, or symbol). Concepts can be related to each other thanks to relations. Some particular, yet useful, relations include inheritance (i.e., "is"), aggregation (i.e., "composed of"); and characterization (i.e., "has"). At the meta-modeling stage, we do know that concepts, relations, and attributes will be manipulated, but we do not know yet of what type. Any UI model, can be expressed in terms of the basic entities as specified in fig. 2.

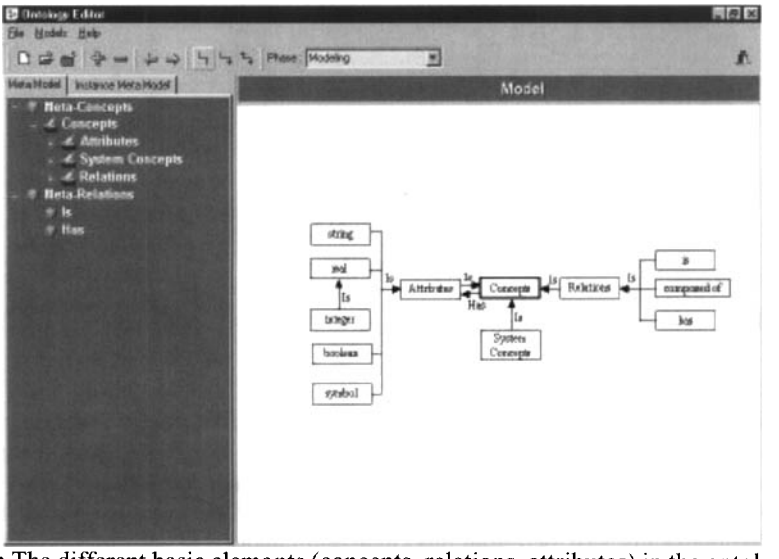

Figure 2: The different basic elements (concepts, relations, attributes) in the ontology editor

3 Concept Graphical Editor

Once the basic elements have been defined, any instance of these elements can be used at the subsequent level to define the models used to capture the specifications of a single UI or for several ones. In particular, it is important to separate, when appropriate, aspects that are independent of the context of use (e.g., a task or a sub-task that needs to be carried out) from aspects that are dependent of this context of use. Therefore, it is likely that each model (i.e., task, domain, user, presentation, dialogue) (Puerta, 1997) will be subject to identifying and separating dependent parts from independent parts. The concept graphical editor is now used to instantiate the context of use, the relationships and attributes of models for the *Medical Attendance* domain involved in patient admission. Fig. 3 graphically depicts the *Urgency Admission* context of use and the attributes of models of task, user and domain. There are two tasks instantiated: *to admit patient* and *to show patient data*. The first one is activated by a *secretary* and uses *patient information* during its execution. For the user model of the secretary, the following parameters are considered: her/his experience layer, input preference, information density with the enumerated values *low* and *high*. The information items describing a patient are the following: date of the day, first name, last name, birth date, address, phone number, gender, and civil status. Information items regarding insurance affiliation and medical regimen can be described similarly. The parameters of an information item of a domain model depend on the UI design process. For instance, parameters and values of an information item used to generate UIs are (Vanderdonckt, 2000): data type (date, Boolean, graphic, integer, real, or alphanumeric), length (n>1), domain definition (know, unknown, or mixed), interaction way (input, output, or input/output), orientation (horizontal, vertical, circular, or undefined), number of possible values (n>1), number of values to choose (n>1), and precision (low or high).

