University of California, Irvine

RESEARCH PROPOSAL
''CAREER'' PROGRAM (CCR-0501)

Improving the Design of Interactive Software

David F. Redmiles

Attention: Caroline Wardle, CAREER Contact, Division of Computer & Computation Research (CCR-0501), National Science Foundation, Arlington, VA 22230

Submitted by: Department of Information and Computer Science, University of California, Irvine, California 92717

Principal Investigator:

Dr. David F. Redmiles, Assistant Professor, Department of Information and Computer Science;
phone: (714) 824-3823; e-mail: redmiles@ics.uci.edu; fax: (714) 824-4056

Planned Period of Support: July 1, 1996 - June 30, 2000

List of Suggested Reviewers

Dr. Leon Osterweil
Computer Science Department
University of Massachusetts
Amherst, MA 01003-4610
phone: (413) 545-2186
email: ljo@cs.umass.edu
area: software process, software environments

Dr. William Curtis
Software Engineering Institute
c/o Carnegie Mellon University
Pittsburgh, Pa 15213
email: curtis@sei.cmu.edu
area: software process, cognitive issues
in software design

Dr. Judith Olson (Professor)
University of Michigan
2084 Delaware
Ann Arbor, MI 48103
phone: (313) 747-4606
email: judy_olson@ub.cc.umich.edu
area: interactive systems,
computer-supported cooperative work

Dr. Walter Scacchi (Associate Professor)
Center for Operations Management & Research
School of Business
University of Southern California
Los Angeles, CA 90089
phone: (213) 740-4782
email: wscacchi@alnitak.usc.edu
area: collaboration technology,
software engineering

Dr. Herb Krasner
Krasner Consulting
1901 Ringtail Ridge
Austin, TX 78746
phone: (512) 328-4264
email: hkrasner@cs.utexas.edu
area: cognitive issues in software design

Dr. Peter G. Selfridge (Member Technical Staff)
AT&T Bell Laboratories
Room 2B-425
600 Mountain Avenue
Murray Hill, NJ 07974
phone: (908) 582-6801; fax: (908) 582-7550
e-mail: pgs@research.att.com
area: knowledge-based systems, interactive systems

A. Project Summary

Objectives. For many years, the poor design of interactive systems has been attributed to insufficient communication between developers and end users [Gould, Lewis 85; Fischer 89]. The problem is compound. Usability practices from the field of human-computer interaction are not integrated into the theory and practice software engineering. Therefore many software developers are not aware of what constitutes good communication between them, end-users, and other stakeholders. Conversely, software developers who attempt to learn and employ usability practices are often faced with high learning curves and expensive-to-implement methods [Olson, Moran 95]. This proposal has three objectives: 1) to develop a model of software development as a process of on-going communication; 2) to support this model through active mechanisms in software tools; and 3) improve the accessibility and acceptance of usability methods by software practitioners. In general, the objectives reflect a process and philosophy of *human-centered software development.*

Approach. The challenge is to successfully integrate the results from research in human-computer interaction with software engineering. The approach taken by this proposal, is that such integration can only be successful if it is actively supported by the tools software developers use day-to-day. Specifically, tool support enables improved software processes to be more readily adopted. Tool support will also reinforce education developers receive on usability methods. A specific kind of active support for communication will be presented, namely, a mechanism called *expectation agents*. Expectation agents will be implemented as software mechanisms that monitor the usage of prototype systems and detect mismatches between actual usage and developers' design assumptions, or, expectations. Such breakdowns are opportunities for communication which make explicit the expectations of both developers and end users. Thus, expectation agents will serve as a test vehicle for supporting on-going communication in software development, for adapting usability practices to software practice, and for reinforcing developers' understanding of human-computer interaction.

Impact on Software Engineering Tools and Theory. Results are expected along three dimensions: theoretical, technical, and empirical. In theory, the research develops a model of on-going communication in design, refines models of the software processes involved in developing interactive systems, and improves the understanding of the role of usability methods in workplace settings. In the technical dimension, the research provides new algorithms and representations for capturing and communicating design intent and active mechanisms such as expectation agents to operate over design intent. These mechanisms involve both formal and semi-formal representations. The empirical contribution includes a critical analysis of the approach as a means of gathering usability information and, in general, enabling the essential elements of communication in systems development.

Impact on Software Engineering Curriculum. The heightened concern for the usability of interactive software has let to efforts to evolve university curricula in human-computer interaction [Strong 94]. Although the required topics might be adequately addressed in speciality programs, curricula aimed at producing human-computer interaction practitioners would in fact overlook the students who need the information the most—software practitioners! This proposal describes three ways software engineering curricula may be positively impacted. First and foremost, the proposed work seeks to augment the software engineering curricula in ways that enable software engineers to accept and appreciate methods of human-computer interaction. Second, the culture of software engineering is such that practice is accompanied by tools and methods. Hence the proposal seeks to to adapt, and where possible, automate methods. Third, the proposal will explore the notion that software development environments which support the automation of usability methods can also support practitioners in sharing new knowledge about methods.

B. Table of Contents

C. Project Description/Career Development Plan

C.1 Results from Prior NSF Support

Between 1992 and 1993, the proposer (working as a PostDoctoral Fellow) was funded in part under an NSF research grant to G. Fischer and R. McCall. His contribution to this grant is documented below. He has not yet been funded directly as a principal investigator by NSF.

C.1.1 National Science Foundation (IRI-9015441)

G. Fischer and R. McCall: ‘‘Supporting Collaborative Design with Integrated Knowledge-Based Design Environments,’’ 1990-1993, amount: $ 700,000. The goal of this project was the development of a conceptual framework and a demonstration system for collaboration among members of design teams when direct communication among these members was impossible, impractical, or undesirable. The work extended Fischer’s theoretical framework for domain-oriented design environments and, in general, knowledge-based systems, in the direction of collaboration technology. Retrieval of information through pro-active agents was one aspect which was explored. D. Redmiles’ contribution included results on people’s comprehension of design knowledge—when information is made available to designers, comprehension is an essential step towards its application.

For this work, publications contributed by D. Redmiles include:

- ‘‘Improving the Explanatory Power of Examples by a Multiple Perspectives Representation’’ (with C. Rathke), Proceedings of the 1994 East-West Conference on Computer Technologies in Education (EW-ED’94) (Crimea, Ukraine), International Centre for Scientific and Technical Information, Moscow, Russia, September 1994 (best paper award).
- ‘‘Observations On Using Empirical Studies in Developing a Knowledge-Based Software Engineering Tool,’’ Proceedings of the 8th Annual Knowledge-Based Software Engineering (KBSE-93) Conference (Chicago, IL), IEEE Computer Society Press, Los Alamitos, CA, September 1993, pp. 170-177.
- ‘‘Reducing the Variability of Programmers’ Performance Through Explained Examples,’’ Human Factors in Computing Systems, INTERCHI’93 Conference Proceedings (Amsterdam, The Netherlands), ACM, 1993, pp. 67-73.
- ‘‘Supporting Software Designers with Integrated, Domain-Oriented Design Environments’’ (with G. Fischer, A. Girgensohn, and K. Nakakoji), IEEE Transactions on Software Engineering, Special Issue on Knowledge Representation and Reasoning in Software Engineering, Vol. 18, No. 6, 1992, pp. 511-522.

C.2 Problem, Objectives, and Significance

Support for determining the requirements of interactive systems—determining what functionality is needed to meet users’ needs—remains an open problem [Davis, Hsia 94]. Systems continue to fail with regards to mismatches between the tool that is developed and the situation in which the tool is used. The problem comes as little surprise to researchers in human-computer interaction. Real work situations are complex to the degree that a-priori planning for them breaks down [Suchman 87]. Researchers in software engineering have responded by creating the specialty field of ‘‘requirements engineering’’ [IEEE Software 94]. Researchers in human-computer interaction have responded with the area of ‘‘usability engineering’’ (e.g., see [Nielsen 93] and [Lewis, Rieman 93]).

