Support for Iterative User Interface Prototyping: The Sherlock Guideline Management System

D. Grammenos, D. Akoumianakis, C. Stephanidis
Institute of Computer Science,
Foundation for Research and Technology - Hellas
Science and Technology Park of Crete,
Heraklion, Crete, GR-71110 GREECE

Abstract: This paper is about supporting the difficult and non-trivial task of user interface design by providing effective human factors input to early stages of system development. The work presented in this paper is motivated by the normative perspective that tools for working with guidelines should provide a collaborative, extensible and evolutionary medium, offering more than mere access to guideline reference manuals or hypertext retrieval, for early human factors design input. To this effect, this paper presents a novel method for working with guidelines and a supporting tool environment, namely the Sherlock Guideline Management System. Sherlock provides an integrated environment for articulating and depositing guidelines, accessing past experience and propagating guidelines in the form of recommendations, to the user interface development life-cycle. In this manner, persistency of organisational knowledge on guidelines and evolution of the accumulated wisdom are supported. Moreover, Sherlock provides facilities for the automatic usability inspection of tentative designs. Finally, the paper describes the results of a preliminary evaluation of Sherlock.

Key words: Design support, Usability, Guideline Management Systems

---

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35349-4_22
S. Chatty et al. (eds.), *Engineering for Human-Computer Interaction*

1. INTRODUCTION

The proliferation of information technology products and services in everyday life, has created a compelling need for more usable and friendly user interaction. User-centred design (ISO, 1997) has emerged as a process for matching technological characteristics to user needs away from the technology-centred practices of the past. Usability engineering (Bevan et al., 1994) is at the core of the user-centred design process, focusing on the development of highly usable user interfaces of interactive products and telematic services, providing the necessary methods and techniques and emphasising iteration between the design and evaluation stages.

The adoption of user-centred design and usability engineering gave rise to an increasing need for methods and tools that bridge the gap between design and evaluation by offering practical, comprehensive, and, most of all, cost-effective support, during the early phases of design. Tools for working with guidelines (Cohen et al., 1995) is a methodological approach supporting an iterative development life-cycle and providing early and direct evaluation feedback. This paper reviews currently available systems and identifies their shortcomings; and describes the development and evaluation of a new software tool that overcomes existing problems highlighted by recent practice and experience and takes into account the results of previous research efforts.

For a number of years, the primary medium for propagating guidelines-based input to interactive system development has been paper-based guidelines reference manuals (Lim et al., 1994). However, in the recent past, a number of tools for working with guidelines have emerged to ease the tasks of accessing and retrieving guidelines, applying recommendations to design prototypes and allowing more effective human factors input to early stages of system development. The current generation of tools for working with guidelines exhibits several shortcomings which impede their wider use and adoption by practitioners (e.g., designers or developers of interactive software components).

The lack of adequate tools for supporting design, as well as the reported shortcomings and obstacles of previous research efforts, motivated the development of a new method for working with guidelines and a supporting tool environment, namely the Sherlock Guideline Management System. Sherlock provides an integrated environment for articulating and depositing guidelines, accessing past experience and propagating guidelines / recommendations to the user interface development life-cycle, thus supporting persistency of organisational knowledge on guidelines and evolution of the accumulated wisdom. Moreover, Sherlock provides facilities for the automatic usability inspection of tentative design.

2. COMPUTER-AIDED USER INTERFACE EVALUATION

2.1 Related work

A growing number of systems have addressed the issue of guideline management during design activities. Their underlying assumption is that the prevailing paper-based medium for propagating human factors knowledge (i.e., guidelines) to user interface design is insufficient and ineffective to provide the type and level of support designers require. In response, a number of systems have been developed to provide on-line hypertext access to guideline reference manuals, integrate a subset of relevant guidelines into knowledge bases that could subsequently augment the design phase, or to automate the evaluation of certain components of a user interface according to recommendations resulting from general or context specific guidelines.

Indicative systems which have been developed to pursue this line of work include Reisner's work (Reisner, 1981) on assessing simplicity and consistency of commands represented in a BNF grammar, the work by Blesser and Foley (Blesser et al., 1982), the EXPOSE system (Gorny, 1995), SIERRA (Vanderdonckt, 1995), GuideBook (Ogawa, 1994), HyperSAM (Iannella, 1994).

A more recent development within this line of work is the effort to develop tools for experience-based usability guidelines. This approach extends the scope of tools for working with guidelines to facilitate depositing and retrieval of design experiences and the construction of "living" design repositories. Though such a concept is still in its infancy, there have been some examples demonstrating the basic principles of the approach in selected application domains, such as software engineering (Terveen et al., 1995; Henninger et al., 1995) and accessible user interface design (Stephanidis et al., 1997).

