Computer Aided Improvement of Human-Centered Design Processes

ABSTRACT

With the increasing relevancy of usability as a software quality factor, a growing demand for tools that support the special activities of HCD (human-centered design) processes within the industrial software development community is expected to arise. To elicit initial requirements for tool support, we analyzed the real tasks of developers of four major companies engaged in the development of highly interactive systems, the problems incurred and the burning support issues. Inspired by the results of this survey we developed REUSE, a system that facilitates the elicitation, organization and effective reuse of best practices and artifacts concerning HCD activities and makes their performance more efficient. An initial formative evaluation with future users showed that REUSE is able to improve the utilization of HCD methods within the development process of interactive systems and overcomes some limitations of related approaches.

1 INTRODUCTION

HCD methodologies like Usability Engineering [1, 2], Contextual Design [3] or Usage-Centered Design [4] have been successfully applied through various projects. They have proved that they have dramatically been able to improve usability and user acceptance of interactive software systems if they are applied in the correct development context.

On the other hand specific knowledge about exactly how to most efficiently and smoothly integrate HCD methods into established software development processes [2] is still missing. Little research has been done on integrating methods and tools of HCD in the development process and gathering knowledge about interface design in a form that can capture relationships between specific development contexts and applicable methods, tools and heuristics [5].

One vital step toward bridging this gap is to provide the people involved in HCD processes with a tool to effectively support their activities. Existing tools for supporting HCD processes beyond computer-aided design of user interfaces (e.g. (semi-) automatic GUI builders [6]), largely focus on two aspects of the process: supporting the performance and evaluation of usability tests, e.g. [7-9], or supporting developers accessing guidelines, e.g. [5, 10, 11].

Most of these approaches assume that there is already a well-established HCD process in place that is practiced by the development team and that provides use of the proposed methods and tools.

To examine this assumption, we initiated a survey to find out what kind of development process for interactive systems is practiced by development organizations engaged in engineering highly interactive systems. The study comprised an analysis of the typical tasks the members of the development teams have to perform and the common difficulties encountered as related to HCD activities.

Taking the results of this survey into account, we had to revise some of the assumptions about requirements for a tool to support HCD processes. These new findings have influenced the development of REUSE, a tool to support the introduction, establishment and continuous improvement of HCD processes under the constraints discovered in the survey.

In this paper we present the structure, performance and results of the survey. We explain the conclusions drawn and their impact on the design and development of the REUSE system. We present the main concepts and major components of REUSE and show how the system addresses those problems identified in our survey. Initial results of a formative evaluation and their effects on future developments are also presented. Finally, we discuss work related to our approach and take a look at future research issues.

2 EXAMINING HCD PROCESSES IN PRACTICE

To be able to construct a tool for the effective support of HCD processes, we needed in-depth knowledge of the future users of such a tool and their requirements. This led to the following central questions:

- What kind of development process for interactive systems is practiced by the development organizations in their projects?
- What typical tasks do the developers have to solve?
- What problems are typical for the development process?
- What kind of support is needed?

2.1 Performance of the survey

The survey was elaborated, performed and evaluated in collaboration with industrial psychologists[12]. A total of 16 employees from four major companies[1] involved in the development of highly interactive software systems were selected. The respondents are engaged in developing these systems in projects from diverse domains: military systems, car driver assistance technology or next-generation home entertainment components. The questioning was performed by a single interviewer and the answers were recorded by a second person in a pre-structured protocol document. Each interview took between 90-150 minutes.

2.2 Results

The results of the survey on tool support for HCD processes can be summarized as follows:

The organizations examined are practicing highly diverse individual development processes, however non of the HCD development models proposed by [1-4] are exactly used.

The persons who are entrusted with the ergonomic analysis and evaluation of interactive systems are primarily the developers of the products. External usability or human factors experts or a separate in-house ergonomics department are seldom available. Furthermore, few of the participants were familiar with basic methods like *user profile analysis* or *cognitive walkthrough.*

[1] DaimlerChrysler Aerospace (DASA) in Ulm, Sony in Fellbach, Grundig in Fuerth and DaimlerChrysler in Sindelfingen (all sites are located in Germany)

The HCD methods that are considered to be reasonable to apply by the respondents are often not used for the following interrelated reasons:

- There is no time allocated for HCD activities: they are neither integrated in the development process nor in the project schedule.
- Knowledge needed for the performance of HCD tasks is not available within the development team.
- The effort for the application of the HCD tasks is estimated to be too high because they are regarded as time consuming.