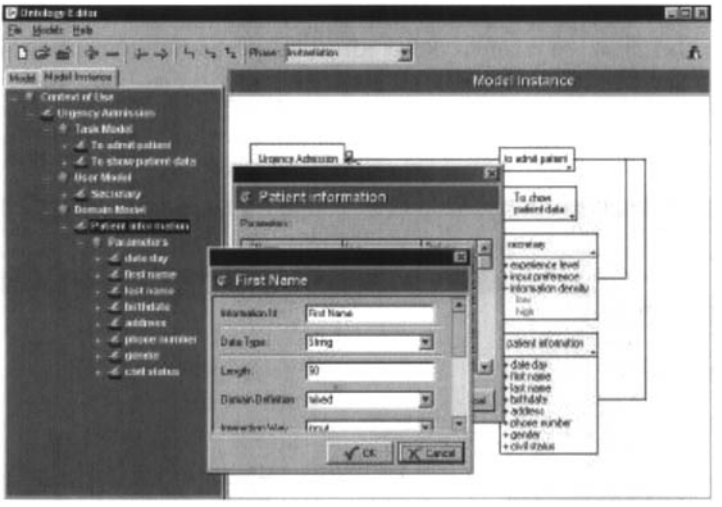

Figure 3: Instantiating the basic elements for a particular model in the concept graphical editor

4 Model-based generator

Once a particular model is obtained, each model can initiate a UI generation process. Here, the SEGUIA (Vanderdonckt and Bodart, 1993) tool is used (fig. 4): it consists of a model-based interface development that is capable of automatically generating code for a running UI from a file containing the specifications defined in the previous step. For this purpose, any model stored in the graphical editor can be exported from this editor and imported in SEGUIA, as a simple file or a DSL file (Dynamic Specification Language) (fig. 4a). Of course, any other tool which is compliant with the model format and/or which can import the specification file may be intended to produce running UI for other design situations, contexts of use, user models, or computing platforms. SEGUIA is able to automatically generate several UI presentations to obtain multiple UIs. These different presentations are obtained

- In an automated manner, where the developer only launches the generation process by selecting which layout algorithm to rely on (e.g. two-column format or right/bottom strategy).
- In a computer-aided manner, where the developer can see at each step what are the results of the generation, can cooperate in a mixed-initiative manner, and govern the process before reaching a final status.

Once a file is opened, the designer may ask the tool to generate a presentation model, if not done before by relying on the concepts of presentation units and logical windows (fig. 4b). A presentation unit is a set of logical windows all related to a same interactive task, but not all these logical windows should be presented simultaneously. For instance, a first input logical window may be presented and a second one afterwards, depending on the status of information acquired in the first one. Logical windows (e.g., a window, a dialog box, a tabbed dialog box) can be determined according to various strategies: maximal, minimal, input/output, functional, and free. Once a presentation model is finalised, the generation process is performed into 4 steps (fig. 4c):

1. Selection of Abstract Interaction Objects: the tools selects interaction objects to input/output information items contained in the model. The objects are tried to be independent from any context of use.
2. Transformation of Abstract Interaction Objects into Concrete Interaction Objects: once a context of use is known, the abstract part can be mapped onto a concrete part that is context-dependent. For instance, once a computing platform is targeted, it univocally determines the possible objects, turned into widgets, upon availability on this platform.

3. Placement of Concrete Interaction Objects: once the context is fixed, the physical constraints imposed by this context can restrict the design space for producing the running UI. Widgets can be laid out in containers, containers can be arranged together so as to create a final UI. Several algorithms exist that lay these widgets out.
4. Manual edition of Concrete Interaction Objects: after being laid out by the algorithm, the layout can be manually retouched using any traditional graphical editor, such as the one we can find in the Microsoft Visual Basic Development environment. This step is sometimes referred to as *beautification*, as it attempts to improve the final layout. Fig. 5 shows a sample of UIs generated for the example above.

