User Interface Evaluation and Empirically-Based Evolution of a Prototype Experience Management Tool

Carolyn B. Seaman, Manoel G. Mendonça, Victor R. Basili, *Fellow*, *IEEE*, and Yong-Mi Kim

Abstract—Experience management refers to the capture, structuring, analysis, synthesis, and reuse of an organization's experience in the form of documents, plans, templates, processes, data, etc. The problem of managing experience effectively is not unique to software development, but the field of software engineering has had a high-level approach to this problem for some time. The Experience Factory is an organizational infrastructure whose goal is to produce, store, and reuse experiences gained in a software development organization [6], [7], [8]. This paper describes The Q-Labs Experience Management System (Q-Labs EMS), which is based on the Experience Factory concept and was developed for use in a multinational software engineering consultancy [31]. A critical aspect of the Q-Labs EMS project is its emphasis on empirical evaluation as a major driver of its development and evolution. The initial prototype requirements were grounded in the organizational needs and vision of Q-Labs, as were the goals and evaluation criteria later used to evaluate the prototype. However, the Q-Labs EMS architecture, data model, and user interface were designed to evolve, based on evolving user needs. This paper describes this approach, including the evaluation that was conducted of the initial prototype and its implications for the further development of systems to support software experience management.

Index Terms—Experience management, knowledge management, experience reuse, user interface evaluation, empirical study.

1 Introduction

Software development organizations in general strive to develop higher quality systems at a lower cost. Yet the processes used to develop such software are still very primitive in the sense that each new development team has to relearn the mistakes of its predecessors. The reuse of an organizations' own products, processes, and experience is a feasible solution to this problem. But, implementation of the idea, in most cases, has not gone beyond reuse of small-scale code components in very specific, well-defined, situations. True learning within a software development organization requires that organizational experiences, both technological and social, be captured, structured, searchable, analyzed, synthesized, made accessible, and maintained, so that members of the organization can learn from them and apply them to new problems. In other words, experience must be managed. This requires an experience management framework of concepts, a scheme for structuring the experience, procedures for the day-to-day management of the experience, and tools to support all of this.

The goals of experience management include such abstract notions as improving development processes and making development work more productive and satisfying. But, there are also a number of more practical and concrete reasons for pursuing it. The problems of employee turnover are not unique to software development, but most software development organizations have experienced the painful loss of a critical member, along with their valuable, usually undocumented experience. The other side of that coin is the frustratingly long time most new employees take to become productive. Development organizations often also include a number of experts on various programming languages, platforms, or systems. Even if such experts remain in the organization, they quickly become overwhelmed with requests for advice and information (and helping new employees come up to speed), in addition to their regular work. Reinvention, too, is a huge waste of resources in most organizations as employees spend hours re-creating something that already existed but was unknown or inaccessible. Mistakes are also often unnecessarily reinvented.

An obvious area to search for solutions to this problem is knowledge management. This area has made significant contributions in terms of terminology and conceptual frameworks with which to reason about experience management [28], [34], [44]. In addition, numerous systems have been developed to support knowledge management in organizations and these systems provide innovative ideas about how organizations can leverage their knowledge assets [14], [28], [29], [30].

- *C.B. Seaman is with the Department of Information Systems, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, and is also with the Fraunhofer Center for Experimental Software Engineering—Maryland, University of Maryland, 4321 Hartwick Rd., Suite 500, College Park, MD 20742-3290. E-mail: cseaman@umbc.edu.*
- *M.G. Mendonça is with the Computer Networks Research Group (NUPERC), Salvador University (UNIFACS), Av. Cardeal da Silva 747, Salvador, Ba, Brazil, 40220-141. E-mail: mgmn@unifacs.br.*
- *V. Basili is with the Department of Computer Science, University of Maryland College Park, A.V. Williams Building, College Park, MD 20742, and is also with the Fraunhofer Center for Experimental Software Engineering—Maryland, University of Maryland, 4321 Hartwick Rd., Suite 500, College Park, MD 20742-3290. E-mail: basili@cs.umd.edu.*
- *Y.-M. Kim is with the School of Information, University of Michigan, 550 E. University Ave., Ann Arbor, MI 48109. E-mail: kimym@si.umich.edu.*

Manuscript received 20 May 2002; revised 1 Apr. 2003; accepted 14 May 2003. Recommended for acceptance by M. Shepperd.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 116579.

Many of the questions we have encountered in our work on experience management in the software industry, however, must be explored experimentally. That is, the answers to such questions in a given context are best found by deploying well-grounded solutions, carefully monitoring their effects, doing a thorough evaluation that takes into account technical, organizational, and social factors, and then evolving the system to reflect feedback from the user organization. This type of evaluation can also be done on a small scale, by evaluating pieces of the whole solution early in the planning phase. Such careful evaluations and feedback seem to be lacking in the knowledge management literature, where evaluation tends to be sweeping and done only after a full-blown system is operational and deployed.

This paper reports on an effort to empirically derive a solution to the experience management problem and then experimentally evolve it. The prototype system described here, called the Q-Labs Experience Management System (or the Q-Labs EMS), was designed to support experience management in a multinational software improvement consultancy called Q-Labs. Q-Labs specializes in helping its clients improve their software engineering practices by implementing state-of-the-art technologies in their software development organizations. Q-Labs has helped many of its clients implement some of the principles of the Experience Factory concept, an approach to experience management in a software development organization [6], [7], [8], described in more detail in Section 2.

Our system development and evolution approach is aimed at instantiating and evolving a system that is highly dependent upon evolving through use. Throughout, we employ experimentation, by:

1. carefully defining the problem and associated research questions (through gathering data on organizational goals, needs, preferences, and processes),
2. creating a set of evaluation criteria (that spells out the empirical procedures to be followed to carry out the evaluation),
3. building an initial (possibly partial) solution grounded in empirically gathered information on the organization's specific needs and based on a flexible component independent architecture,
4. carefully evaluating the effect of the initial solution, and
5. preparing for continuous evolution of the entire experience management system.

The remainder of the paper is organized around these steps. This work laid the groundwork for what is now a very active line of research, occupying several different projects being carried out by the Fraunhofer Center-Maryland and its research and industrial partners.