2.3 Survey conclusions

The results of the survey led to the following conclusions regarding the requirements of a software tool to support HCD processes:

- Requirement 1: Support flexible HCD process models

The tool should not force the development organization to adopt a fixed HCD process model as the practiced processes are very diverse. Instead, the tool should facilitate a smooth integration of HCD activities into the individual software development process practiced by the organization. Turning technology-centered processes into human-centered processes should be seen as a continuous process improvement task where organizations learn which of the methods available best fit in certain development contexts, and where these organizations may gradually adopt new HCD methods.

- Requirement 2: Support evolutionary development and reuse of HCD experience

It was observed that the staff entrusted with ergonomic design and evaluation often lacks a special background in HCD methods. Yet, as the need for usability was recognized by the participating organizations, they tend to developed their own in-house usability guidelines and heuristics. Recent research [13-16] supports the observation that such usability best practices and heuristics are, in fact, compiled and used by software development organizations. Spencer [15], for example, presents a streamlined cognitive walkthrough method which has been developed to facilitate efficient performance of cognitive walkthroughs under the social constraints of a large software development organization. However, from experiences collected in the field of software engineering [17] it must be assumed that, in most cases, best practices like Spencer's are unfortunately not published in either development organizations or the scientific community. They are bound to the people of a certain project or, even worse, to one expert member of this group, making the available body of knowledge hard to access. Similar projects in other departments of the organization usually cannot profit from these experiences. In the worst case, the experiences may leave the organization with the expert when changing jobs. Therefore, the proposed tool should not only support high-level human factors methods but also allow the organizations to compile, develop and evolve their own approaches.

- Requirement 3: Provide means to contextualize HCD knowledge

HCD methods still have to be regarded as knowledge-intensive. Tools are needed to support developers with the knowledge required to effectively perform HCD activities. Furthermore, the tool should enable software development organizations to explore which of the existing methods and process models of HCD works best for them in a certain development context and how they can refine and evolve basic methods to make them fit into their particular development context.

- Requirement 4: Support efficient performance of HCD activities

There is a definite need for tool support when it comes to enabling efficient performance of otherwise tedious and time consuming HCD activities. Otherwise, these essential activities fall victim to the no-time-to-sharpen-the-saw-because-too-busy-cutting-the-wood syndrome. With respect to requirement 2, this means that the tool has to support the efficient elicitation, organization and reuse of best practices and artifacts relating to HCD tasks. The proposed tool should increase the efficiency of these activities - not introduce additional efforts for questionable benefits.

3 THE REUSE SYSTEM

To transfer the above requirements into a software tool that supports the elicitation, organization and reuse of HCD knowledge the related best practices and artifacts are represented and organized using the concepts of a *context model* and a set of *USEPACKs* (Usability Engineering Experience Package). The REUSE system *(Repository for Usability Engineering Experience*) provides four components for manipulating USEPACKs and context models, shown in the architecture depicted in Figure 1. The components of REUSE will be discussed in more detail after introducing the concepts of USEPACKs and context models. We differentiate between two virtual roles played in the utilization of the REUSE system: *readers* who search, explore and apply USEPACKs and *authors* who create, compile and organize USEPACKs.

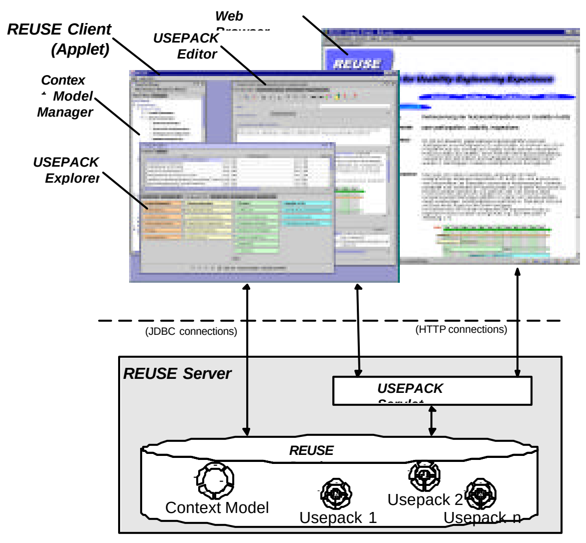

Figure 1: Architecture of REUSE

3.1 The USEPACK concept

A USEPACK is a semi-formal notation for structuring knowledge relating to HCD activities. It encapsulates best practices on how to most effectively perform certain HCD activities and includes the related artifacts like documents, code fragments, templates and tools that facilitate the compliance with the best practice described.