2.2 Shortcomings of the current generation of tools

The current generation of tools for working with guidelines exhibits several shortcomings which impede their wider use and adoption by practitioners (designers or developers of interactive software components). These shortcomings, some of which have been identified in the relevant literature, are briefly discussed below.

Context specificity and guidelines customisation
One well-known shortcoming of the current generation of tools for working with guidelines is their insufficiency to cope with context parameters and customisation issues (Cohen et al., 1995). Specifically, existing tools do not account for context-specific variables that frequently differentiate the implications of a guideline on a particular design, while their support for interpreting and customising the guideline reference manual is limited (if any).

Loose coupling / integration with user interface development systems

Tools for working with guidelines have traditionally not been integrated with popular user interface development systems. This means that the effect of guidelines on a specific design can rarely be automatically articulated (Stephanidis et al., 1997). Instead, the vast majority of the existing tools provide support for hypertext access and retrieval of guidelines, which, though useful, does not provide a sufficient means for augmenting design practices.

Extensibility, maintenance and versioning of guideline knowledge
In currently available tools, guidelines are typically encoded as collections of ergonomic design rules which are subsequently integrated with a user interface management system. These efforts, however, offer no computer-aided support for: (a) maintaining the guideline rule base (e.g., identifying competing guidelines, conflicting recommendations, automatic updates); (b) extending its scope with new rules (e.g., dedicated programming functions for implementing new guidelines; (c) versioning of guideline reference manuals, so as to depict specific requirements of particular design cases.

Design augmentation is beyond guideline access
As already pointed out, tools for working with guidelines can be classified either into systems for access to electronically encoded guidelines, or rule bases that can be integrated with a user interface management environment. In both cases, the level of design augmentation that may be effectively supported is primitive and limited to posterior identification, and sometimes automatic correction of faults in the design, that can be traced through the available rules. There is no way to capitalise on, and reuse past experience, explore alternatives before committing to a particular design option, document problems and design deficiencies, so that they can be referred to by future activities.

Corporate support
Another important shortcoming of existing tools, is their lack for supporting corporate practices. This does not only relate to customising a guidelines reference manual, but also to developing domain-specific styleguides and offering organisation-wide support for appropriating the recommendations of these styleguides. In other words, it is not possible for an organisation to encode a corporate style guide into the representation supported by an existing tool and subsequently provide this representation as an internal company standard to be observed by different business units and development sections. As a result, it is practically impossible to support persistency in the use and application of human factors knowledge.

Reporting
Reporting design defects and alternative solutions is another important issue that needs to be supported by tools for working with guidelines, if they are to provide an effective and efficient medium for integrating human factors knowledge into software design and management. To this end, designers need to be able to effectively document and report the results of their assessments so that they can be communicated to developers, management and other stakeholders.

2.3 Rationale for Sherlock

The above shortcomings provide the motivating rationale for the development of Sherlock, as described in the current paper. Moreover, our prime concern in developing Sherlock has been to provide designers and developers with comprehensive support for iterative prototyping and user interface design. To this effect, we have tried to build upon specific properties of the tools reviewed in section 2.1 and integrate them within one extensible framework for managing guidelines and other interpretable usability heuristics and design principles.

3. THE SHERLOCK GUIDELINE MANAGEMENT SYSTEM

3.1 Overview

Sherlock was implemented as a client/server application. Taking advantage of ActiveX technology (Appleman, 1997), the server and client modules, as well as the extension components, can reside on the same or different computers that are connected through the Internet. The server module runs under Microsoft Windows 95 and the client module is an add-in to the Visual Basic 5.0 Integrated Development Environment (IDE) (Microsoft, 1997).

3.1.1 The Server

The server's role is the inspection of user interfaces according to specific rules and the resulting report of possible rule violations. The rules and the corresponding inspection routines are not embedded in the server module, but they reside in external modules that can be created by any programming language that has the ability of creating ActiveX DLLs. Additionally, the server is also responsible for keeping the clients up-to-date, whenever the rule base is updated, keeping track of guideline violations encountered, as well as for consolidating knowledge about users' solutions to usability problems. Potential users of the server are usability experts and 'programmers' of new rules.

The Sherlock server comprises five basic components (Figure 1):

The User Interface Composer. This component parses a textual user interface description received by a client and converts it to a user interface hierarchical data structure, that will be later used by the inspection routines. The controls and properties recognised by this component can be easily updated and augmented.

