An Organizational Learning Method for Applying Usability Guidelines and Patterns

Scott Henninger

Department of Computer Science & Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588-0115 USA
+1 402 472 8394
scotth@cse.unl.edu

Abstract. As usability knowledge and techniques continues to grow, there is an increasing need to provide tools that disseminate the accumulated wisdom of the field. Usability guidelines are one technique that is used to convey usability knowledge. Another is the emerging discipline of usability patterns. This paper presents an approach that combines these techniques in a case-based architecture and utilizes a process to help an organization capture, adapt, and refine usability resources from project experiences. The approach utilizes a rule-based tool to represent the circumstances under which a given usability resource is applicable. Characteristics of the application under development are captured and used to match usability resources to the project where they can be used to drive the design process. Design reviews are used to capture feedback and ensure that the repository remains a vital knowledge source for producing useful and usable software systems.

1 Introduction and Motivation

As the body of knowledge on the design of interactive software systems becomes mature, the need for disseminating the accumulated wisdom of the field becomes increasingly important to the design of useful and usable software systems. Design for usability is becoming increasingly important to the success of software systems, but software developers are usually poorly trained in human factors, ergonomics, or usability issues. One solution is to always require a human factors specialist on development teams, but this is often impractical as such specialists continue to be in short supply and budgets do not always allow such specialized personnel. Education and iterative development processes aimed at evaluating and improving the user interfaces are necessary solutions to this problem, but techniques are needed that provide software developers with proactive knowledge and techniques for developing high quality user interfaces.

Usability guidelines have been around in various forms for some time, and have had some impact on design practices for user interface software. Yet the full potential of guidelines have yet to be realized [1, 2]. To date, work in these areas have failed to adequately address concerns facing software designers, developers, and managers, focusing on comprehensive usability issues at the expense of determining which

M. Reed Little and L. Nigay (Eds.): EHCI 2001, LNCS 2254, pp. 141–155, 2001.

guidelines should be used under what circumstances. In addition, usability guidelines often become a static document read only by human factors specialists and used to assess an application's conformance to usability standards. Guideline analyzers, which analyze completed interfaces against guidelines or other usability metrics [3, 4], can assess completed systems, but do little to support the development process.

These methods apply usability knowledge as an assessment, which is often too late in the development process. In one example witnessed by the author, an application was submitted to a human factors group in a large IT department that had a screen with 39 seemingly unordered buttons arranged in an array [5], a poor interface that would cause users to engage in lengthy searches to find desired features. This organization had a well-designed on-line style guide [6], but usability approval was so late in the development process that there was inadequate time and resources to fix the problem before it was shipped. The less mature work on usability patterns take a more proactive view of the design process, but add little to the usability guidelines perspective beyond a different format for documenting the pattern and some concerns for establishing the context of a pattern.

Instead of being relegated to a discretionary reference role and/or an after-the-fact human factors certification process, the knowledge contained in usability resources needs to be delivered as an integral part of the entire development process. Guidelines and patterns can be helpful resources for the developer, but tools for finding applicable resources are lacking. Current approaches are document-based, at best supported with hypertext tools, which relies on individual developers to know of the existence of the resources and understand when they should be applied. Given the potentially copious usability guidelines and patterns, and the lack of training in usability issues, this is not a satisfactory solution.

Tools are needed to turn guidelines and patterns into proactive development resources that can be applied throughout the development process. In this paper, a methodology is presented that represents the context of a given guideline or pattern in the form of applicability rules that formally specify the conditions under which a usability resource is appropriate. We present an exploratory prototype, named GUIDE (Guidelines for Usability through Interface Development Experiences), that we have been using to investigate and demonstrate how this methodology can be used to deliver usability resources to software developers when they are needed. The focus of this work is not the creation or discovery of good guidelines or patterns, but the creation of tools that capture and disseminate knowledge of user interface design principles and experiences at the right time – during the development process. In addition, our organizational learning approach [7] allows the incremental capture of the characteristics of the context of use so the applicability rules and guidelines can evolve as new requirements are encountered, new techniques are used, and new designs are created.

2 Usability Guidelines and Patterns

Usability guidelines have become a widely recognized method of bringing the cumulative knowledge of usability issues to bear on the software development process. It is generally accepted that guidelines cannot replace the "golden rules" of

interface design - user involvement, user feedback from early prototypes, and iterative development [8]. But guidelines can play a role in improving the quality of the iterative steps, leading to an improvement in quality and reduction (but not elimination) of the number of iterations involved in the design-evaluate-redesign cycle of HCI development.

Guidelines have evolved to take on a number of forms. *Style guides* address how different kinds of windows should look and interact with the user for tasks such as choosing from lists [9-13]. Style guides tend to be platform-specific and focus on interface widgets, such as dialogue boxes, pull-down menus, screen layouts, and naming conventions. Other questions, such as when a particular widget should be used or how the interface elements integrate together, are left unanswered.