Section 2 describes the underlying concepts and previous research upon which our work is based. Section 3 describes how Step 1 of our development and evolution approach was carried out, resulting in the initial set of high-level requirements and open research questions. Section 4 briefly discusses the chief evaluation criteria (Step 2), usability and flexibility, that drove the evaluation of the first prototype. Section 5 presents the initial prototype, whose goal was to instantiate the requirements into a flexible system that can change and evolve as requirements become clearer or even change. This represents Step 3 of our approach as applied at Q-Labs. Section 6 offers the first evaluation and feedback for evolving the requirements (Step 4). The intention is that the evaluation process would become part of the regular use of the system so it may continue to evolve to meet the needs of the users and the organization. Finally, Section 7 offers some insight into how this process set the stage for subsequent evolution of the EMS prototype (Step 5).

2 Background

This section discusses some of the underlying concepts behind the Q-Labs EMS and places it in the state of the art.

2.1 Knowledge Management

Tiwana [44] defines knowledge management (KM) as "the management of organizational knowledge for creating business value and generating a competitive advantage." Other definitions [26] enumerate the various activities that comprise KM, including building, renewing, formalizing, making accessible, and applying knowledge for improving the performance of the organization.

The KM literature generally divides knowledge into explicit knowledge (knowledge that is or can be explicitly represented) and tacit knowledge (knowledge that lies inside the minds of individuals [34]). Our work deals strictly with explicit knowledge. The process of knowledge externalization (transforming tacit knowledge into explicit knowledge), while very important, is outside our scope. Instead, our work aims to support knowledge internalization, i.e., helping people to absorb explicit knowledge [34].

Tiwana proposes a seven-layer model describing the different types of technology components necessary to support KM in an organization [44]. These layers are the user interface layer, the access and authentication layer, the collaborative intelligence and filtering layer, the application layer, the transport layer, the middleware and legacy integration layer, and the repositories. Lindvall et al. (and Lawton [25]) also propose a seven-layer model [28]. These layers are the business application layer, personalized knowledge gateway layer, KM services layer, organizational taxonomy layer, document and content management layer, low-level IT infrastructure layer, and information and knowledge sources layer. Although different on some aspects, the Lindvall-Lawton and Tiwana models are very similar in many important aspects. They both state that KM should be supported by an integrated collection of technologies for authoring, indexing, classifying, storing, and retrieving information, as well as for collaboration and application of knowledge. A friendly front-end and a robust back-end are basic requirements for knowledge management tools. The Q-Labs EMS includes elements of all seven of these layers. However, it does not intend to be a comprehensive system. It focuses on the use of information visualization to foster experience internalization. It does not intend to support workflow, to allow collaborative work, or to capture expert reasoning, all of which are other important enabling technologies for knowledge management [14], [28].

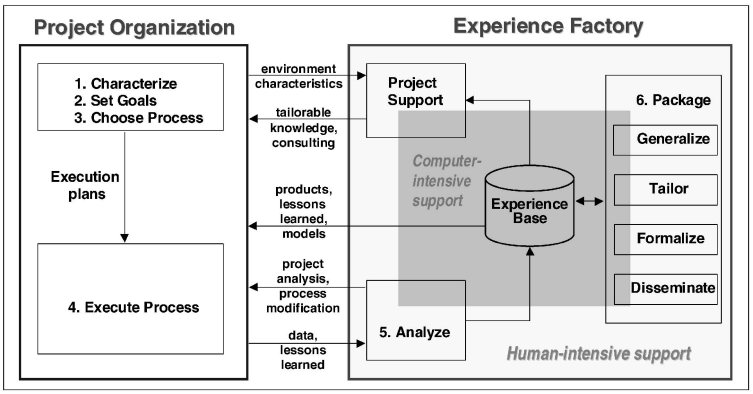


Fig. 1. Experience Factory structure.

Search strategies are also an important issue in designing and integrating a KM infrastructure. Tiwana [44] describes four general types of searching: metasearching (based on broad categories), hierarchical searching (based on increasingly more specific categories), attribute searching, and content searching. These four categories can be combined in the same system. The Q-Labs EMS uses a type of attribute searching, but the search interface provided for manipulating attribute values is innovative, highly user-friendly, and lends itself to information visualization and knowledge internalization. The attributes chosen for use in searches generally fall into the categories of activities, domain, form, type, products and services, time, and location. Taxonomies are defined by classification managers or librarians and can be modified or expanded as needed. Also, in the Q-Labs EMS, packages are categorized into broad "package types," each of which has its own set of attributes tailored to the types of artifacts (e.g., documents, people, etc.) in the package type. The use of attribute searching is highly flexible and can be adapted to support many different underlying paradigms, including case-based reasoning (CBR), which is used to implement a facet-based approach [35] to classify and search for experience packages [43]. The attribute searching paradigm that we use is simply a more general form of facet-based searching.

One final distinction is between "push" and "pull" technologies, either (or both) of which can be utilized by a KM infrastructure. The Q-Labs EMS is strictly a "pull" system, meaning that the user must initiate all activity with the system, and must specify the types of information they want to search for. "Push" technology, on the other hand, allows the system itself to notify or provide information that may be of interest to a user, without an explicit request from the user [23].

The KM literature provides many approaches to evaluating a KM program, once deployed in an organization. These approaches range from frameworks for developing measures and evaluation criteria for KM programs [5], [24], [36], [42] to extracting success criteria from successful KM stories in the literature [15], [16]. In general, this literature does not directly address the underlying technical infrastructure supporting the KM program, but instead mentions the use of already existing, off-the-shelf technologies and tools. Further, this work is concerned with the evaluation of a KM program once it is fully implemented and operational. It does not provide a way to evaluate pieces of the infrastructure for their suitability to support the KM goals of the organization. This is a gap in the literature that this paper addresses by introducing a very targeted approach to evaluating a crucial aspect of the infrastructure for supporting KM, before a full KM program is fully deployed in an organization.

2.2 Experience Management

Experience can be defined, for our purposes, as insight or lessons that have been gained through the practical application of knowledge. Thus, experience management (EM) is closely related to knowledge management (KM). One of the major differences between the two is that EM must process a nearly continuous "stream of knowledge" related to the activities of the organization [3].