This proposal seeks to integrate the approaches of software engineering and human-computer interaction into a perspective of *human-centered software development.* The focus is on interactive software products, the growing majority of applications. The goal is to reconceptualize design of interactive software as a process of communication: communication between developers, end users, other

stakeholders, and on-line information stores. Human-centered software development will simultaneously encompass a theory of evolutionary software design, a prototype software design environment, and a curriculum for educating software developers.

The immediate significance of the proposal is that it will refine the theory of communication and information exchange in software design and it will demonstrate which aspects may be supported in prototype software development environments. The longer term impact is that usability methods will be made more accessible to software developers being trained to produce interactive software. This latter impact will be achieved both through the software design environments and through curriculum development.

C.3 Relationship to the ‘‘State-of-the-Art’’

The analysis below examines the motivation for the perspective that breakdowns in usability are attributable to breakdowns in communication; first, with respect to design in general, and, second, specifically with respect to software design. The analysis continues with an examination of attempts to support communication through active mechanisms such as *agents* and *critics*. Finally, proposals for curriculum revision are reviewed.

C.3.1 Communication among Stakeholders in Software Development

The software development process is permeated by communication [Fischer 90]. Customers contact developers to articulate needs. Developers discuss design (requirements, specification, implementations, etc.) among themselves and, in some cases, with customers. At issue is not *if* communication takes place, but rather, *what* is the nature of communication in good design? In particular, how can software developers and end users arrive at a mutual understanding of an interactive system; how can they make their intentions explicit?

Predating the age of the ‘‘software crisis,’’ researchers in design methodology observed that the design process was, in fact, a process of communication and sought to develop a formal discipline to enhance discussion of design [Kunz, Rittel 70; McCall 79]. Their discipline conceived of design discussions as ‘‘argumentation.’’ This approach has been used by the Olsons and their colleagues to analyze the content of, and measure the process of, design teams [Olson et al. 91; Herbsleb, Kuwana 93], including software design. Potts and colleagues have recently performed similar analyses based on extensions to the argumentation approaches [Potts et al. 94]. Finally, Curtis and colleagues analyzed breakdowns across several software engineering projects [Curtis, Krasner, Iscoe 88]. Many of the problems they identified had simply to do with basic breakdowns in communication.

Communication is inherent in the Scandinavian approach to design, known as participatory design [Greenbaum, Kyng 91]. Moreover, this approach assumes that design involves all ‘‘stakeholders.’’ Specifically, design of new systems involves users of the technology working with developers. This is in contrast to the predominant U.S. paradigms of software development in which managers or other company officials contracting the software replace users or in which end users are approximated by market surveys.

In a participatory design approach, or in interdisciplinary teams in general, team members must develop a common language of communication. One technique is to focus communication around a design mock-up (see, e.g., [Ehn, Kyng 91]). Either a mock-up or an executable prototype has the additional advantage of providing an ‘‘object to think with’’ [Papert 86]; it provides designers with a concrete object to reflect upon and react to [Schoen 83]. A prototype may be incrementally modified to reflect a growing understanding. It serves as a means of drawing out tacit knowledge [Polanyi 66]. On the part of developers, tacit knowledge includes assumptions about a system solution to end users’ problems. On the part of end users, tacit knowledge includes assumptions about their tasks, work environment, and needs, as they are described to developers. Intentions of developers and end users become explicit incrementally. Potts and colleagues arrive at similar conclusions [Potts, Takahashi, Anton 94].

In short, software design is a process of communication in which many stakeholders need to participate. Design discussions will include consideration of many issues, design alternatives, and arguments for and against alternatives. However, the discussion should not take place in a vacuum; but, in the context of an object to think with, an evolving prototype which can draw out tacit assumptions designers make about a system and tacit assumptions users maintain about their work. The process makes explicit the intentions of the developers and end users for a system.

C.3.2 Supporting Communication with Agents and Critics

Previous attempts to support ‘‘conversations of design,’’ including software design, have led to systems that rely on heavily structured communication models [Winograd, Flores 86; Conklin, Begeman 87]. These approaches have fallen short in practice. A general analysis of many such systems indicates that the formality imposed upon the users was the defeating factor [Shipman, Marshall 94; Terveen, Selfridge, Long 93]. Communication needs to facilitated, not constrained. In particular, opportunities for communication need to be integrated into the normal working environments.

Agents have been proposed as one means of facilitating the communication of information both more naturally and with less need for formal structure. Malone employed agents to draw users’ attention to pieces of electronic mail which were of interest to them [Malone, Lai, Fry 92] (see also [Fischer, Stevens 91]) and for supporting cooperation among people [Malone et al. 88]. By advertising relationships between items of information, agents can support communication and collaboration 1) among members of a development team, 2) between developers and end users, and 3) among all participants and on-line information stores.

Other work on agents has focussed on anthropomorphic aspects [Kay 92]. Laural defines interface agents as ‘‘characters, enacted by the computer, who act on behalf of users in a virtual (computer-based) environment [Laurel 90].’’ Responsiveness, competence, accessibility, and the capacity to perform actions on our behalf are defined as essential characteristics of agents. Laurel’s agents act as presenters of stories from a specific, predefined perspective. The anthropomorphic dimension of agents may or may not facilitate users’ interaction. What is more interesting is the concept of multiple perspectives. In particular, because the design of interactive software should involve stakeholders with many different backgrounds, information must be communicated in a perspective most relevant to the individual’s work [Redmiles 92].

Maes has incorporated the notion of user modeling into agents [Maes 94]. Her research focuses on agents which observe and adapt to users’ behavior and preferences. In general, when more is understood about tasks which software users are working on, a better starting point for active mechanisms such as agents can be provided. With respect to software design, an understanding of the overall design process and more specific tasks may be represented [Osterweil 87; Young, Taylor 94].

This same criticism applies to the different critiquing schemes developed within the context of domain-oriented design environments (e.g., [Fischer 87; Fischer et al. 91]). Critics are able to analyze a work product created by a designer and suggest some of its shortcomings. Thus, they successfully demonstrate the potential of active mechanisms to deliver information to designers. However, these critics are not linked to an overall model of a design process and hence fall short of their full potential.

Agents and critics have successfully demonstrated the ability of active mechanisms to support communication and collaboration [Bobrow 91]. The effectiveness of these active mechanisms is often improved by combining them with specific knowledge about a domain of design. A direction for further improvement lies in a greater understanding of the process of design of interactive software and in making that process a basis for the behavior of agents and critics.

C.3.3 Educating Designers of Interactive Software

Gould and Lewis reiterate the centrality of communication for avoiding usability breakdowns [Gould, Lewis 85; Gould, Boies, Lewis 91]. Their usability design process requires constant communication between users and developers. Usability techniques provide guidance as to what information should be considered in developing interactive systems—what needs to be communicated. However, despite the known benefits, usability techniques are not well integrated into software development practice:

- they are not well understood and simply not generally accepted by software developers—a cultural barrier;
- they provide mixed results because of variation in individuals—a methodological barrier;
- they are expensive to use, requiring experimenters' time, subjects time, and, sometimes, special settings—a cost barrier; and
- they require contact between developers and end users (rarely the case because of organization)—an opportunity barrier.

The last two barriers for applying usability methods, cost and opportunity, can be alleviated in part by facilities for communication, such as the agents and critics mechanisms. However, the cultural and methodological problems need to be addressed by educational innovation.