While style guides are usually platform-specific, universally applicable *interface guidelines* have also been explored to provide higher-level guidelines on various aspects of human-computer interfaces [14-18]. These guidelines dispense general advice, such as "Allow the user to control the dialogue," "Provide displayed feedback for all user actions during data entry," or "Reduce the user's memory load." At some level, this is sound advice, but this kind of information lacks important contextual information that would allow designers and developers to assess how and when to apply the guidelines to a specific set of circumstances or system requirements.

Usability standards also take on the character of guidelines, opting to specify general principles rather than mandating specific techniques, widgets, or tools. International standards have been created [19], and standards have been used within organizations to ensure a degree of consistency across applications [6, 20, 21]. *Domain-specific guidelines* have also been created, the most prominent being guidelines for designing Web pages [22, 23].

All of these efforts have largely focused on the content and structure of the guidelines themselves. On-line versions of guidelines have been created, but have used simple hypertext-based search systems for accessing the guidelines [24-28]. But these systems have done little to address the problems demonstrated in studies using guidelines, such as the time to find relevant guidelines, problems with interpreting guidelines for the task at hand, and generally being too abstract to directly apply [29-31].

While usability guideline have become voluminous to the point that it is difficult to determine which principles are applicable to a given design problem. And the continuing proliferation of technology only exacerbates the problem. Little thought has been given to defining when guidelines are applicable or how guidelines can be refined to meet user task requirements for a specific set of users and a specific type of application. In addition, little research has been done to accumulate knowledge about interface design in a form that can capture relationships between specific contexts and applicable guidelines.

2.1 Context and Usability Patterns

A usability patterns community, inspired by the recent work on software patterns, has begun to explore how patterns can be used to provide an intermediate perspective between universally applicable usability guidelines and component-specific style guides [32-35]. The essential idea of a design pattern is to capture recurring problems

along with the context and forces that operate on the problem to yield a general solution. Collections of patterns can be organized in a network of higher-level patterns that are resolved or refined by more detailed patterns, resulting in a *pattern language* [36].

Usability patterns explicitly represent context, although approaches vary from a one-sentence description of the design goals [37] to viewing context as the explicit focus of patterns, telling "the designer *when*, *how* and *why* a solution can be applied." [32]. Usability patterns represent context through text fields such as context and forces that respectively describe how the problem arises and other issues that may impact the outcome. For example, the "Shield" pattern [32] describes the problem of protecting users from accidental selection of a function with irreversible effects. The context states that users need protection against undesired or unsafe system actions, and that the pattern should not be used for easily reversible actions. The forces include severity of the unintended actions and the user's need to work quickly while avoiding mistakes. Patterns can also represent context through a network of linked patterns, the pattern language, from high-level issues to low-level choices, although examples of these networks and tools for traversing the links are currently lacking.

2.2 Integrating Usability Guidelines and Patterns

Differences between usability guidelines and usability patterns lie primarily in perspective and representation of the information. In fact, many of the proposed patterns replicate much of the information contained in existing guidelines. The major difference is that pattern languages are intended to be used as a design method. The pattern community is therefore concerned with using patterns to communicate between designers and customers or users [38, 39], a perspective not often seen in guidelines.

Because of this main difference, the perspective of usability patterns tends to be more problem-oriented, focusing on describing a problem and solution, than the more general information or advice perspective of guidelines. Although templates and data structures for describing guidelines and patterns can easily be reconciled, the fields commonly seen in patterns are indicative of the problem-oriented perspective. In addition to the problem-solution (or title-solution) format seen in most guidelines, patterns add fields to describe the context of the problem and the forces that shape the problem and its variants.

The goals of both these approaches are essentially the same: to document and manage experience about usability design issues in a format that is easily disseminated and understood. But much of the work in these fields focus on the development of patterns and guidelines (a notable exception is the recent Tools for Working With Guidelines workshop series [40]). Creating and disseminating this knowledge is important, particularly where empirical validation is present, but little work has been done on creating the computational framework that will supply this information in an effective manner. Our focus is different in that we begin from the perspective of creating resources and tools for software developers. Instead of teaching developers specific usability principles (the proverbial fish), we aim to provide the tools that allow development organizations to create and disseminate

usability resources in a manner that helps developers design and develop usable applications (the proverbial teaching them how to fish).

3 An Organizational Learning Approach for Usability Guidelines and Patterns

Current research and practice for both guidelines and patterns primarily rely on an educational or information retrieval model. It is entirely up to the designer to either know that usability resource (collectively we will refer to patterns and guidelines as "resources") exists or at least know enough about the repository to realize they should search for applicable resources. One must have an overall understanding of existing guidelines and patterns to recognize when a given resource should be applied. Given the lack of formal usability training, the potential size of a comprehensive pattern or guideline repository, and the frailty of human memory, this assumption does not always hold.