The Client / Server Communication Module. This module is built using Window Sockets and is responsible for the communication between the client(s) and the server.

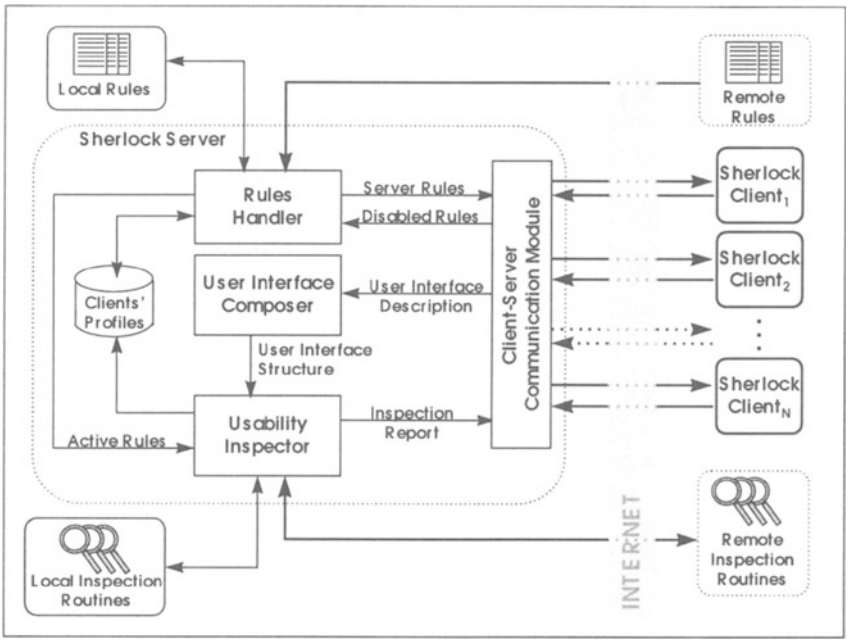

*Figure 1.*Sherlock server architecture

The Clients' Profiles Database. A repository of client-related information, such as identification details, but also rules preferences and evaluation history.

The Rules Handler Module. The *Rules Handler Module* is responsible for handling and integrating rules and inspection routines from a variety of sources. Additionally, this module offers tools for maintaining and extending the rule base and the relevant set of inspection routines.

The Usability Inspector Module. During the evaluation phase of a particular user interface, the *Usability Inspector Module* activates inspection routines based on: (a) which rules are active, and (b) user preferences. This module collects instances of rule violations detected. Upon the end of an evaluation, a comprehensive report of usability problems is compiled and sent to the client.

3.1.2 The Client

The client's main role is to compile a textual description of a user interface created in the Visual Basic 5.0 Integrated Development Environment (VB 5.0 IDE), send it to the server for inspection and then report the inspection results to the user.

Sherlock clients can be used by designers, developers and usability evaluators. The main characteristic of the client's design is simplicity, since Sherlock is intended to be as easy to use as a common spell checker.

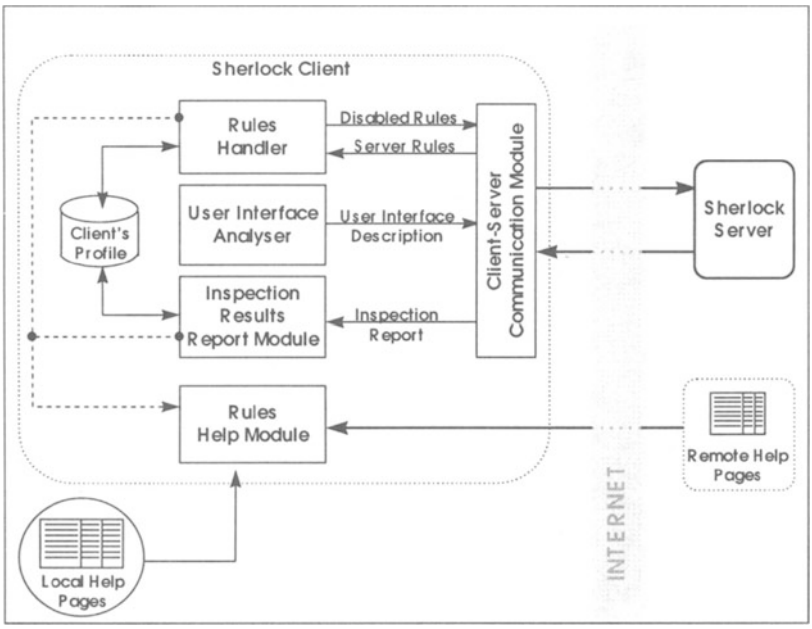

Figure 2. Sherlock Client architecture

The Sherlock client comprises six basic components (Figure 2):

The Client / Server Communication Module. This module is responsible for the communication and exchange of information between the client and the server.