The heightened concern for the usability of interactive software has let to efforts to evolve curricula for universities [Strong 94]. However, education about usability techniques and, in general, human-computer interaction issues, is a costly undertaking [Olson, Moran 95; John, Packer 95]. Many methods require background much time to apply and practice as well as require students to have backgrounds in cognitive psychology. Although the required topics might be adequately addressed in speciality programs, curricula aimed at producing human-computer interaction practitioners would in fact overlook the students who need the information the most—software practitioners! To date, attempts to reach these individuals with methods of discount usability remain controversial [Nielsen 93].

First and foremost, software engineering curricula must be augmented in ways that enable software engineers to accept and appreciate methods of human-computer interaction. Second, the culture of software engineering is such that practice is accompanied by tools and methods. Thus further work is needed to adapt, and where possible, automate methods. Third, the next section explores the notion that software development environments which support the automation of usability methods may also support practitioners in sharing new knowledge about methods.

C.4 General Plan of Work

Theoretically, the problem of engineering usable, interactive software is analyzed as a problem of communicating design intent between stakeholders. The focus of the implementation is on how active mechanisms such as agents can support this communication, especially between members of a development team and end users. Usability techniques help provide information that will contribute to a representation of design intent, helping developers converge on a system that meets the needs of its end users.

Based on the discussion from the previous section, goals in three major areas are addressed:

- challenge in software development—understanding the processes specific to developing interactive systems, in particular communication and feedback relevant to usability;
- challenge for providing active support for communication—developing mechanisms for representation that enable agents, critics, and other kinds of active mehcanisms to locate and present information relevant to a design issue; and
- challenge for educating software practitioners—integrating usability techniques into the software

development process and, more broadly, adapting them to the collection of data from workplace settings.

C.4.1 Software Development Process

The challenge in software engineering is to establish a process that provides for constant communication between developers and end users from the start of a project. The content of that communication is information relevant to the usability of a system. The process must be evolutionary supporting iterative refinement of requirements and prototype.

Figure C-1 illustrates the components and participants in such an evolutionary software development process. An evolving artifact begins as a concept and is cooperatively developed into a working prototype. Stakeholders representing many perspectives participate: end users, managers, software developers, and others. Problem domains reflect diverse perspectives of the stakeholders and contribute to a complete understanding of the problem. Solution domains come from previous projects and provide a framework for the new project. Feedback comes from users, developers and other stakeholders reflecting on the evolving artifact.

The focus of the feedback and corresponding communication is expected to change depending on the state to which an artifact has evolved. Figure C-2 outlines the expected communication between developers and end users based on methods of human-computer interaction. Namely, it follows an iterative model of development in which an artifact is refined based on feedback from users' and developers' communications. It highlights what information usability techniques might contribute to the refinement of different aspects of an evolving system. Further, it identifies what kinds of information the active support from agents or other mechanism might supply.

These figures outline starting points for the proposed research. They are grounded in real projects which attempted to focus on user-centered task analysis, evolving prototypes, user participation, and constant communication between users, developers, and other stakeholders [Girgensohn, Redmiles, Shipman 94]. They are also grounded in the results reported in the literature as discussed in the earlier theoretical analysis of the problem.

Figure C-1: The Evolutionary Process of Human-Centered Software Development

Software Phase	Communication Focus	Current Manual Approach	Potential Active Support
Problem Definition	conceptual framework	design meetings, interviews	retrieving previous design examples
Requirements	conceptual framework, proposed system	site visits for protocols and interviews with end user	retrieving previous design examples
Design (Mock-ups and Prototypes)	system, conceptual framework	site visits, cognitive analysis	collecting & reporting protocols (usage scenarios), usage measures (statistical), end user comments, automated cognitive analysis
Implementation & Maintenance	system		usage measures (statistical), protocols (usage scenarios), end user comments

Figure C-2: Usability Techniques and Communication Focus

The table focuses on the need for communicating information relevant to designing usable systems. The ''phases'' are not isolated steps in a process, but indicate the shifting of focus as a design evolves; in this case, from concept, to prototype, to implementation and on-going maintenance.

C.4.2 Active Support for Communication

Section C.3.2 presented an argument for actively supporting communication during development. General capabilities of active mechanisms such as agents and critics were presented. To help clarify the kind of support that is possible with respect to usability of interactive software, the following section focuses on one kind of active support, called an *expectation agent.* This kind of agent records design assumptions developers make about the expected use and usability of a system, monitor actual usage in prototypes, and communicate failed expectations to developers.

Figure C-3: Bridget Service Provisioning Application

Figure C-4: Dialog Box of an Expectation Agent

Scenario in the Domain of Service Provisioning. The development of a new system at the NYNEX Corporation, called Bridget (see Figure C-3), serves as a basis for illustrating how expectation agents could be used. Bridget is specifically an application for providing new telephone service to business customers but is representative of many form-based interactive systems at NYNEX. The context for the examples is that system developers have already implemented a prototype version of Bridget (such as the version in Figure C-3). The prototype is the result of establishing an initial set of requirements through discussions with end users (service representatives) and their managers, as well as an analysis of observing existing tasks the end users perform.

First, agents could monitor the trial use of this prototype and support communication between developers and end users with the goal of refining the prototype and the requirements it implements. For instance, in developing the initial requirements and prototype, the developers made an assumption that the customer representatives would initiate the address validation query by clicking on the ‘‘Validate Address’’ button (at the top of the screen in Figure C-3) when they finished filling out that section and before proceeding to the next section. Doing so gives time for the query to be evaluated before the additional configuration information is needed.

This assumption would be expressed by the developers in an agent. If the customer representative opened another section (e.g., ‘‘Listing’’) before starting address validation, a discrepancy between expected and actual use would occur; the agent would be triggered. As a result, the representative would be presented with rationale and the possibility to respond, shown in Figure C-4. Responses would be recorded in a form of requirements document (not illustrated) where they could be brought to the attention of developers.

A second use for an expectation agent in Bridget could be to recognize unexpected order of subtasks, implying a need for a new ordering of the sections in the form. In particular, while customer representatives may fill out sections in any order, they can be more efficient if the sections are placed in the normal usage order. An agent can keep statistics about the order followed by many representatives. Such usage patterns will be reported back to developers. Patterns found can then be used to engage users in discussions about whether a different order would be better or whether some fields should be put in different sections.

Third, a user-initiated mechanism, such as a ‘‘suggestion’’ button, could be used to support the volunteering of comments and suggestions by Bridget users. Transcending traditional electronic mail, the suggestion button could combine the user’s suggestion together with the user’s current context, such as recent actions, to be sent to the developers. This information can improve the developer’s understanding of the user’s situation and suggestion.

```
events
    C: field_change() &&
         in_section("Customer");
    V: is_command("Validate_Address");
    L: field_change() &&
       ! in_section("Customer");
    D: is_command("Complete_Order");
global
    D: done;
start
    C: customer;
customer
    V: validated;
    L: unvalidated;
validated
    C: customer;
unvalidated
    action {
      user_dialog("Address ...");
      exit();
    }
done
    action { exit(); }
```

Figure C-6: Representing Expectation Agents in EARL

In sum, this type of agent initiates a dialog between users and developers in which the developers communicate their intent to the users and the users have the opportunity to respond. Additional follow-up discussions can take place outside of the agent-based software development environment.

Figure C-5: State Transition Diagram for Example Agent

Representation of Agents Supporting Communication. An important component of an Expectation Agent is the specification of the end user behavior that leads to its invocation. State transition networks are well-suited for describing interaction histories because they provide convenient means for describing sequences of behavior in a step-by-step fashion. Each network state represents some set of system states that can be considered identical from the point of view of the Expectation Agent. Transitions map the user actions that change an agent's state.