In past research, we have created tools to support organizational memory for usability guidelines. The objective of organizational memory tools is to provide information relevant to organizational practices that "you can't learn in school" [41], such as local terminology, organization and project-specific conventions, lessons learned, policies and guidelines, individuals with expertise, and many others. The Mimir Guidelines system illustrated how organizational memory techniques could be used to collect and disseminate usability guidelines [31]. Project experiences were captured using case-based decision support technology [42], where cases were attached to guidelines as examples of how the guideline had been applied. Dissemination of guidelines was supported through hypertext and searching techniques where users matched project characteristics to appropriate guidelines.

Creating a repository of project experiences, an organizational memory, is valuable in and of itself. But the overarching objective is to learn and improve on past performance. Emphasis must be placed on establishing a continuous improvement process that enhances product quality and developer productivity, while recognizing past experiences as a catalyst for the learning process. We call this an organizational learning approach [7] to emphasize that the knowledge is used as the *basis* for improvement, not just memorizing past experiences [43-45].

3.1 The GUIDE Process for Applying Usability Resources

Our approach to supporting organizational learning to usability resources is a combination of tool and process to capture knowledge as it emerges in practice, review and/or otherwise validate that knowledge, and ensure that previous knowledge and known best practices are applied where they exist. The GUIDE (Guidelines for Usability through Interface Development Experiences) methodology supports an organizational learning approach to developing context-specific usability resources through the process shown in Figure 1. A key component of the methodology is a hierarchical structure of usability guidelines delivered in the GUIDE tool shown in

Figure 2. We have chosen to seed our repository with a Web-enhanced Smith and Mosier 944 guidelines corpus [16], although any set of initial guidelines would work equally well.[1] A rule-based system is then used to match project characteristics (user populations, tasks, GUI tools, etc.) to specific usability resources that project personnel should apply during development. The result is a set of project activities that are assigned to the project. For example, if the system is being accessed by users over the Internet and involves access to sensitive data, then guidelines and/or patterns for login interfaces and access to sensitive data will be given to the project as an activity to be considered.

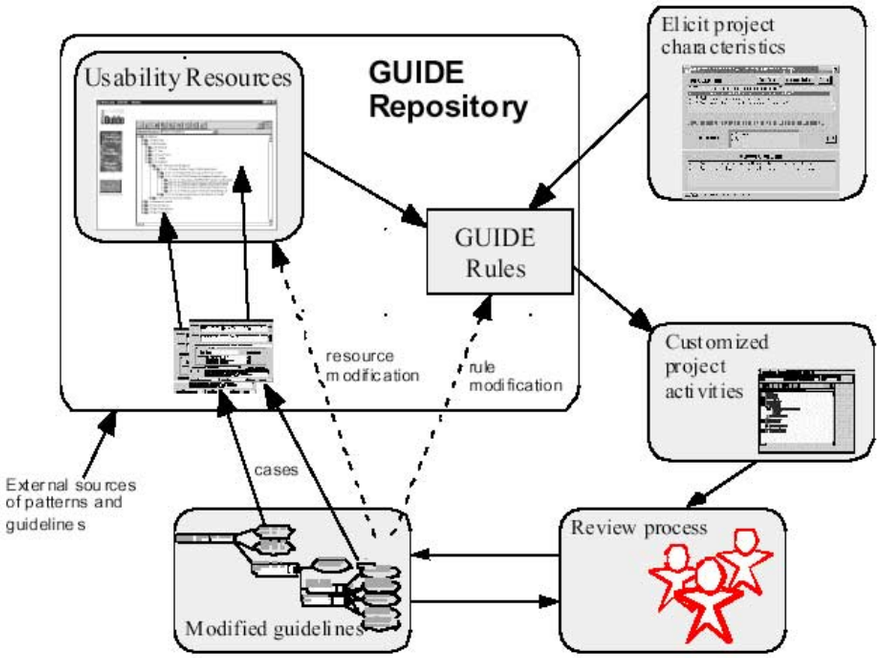


Fig. 1. Using and modifying usability resources.

The next step of the process in Figure 1 is the most critical element of the organizational learning approach. A review process is used to inspect how project personnel answered the options posed by GUIDE and discuss whether the assigned resources are appropriate for the project in question. This review creates an important feedback loop that is used to learn emergent user interface needs. If there is a mismatch between project needs and the resources assigned by GUIDE, reviewers can recommend that either a different option is chosen (options are described below) or that the knowledge in the repository needs to be updated to meet the needs of this project. The latter of these two options creates an opportunity for learning. As shown

[1] We are currently investigating the possibility of basing our structure on patterns, adopting one of the usability pattern languages currently under development and augmenting with guidelines from various sources.

in Figure 1, not only are modified project guidelines created, but the conditions for this modification can optionally be fed into the repository, essentially "blazing a trail" for subsequent projects.