The Client's Profile database. A database used to store information about the rules known to the client, user preferences and information about frequently encountered usability problems and corrections provided by the user.

The Rules Handler. This part of the system is responsible for the visualisation and handling of the rules hierarchy. Rules are presented in tree or list form. Through these visualisations the user has the ability to activate or deactivate rules and classes of rules, as well as to get a short description of each rule or rule class.

The User Interface Analyser. This is a non-interactive module of the client. Its function is to create a textual description of a user interface that was created in the VB 5.0 IDE in a predefined format and send it to the server for evaluation. In order

to minimise the descriptions created, a mechanism for assigning default values is used.

Stages	*Activities*
PREPARATION PHASE	⇨ Identification of relevant design input materials ⇨ Development of the rules ⇨ Integration of the rules into the server
EVALUATION PHASE	⇨ Use of the guidelines server to assess tentative designs and identify design defects ⇨ Presentation of the problems detected ⇨ Problem analysis
PROPAGATION PHASE	⇨ Problem classification ⇨ Problem correction ⇨ Provision of information about the correction of each problem

Figure 3. Stages during iterative prototyping with Sherlock

The Inspection Results Report Module. This module is responsible for presenting the evaluation results. The user can have an overview of the rule violations that were detected by the system, and browse and sort them according to a set of different attributes, such as rule name, severity, class, etc. Furthermore, an in-depth analysis of each rule violation is provided along with background information (such as related theory and examples) and a history of previous solutions to the same problem. Finally, a classification mechanism is provided for separating problems that were fixed, or were not applicable to the specific interface, from those that are still pending.

The Rules Help Module. A customised web browser that presents rule-related information to the user. Each rule can be associated with a web page, or even a whole web site, that contains relevant information, such as, theoretical background, in-depth description, violation and correction examples. The main problem with this type of information is that, since it comes from different sources, it does not have a specific structure or format. The format can be "homogenised" through the use of common web page design guidelines, but there is no way for creating an explicit structure, since the related data is changing dynamically and may be distributed over the Internet. This is why, this module creates 'on the fly' a "*table of contents page*", based upon the *Class* information in the rules' profiles, which presents the underlying (implicit) structure of the information.

3.2 Phased process model

Sherlock follows an iterative prototyping paradigm (Grammenos et al., 1999) and supports a phased process model for assessing tentative designs and propagating the results back to user interface development. The phased approach comprises three main stages which are depicted in the diagram of Figure 3 and which can be initiated from the basic toolbar of the system (see Figure 4). It is important to mention that these stages may be performed more than once, thus leading to design-evaluation cycles and a user-centred perspective to the overall design approach. This tight evaluation feedback loop, which is further discussed in the following sections, ensures that design defects are identified early enough, when their cost of repair is minimal.

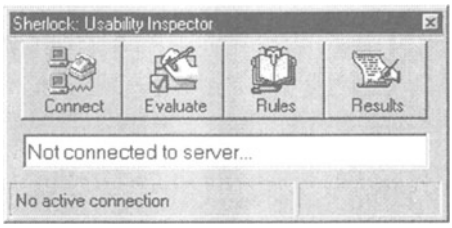

Figure 4. The Sherlock usability inspector

Preparation phase
This is a critical step in the use of Sherlock and an important determinant of the quality of the results. During this phase, two alternatives may be pursued, depending on the availability and sufficiency of the usability knowledge accumulated within the guidelines server. For purposes of simplicity, we assume that the guidelines server contains a sufficient set of guidelines, reflecting the organisation's accumulated wisdom and that the only preparatory activity that is needed is that of selecting the parts within this knowledge component that are relevant to the current design step. The illustration of Figure 5 depicts the dialogue through which the designer may activate or deactivate a relevant subset of the rules. Such rules are classified into clusters of related content and may be represented either through a tree or a list view.

Evaluation Phase
The evaluation phase entails use of the guidelines server to assess tentative designs, identify design defects and support the developers in correcting them. It is important to mention that the same guidelines server may be used by different developers in a project, thus guaranteeing consistency of the designs produced. In order to make use of the evaluation module, a developer should produce a tentative design and should activate the corresponding inspection module of Sherlock (see Figure 4).