The originating concept behind EM, The Experience Factory, predates the popularity of the term knowledge management. The Experience Factory is an organizational infrastructure whose goal is to produce, store, and reuse experiences gained in a software development organization [6], [7], [8]. The Experience Factory organizes a software development enterprise into two distinct organizations, each specializing in its own primary goals. The Project Organization focuses on delivering the software product and the Experience Factory focuses on learning from experience and improving software development practice in the organization. Although the roles of the Project Organization and the Experience Factory are separate, they interact to support each other's objectives. The feedback between the two parts of the organization flows along well-defined channels for specific purposes, as illustrated in Fig. 1.

Experience Factories recognize that improving software processes and products requires: 1) continual accumulation of evaluated and synthesized experiences in *experience packages*, 2) storage of the experience packages in an integrated *experience base* accessible by different parts of the organization, and 3) creation of *perspectives* by which different parts of the organization can look at the same experience base in different ways. Some examples of experience packages might be the results of a study investigating competing design techniques, a software library that provides some general functionality, or a set of data on the effort expended on several similar projects.

The Experience Factory concept has been implemented in a number of software development organizations that have addressed the above questions in various ways (e.g., [9], [10], [22], [38]). The Software Engineering Laboratory (SEL) [9] is one example of an Experience Factory. The SEL Process Improvement Paradigm provides a practical method for facilitating product-based process improvement within a particular organization, based on effective use of that organizations' own experience. Because it directly ties process improvement to the products produced, it allows an organization to optimize its process for the type of work that it does. Using this approach, the SEL has reduced development costs by 74 percent, decreased error rates by 85 percent, and increased reuse by over 300 percent over the past 15 years [9].

There is a significant amount of literature on EM, which has come to refer to a focused subarea of knowledge management that grew out of the Experience Factory concept. Experience management generally refers to efforts by a software development organization to capture and reuse its experience [37]. The EM literature includes methods for representing software engineering experience [43], retrieval paradigms [46], and approaches to building and initially populating an experience base for an organization [43].

Much of this literature focuses on the COIN system at the Fraunhofer IESE [3], its underlying and associated tools [43], and their applications to different domains [12]. COIN is based on the case-based reasoning paradigm [47] for search and retrieval. It has been augmented with such technologies as moderated discourse analysis [4]. The work described in this paper predates much of the literature cited here, and also gave rise to continuing work in this area, including [11] and [27].

Some of this literature focuses specifically on the technical infrastructure (i.e., all the software systems needed to store, retrieve, evaluate, maintain, disseminate, and utilize the information of the experience base [43]) supporting EM. This includes technical requirements for this infrastructure [13], [43] and architectures to support it [1]. An important piece of technical infrastructure is the user interface, one of the major concerns of our work. The EM literature, unfortunately, does not address this in any great detail. In [2], the user interface is recognized as a possible barrier to usage, and thus can degrade perceived usefulness. In [43], an extensive evaluation of COIN revealed, among other things, that the user interface (which was simple a "general-purpose browser") needed more attention. However, there are no techniques proposed in this literature for designing or evaluating the user interface specifically.

The literature on evaluation of EM systems is of particular interest to our work. In [2], Althoff and his coauthors describe a method for evaluating and improving an organizational memory, based on measures of perceived usefulness. The model of perceived usefulness is quite comprehensive, using a cause-effect model to incorporate the many different factors that contribute to this complex concept. Like the KM literature on evaluation surveyed above, Althoff et al.'s evaluation approach assumes a fully operational EM infrastructure before the evaluation can take place.

Feldman et al. [18] also present work on using measurement to evaluate an operational experience repository. In Carsten Tautz's dissertation [43], an extensive formal experiment is described in which the use of an experience base is compared to reliance on human sources of information, in terms of usefulness of information and efficiency of information gathering. While this experiment was valuable in developing many insights into the use of a well-designed experience base with sophisticated supporting infrastructure, it required the use of a fully operational system, and does not provide any insight into how parts of the infrastructure, or prototypes, could be evaluated earlier in the development process.

2.3 Visual Exploration of Information

A crucial aspect of achieving success in implementing systems for experience reuse is acceptance. In this scope, the system's user interface is critical, as even minor usability problems will demotivate users, thus undermining the use and success of the system, as discussed in the previous section. The Q-Labs EMS uses a search and retrieval interface that is based on information visualization and visual data mining tools.

Humans have poor short-term memory, i.e., they have a limited ability to search and interpret textual and/or tabular data [33]. On the other hand, humans can interpret a visual scene in a matter of milliseconds. Information visualization tools play with this human ability to allow domain experts–usually lay people–to view data in creative ways. Most such tools allow active navigation on the visual screen, enabling zooming, rotation, repositioning, and sweeps over the visible areas. They also allow the interactive control of the presentation formats and visible visual attributes. Interactive control of the information being shown is also usually an element of visualization tools, enabling users to look at data from a high-level perspective or quickly diving into more detailed subsets of data.

This type of functionality can be very effectively used to explore, interpret, and search information. It is our belief that this approach is key for experience management tools. It allows users not only to find information but also to visualize the kind of information that is stored in the repository. Combined with the right querying devices, this type of functionality can be used to create fuzzy and nonzero hit queries in which the number of results satisfying a query can be visually previewed before any information is retrieved from experience repositories [20]. It also aids novices to learn by themselves about the organization's experience classification schema and its available body of knowledge.

3 Defining the Problem, the Requirements, and Open Questions

The first step of our approach, as with any development effort, is to gather the customer's requirements. In initial interviews with Q-Labs, they expressed their goals for this project as providing a "virtual office" for the organization, which is spread across two continents, and allowing each

Q-Labs consultant to benefit from the experience of every other Q-Labs consultant. Subsequent, iterative, ongoing discussions revealed more detail both about Q-Labs' vision for EM and their current needs and practices. These discussions included drawing object models and prototype screens, working through scenarios, and telling war stories about current problems. Based on these conversations and the goals that were derived from them, we defined a set of high-level requirements aimed at making the Q-Labs EMS reliable, easy to use, and flexible enough to support the Experience Factory concept and meet the needs of Q-Labs.

R1. The system shall support geographically distributed organizations, allowing them to share and manage experience packages remotely.

R2. The repository shall be portable to all standard computer platforms (Windows, NT, Unix) and be equally robust and reliable on each of them.

R3. The user interface level shall be platform independent.

R4. The data model shall be simple enough that Q-Labs EMS users are able to correctly categorize their experience packages, but powerful enough to model unanticipated types of packages. The idea here is that the system must adapt to the current practices, processes, and products of different organizations and not vice-versa.