Fig. 2. Guide interface and a usability case.

For example, suppose a project is the first to have a requirement for both cross-platform and cross-browser Java delivery. A project performs some studies and determines that using the Java Plug-in is the best choice in this instance. While this decision can later be augmented (for example, another project's users may be using 28k modems and could deem downloading the rather sizable Plug-in infeasible), extended, and eventually replaced by subsequent efforts or outside changes in technology, it represents a form of intellectual capital that has considerable research and effort behind it that the organization probably does not want to replicate unnecessarily.

3.2 The GUIDE Architecture

GUIDE is an exploratory prototype that has been used to investigate and demonstrate how usability guidelines can be integrated into the software development process. GUIDE borrows from the case-based architecture of the BORE (Building an Organizational Repository of Experiences) project [7], but focuses exclusively on usability issues. Although initial efforts have explored the different issues arising in software engineering processes and usability separately, GUIDE is currently being re-designed to become part of BORE. As with BORE, GUIDE is a Web-based

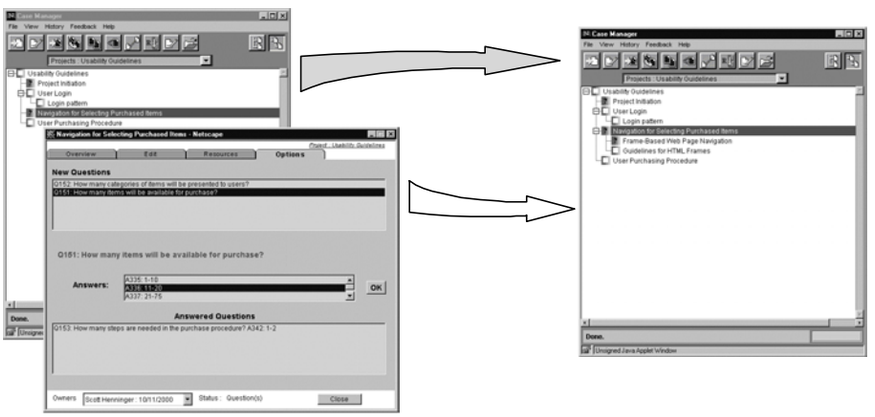
application, using HTML and Java for the user interface, Java for processing, and a database back-end to store information.

Representation of usability resources in GUIDE embodies the intersection of three closely related technologies. Patterns and guidelines, as discussed above, are related by common goals and similar formats. Case-based technology, which uses a problem-solution structure similar to patterns and guidelines, adds an instance-of relationship (cases are context-specific instances of usability resources, which can be patterns or guidelines).

Figure 2 shows a hierarchical view of usability resources in GUIDE and a window showing a specific resource on frame-based Web page navigation. The fields shown on the resource window (the window on the right in Figure 2) include the canonical fields found in usability patterns work, although other formats can easily be integrated into GUIDE's architecture. The system architecture is flexible enough to accommodate many of the different fields suggested in usability guideline and pattern research.

3.3 Using GUIDE to Develop Interactive Software Systems

GUIDE uses a case-based structure to associate usability resources to project activities (see the Case Manager windows in Figure 3a and 3b) that document a specific project's use of the resources. At the start of a software development effort, a GUIDE project is created. This will create a number of project initiation cases, some of which will have usability options associated with them, delimited by a '?' inside the icon in the project hierarchy. Clicking on the "Options" tab of the case displays one or more questions about the characteristics or requirements of the project (bottom window, Figure 3a). Selecting a question displays possible answers in the Answers box. Selecting an answer will trigger applicability rules (described in the next section) for resources that are assigned to the project to inform developers of usability principles that need to be followed.



3a **3b**

Fig. 3. Documenting Contextual factors through project requirements.

For example, the left window in Figure 3a displays a project named "Usability Guidelines." This project has a few initial project cases associated with it that are used to identify tools, techniques, and usability issues that developers should be considering during design. Previous questions have determined that the project involves designing an e-storefront that allows multiple items to be purchased during the same session. This causes a number of cases to be assigned to the project, such as "Navigation for Selecting Purchase Items," that represents recommended project activities. The user has selected the Options tab for that case (bottom window of Figure 3a), revealing a series of questions to disclose further project requirements. One question has already been answered, leading to new questions that explore further project requirements.