EARL is a language for representing Expectation Agents. Unlike the state transition networks used in systems for pattern matching, user interface specification, and dialog descriptions (e.g., [Wasserman et al. 86]), state transitions in EARL are described by predicates. This provides an additional layer of abstraction that can be used for application-domain specific support. To simplify the specification of expectations, only events causing a state change or side-effects (such as the logging of a user action) that are of interest to the Expectation Agent need to be specified. All other events leave the network in the same state. Hence, many events will not result in a state change in the Expectation Agent network because a single Expectation Agent state represents many system states. This behavior is also different from normal state transition networks, which fail when an unspecified event occurs.

Figure C-5 shows the transition network for describing the Expectation Agent from the Bridget system. This simple agent can be written in EARL as shown in Figure C-6. The description of an agent in EARL starts with the definition of event classes in the ‘‘events’’ section. Event classes are not required for defining agents and are included to increase the readability of the Expectation Agent specification and to provide shortcuts for expressing predicates. Predicates specific to Bridget enable the developers to specify the agents in terms of form interactions.

To keep the specification simple, there are global transitions (indicated by the keyword ‘‘global’’ in Figure C-6) for transitions that apply to every state. Such transitions are usually used to represent termination conditions. In the example, when the end user issues the command for a completed order, the agent terminates. Since the design decision was made to leave an agent in the same state when an unspecified events occurs, the event that terminates the agent needs to be explicitly indicated by a termination transition.

The remainder of the specification of an agent is a list of named states with transitions to other states. A transition is a boolean condition or an event class that is separated from the target state by a colon. States can also have actions (indicated by the keyword ‘‘action’’) that are executed when the state is reached. Such actions include starting dialogs between the end user and developer (see Figure C-4), collecting data, getting feedback from users, reporting back to the developer, or terminating an agent. Just like the predicates used in the transitions, actions can be added to extend the Expectation Agent system to new application domains.

Representation of Design Intent. The basic challenge is to represent many kinds of design knowledge in a fashion that enables agents to locate and present to developers and end users, information relevant to a design issue. As stressed throughout this proposal, developing interactive systems requires communication among all stakeholders in order to create the collaborative requirements, specifications and design of the system. This communication can take various forms as the development process progresses. Initially this communication may be in the form of text, pictures, video, etc. as the shared understanding of the domain and the problems are understood. Notes from meetings, existing documents, results from studies, development contracts, are all part of the growing body of design knowledge referred to as design intent. Individual pieces of information are referred to as *snippets*. As development advances and prototypes are created, this communication and collection of design snippets may be assisted by the expectation agents. The information space grows and the structure of that information becomes more important and difficult to manage.

The approach is to create a hypermedia space with an underlying associative representation supporting the appropriate links between related snippets [Anderson, Taylor, Whitehead 94; Shipman 93]. In the early stages of a project’s life cycle, many links will have to be created manually. However further work with heuristics by Shipman may support semi-automatic connections. Snippets collected by agents will have their initial destination defined by the agent. In particular, in the later stages of a project, snippets will be associated with aspects of the evolving prototype system. Previous work on associative representations of explanation provide a foundation for this approach [Rathke, Redmiles 94; Redmiles 92; Reeves, Shipman 92a; Mannes, Kintsch 91], including the attribute-value representation employed in HOS [Shipman 93]. These approaches indicate the need for supporting different perspectives of design corresponding to different stakeholders [Stahl 93] and use situations [Rathke, Redmiles 94].

Cost and Benefits. Any ‘‘new’’ approach to a problem faces the challenge of whether it is cheaper or more cost-effective, in some sense, than an existing approach. Approaches requiring people to work within a framework to structure knowledge for design, to any degree, have an especially poor history in this regard. In defense of the analysis and approach outlined in this proposal, it is noted that usability of interactive systems does not come for free. Thus, judging the cost effectiveness of this approach requires a comparison against methods where usability has been incorporated already as a serious goal. From this perspective, the approach of the research creates a positive benefits-for-cost balance by

- reducing of the costs of making observations—a) enabling observations without designers always traveling to users' sites; and b) allowing more observations to be made; and
- making the input of expectations and other rationale more natural and less difficult.

The first idea is straightforward and is discussed below as one criteria for evaluation. The second issue involves additional work on the definition and management of agents.

Evaluating the Effectiveness of Agent-Based Approaches. The most obvious evaluation strategy would be to compare the development of a small interactive system under an agent supported process and under a manual process. Usability of the resulting interface could be judged according to common criteria, such as those given in [Nielsen 93]. However, this approach is, unfortunately, prohibitively costly within the scope of this proposal. It would require several teams working according to each method. Only one such experiment has been reported to date (see Curtis' survey [Curtis 91]).

A compromise is to assess the agent-supported process against current software practice and usability theory. Assuming that one or two development projects can be performed with the agent-based approach, an assessment could be made as follows. First, the semi-automated process could be compared against the Bridget prototype experience and current software practice with respect to the following criteria:

- the ability to detect breakdowns in the field with users performing tasks—assessing whether the process succeeds in supporting convergence to a usable prototype;
- the quality of communication—whether the agent-mediated communications contributed more or less to the design of a prototype compared to face-to-face communications and observations;
- cost savings—whether fewer site visits were required after prototypes began to emerge;
- the quality of the usability technique—whether effective data was collected, users and developers were comfortable with the collection method, and whether more data was available than in the manual collection process.

Second, the system produced by each development effort may be assessed against usability criteria:

- learnability; and
- accuracy of task performance—whether many breakdowns persist in the field, an indication of failure of convergence between designer and end users' intent; etc.

C.4.3 Curriculum

Section C.3.3 concluded with three challenges to the state-of-the-art in education:

- challenge to methods and presentation,
- challenge of tool integration, and
- challenge of tool support for learning.

The work proposed here will address these challenges both in content of curriculum as well as in the mechanism for learning.

New methods for the curriculum. As noted earlier, Nielsen has advocated the adaptation of evaluative methods from human-computer interaction to methods that are less costly for practitioners to apply. This work remains controversial with the primary objection being the validity of the conclusions users of the new methods will draw. Some evaluation of methods themselves has been attempted (see e.g., [John, Packer 95]). However, much more is required to tune known techniques to software practice. The principal investigator has had success in introducing one method, the cognitive walkthrough [Wharton et al. 94] into a graduate seminar in software engineering and subsequently into an undergraduate course and industry tutorial. He seeks to continue this general approach with other methods as mentioned in the discussion of Figure C-2.

New presentation/interpretation of methods. Incorporating human-computer interaction theories and techniques into software engineering curriculum can in some cases be achieved by a fresh presentation or reinterpretation of the material in terms and experiences software engineers can relate to. A simple example is to explain ‘‘usability engineering’’ as improvements to ‘‘requirements engineering.’’

A more serious undertaking in this regard is to leverage off of the software engineering theory of ‘‘process.’’ Process theory is about describing the anticipated steps software developers should follow in carrying out a specific project [Osterweil 87]. Under this proposal, the investigator intends to broaden the understanding of software processes to accommodate research in cognitive process of design. The investigator has begun this approach by mapping the cognitive theories of opportunistic design [Guindon, Krasner, Curtis 87], reflection-in-action [Schoen 83], and problem solving [Kintsch, Greeno 85; Pennington 87], onto process models [Robbins, Redmiles 95].

Integration as an electronic design studio. In addition to the recent work reviewed earlier, two seminal steps toward curriculum revision were made over a decade ago by Herbert Simon and Donald Schoen [Simon 81; Schoen 83]. Their discussions about engineering of artificial systems parallel identically issues which arise in software engineering. In previous work, the principal investigator collaborated in building elements of these theories of design into domain-oriented design environments [Fischer 92; Fischer et al. 92]. These environments focused on evolutionary design in which designers received feedback on partial design from critics and could review previous solutions through an explanation tool [Redmiles 93a]. Because of the origin of Schoen’s work in architecture, many think about design environments as on-line or electronic design studios. The image is pertinent to the extensions in the work proposed here. Namely, the extension of critics, explanation, and other active mechanisms to support communication among developers, end users, and other participants in a development project.