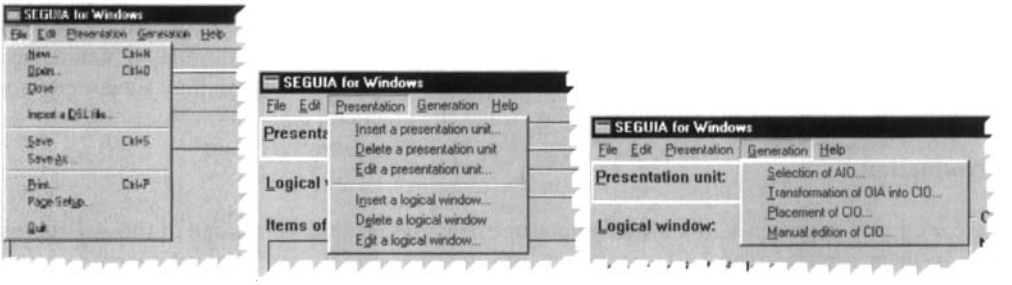

Figure 4: Menus of the model-based generator to generate a running UI

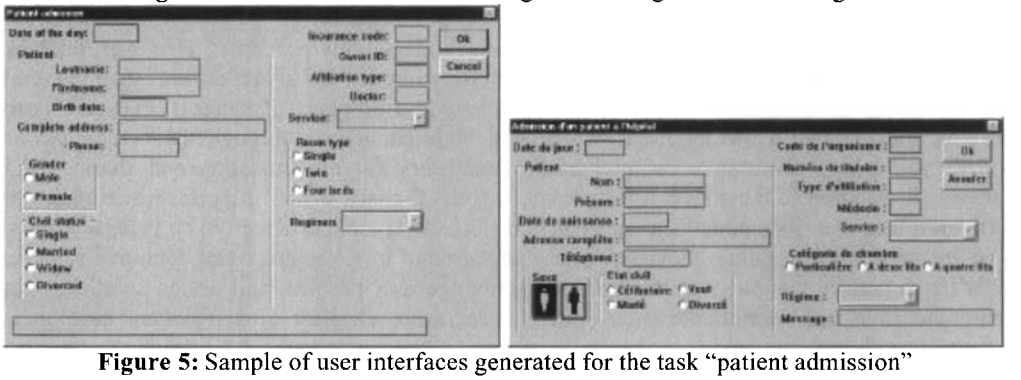

Figure 5: Sample of user interfaces generated for the task "patient admission"

References

Guarino, N. (1995). Formal Ontology, Conceptual Analysis and Knowledge Representation: The Role of Formal Ontology in the Information Technology. *International Journal of Human-Computer Studies*, 43(5-6), 625-640.

Savidis, A., Akoumianakis, D., and Stephanidis, C. (2001). The Unified User Interface Design Method. Chapter 21. In C. Stephanidis (Ed.), *User Interfaces for All: Concepts, Methods, and Tools* (pp. 417-440). Mahwah: Lawrence Erlbaum Associates Pub.

Puerta, A.R. (1997). A Model-Based Interface Development Environment. *IEEE Software*, 14(4), 41-47.

Top, J. and Akkermans, H. (1994). Tasks and Ontologies in Engineering Modelling. *International Journal of Human-Computer Studies*, 41(4), 585-617.

Vanderdonckt, J. and Bodart, F. (1993). Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection. in S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, and T. White (Eds.), *Proceedings of the ACM Conf. on Human Factors in Computing Systems INTERCHI'93 (Amsterdam, 24-29 April 1993)* (pp. 424-429). New York: ACM Press.

Vanderdonckt, J. (2000). A Small Knowledge-Based System for Selecting Interaction Styles. In *Proceedings of International Workshop on Tools for Working with Guidelines TFWWG'2000 (Biarritz, 7-8 October 2000)* (pp. 247-262). London: Springer-Verlag.

A Pattern Framework for Eliciting and Delivering UCD Knowledge and Practices

A. Gaffar, H. Javahery and A. Seffah
Human-Centered Software Engineering Group, Concordia University, Montreal
{gaffar, h_javahe, seffah}@cs.concordia.ca

Abstract: In the software and usability engineering community, there exist various tools for gathering and disseminating design knowledge. These tools aim to capture the best practices about the design of usable systems and to disseminate, or distribute, this knowledge. Examples of such tools include guidelines, patterns, various databases, and repositories of information. Guidelines and patterns are by far the most popular tools. This paper proposes a methodology supported by a tool for capturing and disseminating UCD knowledge and practices.