A USEPACK is structured into five sections:

- The *core information* permits authors to describe the main message of a USEPACK. It is organized according to the pyramid principle for structuring information [18]. The information first presented to the reader has a low level of complexity, allowing the reader to quickly decide if the USEPACK is worth further exploration. With further reading, the degree of complexity rises, introducing the reader to the experience described. The core information section includes the fields *title*, *keywords*, *abstract*, *description* and *comments.*
- The *context situation* describes the development context related to the experience in question. The context situation is generated by using the context model, allowing the authors and readers of USEPACKs to utilize a shared vocabulary for contextualizing and accessing USEPACKs.

- A set of *artifacts,* such as checklists for user profile analysis or templates for usability test questionnaires, facilitates the efficient compliance with the best practice. They represent an added value to the readers of a USEPACK. Artifacts allow readers to regain time spent on exploring the package by using the supplied artifacts to simplify their work.
- A set of *semantic links* pointing to other USEPACKs or external resources which, for example, support or contradict the best practice described. By interlinking USEPACKs and connecting them to external resources, like web pages, a net of related experiences can be created to provide more reliable information than through the isolated sets of USEPACKs.
- The *administrative data* section is used to store data like the *author(s) of the package, the date of creation, access rights* and statistical data such as the *number of accesses* to the package and a *user rating*.

3.2 The context model concept

The context model serves as a template to construct the *context situation* for USEPACKs – a semi-formal description of the context in which the information of a USEPACK can be applied. It is organized in a tree structure, divided into sections which contain groups of *context factors*. On the one hand, authors can use the context model to easily construct a description of the context in which the information of a USEPACK can be applied by selecting appropriate context factors from the model. On the other hand, readers can use the context model to specify a context situation which reflects the development context for which they need support in the form of USEPACKs.

Currently a context model containing the following five sections is used:

- The *process context* section provides context factors to describe elements of the development process used, such as process phases (e.g. 'User Interface Design') as well as roles (e.g. 'Usability Engineer') and deliverables (e.g. 'User Interface Styleguide') related to the experience in question.
- The *project context* section provides context factors to describe project constraints like the size of the development team, budget or project duration which are related to the experience cited.
- The *domain context* section provides context factors to describe elements of the domain related to the experience described. Top-level context factors of this section specify domains in terms like 'home entertainment systems' or 'car driver assistance systems', which can be subsequently refined to capture more detailed domain attributes.
- The *technology context* section provides context factors to describe features of technologies related to the experience described like 'gesture recognition' or 'speech input'.
- The *quality context* section provides context factors to describe quality factors of standards (e.g. ISO9241-11 [19] with quality factors such as self-descriptiveness or error tolerance) related to the experience in question.

Users never work directly on the context model, instead they manipulate the model by interacting with components of the REUSE system. These components hide much of the complexity of the context model and provide appropriate means for the manipulation of each section of the context model.

3.3 Components of REUSE

The *USEPACK editor* component makes it possible for authors to create a new USEPACK, to delete packages and to change or comment on existing packages. To support this task, the editor includes specialized assistants for work on the related sections of a USEPACK.

The *USEPACK explorer* provides different views of the set of USEPACKs available and various filters to facilitate convenient browsing and retrieval of those USEPACKs. The USEPACK explorer facilitates the search for USEPACKs that share certain context factors specified by the reader. An example for one of these filters is the *process context filter*, which builds a graphical representation of the HCD process model defined in the process context section of the context model. By directly selecting process elements in the graphical representation, the reader can easily create a context situation for a query.

The *context model manager* component enables users to edit and extend the currently used context model with new context factors, e.g. to adapt REUSE to new process models or application domains. By this means, the REUSE module and the underlying models about processes, projects, domains, technologies and quality standards can be evolved in concert with the growing amount of experience to meet the needs of the organization.

The USEPACK *servlet* displays USEPACKs on a standard web browser in read-only mode and allows convenient browsing of the net of linked USEPACKs.

3.4 Use Cases for the REUSE system

The following two use cases show the utilization of the REUSE system.