3.4 Representing Context with Applicability Rules

Usability resources in GUIDE are assigned to a specific project through applicability rules that match project characteristics to appropriate resources. This is accomplished by a forward-chaining inference engine using production rules with an if-then structure. Preconditions are defined as question-answer pairs. Each time a question is answered, the inference engine checks the database to see if any of the rules are satisfied. When this occurs, a set of actions are fired. Actions can cause a variety of events, including placing questions in or taking questions out of the New Question stack, assignment of system variables, and attaching usability resources to the project.

For example, selecting the answer "11-20" in Figure 3a will fire a rule that will place cases in the project that points to frame-based navigation resources, shown as child cases under the "Navigation for Selecting Purchase Items," shown in the selected case in Figure 3b. In essence, the rule base is stating that using frame-based navigation is recommended when the purchasing procedure takes between 1 and 2 steps and there are between 11 and 20 items available for purchase. Rationale for this recommendation is provided in the resource, which states that frames can be used as a solution to the problem of needing to keep a context while accessing multiple pages[2]. Note that these rules encode the context of the resources, a major features of pattern languages [38].

It should also be noted that GUIDE is designed so that questions can be associated with any project case, allowing development teams to incrementally disclose usability issues when they are ready, instead of having to answer all questions at the beginning of a project. In Figure 3b, the "Frame-Based Web Page Navigation" case has further options associated with it, allowing further decomposition to more detailed guidelines or patterns.

Figure 4 depicts a partial decision tree of the kind that can be represented by our forward-chaining inference engine. Through these rules, which are developed by usability professionals (see the following section), the GUIDE system is placing the accumulated wisdom of usability issues at the fingertips of software developers in the context in which they are applicable.

---

[2] This example is intended to demonstrate our approach, not advocate any specific usability principles.

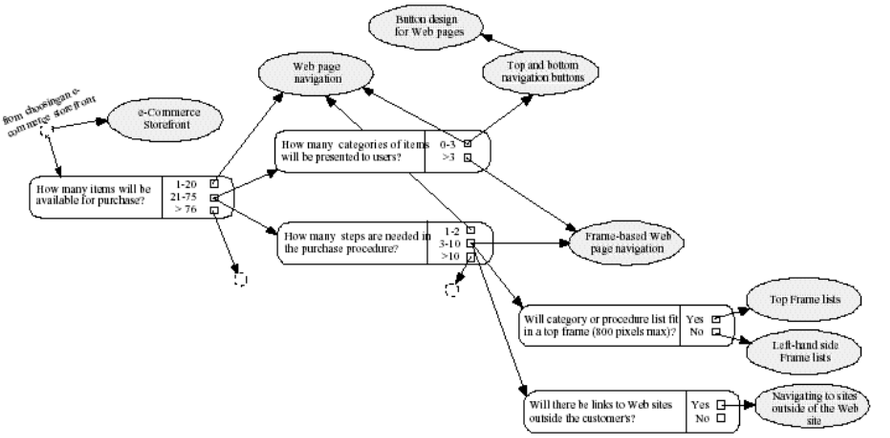


Fig. 4. A partial GUIDE decision tree.

3.5 Incremental Acquisition of Design Knowledge

Applicability rules in GUIDE are meant to provide a more proactive alternative to the "build a repository and let them search" philosophy that current approaches to usability guidelines and patterns employ. Rules are not meant to automate the design and development of interfaces, but to provide a match to resources that can inform developers of usability issues. They are intended to act as a medium for discussion and debate. Indeed, we view rules as a means to formally state, evolve, and improve the current understanding of usability guidelines and the conditions under which they can provide helpful information to software developers. An important aspect of this philosophy are tools and processes to modify rules to meet the dynamic needs of interactive software systems, as described by process depicted in Figure 1.

The overall objective and philosophy of this research is not to derive universally applicable rules or usability resources. Rather, the aim is to provide the rules and infrastructure that allow an organization or group to accumulate knowledge based on the collective experiences of the organization. This is accomplished by instituting a process that reviews the recommendations given by GUIDE during project design reviews. Suggestions on modifications and improvements to the knowledge base are then forwarded to human factors and GUIDE librarian personnel for consideration. This kind of process has been demonstrated to work in practice [43], provided the repository remains up to date and remains an important corporate asset that provides benefit to the developers.

During periodic design reviews, teams will review and critique project answers to GUIDE questions. Different answers could be negotiated and found to be more appropriate for the project and can easily be changed in GUIDE (the system supports rule backtracking). Review teams could also determine that GUIDE recommendations are either inappropriate or missing. This is seen as an opportunity to improve the knowledge base. The review team provides rationale for why

deviations are necessary. Human factors specialists and/or other GUIDE curators review the requests and either refuses the request, allows a once-only deviation, or turns the rationale into GUIDE rules and modifies the repository with this new knowledge. New guidelines, examples, and other information could also be created and placed in the repository to document the project's experiences.