Introduction

There are many approaches to knowledge management and patterns is just one of those. Patterns do more than capture good UCD practices; they are also useful for documenting process and organizational strategies. They are an ideal vehicle for transferring, by means of software development tools, the design expertise of human-factor experts and UI designers to software engineers, who are usually unfamiliar with UI design and usability principles. However, the lack of common fulcrum and central repository for patterns make it hard to achieve this goal. Several pattern writers have introduced their own terminology and ontology of patterns. Our goal is to develop and validate a methodology, which will highlight a path through the heterogeneous pattern world, and result in a comprehensive framework for disseminating and sharing HCI patterns. In addition to the above, a further challenge is the lack of tool support, which makes it difficult to capture, disseminate and apply patterns effectively and efficiently [7]. Pattern writers, who are most often usability engineers with a background in psychology and cognitive science, must try to convey complex information to describe the user problem and design solutions in a comprehensible and comfortable way. Pattern users, who are most often software developers unfamiliar with usability engineering techniques, need also tools to understand when a pattern can be applied (context), how it works (solution), why it works (rationale), and how it should be implemented. To address this problem, our team is developing MOUDIL (Montreal Online Usability Patterns Digital Library) which is a method + a tool for capturing and disseminating patterns.

UCD Patterns Variety

Like the whole software engineering community [Gamma et al., 1995], the user interface design community has been a forum for a vigorous discussion on pattern languages for user interface design and usability engineering. It has been also reported that patterns are useful for a variety of reasons [10, 7].

Within our approach, we have been using three different categories of patterns:

- Interactive system design patterns. The aim of these patterns is to discuss and show a number of object-oriented design techniques and architectures for building flexible, portable, modifiable and extensible systems. A classical pattern of this category is the Observer that decouples between the user interface and the model [4]. For example, you might have a spreadsheet that has an underlying data model. Whenever the data model changes, the spreadsheet will need to update the spreadsheet screen and an embedded graph. In this

example, the subject is the data model and the observers are the screen and graph. When the observers receive notification that the model has changes, they can update themselves.

- Human computer interaction design patterns. These are proven user experiences and solutions to common usability problems. Different pattern languages have been developed and are being used as an alternative/complementary design tool to guidelines [3, 5, 9, 10].
- Process patterns [8] that describe user-centered design best practices. Such patterns have the potential to support UCD practices such as iterative design, low-fidelity prototyping as well as conducting a user satisfaction-oriented test using a questionnaire.
- Organizational patterns that depicted an effective organizational strategy for establishing the usability function in an organization. Basic patterns include training a champion, motivating developers, building a usability group.
- Software process improvements that describe a maturity scale for the assessment of an organization's progress towards human-centredness in system development and management. The scale is based on a number of models including Flanaghan's Usability Leadership Maturity Model, Sherwood-Jones' Total Systems Maturity Model and ISO 13407 human-centered Design Processes for Interactive Systems.
- UCD to software engineers' communication patterns which describes proven, successful approaches for organizing and managing the multidisciplinary team generals needed. Examples of this category of patterns include engage user patterns, build and use a persona, etc.
- Pedagogical patterns that implement effective approach for training developers in UCD skills. For example, the DIRR (Design-Implement-Redesign-Reimplement) pattern described in Table 6 attempts to explain new concepts and methods based on legacy concepts (for instance, learning object-oriented fundamental concepts using structural software design methods).