3.4.1 Use Case 1: Capturing an new experience

A usability expert of a development organization conceived a new approach for efficient performance of cognitive walkthroughs in large software development projects, called the streamlined cognitive walkthrough method. He captures his experience by using the USEPACK editor to generate and describe a new USEPACK. Table 1 shows an excerpt of the information which is to be recorded in this USEPACK.

Title:	*Performing streamlined cognitive walkthroughs (SCW)*
Abstract:	Contains a short outline of the advantages of the SCW over the original cognitive walkthrough
Description:	Contains a description of the SCW and its four ground rules.
Artifacts:	Contains a template for the agenda of a SCW session, a template for recording the critical output of a SCW and a document describing the details of the SCW method.
Context-Situation:	The described experience is linked to the process phase 'User Interface Design and Evaluation' and the process step 'Iterative User Interface Walkthroughs' as well as to the process phase 'Evaluation an tests'. Furthermore the experience was gained in a large development organization with a large development team. In the context situation of this USEPACK

	these context factors are marked.

Table 1 : Sample excerpt of the information of a USEPACK

Figure 2 shows the how the core information section and the artifacts of the USEPACK are presented in the USEPACK editor. Figure 3 shows how the description of the context situation is done in the USEPACK editor. The user can visually manipulate the context situation by using controls like check boxes and sliders – no textual input is required.

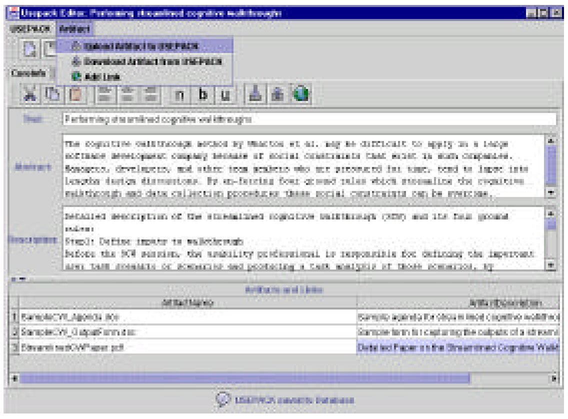
USEPACK
Artifact
Upload Artifact to USEPACK
Download Artifact from USEPACK
Add Link
Performing streamlined cognitive walkthroughs
Artifacts and Links

Figure 2 : Editing the core information

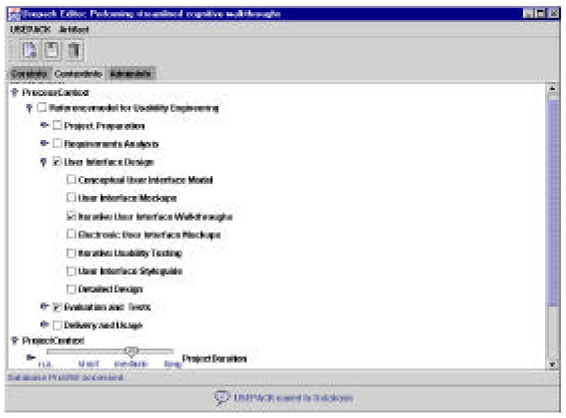

Figure 3 : Editing the context information

3.4.2 Scenario 2: Reusing an existing experience

A usability novice within the development organization has to perform a streamlined cognitive walkthrough because the usability expert is not available in his project. In order to look for experiences how to perform this task he utilizes the USEPACK explorer to search for adequate information. Starting with the graphical overview of the usability process model, he focuses on experiences in the field of iterative user interface walkthroughs by selecting the process step ‘Iterative User Interface Walkthroughs’ in the process context filter. As a result the USEPACK with the title ' Performing streamlined cognitive walkthroughs’ is shown to him. By exploring the USEPACKs core information and its context situation he can easily decide if this USEPACK is useful in his development context and he can exploit the USEPACKs artifacts to perform streamlined cognitive walkthroughs. Later on he is able to extend the existing USEPACK by adding his own experiences.

In this way an organization is able to augment its development processes with REUSE to facilitate an organizational learning process in HCD.

4 EVALUATION

A formative evaluation [1] was conducted in order to gain feedback about the usefulness of the functionality of REUSE and the usability of its components. Thinking-aloud [21] was selected as test method.

4.1 Participants

Six participants who were selected to cover several roles of potential users of the REUSE system were identified for the evaluation. The group consisted of one usability engineer, one user interface designer, one usability inspections specialist and three UI developers, each with professional experience in the development of highly interactive systems.