R5. There must be no usability obstacles to the understanding and use of the system. The criteria here is that users, when given the option, will use Q-Labs EMS to find potentially useful packages rather than creating artifacts from scratch without benefit of prior experience.

The motivation behind these requirements is a combination of Q-Labs characteristics (geographically dispersed, variety of computing platforms, and employees under time pressures) and goals (better sharing of information and learning, high-level reuse of artifacts), and what we know about how Experience Factories are successfully deployed (data organization must be flexible and simple enough to accommodate future needs, users must be able to make use of the system with little overhead). These requirements are not as extensive as those elaborated in the literature [13], [43], but reflect our own group's experience and, more specifically, Q-Labs' primary concerns.

Based on these initial requirements and an ongoing collaboration with Q-Labs, the Experimental Software Engineering Group (ESEG) at the University of Maryland began working to build the infrastructure to support a true Experience Factory within Q-Labs [31], [39], [48]. There was a big gap, however, between these high-level requirements and a workable strategy for building a useful system. Many questions remained open. What is the best architecture to use for such a system? How should the "experience" be electronically represented and organized? What was needed of the user interface to make it acceptable to users? Most importantly, how can we keep the approach flexible to facilitate future evolution? Our initial answers to these questions, embodied in the initial prototype, are described in Section 5. Our method for evaluating some parts of our solution is described in Section 6.

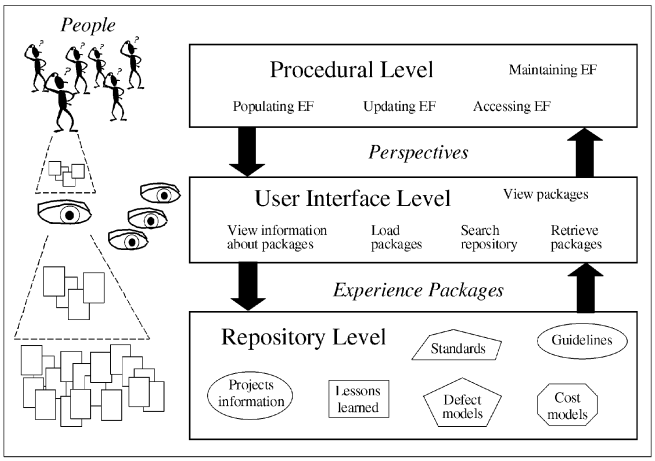


Fig. 2. The three levels of an Experience Management System.

4 Evaluation Criteria

The user interface for a system such as the Q-Labs EMS is a crucial component. The fifth requirement states that there must be no usability obstacles, clearly a weighty requirement. The long-term success of the Q-Labs EMS depends heavily on the willingness of users to start using it early on, thus providing both feedback on the contents of the repository and new experience packages. Further, the job of motivating early Q-Labs EMS users will depend largely on the user interface. Q-Labs EMS users will not be compelled to use the system because there are no critical activities that cannot be accomplished without it. The users must be motivated to use the Q-Labs EMS because the repository contains useful experience and because the interface is as easy to use as possible. The smallest usability obstacles would discourage early users and thus jeopardize the success of the system. Thus, we chose usability as the overriding evaluation criterion for the intial Q-Labs EMS prototype.

5 The Initial Prototype

We have found it useful to discuss the problem of software experience capture and reuse, and our approach to addressing it, in terms of the three-layer conceptual view shown in Fig. 2. This view shows three aspects of the problem, all of which need to be addressed before a complete solution can be implemented.

At the lowest level, there are issues of how experience should be electronically stored in a repository and made accessible across geographical boundaries. The middle-level deals with user interface issues, including how experiences are best presented to a user and how the user interacts with the system to manipulate, search, and retrieve experience. At the top level, the organizational issues of how experience reuse will fit into the work of the organization, how the experience base will be updated and maintained, and how experiences will be analyzed and synthesized over time, are addressed. The bottom two levels of Fig. 2 define the computer-intensive support pictured in Fig. 1. The top level of Fig. 2 defines the interface between the human-intensive and the computer-intensive areas.

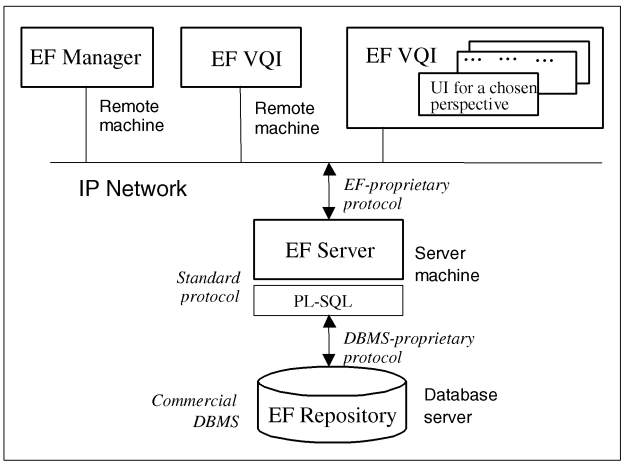


Fig. 3. Q-Labs EMS architecture.

5.1 System Architecture

In order to fulfill the first requirement presented in Section 3, to support geographically distributed organizations, the Q-Labs EMS is a client-server system. The architecture is shown in Fig. 3. It follows a three-tier model. At the top level, we have the EMS Manager and EMS Visual Query Interface (VQI) applications. They work as client applications sending requests to a "middle tier" of services. This EMS Server receives messages from the client applications and translates them into low-level SQL (Standard Query Language) calls to an EMS Repository. In order to fulfill the second and third requirements, the EMS Repository is managed by a commercial DBMS (Data Base Management System), and the client applications are implemented in Java.

5.2 Data Model

In order to fulfill our fourth requirement, we adopted the following data model. An experience package isdescribed by three parts: a characterization part used to classify packages in the experience base, a relationship part used to establish relations between packages, and a body part that contains the content of the experience package itself. Each package is associated with a **package type** that defines the type of **attributes** that will be used in its characterization part, the type of **links** that will be used in its relationship part, and the type of **elements** that will compose its body. In other words, each **package type** establishes a well-defined set of attributes for classification, links for establishing relationships, and elements for storing the contents of the instantiated packages.

Package elements are typed as a **file** or as a **list of files**. In order to facilitate the later usage of retrieved files, each file is kept associated with its original name extension. This way, the retrieved files can be opened directly by the client machine if its file extension is consistent with its OS registry.