An evaluation step entails the assessment of an internal textual description of the object hierarchy against the current version of the guidelines server. The evaluation module per se acts as a passive critic, which collects usability errors and presents

them to the developer. A typical example of the dialogue used to report a usability problem is depicted in Figure 6. The *Inspection Results Report* window comprises three parts. In the upper part, a single problem is presented using all the available information. In the middle of the window, a set of push buttons allows the user to: (a) move to the next/previous problem; (b) access background information about the problem (such as related theory and examples); (c) view a history of previous solutions to the same problem (Figure 7); and (d) classify the current problem as '*Fixed*' or '*Not Applicable*'.

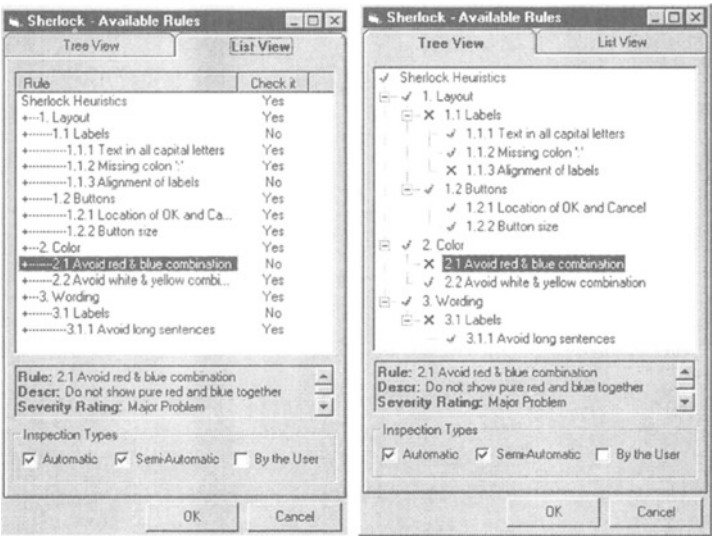

Figure 5. Alternative views for rule handling

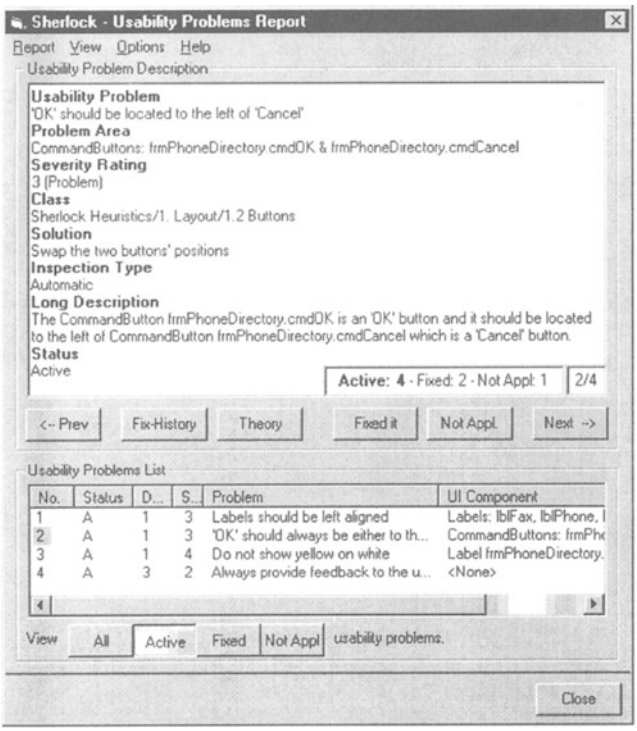

Figure 6. Inspection Results Report

The lower part of the *Inspection Results Report* window is a list (titled *Usability Problems List*) that contains one of the following, depending on the user's choice: (a) all the problems found; (b) the '*Active*' problems; (c) the '*Fixed*' problems; (d) the '*Not Applicable*' problems. Each list entry represents a single problem detected, and contains a short description of the problem, its status, diagnosis type and severity and the user interface component(s) related to the violation. The list can be sorted by any one of these attributes. The user can retrieve more details about a specific usability problem simply by clicking on it.

Propagation Phase

When a rule violation is detected, a usability problem is reported that is automatically classified as '*Active*'. Browsing through the inspection report, the user can explicitly declare a problem '*Fixed*' and optionally provide information on the actual steps taken to tackle the problem (Figure 8); alternatively, the user may decide that the violation reported was '*Not Applicable*' to the specific interface.