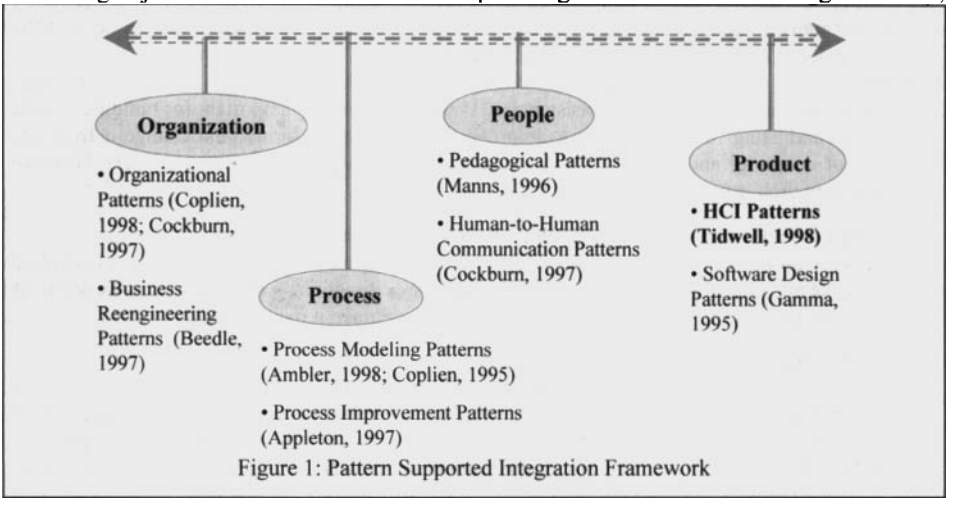

Figure 1: Pattern Supported Integration Framework

We use the term UCD pattern to refer to any of these categories of patterns. We also define a pattern supported integration framework as a collection of patterns and the relationships between them. A UCD pattern supported integration framework is a proven solution for integrating UCD practices and knowledge at the organizational, process, product and people levels (Figure 1). Each pattern is a three-part rule, which expresses a relation between a certain context, a certain system of forces that occurs repeatedly in that context, and a solution that allows these forces to resolve themselves.

The Seven C'S Methodology

However, each of these pattern collections introduces its own terminology, classification system and notation/format [8]. Our methodology called "The seven C's" aims of centralizing and organizing patterns into one repository, as well as disseminating pattern knowledge to the HCI community. Our methodology distinguishes seven steps:

- Collect: Place Different Research Work on Patterns in One Central Data Repository
- Cleanup: Change from Different Formats/Presentations into One Style
- Certify: Define a Domain and Clear Terminology for Our Collection
- Contribute: Receive Input from Pattern Community
- Categorize: Define Clear Categories and relationships for our Collection
- Connect: The Second Level of Complexity – Establishing Semantic Relationships between Patterns in a Relationship Model
- Control – Machine Readable Format for Future Tools

Collect: Place Different Research Work on Patterns in One Central Data Repository
Numerous works on patterns have been developed in the HCI community, however they are scattered in different places. A central repository of patterns will allow the user to concentrate on knowledge retrieval, rather than wasting time on searching for patterns. For this reason, we are collecting known references on patterns into one corpus. Currently our corpus includes more than 300 patters.

Cleanup: Change from Different Formats/Presentations into One Style
Ideally, different works on patterns deal with different problems. However, as we went through step 1, we were able to identify that some patterns are dealing with different sides of the same problem (correlated patterns), some patterns are offering different solutions to the same problem (peer patterns/competitors) and some are even presenting the same solution to the same problem (similar), only in different collections with different presentation formats (redundant patterns). Since a large number of patterns have differing presentation formats, it is difficult to detect these and their relations with other patterns.
Putting patterns in a unified format will help discover these relationships, put related patterns closer together, and possibly remove the redundancies/inconsistencies. This is an extremely important for building a common ground. We are conducting further research to improve the presentation format and change it from an *art*, requiring a lot of creativity and expertise, to a systematic and possibly automated approach. This will be developed further in the *Control* step.

Certify: Define a Domain and Clear Terminology for Our Collection
The available work on patterns is tremendous; it is not wise, feasible or even useful to collect everything in one place. To make any collection useful, it has to focus on a specific domain. For this reason we are working on terminology to clearly define what belongs in our collection to make it inclusive and concise.

Contribute: Receive Input from Pattern Community
New patterns emerge all the time in all areas of the scientific community, including HCI. It is very difficult to keep track of these emerging patterns. Typically, it would take years before an expert can come up with a thorough collection of patterns [1, 2, 4] or have time to update an existing collection [9, 10]. Having one central repository for patterns will help to unify pattern knowledge captured by different individuals. Furthermore, such a repository will help to add emerging patterns quickly, so that they are made available to the community. We will therefore have a continuously evolving collection of patterns.

Categorize: Define Clear Categories for our Collection
Within our collection, we need to be able to create a hierarchy of categories to make them manageable. The first goal of categorization is to reduce the complexity of searching for, or understanding, the relationship between patterns. The second, and more important goal, is to build a model for our categories. We are inspired by the evolution process in other domains like C++ (hierarchies in I/O classes, STL hierarchies, etc) and Java (evolving hierarchies in event handling models, etc).