Agents to reinforce the curricula. Simply educating software engineers about usability methods is insufficient if these methods are not part of their routine work habits. However, changes in software processes are avoided by companies as well as individuals: changes require expensive retraining and changes are perceived as risky. The approach espoused in this proposal is that the resource for applying human-computer interaction methods should come from the software environment in which practitioners work. For example, the expectation agents may be augmented with explanation resources. They may also be augmented to detect and suggest their usefulness in certain evaluative situations. Again, this approach is routed in previous work done on software critics [Fischer et al. 93; Robbins, Redmiles 95].

C.5 Prior Research and Educational Experiences

C.5.1 Software Engineering

For several years, the principal investigator has researched issues of software reuse and software engineering from the perspective of the human participants. Specifically, software environments were developed and evaluated in terms of the support they provided software developers in the design process [Redmiles 90; Fischer, Henninger, Redmiles 91; Redmiles 93a]. This work led to an evolutionary model of object-oriented software development [Fischer et al. 95]. This model focuses on three aspects of development: (1) evolution, (2) reuse and redesign, and (3) domain-orientation.

C.5.2 Human-Computer Interaction

Previously, the principal investigator has explored the role of usability techniques in the software life cycle [Redmiles 93b]. This analysis was based on a several year experience consulting in a scientific software group and in building and evaluating a prototype, knowledge-based tool to capture aspects of that experience. Usability techniques provide a starting point for semi-automated support. However, some adaptation is required to extend them for workplace evaluation.

C.5.3 Tools for Explanation and Learning

Man-made artifacts, including executable software systems, are not self-explanatory [Rittel 84]. Some mechanism for explanation is essential to enable human understanding of design. In the past, the principal investigator and colleagues have investigated associative representations for representing design knowledge [Redmiles 92; Redmiles 93b], including support for multiple perspectives and reinterpretation of explanations [Rathke, Redmiles 94; Stahl 93]. A unique feature of these investigations has always been the central role of a design artifact or prototype to make retrieval and entry of design knowledge possible with less overhead than traditional design rationale approaches [Reeves, Shipman 92b]. With respect to the content or kinds of knowledge that need to be represented, a previous development project indicated that design information consisted not only of textual argumentation supporting design decisions, but also mock-up's of potential interfaces, partial prototypes, sketches from design meetings, and informal discussions between members of a design team [Burns et al. 93]. Thus, the representation scheme, associative or otherwise, must accommodate indexes to multiple information media.

C.5.4 Software Curriculum for Academia and Industry

In the first year of his academic appointment, the principal investigator sought to combine aspects of human-computer interaction with software engineering in a graduate seminar entitled ''Human-Centered Software Development.'' The seminar looked at cognitive theories of design and use of artificial artifacts, practical formative and evaluative techniques, and prototype software environments for supporting better design. This seminar provided a basis for many of the proposed ideas. The investigator has been encouraged by 1) the positive reception of the seminar by students of human-computer interaction and software engineering discipline and 2) the successful adaptation of some of the material into an undergraduate course on software engineering and into seminars for industrial affiliates of the university.

Finally, as CASE tools (computer-aided software engineering) are often part of lab sessions for university courses in software engineering, the investigator is hopeful that design tools embedding some kind of active support for learning and applying human-computer interaction principles will be eventually another means of integrating aspects of the two fields. The investigator had a positive result with a related trial several years prior [Majidi, Redmiles 91].

C.6 Departmental Endorsement

The applicant's first, full-time, tenure-track appointment began on July 1, 1994.

I have read and endorse this Career Development Plan.

_____

Michael Pazzani, Chair
Information and Computer Science

Date October 16, 1995

D. Bibliography

[Anderson, Taylor, Whitehead 94]
K. Anderson, R. Taylor, E.J. Whitehead, *Chimera: Hypertext for Heterogeneous Software Environments*, Proceedings of Hypertext '94 (Edinburgh, Scotland), ACM, New York, September 1994, pp. 94-107.

[Bobrow 91]
D.G. Bobrow, *Dimensions of Interaction*, AI Magazine, Vol. 12, No. 3, 1991, pp. 64-80, (AAAI-90 Presidential Address).

[Burns et al. 93]
B. Burns, M. Atwood, A. Girgensohn, B. Zimmermann, J. Ostwald, *EVA: Tools to Support the Evolution of Complex Systems*, Technical Report S&T Technical Report #TM 93-0012, NYNEX Science and Technology, Inc., White Plains, NY, 1993.

[Conklin, Begeman 87]
J. Conklin, M. Begeman, *gIBIS: A Hypertext Tool for Team Design Deliberation*, Hypertext'87 Papers, University of North Carolina, Chapel Hill, NC, November 1987, pp. 247-251.

[Curtis 91]
B. Curtis, *Five Paradigms in the Psychology of Programming*, in M. Helander (ed.), *Handbook of Human-Computer Interaction*, North-Holland, Amsterdam, 1991, pp. 87-105, ch. 24.

[Curtis, Krasner, Iscoe 88]
B. Curtis, H. Krasner, N. Iscoe, *A Field Study of the Software Design Process for Large Systems*, Communications of the ACM, Vol. 31, No. 11, November 1988, pp. 1268-1287.

[Davis, Hsia 94]
A. Davis, P. Hsia, *Giving VOICE to Requirements Engineering*, IEEE Software, Special Issue on Requirements Engineering, Vol. 11, No. 2, March 1994, pp. 12-16.

[Ehn, Kyng 91]
P. Ehn, M. Kyng, *Cardboard Computers: Mocking-it-up or Hands-on the Future*, in J. Greenbaum, M. Kyng (eds.), *Design at Work: Cooperative Design of Computer Systems*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1991, pp. 169-195, ch. 9.

[Fischer 87]
G. Fischer, *A Critic for LISP*, Proceedings of the 10th International Joint Conference on Artificial Intelligence (Milan, Italy), J. McDermott (ed.), Morgan Kaufmann Publishers, Los Altos, CA, August 1987, pp. 177-184.

[Fischer 89]
G. Fischer, *Human-Computer Interaction Software: Lessons Learned, Challenges Ahead*, IEEE Software, Vol. 6, No. 1, January 1989, pp. 44-52.

[Fischer 90]
G. Fischer, *Communications Requirements for Cooperative Problem Solving Systems*, The International Journal of Information Systems (Special Issue on Knowledge Engineering), Vol. 15, No. 1, 1990, pp. 21-36.

[Fischer 92]
G. Fischer, *Domain-Oriented Design Environments*, Proceedings of the 7th Annual Knowledge-Based Software Engineering (KBSE-92) Conference (McLean, VA), IEEE Computer Society Press, Los Alamitos, CA, September 1992, pp. 204-213.

[Fischer et al. 91]
G. Fischer, A.C. Lemke, T. Mastaglio, A. Morch, *The Role of Critiquing in Cooperative Problem Solving*, ACM Transactions on Information Systems, Vol. 9, No. 2, 1991, pp. 123-151.

[Fischer et al. 92]
G. Fischer, A. Girgensohn, K. Nakakoji, D. Redmiles, *Supporting Software Designers with Integrated, Domain-Oriented Design Environments*, IEEE Transactions on Software Engineering, Special Issue on Knowledge Representation and Reasoning in Software Engineering, Vol. 18, No. 6, 1992, pp. 511-522.