For example, suppose that previous projects have noted that users have problems keeping track of where they are in complex procedures and clicking on links to external Web pages that end up in the viewing part of the frame (the second problem is a common HTML frame issue, especially when framed pages are displayed inside of frames).[3] The review team identify that frames are needed in this project and want to use the review as an opportunity to document and apply the lessons learned from previous efforts. The external link problem could be handled in a number of ways, including adding sections in the "Guidelines for HTML Frames" document (see Figure 3b) or adding a new guideline and rule stating that if frames are used the new guideline on external page links needs to be followed. The tracking complex procedures issue could be addressed by creating a new guideline stating, for example, that completion of steps in a procedure are tracked by changing the color or otherwise highlighting completed steps. Then an action is added to the rule used in Figure 3 so that any project having 3 to 10 steps in the procedure is also given this new guideline.

This process ensures that the repository will evolve to meet the changing needs of the organization while foraging new paths for subsequent projects to follow. As the repository grows it will accommodate greater portions of projects, minimizing the number of deviations while increasing knowledge reuse.

The rule base can be as specific or vague as needed by the organization. For example, an organization that services only the medical community will probably have guidelines that are specific to medical terms and procedures, while an organization servicing a broader customer base will probably want guidelines at a higher level of abstraction. The level of detail is determined by the rules and the amount of effort desired by the organization, not by any limitation or mandate placed in GUIDE itself.

4 Conclusions and Future Research

The objective of this research is not an attempt to automate user interface design. To the contrary, it is recognized that effective user interface design take a degree of talent and careful work with the end users that cannot be captured through rules, patterns or any information system. Nonetheless, there is recognized knowledge and conventions that can help some designers reach higher levels of competency and help accomplished designers extend their knowledge to areas they have not yet experienced. This research is an exploration of how resources can be delivered to software developers through a rule-based structure that provides the basis for an organizational memory – capturing the collective intelligence of an organization with respect to usability design issues. Rules in this context serve as a medium, a formal

[3] Empirical studies may have been conducted confirming this, and would be linked to the specific guidelines or patterns addressing these issues.

mechanism for communicating design knowledge and establishing relationships between context and usability resources.

Rule-based systems are often criticized for their inflexibility, which we ameliorate through a process that reviews the relationships established by the rules throughout the development process to ensure that real project experiences are represented. Our integration of rule-based and case-based systems comes closer to the spirit of American Case Law, where statutory law (rules) are contextualized by case law (cases, guidelines, patterns). People then use this structure to argue which cases come closest to the current situation and apply the attached rule, or set new precedents if none of the cases are applicable.

The result is a web of knowledge on usability issues that is continuously updated to meet the evolving needs of the organization. As the repository grows, it will become an important piece of intellectual capital that puts knowledge of proven usability techniques and wisdom at the fingertips of software developers. This approach does not replace the need for iterative software development methodologies and user studies (although knowledge of how to conduct those processes should be contained in the repository), but can reduce the number of iterations by assuring that certain classes of errors are avoided.

This approach requires that some software development staff devote some percentage of their time maintaining the knowledge base. Personnel knowledgeable about usability issues and the structure and content of the usability guidelines and patterns are necessary for this approach to work. In addition, rule-based expertise is necessary to ensure that GUIDE rules are well-structured and operate properly. Such a structure resembles the concept of software factories [46], aligns well with current trends to involve human factors in the design process [6], and can easily be applied to mid-size or small development firms that can only afford to staff usability consultants on a part-time basis.

The intersection of guidelines and patterns needs further investigation. We are currently exploring structures that better integrate the different kinds of knowledge contained in patterns, guidelines, style guides, etc. We are also interested in providing examples as a significant knowledge resource. Given the number and diverse composition and content of existing resources, particularly usability guidelines, finding a proper “seed” [47] for the repository is problematic. A study revealed that there was minimal overlap between 21 different guideline corpuses [48], further underscoring the need for supporting a diverse initial set of usability resources.

Continued research is needed to further understand the issues of using organizational memory repositories as advocated here. Many empirical questions remain, such as whether variance between projects is too great to apply past experiences, whether past experiences stifle creativity or enables it by shifting attention away from re-creating previous solutions, whether the approach is useful for only certain types of organizations, and whether the documentation burden of constructing rues is too great for practical application. The contribution of this work thus far is to provide tool support to turn usability guidelines and patterns into a proactive design tool and design organizational structure and process to capture and disseminate project experiences on usability issues.

The GUIDE and BORE projects will be evaluated through use in the Software Design Studio in the JD Edwards Design Studio at the University of Nebraska-Lincoln. This program, which integrates a combination of business and computer science subjects, has been built around a design studio concept [49] where students

are engaged in long-term projects from paying customers external to the University. BORE will be used to deliver and manage the studio's defined software development process. GUIDE will be employed to deliver usability resources as part of scenario-based and other design methodologies. This and other research efforts will further refine the system while further populating the repository with usability knowledge and experiences. We will also seek to apply the tool and technique to pilot studies in industry, as an effort to further study the issues involved with employing an experience-based methodology to the usability design process.

