Knowledge Management for Software Design

David F. Redmiles

with

David Hilbert, Michael Kantor, and Jason Robbins,

Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425

email: redmiles@ics.uci.edu
voice: (714) 824-3823
fax: (714) 824-1715
web  http://www.ics.uci.edu/~redmiles/

Abstract

The poor design of interactive systems has been attributed to insufficient communication between developers and end users. Critical forms of communications include soliciting requirements and feedback on prototype systems. Our research develops a software lifecycle emphasizing feedback from end users and communication among end users, developers, and other stakeholders in a software development project. Communication implies the transferal, storage, and review of knowledge about a software project. Three software systems — Argo, EDEM, and the Knowledge Depot — illustrate one kind of support that can be provided for the management of knowledge to improve the design of interactive software. Specifically, Argo provides support for the design of systems; EDEM provides support for collecting feedback and usage data from end users; and the Knowledge Depot provides support for reviewing feedback and other information about a system.
Knowledge Management for Software Design

Introduction

For many years, the poor design of interactive systems has been attributed to insufficient communication between developers and end users [Gould, Lewis 85; Fischer 89a]. The problem is compound. Usability practices from the field of human-computer interaction are not integrated into the theory and practice of software engineering. Therefore many software developers are unaware 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]. Our research 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 theory of human-centered software development.

Human-Centered Software Development Lifecycle

Figure 1 shows one way to illustrate a human-centered software development lifecycle. Namely, prototype systems are designed, evaluated in a usage situation, and the results of the evaluation are reviewed and incorporated back into a revised design.

The human-centered software development lifecycle presented above is being explored through three systems. The Argo Software Design Environment implements the design theory component illustrated in Figure 1. The EDEM (Expectation-Driven Event Monitoring) system substrate implements the observation of usage component. The Knowledge Depot implements the review component. The three systems are briefly described below. A more complete version of this paper will provide an integrated scenario. In the meantime, the reader is referred to existing papers.
The Argo Software Design Environment

Argo is our software architecture design environment. It is based on the cognitive theories of reflection-in-action (Schoen, 1983, 1992), opportunistic design (Guindon, Krasner, Curtis 1987; Visser, 1990), and comprehension and problem solving (Kintsch, Greeno, 1985; Fischer, 1987; Pennington 1987). Argo offers basic CASE tool features for entering a software architecture. Argo contains critics that automatically provide design feedback, a user interface for browsing design feedback, a process model to guide the architect and help control critics, and multiple coordinated design perspectives.

Our current Argo implementation support critics that run in a background thread of control to continuously analyze a software architecture as it is being manipulated. Critics are made up of an analysis predicate, a design feedback item, and various attributes used to determine if the critic is timely and relevant. Criticism control mechanisms are objects that define Argo’s policies on when to apply individual critics to a given part of the architecture. For example, one criticism control mechanism selects critics that are relevant to stated design goals, while another selects critics that are timely (relevant to design decisions that the architect is currently considering). Argo associates critics with specific types of design materials (i.e., elements of the architect model) rather than storing all critics in a central knowledge-base. This allows us to keep Argo’s kernel simple and flexible, and will soon allow up to load models and their associated critics over the Internet as needed.

EDEM Software for Usage Data Collection

We have developed prototype software for detecting and resolving mismatches in usage expectations by allowing developers to specify their expectations in terms of user interface events. These expectations are encoded in the form of expectation agents that continually monitor usage of the application and perform various actions when encapsulated expectations are violated.

Other researchers have proposed event monitoring as a means for collecting usability data, however, existing approaches suffer from one or more of the following limitations: (1) low-level semantics—events are captured and analyzed at the window system level, or just slightly above; (2) decontextualization—analysis is done post-hoc on raw event traces, potentially relevant contextual cues are lost; (3) “one-way” communication—data flows from users to developers who must then infer meaning, no “dialogue” is established to facilitate mutual understanding; (4) batch orientation—hypothesis formation and analysis is performed after large amounts of (potentially irrelevant) data have been collected, no means for hypotheses to be analyzed and action taken while users are engaged; and (5) privacy issues—arbitrary events are collected without any explicit constraints on the purposes of collection, no way to provide users with discretionary control over what information is collected and what information is kept private.

The EDEM software addresses these issues and goes beyond existing approaches in supporting user involvement in development. Our system provides: (1) a multi-level event model—allowing
agents to compare usability expectations against actual usage at reasonable levels of abstraction; (2) contextualization—taking action and collecting information while users are engaged in using the application; (3) two-way communication—helping initiate dialog between users and developers when breakdowns occur; and finally, (4) specializable monitoring and analysis—promoting a shift from batch-oriented data collection and analysis to hypotheses-guided collection and analysis.

The Knowledge Depot for Information Review and Project Awareness

Groups need a mechanism for communicating about usability and other data, design decisions, and general progress. The Knowledge Depot, an enhanced version of GIMMe (Zimmermann, Lindstaedt, Girgensohn, 1997), captures email conversations, and other information, and organizes this information, allowing users to browse through the information to rediscover (or learn for the first time) why different decisions were made, what problems were encountered as a result of those decisions, and allowing the user to regain some of the context in which those decisions were made.

The system organizes its knowledge around a set of topics defined over time by all users of the system. A topic consists of four things:

1. A phrase describing a concept, task or activity representing aspects of the group’s work;
2. A place where people go to find information;
3. A definition of the type of information the system looks for to determine whether something belongs in the topic; and
4. A destination that people will aim their messages at in order to have the message stored correctly for later retrieval.

At any time, any user can add a new topic, and based on the definition of the topic, all existing messages will be checked to see if they belong in the topic (in addition to any other topics that the message is already in). Potentially, the entire topic structure could be removed, and replaced with a new topic structure, with messages automatically reorganizing themselves to account for the new hierarchy. While a newsgroup acts as an oversized shared email box, Knowledge Depot acts as a shared database. The database is used to organize all of a groups discussions and documents, and provides standard types of database queries and reorganizations.

References


Hilbert, D., Redmiles, D. *Agents for Collecting Application Usage data Over the Internet*, Pro-