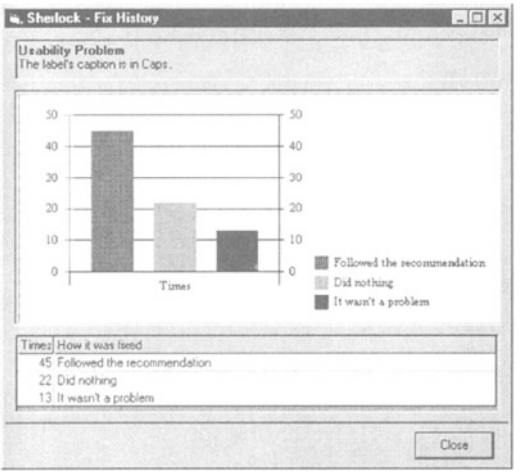

Figure 7. Reviewing deposited information

This classification data is stored in the user's profile, as part of the inspection history. All the classification actions are reversible. The above approach to problem classification was selected in order to assist the developer in organising the task at hand, by providing a quick overview of the problems found, the ones which have been taken care of, and those ignored.

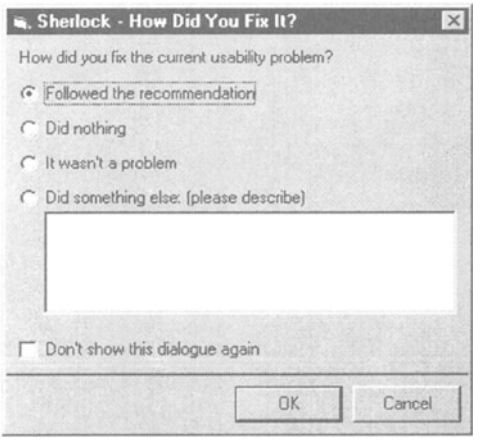

Figure 8. Depositing information about fixing a problem

4. SHERLOCK EVALUATION

Sherlock was evaluated using the thinking aloud method in combination with a widely used questionnaire measuring user satisfaction. The rationale of this decision is as follows: first of all, there are no available heuristics or guidelines, in the available literature, for evaluating a design support system from the perspective of the designer. Additionally, Sherlock is a unique tool, i.e., there are no other existing systems depicting equivalent functionality that can be used for comparative assessments. These two facts have ruled out the use of a non-empirical evaluation method. The existence of a high fidelity prototype made feasible the use of a user-based method. The available options were two: (a) measure user performance, or (b) assess the users' subjective opinion. Given the intention of the current effort (to demonstrate the technical feasibility for such a tool, as opposed to developing a commercial product), the subjective measurement was selected, as this would be more informative of the opinion of designers.

Thinking aloud would be used to observe the user interacting with the system, asking vocalisation of thoughts, opinions and feelings while working with the interface, in order to understand the user's mental model, the way he thinks and performs tasks and find out any mismatches between the mental model and the system model. The questionnaire would be used in order to assess the users' opinion in a more formal way about the perceived usability of the system.

Two questionnaires were used (*IBM Computer Usability Satisfaction Questionnaires* (Lewis, 1995)); the first, namely *After-Scenario Questionnaire* (ASQ), is filled in by each participant at the end of each scenario (so it may be used several times during an evaluation session), while the other one, namely *Computer System Usability Questionnaire* (CSUQ) is filled in at the end of the evaluation (one questionnaire per participant). The result of the subjective evaluation with the IBM Computer Usability Satisfaction Questionnaires is a set of metrics which can be summarised as follows:

- ASQ metric provides an indication of a participant's satisfaction with the system for a given scenario;
- OVERALL metric provides an indication of the overall satisfaction score;
- SYSUSE metric provides an indication of the system's usefulness;
- INFOQUAL metric is the score for information quality;
- INTERQUAL metric is the score for interface quality.

Sherlock was evaluated by eight expert users with substantial experience in user interface design. All users had a University degree in Computer Science or related subject and some of them postgraduate education; all of them had at least a few years experience (typically, four to six years) in the field of human-computer interaction. The user group consisted of five males and three females, whose age ranged from twenty-five to forty years.

Each subject had to execute two scenarios. The first scenario required minimal interaction with the system, and included the evaluation of a very simple interface, as well as the correction (by the user) of the usability problems detected. The second scenario required the interaction of the subject with most parts of the system. The

subject had to configure the rules used for the evaluation, to retrieve background information about the usability problems detected, to classify those problems in the categories supported by Sherlock and finally to record the way he/she had corrected them.