Acknowledgments

I gratefully acknowledge the efforts a number of graduate students that have helped develop BORE, including Charisse Lu, Kurt Baumgarten, and Peter Hsu. Osama Al Shara has also contributed. This research was funded by the National Science Foundation (CCR-9502461, CCR-9988540, and ITR/SEL-0085788), and through contracts with Union Pacific Railroad.

References

1. J. D. Gould, S. J. Boies, and C. H. Lewis, "Making Usable, Useful, Productivity-Enhancing Computer Applications," *Communications of the ACM*, vol. 34, 1991, pp. 75-85.
2. F. de Souza and N. Bevan, "The Use of Guidelines in Menu Interface Design: Evaluation of a Draft Standard," *Human-Computer Interaction - INTERACT '90*, 1990, pp. 435-440.
3. M. Y. Ivory, R. R. Sinha, and M. A. Hearst, "Empirically Validated Web Page Design Metrics," *Proc. Human Factors in Computing Systems (CHI 2001)*, Seattle, WA, 2001, pp. 53-60.
4. D. Scapin, C. Leulier, J. Vanderdonckt, C. Mariage, C. Bastien, C. Farenc, P. Palanque, and R. Bastide, "A Framework for Organizing Web Usability Guidelines," *6th Conference on Human Factors and the Web*, Austin, TX, 2000.
5. S. Henninger, "A Methodology and Tools for Applying Context-Specific Usability Guidelines to Interface Design," *Interacting With Computers*, vol. 12, 2000, pp. 225-243.
6. P. A. Billingsley, "Starting from Scratch: Building a Usability Program at Union Pacific Railroad," *interactions*, vol. 2, 1995, pp. 27-30.
7. S. Henninger, "Case-Based Knowledge Management Tools for Software Development," *Journal of Automated Software Engineering*, vol. 4, 1997, pp. 319-340.
8. J. D. Gould and C. H. Lewis, "Designing for Usability - Key Principles and What Designers Think," *Communications of the ACM*, vol. 28, 1985, pp. 300-311.
9. Apple Computer Inc., *Macintosh Human Interface Guidelines*. Reading, MA: Addison-Wesley, 1992.
10. Microsoft Corporation, *The Windows Interface: An Application Design Guide*. Redmond, WA: Microsoft Press, 1992.
11. IBM, *Object-Oriented Interface Design: IBM Common User Access Guidelines*. Carmel, IN: Que, 1992.
12. OSF, *OSF/Motif Style Guide: Revision 1.2*. Englewood Cliffs, NJ: Prentice Hall, 1993.
13. Sun Microsystems, *Open Look Graphical User Interface Application Style Guidelines*. Reading, MA: Addison-Wesley, 1989.
14. C. M. Brown, *Human-Computer Interface Design Guidelines*. New Jersey: Ablex, 1988.
15. P. Heckel, *The Elements of Friendly Software Design*. San Francisco: Sybex, 1991.