Package attributes have well-defined naming and typing. These attributes build a classification taxonomy for each package type. The attributes effectively define facets that are filled in by the author of an experience package to characterize the package being submitted to the repository. These attribute values are then used to search the repository for packages of interest. In this respect, our approach is similar to Prieto-Diaz's faceted classification schema for software reuse [35]. Package attributes are typed as numbers, strings, dates, or a list of one of those. Package links also have well-defined naming and typing. As they are used to establish relationships between packages, a package link can be typed as a **pointer to** a package of a certain **package type**, or a list of them. The system also supports URL addresses as pointers to "external" packaged information. For this reason, a link can also be typed as a URL address or a list of them.

A package is created by giving values to the attributes, links, and elements defined by its package type. This arrangement provides a simple yet powerful data model, as per requirement 4. An example of a package type and some package instantiations are shown in Fig. 4. In this example, a package type named "Document" is shown along with its *attributes*, *links*, and *elements* in the top-left dashed box of the diagram. A package named "Document 10" in which all those parameters are instantiated is shown in the bottom-left box of the diagram. Another package named "Consultant 12" is shown to exemplify the instantiation of the link "Produced by" of the package type "Document." The actual content of these packages is contained in the files listed in the "Body" portion of each package. Each package type defines how its content is organized among its files. For example, the package type Document in Fig. 4 specifies that each document must consist of one file containing the abstract and another file or set of files containing the main text. A "Project" package type, on the other hand, might specify in its Body definition, that the "content" of a "Project" consists of files containing a "project plan," a set of "deliverables," and a "final report."

An overriding design consideration for the data model, and to some extent the architecture, was flexibility and generality. From a research perspective, our main interest in this initial prototype was the user interface, described below, so we chose a simple design for the underlying technology in order to isolate and concentrate on user interface aspects. Also, we wanted to create a prototype that could later serve as a testbed for other data organization and architectural paradigms. For example, goal-based and similarity-based retrieval [46] could easily be incorporated into our basic, attribute-based search platform without changing the basic design.

5.3 Visual Query Interface

As explained earlier in Section 4, the usability requirements for a system such as the Q-Labs EMS are heavy. To fulfill this crucial requirement, we adopted a visual query interface (VQI) concept. As proposed by Shneiderman [41], visual query interfaces let users "fly through" stored information by adjusting query devices (checkboxes and slider bars) and viewing animated results on the computer screen. In the Q-Labs EMS, VQI's allow easy interactive querying of the repository based on multiple attributes of the experience packages. Built in to the interface is the set of attributes defined for the perspective currently being viewed. Upon login, a user will have a set of perspectives from which he/she can look at stored experience packages. A user will fire a VQI by selecting one of those perspectives. The VQI will display the packages that are associated with

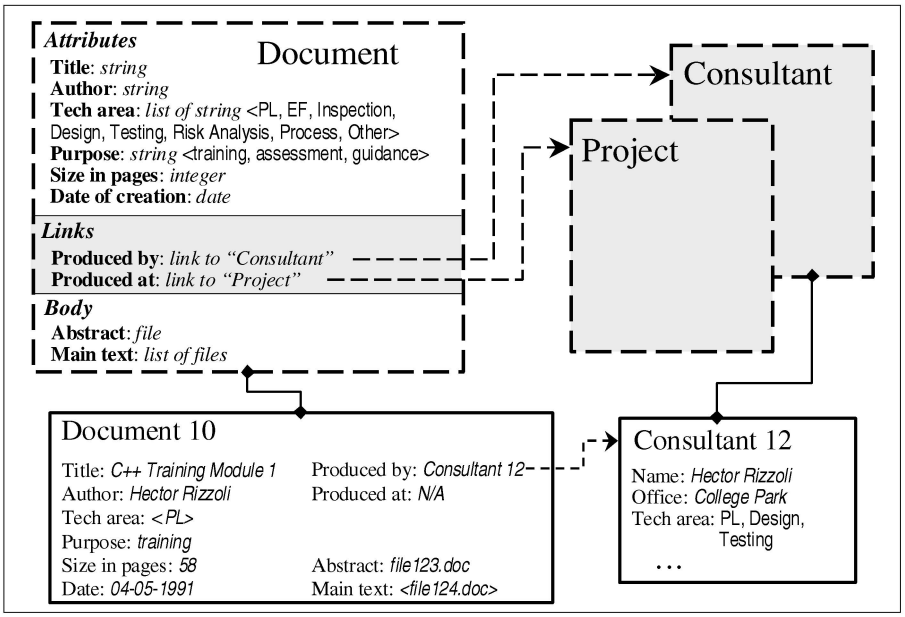


Fig. 4. Examples of Package Types and Package Instantiations.

this perspective together with all the attributes and query devices used to search and browse those packages. Fig. 5 shows such a VQI. A query device is defined on the fly and placed on the VQI for each attribute, based on the data type and number of different values associated with that attribute. Using the VQI, the user can interactively search the experience packages associated with a certain perspective by manipulating any combination of the query devices on the right and observing the number of selected packages (dots) on the two-dimensional grid on the left. Although only two dimensions are shown at a time on the grid, the user can use as many attributes as desired to perform the query that results in the grid display. Some of these query devices are slider bars, which can be used to narrow down the range of values of the attribute. Other query devices are check boxes, where the user checks off the values in which they are interested. The grid has two axes, each of which corresponds to one of the attributes in the perspective. The user can choose which two attributes to use on the X and Y axes and can change them at any time, resulting in a real-time update of the grid. This allows the user to view a set of packages in a variety of different ways very quickly. Once a small subset of packages is selected, the user can quickly examine a specific package by clicking on the corresponding dot in the grid. This will fire a Web page with a complete description of the selected package, including its links and elements. If the selected package corresponds to the user's expectations, he/she can click on the desired elements to retrieve the files containing the content of the package.

The VQI has two features that we believe are fundamental to the Q-Labs EMS. First, its search is interactive and controlled by the user. This allows the user to easily control the number of matches by widening or narrowing the search scope with a few mouse clicks on the VQI's query devices. This is a clear advantage over keyword-based search—such as those executed by World Wide Web search engines. This approach is also infinitely flexible in that the user has full control over which and how many attributes to use and may experiment at will with different combinations of attributes. The user does not a priori have to assign any sort of weights to the attributes. We hypothesize that this will significantly help users to find packages that are useful to them even when an exact match is not available.