Connect: The Second Level of Complexity – Establishing Semantic Relationships between Patterns in a Relationship Model

A significant part of knowledge associated with patterns lies in the relationships between them. Finding and documenting these relationships will allow developers to easily use patterns as one integrated part to develop an application, instead of relying on their common sense and instinct to pick up some patterns that seem to be suitable. A proven model for the pattern collection will help to define an ontology for the pattern research area with all proper relationships such as inference, equivalence and subsumption between patterns.

Control – Machine Readable Format for Future Tools

Once a model is established, it will enhance the process of automating the UI design. The ultimate goal of many applications is to interact with the machine as a viable partner that can read, understand our work, and then contribute to it in an intelligent way. In short, having a machine-readable format can help automate the process of UI design using patterns.

Tools Support

In order to accelerate the implementation of our seven C's methodology, tool support is necessary. Each step requires a certain tool. Our research team is currently developing an Integrated Pattern Environment (IPE), called MOUDIL, which will unite all necessary functionality. MOUDIL was originally designed with two major objectives: Firstly, as a service to UI designers and software engineers for UI development. Secondly, as a research forum for understanding how patterns are really discovered, validated, used and perceived. One last objective for MOUDIL, in addition to the above two, is to use it as a prototypical implementation of an IPE. It will be able to provide functionality that supports every step of the seven C's methodology.

In particular, MOUDIL as medium for delivering patterns provides the following key features:

- At the heart of MOUDIL is a database designed specifically to serve as a central repository for patterns. These patterns can be backed up with examples and supported by other content areas – guidelines, case studies, checklists, reusable assets and components, templates, and resources.
- MOUDIL has been designed to accept proposed or potential patterns in many different formats or notations. Therefore patterns in versatile formats can be submitted for reviewing.
- Reviewing tools that can allow an international editorial board to evaluate patterns. Before publishing, collected and contributed patterns must be accessed and acknowledged by the editorial committee.
- Pattern Ontology editor captures our understanding of pattern concepts and puts them into relation with each other (Taxonomy).
- The MOUDIL Pattern Editor allows us to attach semantic information to the patterns. Based on this information and our ontology, patterns will be placed in relationships, grouped, categorized and displayed.
- The pattern navigator provides different ways to navigate through patterns or to locate a specific pattern. The pattern catalogue can be browsed by pattern groups or searched by keyword. Moreover, a pattern wizard will find particular patterns by questioning the user.
- The pattern viewer provides different views of the pattern, adjusted to the preferences of the specific user.

MOUDIL should include the following features:

- Quality assurance checklists
- A glossary of pattern terminology
- Booklist defined by category and includes ratings and links to an internal company book site or external book site, such as amazon.com
- Resources for developing patterns-oriented designs. Examples of resources available to developers and users include graphics, tables, and templates
- Case studies that add contextual meaning. These patterns can be backed up with examples and supported by MOUDIL other content areas such as case studies, checklists, and resources.