4.2 Procedure

First, the participants were allowed to play and experiment with the system for some time. Then the participants had to read a global scenario introducing them to the test tasks. The next steps included reading the three main test tasks. To fulfill these tasks, the participants had to make extensive use of all the features and components of REUSE.

A usability lab was deployed to simultaneously record the interactions on the screen, the participant and the comments from the participant. After all usability test sessions had been performed, we analyzed the video tapes and documented the usability problems, new design ideas and suggestions for improvement identified by the participants during the operation of REUSE.

4.3 Results

In the first phase of exploration of the system some participants encountered difficulties in understanding the concepts of context models and context situations. However, these difficulties disappeared when the participants were introduced to the system by reading the scenarios and test tasks. All participants were able to completely solve all test tasks without interventions by the test staff.

Most usability problems were caused by the context model manager component. This component offered the users only a low level of abstraction concerning the concept of the context model. For the related activities of maintaining and extending context models the users demanded for a more proactive support.

No major usability problems were discovered in the USEPACK editor. Users easily constructed USEPACKs guided by the USEPACK template and contextualized USEPACKs supported by the context situation assistant.

The participants highly appreciated the browsing features of the USEPACK explorer which facilitated easy exploration of the information space. They were especially enthusiastic about the overall idea of sharing their knowledge with their colleagues and profit from experiences of previous projects.

5 RELATED WORK

A line of research related to the approach presented concentrates on augmenting the design process by offering tools for working with guidelines. Most of this work focuses on the effective organization and presentation of existing guidelines, e.g. Smith & Mosier’s ‘Guidelines for Designing User Interface Software’ [23], styleguides, e.g. the ‘OSF/Motif Styleguide’ [24] and national or international standards like ‘ISO9241-11’ [19] by using tools like SIERRA [25] , GuideBook [26] or HyperSAM [27]. A comprehensive summary of this work was compiled by Vanderdonckt [10].

One major concern regarding these approaches is the danger of getting ‘decontextualized’ guidelines [5, 10, 28]. Decontextualization in this regard means that it often remains unclear in which development context the described guidelines should be applied, thus complicating their access, interpretation and the estimation of their potential utility. REUSE offers a comprehensive solution to these problems by providing a context model shared between authors and readers of USEPACKs, thus explicitly facilitating contextualization of guidelines. By their open, evolving nature, context models enable the users to keep track of the application context of guidelines even if the underlying processes, technologies, domains and quality standards are still evolving.

6 CONCLUSIONS

We performed a survey to examine the development processes of companies engaged in the development of highly interactive software systems and found a high potential for improvement regarding HCD activities. The concepts and components implemented in REUSE represent an initial step towards an open tool geared for supporting HCD processes to facilitate efficient, painless performance of HCD activities. First results of a formative evaluation showed that REUSE meets the requirements elicited in the survey, is well accepted by the users and overcomes some limitations of related approaches.

7 ACKNOWLEDGMENTS

This research was sponsored in part by the BMBF[2] award #01 IL 904 B7 of the EMBASSI[3] project. We would like to thank all participants of our studies.

8 REFERENCES

1. J. Nielsen, *Usability Engineering*: Morgan Kaufman Publishers, 1994.
2. D. J. Mayhew, *The Usability Engineering Lifecycle: A Practioner's Handbook for User Interface Design*: Morgan Kaufman, 1999.
3. H. Beyer and K. Holtzblatt, *Contextual Design: Defining Customer-Centered Systems*: Morgan Kaufmann Publishers, 1998.
4. L. L. Constantine and L. A. D. Lockwood, *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*: Addison-Wesley, 1999.
5. S. Henninger, “A Methodology and Tools for Applying Context-Specific Usability Guidelines to Interface Design,” *Interacting with Computers*, vol. 12, 2000.
6. B. A. Myers, “User Interface Management Systems,” in *Wiley Encyclopedia of Electrical and Electronics Engineering*, vol. 23, J. G. Webster, Ed. New York: John Wiley & Sons, 1999, pp. 42-58.
7. M. Macleod and R. Rengger, “The Development of DRUM: A Software Tool for Videoassisted Usability Evaluation,” presented at BCS Conference on People and Computers VIII HCI'93, Lougborough, 1993.
8. D. Uehling and K. Wolf, “User Action Graphing Effort (UsAGE),” presented at ACM Conference on

[2] Bundesministerium für Bildung und Forschung: German Ministry for Education and Research

[3] Elektronische, Multimediale Bedien- und Service Assistenz: Electronic, Multimedia Operating and Service Assistance

Human Aspects in Computing Systems CHI'95, Denver, 1995.