The second key feature of this type of interface is that it allows people to visualize the amount of stored experience and the classification schema used by the organization. We believe that this will significantly help new users to get used to the Q-Labs EMS and is also an important learning medium for new team members.

The user interface also has functionality for submitting new experience packages to the experience base. This functionality uses the attributes, links, and elements associated with the perspectives to produce the simple forms that a user must complete to describe new packages.

6 Interface Protoype Evaluation

The initial prototype, which focused on the user interface, consisted of the VQI (pictured in Fig. 5), a simple data entry interface used to submit experience packages and a small repository populated with a collection of real Q-Labs documents and project descriptions. Two perspectives were also provided with this prototype. The *documents* perspective used attributes of documents (e.g., author, date, title, etc.) as the search mechanisms, while the *projects* perspective used attributes of projects (e.g., project lead, customer, start date, finish date, total effort, etc.). Some attributes were common to both perspectives (e.g., technical area). The evaluation was carried out at this point in the project (before having a full working system) because it was essential to get user feedback on the basic paradigms we had chosen before we proceeded further. The goal, then, of

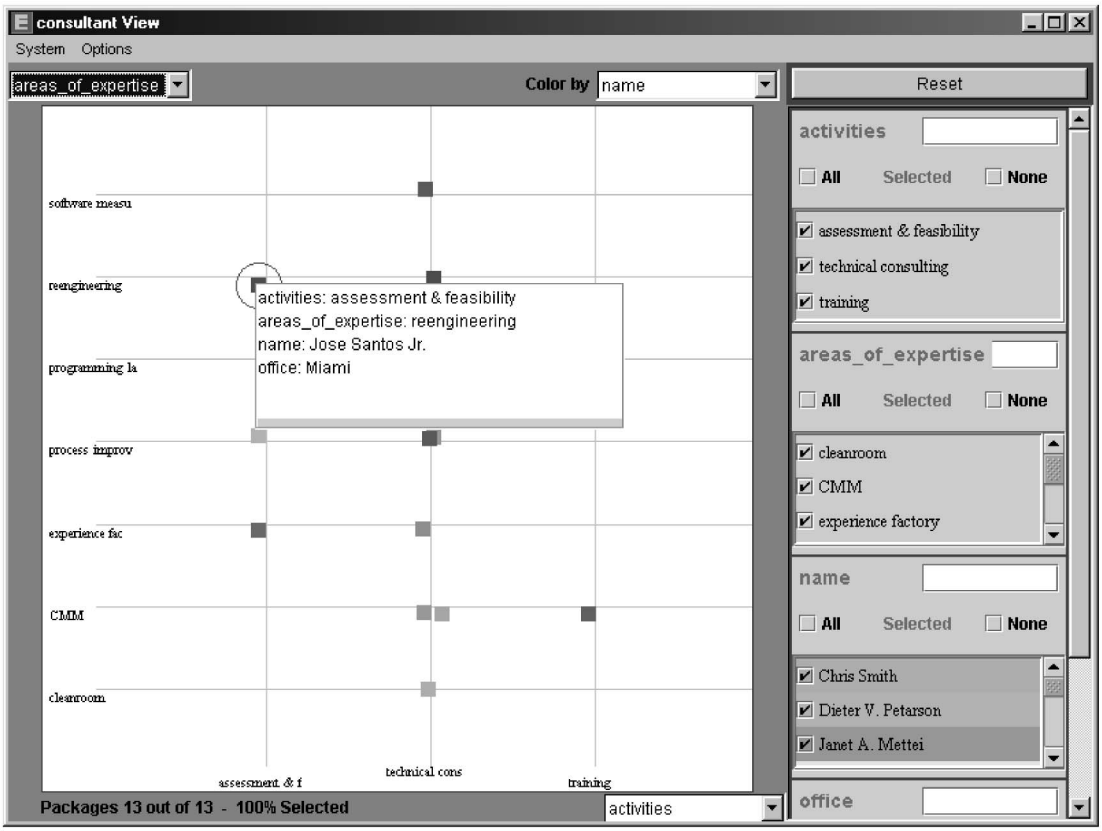


Fig. 5. Q-Labs Visual Query Interface (VQI).

this evaluation was to determine if the current prototype was meeting the requirements (in particular R4 and R5) and would be able to support Q-Labs' goals.

6.1 Study Design

The interface evaluation study was based on qualitative methods [19], [32]. The importance of such methods in validating software engineering technology is discussed by Seaman in [40]. The specific objectives of the interface evaluation study were:

1. To evaluate the current set of attributes (in both the "projects" and "documents" perspectives) in terms of completeness, usefulness, and clarity.
2. To evaluate the visual query search and retrieval interface in terms of usefulness, usability, and appropriateness.
3. To evaluate the data entry interface in terms of feasibility, usability, and impact on working procedures.

These goals were refined into a set of questions that guided the design of the study. To answer these questions, two types of data were collected. The first data source consisted of detailed notes concerning how the subjects used the prototype and the comments they made while using it. The second data source came from a set of interviews that were conducted at the end of each evaluation session.

Interface evaluation sessions were held with five different Q-Labs consultants (none of whom had previously been involved in the EMS project) from three different offices over the course of two months. In each session, the subject was given a short hands-on training by one of the researchers, then given a set of exercises that represented common Q-Labs work scenarios. The exercises were taken from the set of use cases we had collected as part of the initial requirements gathering activity for the Q-Labs EMS. Some examples of the exercises used are shown below:

- You are planning a project, which you will be managing, that is conducting a CMM self-assessment for a customer. Everyone has told you that these types of projects are difficult to manage, so you would like to avoid as many problems as possible. Search the experience repository for any type of lessons learned that might help you prepare for this project.
- A potential customer has called and asked for a detailed list of CMM-related services that Q-Labs can offer. Search the experience repository for materials that might help you compile such a list.

The subjects were asked to choose some of the exercises (each subject went through about three to four of the 15 exercises provided) and then to use the Q-Labs EMS prototype to gain information relevant to the scenario described in each exercise, while being observed by one of the researchers. They were also asked to verbalize their thoughts and motivations while working through the exercises. This technique, called a "think aloud" protocol [21], is often used in usability studies

(and occasionally in other software engineering studies [45]) to capture a subject's immediate impressions, thought processes, and motivations while performing a task. The subjects could and did ask questions of the researcher conducting the session.