[Fischer et al. 93]
G. Fischer, K. Nakakoji, J. Ostwald, G. Stahl, T. Sumner, *Embedding Critics in Design Environments*, The Knowledge Engineering Review Journal, Vol. 8, No. 4, December 1993, pp. 285-307.

[Fischer et al. 95]
G. Fischer, D. Redmiles, L. Williams, G. Puhr, A. Aoki, K. Nakakoji, *Beyond Object-Oriented Technology: Where Current Object-Oriented Approaches Fall Short*, Human-Computer Interaction, Vol. 10, No. 1, 1995, pp. 79-119.

[Fischer, Henninger, Redmiles 91]
G. Fischer, S.R. Henninger, D.F. Redmiles, *Cognitive Tools for Locating and Comprehending Software Objects for Reuse*, Thirteenth International Conference on Software Engineering (Austin, TX), IEEE Computer Society Press, ACM, IEEE, Los Alamitos, CA, 1991, pp. 318-328.

[Fischer, Stevens 91]
G. Fischer, C. Stevens, *Information Access in Complex, Poorly Structured Information Spaces*, Human Factors in Computing Systems, CHI'91 Conference Proceedings (New Orleans, LA), ACM, New York, 1991, pp. 63-70.

[Girgensohn, Redmiles, Shipman 94]
A. Girgensohn, D. Redmiles, F. Shipman, *Agent-Based Support for Communication between Developers and Users in Software Design*, Proceedings of the 9th Annual Knowledge-Based Software Engineering (KBSE-94) Conference (Monterey, CA), IEEE Computer Society Press, Los Alamitos, CA, September 1994, pp. 22-29.

[Gould, Boies, Lewis 91]
J. Gould, S. Boies, C. Lewis, *Designing for Usability: Key Principles and What Designers Think*, Communications of the ACM, Vol. 34, No. 1, 1991, pp. 75-85.

[Gould, Lewis 85]
J.D. Gould, C.H. Lewis, *Designing for Usability: Key Principles and What Designers Think*, Communications of the ACM, Vol. 28, 1985, pp. 300-311.

[Greenbaum, Kyng 91]
J. Greenbaum, M. Kyng (eds.), *Design at Work: Cooperative Design of Computer Systems*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.

[Guindon, Krasner, Curtis 87]
R. Guindon, H. Krasner, B. Curtis, *Breakdown and Processes During Early Activities of Software Design by Professionals*, in G.M. Olson, E. Soloway, S. Sheppard (eds.), *Empirical Studies of Programmers: Second Workshop*, Ablex Publishing Corporation, Lawrence Erlbaum Associates, Norwood, NJ, 1987, pp. 65-82.

[Herbsleb, Kuwana 93]
J. Herbsleb, E. Kuwana, *Preserving Knowledge in Design Projects: What Designers Need to Know*, Human Factors in Computing Systems, INTERCHI'93 Conference Proceedings, ACM, 1993, pp. 7-14.

[IEEE Software 94]
IEEE Software, Special Issue on Requirements Engineering, Vol. 11, No. 2, IEEE Computer Society, March 1994.

[John, Packer 95]
B. John, H. Packer, *Learning and Using the Cognitive Walkthrough Method: A Case Study Approach*, Human Factors in Computing Systems, CHI'95 Conference Proceedings (Denver, CO), ACM, New York, May 1995, pp. 429-436.

[Kay 92]
A. Kay, *The Knowledge Navigator*, ACM SIGGRAPH Video Review (CHI'92 Special Video Program), No. 79, 1992.

[Kintsch, Greeno 85]
W. Kintsch, J.G. Greeno, *Understanding and Solving Word Arithmetic Problems*, Psychological Review, Vol. 92, 1985, pp. 109-129.

[Kunz, Rittel 70]
W. Kunz, H.W.J. Rittel, *Issues as Elements of Information Systems*, Working Paper 131, Center for Planning and Development Research, University of California, Berkeley, CA, 1970.

[Laurel 90]
B. Laurel, *Interface Agents: Metaphors with Character*, in *The Art of Human-Computer Interface Design*, Addison-Wesley Publishing Company, Reading, MA, 1990, pp. 355-365.

[Lewis, Rieman 93]
C. Lewis, J. Rieman, *Task-Centered User Interface Design: A Practical Introduction*, FTP On-line Document, Department of Computer Science, University of Colorado, Boulder, CO, 1993, Available via anonymous ftp from ftp.cs.colorado.edu.

[Maes 94]
P. Maes, *Agents that reduce work and information overload*, Communications of the ACM, Vol. 37, No. 7, July 1994, pp. 30-40.

[Majidi, Redmiles 91]
M. Majidi, D. Redmiles, *A Knowledge-Based Interface to Promote Software Understanding*, Proceedings of the 6th Annual Knowledge-Based Software Engineering (KBSE-91) Conference (Syracuse, NY), IEEE Computer Society Press, Los Alamitos, CA, September 1991, pp. 178-185.

[Malone et al. 88]
T.W. Malone, K.R. Grant, K.-Y. Lai, R. Rao, D. Rosenblitt, *Object Lens: a ''spreadsheet'' for cooperative work*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'88), ACM, New York, September 1988, pp. 115-124.

[Malone, Lai, Fry 92]
T.W. Malone, K-Y. Lai, C. Fry, *Experiments with Oval: A Radically Tailorable Tool for Cooperative Work*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'92), ACM, New York, November 1992, pp. 289-297.

[Mannes, Kintsch 91]
S. Mannes, W. Kintsch, *Routine Computing Tasks: Planning as Understanding*, Cognitive Science, Vol. 3, No. 15, 1991, pp. 305-342, also published as Technical Report No. 89-8, Institute of Cognitive Science, University of Colorado, Boulder, CO.

[McCall 79]
R. McCall, *On the Structure and Use of Issue Systems in Design*, Doctoral Dissertation (1978), University of California, Berkeley, University Microfilms, 1979.

[Nielsen 93]
J. Nielsen, *Usability Engineering*, Academic Press, Reading, MA, 1993.

[Olson et al. 91]
G.M. Olson, J.S. Olson, M. Storrosten, M.R. Carter, *Small Group Design Meetings: An Analysis of Collaboration*, Human Computer Interaction, Special Issue on Computer Supported Cooperative Work, Vol. 7, No. 4, 1991.

[Olson, Moran 95]
J. Olson, T. Moran, *Mapping the Method Muddle: Guidance in Using Methods for User Interface Design*, in C. Lewis, P. Polson, L. Gugerty, M. Rudisill (eds.), *Human-Computer Interface Design: Success Cases, Emerging Methods and Real World Controls,* , 1995, (in press).

[Osterweil 87]
L. Osterweil, *Software Processes are Software too*, Proceedings of the 9th International Conference on Software Engineering (Monterey, CA), IEEE Computer Society, Washington, D.C., March 1987, pp. 2-13.

[Papert 86]
S. Papert, *Constructionism: A New Opportunity for Elementary Science Education*, Proposal to The National Science Foundation, MIT - The Media Laboratory, Cambridge, MA, 1986.

[Pennington 87]
N. Pennington, *Stimulus Structures and Mental Representations in Expert Comprehension of Computer Programs*, Cognitive Psychology, Vol. 19, 1987, pp. 295-341.

[Polanyi 66]
M. Polanyi, *The Tacit Dimension*, Doubleday, Garden City, NY, 1966.

[Potts et al. 94]
C. Potts, K. Takahashi, J. Smith, K. Ota, *An Evaluation of Inquiry-Based Requirements Analysis for an Internet Service*, Technical Report, Georgia Institute of Technology, College of Computing, Atlanta, GA, 1994.

[Potts, Takahashi, Anton 94]
C. Potts, K. Takahashi, A. Anton, *Inquiry-Based Requirements Analysis*, IEEE Software, Vol. 11, No. 2, March 1994, pp. 21-32.