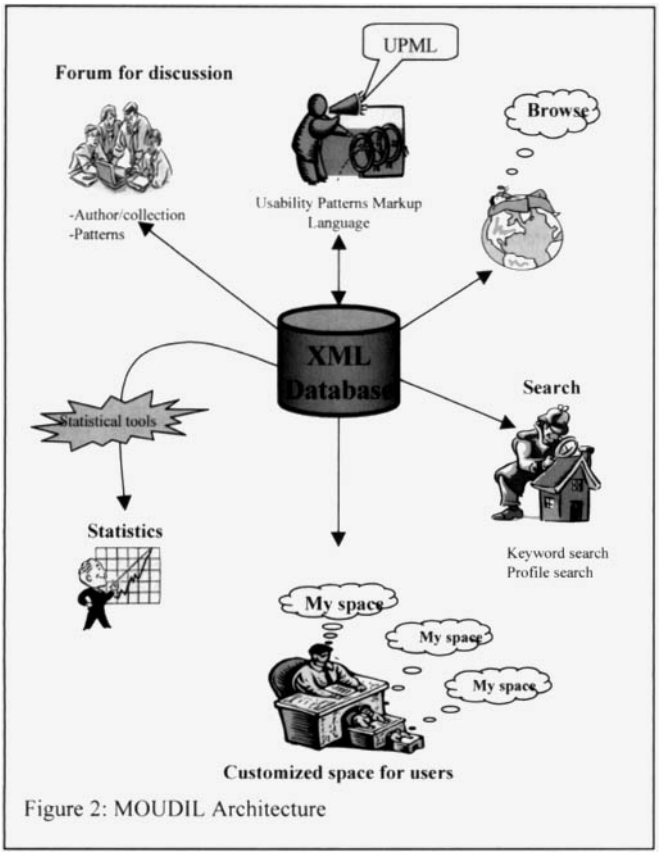

Figure 2: MOUDIL Architecture

Conclusion

Patterns are useful in gathering and documenting experiences for future developers. A great deal of work has been done on HCI patterns by many different individuals. Patterns do not only provide help in applying UCD practices, but also provide support for understanding and mastering UCD concepts and applying related methods. The lack of knowledge centralization, however, requires users to hunt for suitable patterns, and extract them for their own use. Gathering relevant patterns in one repository will help overcome this difficulty. In our future research, we want to use the gathered information from the collected and analyzed patterns to come up with a formal pattern notation. Such a notation will help capture and disseminate pattern knowledge effectively.

References

1. Alexander, C. *The Timeless Way of Building*. New York: Oxford University Press, 1979.
2. Alexander C., Ishikawa S., Silverstein M., Jacobson M., Fiksdahl-King I., and Angel S., *A Pattern Language: Towns, Buildings, Construction*. New York: Oxford University Press, 1977.
3. Coram, T., Lee, J. *Experiences - A Pattern Language for User Interface Design*. Available at http://www.maplefish.com/todd/papers/experiences/Experiences.html.
4. Gamma, E., Helm, R., Johnson, R. and Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.
5. Griffiths, R. *Brighton Usability Pattern Collection*. Available at http://www.it.bton.ac.uk/cil/usability/patterns/.
6. HCI Department, Concordia University, Montreal. *MOUDIL: Montreal Online Usability Digital Library*. Available at http://hci.cs.concordia.ca/moudil/homepage.php.
7. Seffah, A., Javahery, H. *A Model for Usability Pattern-Oriented Design*. In Proceedings of TAMODIA 2002, (Bucharest, Romania, July 2002).
8. Seffah, A., Javahery, H. *On the Usability of Usability Patterns*. Workshop entitled *Patterns in Practice*, CHI 2002, (Minneapolis, Mi, April 2002).
9. Tidwell, J. *COMMON GROUND: A Pattern Language for Human-Computer Interface Design*. Available at http://www.mit.edu/~jtidwell/interaction_patterns.html
10. Welie, M. *Interaction Design Patterns*. Available at http://www.welie.com/patterns/.

Towards Virtual Intuitive Tools for Computer Aided Design

Yvon Gardan

CMCAO Team / IFTS
Pôle de Haute Technologie
08000 Charleville-Mézières, France
gardan@infonie.fr

Erwan Malik

CMCAO Team / IFTS
Pôle de Haute Technologie
08000 Charleville-Mézières, France
erwan.malik@univ-reims.fr

Estelle Perrin

CMCAO Team / University of Metz
Île du Saulcy
57045 METZ Cedex 01, France
perrin@sciences.univ-metz.fr

Abstract

The main objective of the DIJA project is to provide a CAD system through the internet and to help designers from the early phases of the design with an intuitive interface. Distributing a CAD system over the World Wide Web implies that this system has not any knowledge about its user (his computer and CAD skills, his trade, etc.). The particularity of our system is that it is based on a new design method, called the "synthetic" approach. In this paper, we focus our attention on the man-machine interface part of the DIJA project: our goal is to provide the user with the ability to express his intent. We propose a set of virtual tools used to control the design and we consider the importance of the media in the user-system dialog.