After several exercises had been completed, a short interview was conducted by the researcher. Examples of some of the questions used in the interviews are:

- What did you like most about the search and retrieval interface?
- Was there anything really annoying about using it?
- What attributes did you use most in searches?
- What did you like most about the data entry interface?

All the sessions were audiotaped and observed by at least one researcher. Each session lasted about 1.5 to 2 hours. Although the tapes were not transcribed verbatim, they were used to write very detailed notes after the fact.

The notes written from the tapes served as the major data source for the analysis part of the study. This data was textual and unstructured and, thus, did not lend itself to transformation to any quantitative form. Therefore, the analysis method used was the constant comparison method [19], [32], a general approach widely used for analyzing all types of qualitative data. This method begins with coding the field notes by attaching codes, or labels, to pieces of text that are relevant to a particular theme or idea that is of interest in the study. Then, passages of text are grouped into patterns according to the codes and subcodes they've been assigned. These groupings are examined for underlying themes and explanations of phenomena. The next step is the writing of a field memo that articulates a proposition (a preliminary hypothesis to be considered) or an observation synthesized from the coded data. In this case, the field memo written as part of this process became the results of the study, which are reported in the next section.

Like any empirical study, this evaluation study involves some validity issues. For example, the external validity of the study is significantly compromised by the fact that all subjects came from the same organization. However, this was consistent with the goals of the study, which were to evaluate the EMS prototype for use at Q-Labs. Within this context, external validity could be seen to be quite high as the subjects came from three different Q-Labs offices. However, the sample size was small. Internal validity, in the context of this study, is mainly concerned with the accuracy of the data and conclusions. This issue is addressed through triangulation, or the use of data from different sources or gained through different means to produce the same conclusions. The study design includes data collected via observation, think-aloud protocols, and interviews. Construct validity is quite strong, as the data was collected during and immediately after actual use of the prototype. Thus, the data collected is strongly believed to accurately describe the phenomen being studied.

6.2 Results

The subjects generally liked the basic elements of the search and retrieval interface. In particular, they seemed to have no trouble mastering the search mechanism and liked how it was easy to negotiate the interface and see the distribution of packages among different attribute values. They also liked the immediate feedback in the graph part of the interface in response to changes made with the search mechanisms. Subjects also said that they felt they were able to find what they were looking for (if it existed in the repository) with a minimal number of steps.

Aside from using the basic elements of the search and retrieval interface, subjects were also able to glean useful information from the interface even when they couldn't find exactly what they were looking for. For example, one subject found a document that was not exactly what she wanted, but she saw the primary author's name and decided that would be a good contact and, so, she felt she had found useful information. In another case, a subject was looking for a tender (an estimate submitted to a potential customer), couldn't find it, but found a final report that would be useful.

Several major annoyances surfaced during the evaluation. One was the use of slider bars. Several subjects had trouble figuring out the mechanics of using and interpreting them. It was hard to tell by looking at the slider bar which values were currently selected and which were not. In response to this annoyance, several subjects suggested using some form of checkboxes instead of the slider bars.

Another general annoyance subjects expressed had to do with the relationship between the two perspectives (documents and projects) and the lack of linkage between them. After finding some relevant project in the projects perspective, subjects had to then start up the document perspective and start a search from scratch in order to find documents related to the project. Another problem was the confusion caused by some attributes and attribute values existing in one perspective but not the other.

As for the data entry interface, the data being collected was seen to be appropriate, but otherwise the interface left a lot to be desired. Subjects in general found the data entry interface unusable because they needed more guidance as to what attribute values to enter in the fields.Almost all of the subjects suggested pull-down menus listing the existing values for each attribute so that the user could just pick one, possibly with an option to enter a "new" value. This was seen as necessary to ensure consistency. Another suggestion was to have automatic fill-ins, where the value entered for one attribute would automatically determine the value for another attribute, or at least restrict the possible values.The "instructions" on the screen needed to be more helpful. In general, the subjects saw this interface as just a skeleton of what was needed.

Many comments were made about the attributes, including suggestions of new attributes to add (planned effort and schedule, project problems, and financial data), current attributes that are not useful (version_id), and attributes whose meanings were not clear (title, project effort, product_type, office, and language). Some attributes were suggested to be made multivalued (service_provided

and project problems). The attributes used most were product_type, service_provided, technical_area, customer_name, and customer_domain.

The learning curve on the search and retrieval interface was fairly short. By the second or third exercise tried, all of the subjects were conducting their searches very rapidly and confidently. For some subjects, it was even quicker. Subjects generally narrowed their searches down to about two to four hits before looking at individual packages. This was seen as a reasonable number of packages to look through.

Some features of the search and retrieval interface were not immediately apparent to the subjects. One subject got stuck in one perspective and did not think to switch to the other perspective until prompted by the tester. When they did switch, they were able to find the information they were looking for. The same thing happened several times with the X and Y axes. Subjects did not think to change the attributes shown on the axes until it was suggested by the tester, but once they did, they found it very useful. The subjects then seemed to remember to use the perspectives and the axes later, after they had done it once.

Some subjects also tended to make inferences from the data that was displayed. For example, one subject inferred from the value for project effort that the project she was looking at was much larger than she was interested in, so she decided to look for another package. In another case, a group of dots clustered around a particular date in the documents perspective suggested to one subject that they were all from the same project.

Several subjects said that the data entry interface would save them time only if it replaced other processes in which they entered the same data. The key was to reduce redundancy, which included using pull-down menus so that a value does not actually have to be typed more than once. One subject thought that probably the Q-Labs EMS would not be used on a daily basis, but for exploratory work and for starting new projects.

The subjects offered a variety of ideas for enhancements to the prototype. One suggestion was a facility for saving searches, so that the same search could be re-executed later without having to manipulate the query devices again. Another suggestion was to show a simple count of the number of dots currently being displayed on the grid, so that the user would know how many "hits" the current search criteria had found. Another idea was to provide a "zoom" feature on the axes of the grid so that only a subset of the values of the attribute for that axis would be shown.

6.3 Discussion

In general, the search and retrieval interface (VQI) proved to be very easy to use and useful for finding information in the repository. There were some significant problems that hindered its usefulness, but the basic VQI paradigm was well received. The data entry interface, on the other hand, was generally found to be unsatisfactory. It was found to be clumsy and the information requests too ambiguous. However, the subjects seemed to agree on the features that needed to be added to make it useful. These were pull-down menus and providing better instructions on the screen.