16. S. L. Smith and J. N. Mosier, "Guidelines for Designing User Interface Software," Technical Report, The MITRE Corporation ESD-TR-86-278, 1986.
17. J. Vanderdonckt, "Towards a Corpus of Validated Web Design Guidelines," *Proceedings of the 4th ERCIM Workshop on 'User Interfaces for All'*, 1998, pp. 16-31.
18. B. Shneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interaction, 2nd ed*. Reading, MA: Addison-Wesley, 1992.
19. ISO/WD 9241, "Ergonomic Requirements for Office Work with visual Displays Units," International Standard Organization 1992.
20. E. Rosenweig, "A Common Look and Feel or a Fantasy?," *interactions*, vol. 3, 1996, pp. 21-26.
21. S. Weinschenk and S. C. Yeo, *Guidelines for Enterprise-Wide GUI Design*: Wiley & Sons, 1995.
22. P. J. Lynch and S. Horton, *Web Style Guide : Basic Design Principles for Creating Web Sites*. Princeton, NJ: Yale Univ Press, 1999.
23. J. A. Borges, I. Morales, and N. J. Rodriacuteguez, "Guidelines for Designing Usable World Wide Web Pages," *Proc. Human Factors in Computing Systems (CHI '96) Short Papers*, 1996, pp. 277 - 278.
24. D. Grammenos, D. Akoumianakis, and C. Stephanidis, "Integrated support for working with guidelines: the Sherlock guideline management system," *Interacting with Computers*, vol. 12, 2000, pp. 281-311.
25. J. Vanderdonckt, "Accessing Guidelines Information with SIERRA," *Proceedings Fifth IFIP International Conference on Human-Computer Interaction INTERACT '95*, Lillehammer, 1995, pp. pp. 311-316.
26. R. Iannella, "HyperSAM: A Practical User Interface Guidelines Management System," *Proceedings of the Second Annual CHISIG (Queensland) Symposium - QCHI '94*, Bond Univ., Australia, 1994.
27. L. Alben, J. Faris, and H. Saddler, "*Making it Macintosh:* Designing the Message When the Message is Design," *interactions*, vol. 1, 1994, pp. 10-20.
28. G. Perlman, "Asynchronous Design/Evaluation Methods for Hypertext Development," *Hypertext '89 Proceedings*, 1989, pp. 61-68.
29. L. Tetzlaff and D. R. Schwartz, "The Use of Guidelines in Interface Design," *Proc. Human Factors in Computing Systems (CHI '91)*, 1991, pp. 329-333.
30. H. Thovtrup and J. Nielsen, "Assessing the usability of a user interface standard," *Proc. Human Factors in Computing Systems (CHI '91)*, New Orleans, LA, 1991, pp. 335-341.
31. S. Henninger, K. Haynes, and M. W. Reith, "A Framework for Developing Experience-Based Usability Guidelines," *Proc. Designing Interactive Systems (DIS '95)*, Ann Arbor MI, 1995, pp. 43-53.
32. M. van Welie, G. van der Veer, and A. Eliens, "Patterns as Tools for User Interface Design," *Workshop on Tools for Working With Guidelines*, Biarritz, France, 2000.
33. M. J. Mahemoff and L. J. Johnston, "Principles for a Usability-oriented Pattern Language," *Proc. Australian Computer Human Interaction Conference OZCJI 98*, Adelaide, 1998, pp. 132-139.
34. G. Casaday, "Notes on a Pattern Language for Interactive Usability," *Proc. Human Factors in Computing Systems (CHI '97)*, Atlanta, GA, 1997, pp. 289-290.
35. J. Borchers, "CHI Meets PLoP: An Interaction Patterns Workshop," *SIGCHI Bulletin*, vol. 32, 2000, pp. 9-12.
36. C. Alexander, *The Timeless Way of Building*. New York: Oxford Univ. Press, 1979.
37. A. Granlund and D. Lafreniere, "UPA 99 Workshop Report: A Pattern-Supported Approach to the UI Design Process,", 1999.
38. T. Erickson, "*Lingua Francas* for Design: Sacred Places and Pattern Languages," *Proc. Designing Interactive Systems (DIS 2000)*, New York, 2000, pp. 357-368.
39. J. Tidwell, "The Gang of Four are Guilty,"., 1999.
40. TFWWG, *Tools for Working With Guidelines Workshop (TFWWG2000)*, Biarritz, France, 2000.

41. L. G. Terveen, P. G. Selfridge, and M. D. Long, "From 'Folklore' To 'Living Design Memory'," *Proceedings InterCHI '93*, Amsterdam, 1993, pp. 15-22.
42. J. L. Kolodner, "Improving Human Decision Making through Case-Based Decision Aiding," *AI Magazine*, vol. 12, 1991, pp. 52-68.
43. L. G. Terveen, P. G. Selfridge, and M. D. Long, "Living Design Memory' - Framework, Implementation, Lessons Learned," *Human-Computer Interaction*, vol. 10, 1995, pp. 1-37.
44. J. P. Walsh and G. R. Ungson, "Organizational Memory," *Academy of Management Review*, vol. 16, 1991, pp. 57-91.
45. E. W. Stein and V. Zwass, "Actualizing Organizational Memory with Information Systems," *Information Systems Research*, vol. 6, 1995, pp. 85-117.
46. V. R. Basili, G. Caldiera, and G. Cantone, "A Reference Architecture for the Component Factory," *ACM Transactions on Software Engineering and Methodology*, vol. 1, 1992, pp. 53-80.
47. G. Fischer, R. McCall, J. Ostwald, B. Reeves, and F. Shipman, "Seeding, Evolutionary Growth and Reseeding: Supporting the Incremental Development of Design Environments," *Proc. Human Factors in Computing Systems (CHI '94)*, Boston, MA, 1994, pp. 292-298.
48. J. Ratner, E. M. Grose, and C. Forsythe, "Characterization and Assessment of HTML Style Guides," *Proc. Human Factors in Computing Systems (CHI '96)*, 1996, pp. 115-116.
49. D. A. Schön, *The Design Studio: An Exploration of its Traditions and Potentials*. London: RIBA Publications Limited, 1985.

Discussion

P. Smith: Your approach is to have a set of guidelines and then examples. Commercial systems use the reverse.
S. Henninger: Pattern work uses this approach, generalise and then examples. There is not too much difference. The end goal is the same. Communicate a design principle to get a better interface.