9. G. Al-Quaimari and D. McRostie, “KALDI: A Computer-Aided Usability Engineering Tool for Supporting Testing and Analysis of Human-Computer Interaction,” presented at CADUI99: Computer Aided Design of User Interfaces, Louvain-la-Neuve Belgium, 1999.

10. J. Vanderdonckt, “Development Milestones Towards a Tool for Working with Guidelines,” *Interacting with Computers*, vol. 12(2), 1999.

11. D. Grammenos, D. Akoumianakis, and C. Stephanidis, “Integrated Support for Working with Guidelines: The Sherlock Guideline Management System,” *Interacting with Computers*, vol. 12, (2),2000.

12. E. Wetzenstein and A. Becker, “Requirements of Software Developers for a Usability Engineering Environment,” Artop Institute for Industrial Psychology, Berlin 2000.

13. S. Weinschenk and S. C. Yeo, *Guidelines for Enterprise-wide GUI design*. New York: Wiley, 1995.

14. P. A. Billingsley, “Starting from Scratch: Building a Usability Programm at Union Pacific Railroad,” *Interactions*, vol. 2, pp. 27-30, 1995.

15. R. Spencer, “The Streamlined Cognitive Walkthrough Method: Working Around Social Constraints Encountered in a Software Development Company,” presented at CHI2000, The Hague, Netherlands, 2000.

16. S. Rosenbaum, J. A. Rohn, and J. Humburg, “A Toolkit for Startegic Usability: Results from Workshops, Panels and Surveys,” presented at CHI 2000, The Hague, Netherlands, 2000.

17. V. R. Basili, G. Caldiera, and H. D. Rombach, “Experience Factory,” in *Encyclopedia of Software Engineering*, vol. 1, J. J. Marciniak, Ed. New York: John Wiley & Sons, 1994, pp. 528-532.

18. B. Minto, *The Pyramid Principle - Logic in Writing and Thinking*, 3 ed. London: Minto International Inc., 1987.

19. ISO/TC 159 Ergonomics, “Ergonomic requirements for office work with visual display terminals - Part 11: Guidance on usability,” ISO International Organization for Standardization ISO 9241-11:1998(E), 1998.

20. ISO/TC 159 Ergonomics, “Human-centered Design Processes for Interactive Systems,” ISO International Organization for Standardization ISO 13407:1999(E), 1999.

21. A. H. Jørgensen, “Using the thinking-aloud method in system devlopment,” in *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*. Amsterdam: Elsevier Science Publishers, 1989, pp. 743-750.

22. DaimlerChrysler Aerospace AG, DaimlerChrysler AG, Siemens Electrocom GmbH, Carl Zeiss Oberkochen, and Technical University of Ilmenau, “OSVA Final Report - Chapter 7.2: Experience Based Usability Engineering,” Technical University of Ilmenau 01 IS 605 A4, Mai 1999 1999.

23. S. L. Smith and J. N. Mosier, “Guidelines for Designing User Interface Software,” The MITRE Coporation, Bedford ESD-TR-86-278 MTR-10090, 1986.

24. Open Software Foundation, *OSF/Motif Styleguide*. Englewood Cliffs: Prentice-Hall, 1992.

25. J. Vanderdonckt, “Accessing Guidelines Information with SIERRA,” presented at IFIIP Conference on Human Computer Intercation Interact'95, Lillehammer, Norway, 1995.

26. K. Ogawa and K. Useno, “GuideBook: design guidelines,” *Special Interest Group on Human Computer Interaction SIGCHI*, vol. 27, pp. 38-39, 1995.

27. R. Ianella, “HyperSAM: a management tool for large user interface guideline sets,” *Special Interest Group on Computer Human Interaction SIGCHI*, vol. 27, pp. 42-43, 1995.

28. F. d. Souza, “The use of Guidelines in Menu Interface Design: Evaluation of a Draft Standard,” presented at 3rd IFIP Conference on Human Computer Interaction (INTERACT'90), Cambridge, 1990.

29. J. Kolodner, *Case-based Reasoning*: Morgan Kaufmann Publishers, 1993.
30. Sun Microsystems, “Forte for Java - Development Tools for the latest Java Technology,” : Sun, 2000.