During the execution of each scenario, users were prompted to vocalise their thoughts, pinpoint any problems encountered and express suggestions for improving the tool. After the execution of each scenario, each subject filled in the ASQ questionnaire, while at the end of the evaluation session each subject filled in the CSUQ questionnaire.

The results of the subjective assessment of the users' opinion are summarised in Table 1 and Table 2. The lower the score, the better the quality being assessed; this is a property of the instrument used for the evaluation which follows a 7-point scale (1; strongly agree - 7; strongly disagree). The slight increase in the scores observed between the two scenarios in Table 1, reflects their difference in required user interaction load.

Table 1. After-Scenario Questionnaire (ASQ) Results

	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8
Scenario 1	2,00	3,00	3,33	1,33	2,33	3,00	3,67	2,67
Scenario 2	2,00	3,33	3,67	2,00	3,00	3,00	4,00	3,33

Table 2. Computer System Usability Questionnaire (CSUQ) Results

	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8
SYSUSE	2,00	2,75	2,63	1,38	1,63	2,13	2,63	2,25
INFOQUAL	2,86	4,00	4,43	1,86	3,14	3,57	4,00	3,23
INTERQUAL	2,00	3,00	2,00	2,33	2,00	4,00	3,00	2,67
OVERALL	2,3	3,21	3,16	1,74	2,26	3,00	3,05	2,58

As shown, the overall score, as well as the specific metrics, illustrate a very positive user attitude towards the system. The thinking aloud protocol verified the quantitative assessment and provided valuable insight as to how the prototype could be improved. Some of the recommendations that were collected reflected the requirement for better documentation, on-line help facilities and design examples, so as to help users become accustomed to the system. In addition, users raised the request to provide undo facilities at various steps of the computer-aided assessment.

In general, this preliminary evaluation of Sherlock, by designers and usability experts working in the field, has confirmed the initial hypotheses of an existing real need for such a tool, and in particular, the actual usefulness of Sherlock in supporting the user-centred design process and its potential in contributing to higher quality of human-computer interaction.

5. SUMMARY & FUTURE WORK

This paper has highlighted the importance of providing adequate and timely user-centred support during the process of designing the user interface of interactive applications. In particular, it described a new methodological approach and a tool

(Sherlock) intended to provide a means for improving current HCI design practices and potentially enhancing the quality of the resulting interactive software products and services. The results of the evaluation of Sherlock, have confirmed the initial hypotheses of an existing real need for such a tool, and in particular, the actual usefulness of Sherlock in supporting the user-centred design process as well as its potential in contributing to higher quality of human-computer interaction.

Future work is seeking to extend the current capabilities of the tool to include support for group collaboration and design rationale. This is in line with available evidence suggesting a pressing need for further work in this research direction. In particular, developments under way concern not only enhancements of the present functional characteristics of Sherlock, but also the identification and development of additional means (methods, techniques and tools) to automate different types and levels of support for designers and usability experts.

6. REFERENCES

Appleman D., (1997). Developing ActiveX Components with Visual Basic 5.0 - A Guide to the Perplexed, Emeryville: Ziff-Davis Press.

Bevan, N., Macleod, N., (1994). Usability measurement in Context, *Behaviour and Information Technology*, vol. 13(1&2), pp. 132-145.

Blesser, T., Foley, J., (1982). Towards specifying and evaluating the Human Factors of User-centered Interfaces, Proceedings of ACM Conference on Human Factors in Computing systems (CHI'82), ACM Press, pp. 309-314.

Cohen, A., Crow, D., Dilli, I., Gorny, P., Hoffman, H.-J., Iannella, R., Ogawa, K. Reiterer, H., Ueno, K., Vanderdonckt, J., (1995). Tools for Working With Guidelines, SIGCHI Bulletin 27(2), pp. 30-32.

Gorny, P., (1995). EXPOSE: An HCI-Counselling tool for User Interface Design, INTERACT'95, pp. 297-304.

Grammenos, D., Akoumianakis and D., Stephanidis, C., (1999). Integrated Support for Working with Guidelines : The Sherlock Guideline Management System, accepted for publication in the International Journal of Interacting with Computers, Special Issue: Tools for Working with Guidelines, vol. 11(2), May.

Henninger, S., Heynes, K., Reith, M., (1995). A Framework for Developing Experience-Based Usability Guidelines, Conference Proceedings of DIS'95, University of Michigan, ACM Press, pp. 43-53.

Iannella, R., (1995). HyperSAM: A management tool for large user interface guideline sets, SIGCHI, vol. 27(2), pp. 42-43.

ISO/DIS 13407, (1997). Human-centred design processes for interactive systems, International Organisation for Standardisation, Geneva, Switzerland.