[Rathke, Redmiles 94]
C. Rathke, D. Redmiles, *Improving the Explanatory Power of Examples by a Multiple Perspectives Representation*, Proceedings of the 1994 East-West Conference on Computer Technologies in Education (EW-ED'94) (Crimea, Ukraine), International Centre for Scientific and Technical Information, Moscow, Russia, September 1994, (in press).

[Redmiles 90]
D.F. Redmiles, *Explanation to Support Software Reuse*, Proceedings of the AAAI 90 Workshop on Explanation (Boston, MA), J. Moore, M. Wick (eds.), AAAI, Menlo Park, CA, July 1990, pp. 20-24.

[Redmiles 92]
D.F. Redmiles, *From Programming Tasks to Solutions—Bridging the Gap Through the Explanation of Examples*, Ph.D. Dissertation, Department of Computer Science, University of Colorado, Boulder, CO, 1992, Also available as TechReport CU-CS-629-92.

[Redmiles 93a]
D.F. Redmiles, *Reducing the Variability of Programmers' Performance Through Explained Examples*, Human Factors in Computing Systems, INTERCHI'93 Conference Proceedings, ACM, 1993, pp. 67-73.

[Redmiles 93b]
D.F. Redmiles, *Observations On Using Empirical Studies in Developing a Knowledge-Based Software Engineering Tool*, Proceedings of the 8th Annual Knowledge-Based Software Engineering

(KBSE-93) Conference (Chicago, IL), IEEE Computer Society Press, Los Alamitos, CA, September 1993, pp. 170-177.

[Reeves, Shipman 92a]
B.N. Reeves, F. Shipman, *Supporting Communication between Designers with Artifact-Centered Evolving Information Spaces*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'92), ACM, New York, November 1992, pp. 394-401.

[Reeves, Shipman 92b]
B.N. Reeves, F. Shipman, *Making it Easy for Designers to Provide Design Rationale*, Working Notes of the AAAI 1992 Workshop on Design Rationale Capture and Use, AAAI, San Jose, CA, July 1992, pp. 227-233.

[Rittel 84]
H.W.J. Rittel, *Second-Generation Design Methods*, in N. Cross (ed.), *Developments in Design Methodology*, John Wiley & Sons, New York, 1984, pp. 317-327.

[Robbins, Redmiles 95]
J. Robbins, D. Redmiles, *Software Architecture Design from the Perspective of Human Cognitive Needs*, 1995. (in preparation).

[Schoen 83]
D.A. Schoen, *The Reflective Practitioner: How Professionals Think in Action*, Basic Books, New York, 1983.

[Shipman 93]
F. Shipman, *Supporting Knowledge-Base Evolution with Incremental Formalization*, Ph.D. Dissertation, Department of Computer Science, University of Colorado, Boulder, CO, 1993, Also available as TechReport CU-CS-658-93.

[Shipman, Marshall 94]
F. Shipman, C. Marshall, *Formality Considered Harmfull: Experiences, Emerging Themes, and Directions*, Technical Report ISTL-CSA-94-08-02, Xerox Palo Alto Research Center, Palo Alto, CA, 1994.

[Simon 81]
H.A. Simon, *The Sciences of the Artificial*, The MIT Press, Cambridge, MA, 1981.

[Stahl 93]
G. Stahl, *Supporting Situated Interpretation*, Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society (Denver, CO), Lawrence Erlbaum Associates, Hillsdale, NJ, July 1993, (in press).

[Strong 94]
G. Strong, *New Directions in Human-Computer Interaction Education, Research, and Practice*, Technical Report, National Science Foundation (NSF) and Advanced Research Projects Agency (ARPA), Washington, D.C., November 1994.

[Suchman 87]
L.A. Suchman, *Plans and Situated Actions*, Cambridge University Press, Cambridge, UK, 1987.

[Terveen, Selfridge, Long 93]
L.G. Terveen, P.G. Selfridge, M.D. Long, *From Folklore to Living Design Memory*, Human Factors in Computing Systems, INTERCHI'93 Conference Proceedings, ACM, April 1993, pp. 15-22.

[Wasserman et al. 86]
A. Wasserman, P. Pircher, D. Shewmake, M. Kersten, *Developing Interactive Information Systems with the User Software Engineering Methodology*, IEEE Transactions on Software Engineering, Vol. 12, No. 2, February 1986, pp. 198-210.

[Wharton et al. 94]
C. Wharton, J. Rieman, C. Lewis, P. Polson, *The Cognitive Walkthrough Method: A Practitioner's Guide*, in J. Nielsen, R. Mack (eds.), *Usability Inspection Methods*, John Wiley & Sons, Inc., New York, 1994, pp. 105-140, ch. 5.

[Winograd, Flores 86]
T. Winograd, F. Flores, *Understanding Computers and Cognition: A New Foundation for Design*, Ablex Publishing Corporation, Norwood, NJ, 1986.

[Young, Taylor 94]
P. Young, R. Taylor, *Human-Executed Operations in the Teamware Process Programming System*, Proceedings of the Ninth International Software Process Workshop (Arlington, VA), IEEE Computer Society Press, Los Alamitos, CA, October 1994, pp. 78-81.

E. Biographical Sketches

E.1 David Redmiles, Principal Investigator

Contact Information

Department of Information and Computer Science; University of California; Irvine, CA, 92717;
Phone: (714) 824-3823; Fax: (714) 824-4056; E-mail: redmiles@ics.uci.edu

Education

1987-1992	Ph.D. Computer Science, University of Colorado, Boulder
1980-1982	M.S. Computer Science, The American University, Washington, D.C.
1976-1980	B.S. Mathematics and Computer Science, The American University, Washington, D.C.

Academic Appointments

1994 - present	University of California, Irvine, Department of Information and Computer Science
	Assistant Professor (July 1994-present)
1987 - 1994	University of Colorado, Boulder, Department of Computer Science
	Postdoctoral Fellow (1992-1994)
	Graduate Research Assistant (1987-1992)

Industrial Appointments

Summer 1989	Gesellschaft fuer Mathematik und Datenverarbeitung mbH
	Guest Scientist
1979-1987	U.S. Department of Commerce, National Institute of Standards and Technology (formerly National Bureau of Standards)
	Computer Scientist (1981-1987)
	Mathematician (1980-1981)
	Mathematics Aide (1979-1980)

Five Publications Most Closely Related to the Proposed Project

‘‘Beyond Object-Oriented Technologies: Where Current Object-Oriented Approaches Fall Short’’ (with G. Fischer, L. Williams, G. Puhr, A. Aoki, and K. Nakakoji), 1995, Human-Computer Interaction, (in press).

‘‘Agent-Based Support for Communication between Developers and Users in Software Design’’ (with A. Girgensohn and F. Shipman), 1994, Proceedings of the 9th Annual Knowledge-Based Software Engineering (KBSE-94) Conference (Monterey, CA), IEEE Computer Society Press, Los Alamitos, CA, September 1994, pp. 22-29.

‘‘Observations On Using Empirical Studies in Developing a Knowledge-Based Software Engineering Tool,’’ Proceedings of the 8th Annual Knowledge-Based Software Engineering (KBSE-93) Conference (Chicago, IL)", IEEE Computer Society Press, Los Alamitos, CA, September 1993, pp. 170-177.

‘‘Reducing the Variability of Programmers’ Performance Through Explained Examples,’’ Human Factors in Computing Systems, INTERCHI’93 Conference Proceedings (Amsterdam, The Netherlands), ACM, New York, 1993, 67-73.