1 Introduction

Current CAD systems are monolithic and their computer-human interface is not easily accessible for a non-expert user. It is common that the latter has to learn about a CAD system before use. The reason is that CAD systems do not integrate properly functions from requirements. In fact, they completely forget and hide functional intent, design intent and trade know-how. They just deal with features, which is inadequate to really take into account design intent. Features are very close to classical geometric models (such as Boundary representation, Constructive Solid Geometry, ...), so they are enclosed in a way of modeling that imposes the user a method to build models, essentially based on basic objects and operations. At the same time, CAD systems have a rustic interface that is based on standard devices (such as mouse) and number of menus soars with increase of services.

We propose, in this paper, an interface to provide the user with the ability to express his intent, based on a modeling method, called the "synthetic method". Both interface and modeling method are parts of the DIJA project (Danesi, Denis, Gardan & Perrin, 2002). The main objective of the DIJA project is to provide a CAD system through Internet and to help designers from the early phases of the design of an object with an intuitive interface. The basic idea of the synthetic method is that the user, even by giving a function or by choosing a form, obtains a very approximate shape. This shape, from a certain point of view, only represents the topology of the object. The user can then deform this shape using some dedicated tools. The initial form can be very close to or far away from the goal. It does not have really influence, the method being always the same, requiring more or less steps. Obviously, an expert user will choose a form necessitating only a little number of steps.

The synthetic method respects a top down methodology which seems very interesting for many applications. It considers basic deformable objects depending on domain of the application. Each domain has its proper rules of design and its own vocabulary. The words are different and even the same words can lead to different signification according to the trade. Then the language which is at the disposal of the user is domain dependent. A word (specifically a verb) of the language can lead to a general behaviour (that means that this word has a shared meaning, which is the case of those we describe hereafter) or to a domain dependent behaviour. Three kinds of behaviour or tool are defined:

- deformation: the object is deformed following some rules described later;
- dividing: an object is divided in several parts of the same type;
- transmutation: the type of the object changes (for instance a cylinder becomes a box).

In this paper, we only focus on the deformation tool of the DIJA project. The user deforms the object, by stretching one of the shape surfaces (surface deformation), or by deforming the shape globally (shape deformation). Both deformations perform the same interactions which are fully described in the next section: selection of a shape, choice of a tool and applying the tool. In the last interaction, the user has to describe how the tool will affect the shape in terms of direction and strength. Such notions are not easily received by a standard device. In this context, we present (section 3) an interface based on a camera that we think to be more intuitive.

2 Virtual Intuitive Tools of Dija Project

Traditionally, with CAD systems, the user has to express deformation by using a geometric or mathematic description (references). On the contrary, by using tools, the user can focus on his design rather than on geometric aspects of the object he wants to achieve. Then he can consider his design more like a virtual object and thus, he can use deformations close to those from real world. In DIJA, to deform an object, the user has to define the shape of the tool he will use – like bending a bar using the sharp edge of a table corner or the smooth shape of a large cylinder. Through this, we want to increase the relation between the user and his working environment by very simple meanings.

We define a tool by associating a shape with some vocabulary terms. For example, the tool shape represented in figure 1a is associated with the term "to swell" in a knowledge based model. This knowledge based model associates a shape to a term in function of a trade context. For example, the maximum or minimum of swelling depends on mechanical considerations. Considering two different trades, the system would not propose the same shape for the same term. This is realised by using fuzzy parameters (Zadeh, 1997). Nonetheless, other terms fully specify the tool shape like "sharp" or "narrow". Obviously, an internal geometric representation is also used by the system (by example, "to swell" is represented by conics such as a circle or ellipse) but the user is not aware of it. He only manipulates a trade vocabulary that protects him from geometric reflections.

The method we use to get the tool shape follows the same synthetic approach than the rest of our project. The user starts the process by choosing a trade and a term which qualifies the deformation. Our system deals with this term to propose a standard shape as we mentioned above. Finally, the shape is refined by using adverbs and/or adjectives belonging to the tool's vocabulary through an iterative process. Once again, the system reactions will depend on the previous user interactions. If the user asks for "more sharp" for the fourth time, the result will not be the same as the first time. As for trade-oriented tool definition, this is realised by fuzzy parameters.