Lewis, R. J., (1995). IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use, International Journal of Human-Computer Interaction, vol. 7(1), pp. 57-78.

Lim, K. Y., Long, J., (1994). The MUSE Method for Usability Engineering, Cambridge University Press.

Microsoft, (1997). Microsoft Visual Basic 5.0 Programmer's Guide, Microsoft Press.

Ogawa, K., Useno, K. (1995). GuideBook: Design Guidelines database for assisting the interface design task, SIGCHI, vol. 27(2), pp. 38-39.

Reisner, P., (1981). Formal grammar and human factors design of an interactive graphics system, IEEE Transactions on Software Engineering, SE-7(2), pp. 229-240.

Stephanidis, C. Akoumianakis, D, (1997). Preference-based Human Factors Knowledge Repository for Designing User Interfaces, International Journal of Human Computer Interaction, vol. 9(3), pp. 283-318.

Terveen, L., G., Selfridge, P., G., Long, M., D., (1995). "Living Design Memory" - Framework, Implementation, Lessons Learned, Human-Computer Interaction, vol. 10(1), pp. 1-37.

Vanderdonckt, J., (1995). Accessing guidelines information with SIERRA, Proceedings of IFIP Conference on Human Computer Interaction (Interact'95), London: Chapman & Hall, pp. 311-316.

7. BIOGRAPHY

Constantine Stephanidis, Ph.D, leads the Assistive Technology and Human-Computer Interaction Laboratory at the Institute of Computer Science, Foundation for Research and Technology - Hellas (ICS-FORTH), Heraklion, Crete, Greece. He is a Visiting Professor at the University of Crete, Department of Computer Science, teaching Human-Computer Interaction. He is the Founding Chairman of the Working Group "User Interfaces for All" of the European Research Consortium on Informatics and Mathematics (ERCIM), and the Founding Chairman of the International Scientific Forum "Towards an Information Society for All".

Demosthenes Akoumianakis is on the research staff at the Assistive Technology and Human-Computer Interaction Laboratory, ICS-FORTH, Greece, with extensive experience in computer-aided user interface design.

Dimitrios Grammenos is on the research staff at the Assistive Technology and Human-Computer Interaction Laboratory, ICS-FORTH, Greece, with expertise in user interface design and usability evaluation.

Discussion

Laurence Nigay: Where does the UI description come from?

Dimitris Grammenos: It is automatically generated by Visual Basic.

Laurence Nigay: What is the control of the UI description? How do you describe the dynamic aspects of the dialogue?

Dimitris Grammenos: SHERLOCK is not a remedy for everything but it can assess many important aspects. Many of the usability problems can be automatically detected by SHERLOCK.

Claus Unger: In your presentation, you give some examples for simple rules. To get an idea of the sophistication and complexity of your system, could you please give an example of a complex rule.

Dimitris Grammenos: The provision of rules is not a major goal of the system. The system mainly serves as an umbrella for embedding user provided evaluation rules. Rules can be expressed in terms of programs and thus can be of arbitrary sophistication and complexity.

Ken Fishkin: Does the system support tools to correct errors rather than just report them.

Dimitris Grammenos: Yes, an earlier version of the system had this feature. In the current version, we had to drop this feature due to time constraints but we plan to re-introduce it in the next version.

Jean Scholtz: Can designers utilise SHERLOCK for partial designs or must the design be complete?

Dimitris Grammenos: Designers can work with it as objects are added to the design. This is the preferred mode for using SHERLOCK.

Jean Scholtz: Can designers input usability problems manually?

Dimitris Grammenos: Problems are kept in the data base and can be managed (or added) through database management tools.

Joelle Coutaz: How is information about context embodied?

Dimitris Grammenos: By customising the rules.

Joelle Coutaz: How do you cope with conflicting rules?

Dimitris Grammenos: There are two cases:

a) Two rules refer to the same guideline. This is identified when the rules are

specified.
b) Each rule has a detailed profile. In the profile is a field labelled "conflict". In this field is a message for the user in the case of conflicting rules. The user can then specify which rule should be used.

Helmut Stiegler: What are the technical prerequisites for your tool? How much effort needs to be invested to adapt it to a new environment? How close are you to a product?

Dimitris Grammenos: It is now based on the Visual Basic environment. Adaptation to a different environment may cost a couple of months of work but the textual description is stored in the server and any environment could use this server. In order to turn SHERLOCK into a product, more rules need to be developed. Right now we have integrated a small set of rules for demonstration purposes.