‘‘Supporting Software Designers with Integrated, Domain-Oriented Design Environments’’ (with G. Fischer, A. Girgensohn, and K. Nakakoji), IEEE Transactions on Software Engineering, Special Issue on Knowledge Representation and Reasoning in Software Engineering, Vol. 18, No. 6, 1992, pp. 511-522.

Five Additional Publications

‘‘Improving the Explanatory Power of Examples by a Multiple Perspectives Representation’’ (with C. Rathke), Proceedings of the 1994 East-West Conference on Computer Technologies in Education (EW-ED’94) (Crimea, Ukraine), International Centre for Scientific and Technical Information, Moscow, Russia, September 1994, (best paper award).

‘‘From Programming Tasks to Solutions—Bridging the Gap Through the Explanation of Examples,’’ Ph.D. Dissertation, Department of Computer Science, University of Colorado, Technical Report CU-CS-629-92, July, 1992.

‘‘A Knowledge-Based Interface to Promote Software Understanding’’ (with M. Majidi), Proceedings of the 6th Annual Knowledge-Based Software Engineering (KBSE-91) Conference (Syracuse, NY), IEEE Computer Society Press, Los Alamitos, CA, September 1991, pp. 178-185.

‘‘Intertwining Query Construction and Relevance Evaluation’’ (with G. Fischer and S. Henninger), Human Factors in Computing Systems, CHI’91 Conference Proceedings (New Orleans, LA), ACM, New York, 1991, pp. 55-62.

‘‘ASM/NBS Numerical and Graphical Database for Binary Alloy Phase Diagrams’’ (with J. Sims and J.B. Clark), in *Computerized Metallurgical Databases*, J.R. Cuthill, N.A. Gokcen, and J.E. Morral (eds.), The Metallurgical Society, 1988, pp. 119-134.

Own Graduate and Postdoctoral Advisors:

Walter Jacobs (American University—deceased), Angela Wu (American University), Gerhard Fischer (University of Colorado).

E.2 Primary Collaborators

The proposal draws strength from the caliber of the faculty and graduate students at the University of California, Irvine, and from on-going research collaborations with faculty and research staff at several universities and research institutes.

University of California, Irvine

- *Richard Taylor* (professor in the Department of Information and Computer Science) has worked for the past several years on software environments for large-scale system development. This has included traditional models of software engineering as well as current research in supporting process models. His systems have emphasized a large human-computer interface component. His research group contributes both working substrates for supporting trial implementations of this research as well as insight into isolating unique aspects of the software process for developing interactive systems.
- *Jonathan Grudin* (associate professor in the Department of Information and Computer Science) is an expert in computer-supported cooperative work and human-computer interaction. His expertise places him in a unique position to critique progress on both the agent-based approach to supporting communication and collaboration as well as comment on the validity of the workplace evaluation methods.

University of Colorado, Boulder

- *Gerhard Fischer* (professor in the Department of Computer Science) has developed innovative techniques of human-computer interaction including critics and active help, both of which are closely related to agent mechanisms. Moreover, his theory of domain-oriented design environments provides a foundation for an integrated system of software development where the underlying metaphor is a ‘‘conversation of design.’’
- *Raymond McCall* (associate professor in the College of Environmental Design) has been one of the pioneers in developing issue-based information systems to support design as an argumentative process. His many years of experience provide invaluable insights to the strengths and weaknesses of argumentation and design rationale structures. His criticisms of those components in this research will strengthen their use.

Texas A&M University, College Station

- *Frank Shipman* (research assistant scientist in the Department of Computer Science) has made contributions in semi-formal representations, hypermedia, and active objects (agents). These experiences are directly applied to the representation of expectation agents and their operation over design intent information in this project.
- *Catherine Marshall* (research associate scientist in the Department of Computer Science) has pioneered work in the area of hypertext and argumentation, including NoteCards, Aquanet, and VIKI. Her experience will provide valuable feedback on the representation of design intent and its use by teams of designers and communities of users.

University of Stuttgart

- *Christian Rathke* (assistant professor in the Institute for Computer Science) has contributed to the field of knowledge representation since his 1986 dissertation on object-oriented, knowledge-based languages. This work has evolved into a frame-based representation language capable of capturing domain knowledge from different perspectives. The language supports various behaviors suitable for representing agents, including constraint propagation.

NYNEX Science & Technology, White Plains

- *Michael Atwood* (Technical Director, Advanced Software Development Environments) has held several positions in research and industry. Throughout his 20-year career, he has had maintained an interest and participation in the human-computer interaction community, including systems development and evaluation. In the projects he has directed, he has demonstrated the value of adapting techniques of cognitive psychology and, in particular, methods of cognitive modeling, to the development of improved user interfaces.
- *Andreas Girgensohn* (Member Technical Staff, Advanced Software Development Environments) has done extensive work in the design and development of object-oriented representations and innovative human-computer interfaces. In relation to this project, he participated in the development of the Bridget system used as a model for part of this proposal and has developed the Form Definition Language that the agent-based architecture will interact with.

F. Budget

F.1 Description and Justification

F.1.1 Time Frame

The proposed time frame is starting 7/1/96 and lasting for 4 years.

F.1.2 Personnel

The summer salary allows the principal investigator to devote full time during that period toward the project.

The two research assistants are expected to make substantial contribution to the systems building and evaluation efforts. These expectations are greater during the summer period of support. The money requested will cover their half-time stipend during the academic year and full-time summer salary.

F.1.3 Equipment

Major aspects of the work are being planned for Apple Macintosh or equivalent platforms. The intention is that the end products be as accessible as possible to general computer users. First, this aim facilitates the recruitment of subjects for evaluating system prototypes. Second, the low cost of these systems implies that they will indeed be the kind of platform for which more end users will need applications support. Third, the cost-performance tradeoff is very favorable for establishing a research group. Fourth, many educational institutions already have Macintosh laboratories for undergraduate software education.

The budget includes a modest request for approximately one workstation per year (e.g., Power Macintosh caliber) and related software (e.g., C++ programming environment, database software, word processing software). They will provide programming platforms for the research assistants, for others working with the research assistants (including undergraduates) on specific components, and for usability evaluations.

F.1.4 Travel

Domestic travel money will be used to enable collaborations for the personnel with the industrial research partner (NYNEX, White Plains) and other university collaborators. Travel money will of course permit attendance and reporting at relevant conferences (e.g., CHI and CSCW). Foreign travel money will be used to interact with researchers at foreign universities and foreign research laboratories and to attend international conferences (e.g., INTERCHI, HYPERTEXT, ICSE).

F.1.5 Materials and Supplies

These funds cover communication costs, video tapes for recording experimental observations and scenarios, and miscellaneous materials (e.g., poster board, tape, glue, pens, etc. for preparing mock-up's) and copying to facilitate discussion of the research among the collaborators.

F.1.6 Publication Costs

These funds cover preparation of reports and papers, including reprints and editing costs.

F.2 Spreadsheet

G. Current and Pending Support

Currently, the principal investigator has no financial support outside of the University of California and no research funding is pending.

H. Facilities, Equipment and Other Resources

H.1 University of California, Irvine

The Department of Information and Computer Science has a large number of Sun Workstations and Macintosh systems. However these are dedicated to other, on-going research projects or reserved in teaching labs. Hence the equipment requests discussed in Section F.1.3.

However, the project will naturally rely on the internal and external networking capabilities and laboratory space provided for faculty and students.

H.2 NYNEX Corporation, White Plains

The collaborating group at NYNEX, White Plains, is using Sun SparcStations, Apple Macintoshes, and Silicon Graphics INDYs. This equipment, in conjunction with several p.c.'s, will be used over the network to enact scenarios of expectation agent support.

I. Special Information and Supplementary Documentation

I.1 Statement of Support from the Chair

I.2 Letter of Cooperation from the NYNEX Corporation

Table of Contents

List of Figures