These observations raise several deeper questions about the design, not only of the user interface, but the underlying data model as well. The idea of perspectives, as we had originally envisioned it, assumed that perspectives would be largely distinct. Although we allowed for the possibility of a package being accessible through more than one perspective, we had assumed that users would generally only use one perspective at a time, and would not need to switch between perspectives frequently. That turned out not to be the case. This implies that a useful enhancement to the current prototype would be a tighter and easier to navigate coupling between different perspectives. One way to accomplish this, which is being investigated, is to use the relationships part of each package (which contains links to other packages) to allow users to jump more easily between perspectives.

Another design issue that arose from the evaluation study was the scheme we had designed for package attributes. In order to enforce consistency between names of attributes in different perspectives, we created a general pool of attributes, each with a name and type, and then decided a priori which attributes were appropriate for each perspective. This seemed to cause some confusion for users, when they expected to see the same attributes in different perspectives. This raises the question of who should decide which attributes should be applicable to which perspectives, and when this should be decided. On one hand, the user could be allowed (i.e., required) to specify which attributes belong in which perspectives, and to modify the scheme at any time, adding and deleting attributes on the fly. At the other end of the spectrum, all attributes and their assignment to perspectives could be determined a priori without the possibility of later modification. The former scheme allows maximum flexibility, while the latter provides greater simplicity and consistency across the organization.

This evaluation provided valuable feedback that has been used in our plans for further development of EMS for Q-Labs and other organizations. Although we knew that the interface we were evaluating was not ideal, we had not anticipated some of the specific problems that our subjects reported. For example, we had not considered the slider bar mechanism to be a problem, but our subjects definitely did. Also, although we knew the data entry interface needed some improvements (many of the suggestions from the subjects were already in our development plans), we had not considered it as completely unusable as our subjects did. In addition, the subjects had some creative ideas for enhancements that we had not considered. On the other hand, the study validated some of our basic choices in the interface design, e.g., the VQI and the use of attributes and perspectives. Thus, we can, with confidence, continue improvement of the interface without changing the underlying structure.

There were also some lessons learned about how the interface evaluation was conducted. Some problems came up related to the limited scope of the repository. Subjects were sometimes frustrated when there was nothing to be found for their search criteria. Subjects were also bothered by inconsistencies in the sample data. In particular, one subject found that there was a document in the documents perspective that had a project name associated with it, but

that project was not to be found in the projects perspective. He felt that the data in the different perspectives should be consistent, like a "transparent cube." If you turn the cube, you want to see the same thing from a different perspective. The same subject also found a document for which he was the primary author, but the primary author attribute was "n/a," which he thought was strange. There were also some problems with the exercises that were used, but these were avoided for the most part after the first session.

The interface evaluation, in general, proved to be a valuable and timely tool for getting feedback from the eventual users of the Q-Labs EMS. The effort involved was relatively small, although finding and scheduling subjects was difficult and caused some delays. Although much remained to be done before an operational system could be delivered, the evaluation assured us that the Q-Labs EMS would eventually be acceptable to its intended users. In addition, the evaluation provided an opportunity to disseminate the aims of our project, and our work thus far, throughout Q-Labs.

As it turned out, due to other circumstances, the Q-Labs EMS was not adopted at Q-Labs. However, our experience with the Q-Labs EMS proved to be valuable input into what is now a diverse and active area of research involving several research and industrial partners at the Fraunhofer Center-Maryland [11], [27]. In particular, the methods used in this evaluation have now become part of the standard, continuous evaluation of other EMS prototypes.

7 Evolution of EMS

The interface evaluation described in Section 6 validated some of our assumptions about what features the interface for such a system should have, but also raised some issues that we had not considered previously. Some of the subjects' suggestions are easily incorporated as enhancements to the user interface, but others touch on underlying design issues that need to be investigated further. For example, what is the proper balance between user customizability and simplicity? Should users be able to create and modify attributes on the fly, for example, or should they be specified a priori by those setting up the experience system? Architecturally, this has implications for the underlying data model. It also raises questions about the role of users of an EMS in the process of capturing, analyzing, and reusing experience. Should the average user be expected to understand the entire taxonomy of attributes and perspectives that describe the contents of the repository, or should their view be limited to what has been deemed of interest to them? These questions and others are being considered as new instances of the EMS are being developed [27].

Parallel and subsequent to the work with Q-Labs, researchers at the University of Maryland and at the Fraunhofer Center for Experimental Software Engineering have significantly evolved the idea of an Experience Management System [11]. EMS is now seen as a logical architecture, a set of tools, and a well-defined methodology to tailor the system to a particular organization. The latter includes a process for designing, implementing, and evaluating new organizational procedures and deployment strategies to ensure the acceptance of the system. The tool currently employed includes:

1. the VQI and other interfaces for navigation of links and keyword search,
2. explicit support for different Experience Factory roles,
3. tools for user cooperation and socialization,
4. tools for transforming tacit knowledge into explicit knowledge, and
5. middleware software for accessing information already stored in corporate databases [11], [27].

The VQI paradigm is still in use for the user interface to the experience package repository, but it has been enhanced as well. It still allows the user to employ any number of attributes for searching, but now incorporates more of them at one time into the grid part of the display by allowing the user to code the color, size, and shape of the dots with the values of selected attributes.

In this scenario, the Q-Labs EMS is now seen as an instantiation of this EMS architecture, that uses the VQI as its central tool. Its main merit was to serve as the original testbed for evaluating hypotheses, ideas, and tools for experience management systems in industrial settings. The Q-Labs EMS was our first crucial step in this direction. The ideas it generated laid the groundwork for the EMS architecture and spun off several other EMS instantiations [11], [27]. But, most importantly, it instantiated the idea of continuously evolving a system, with a flexible architecture and built-in repeatable evaluation processes, to better reflect user needs.

8 Summary

We have described a method that was aimed at instantiating and evolving a system that is highly dependent upon evolving through use. It consists of unifying a set of concepts that relate to the problem, selecting a flexible component independent architecture, and evolving the requirements based upon experimentation in the form of feedback loops. This effort represents a collaboration between the Experimental Software Engineering Group (ESEG) at the University of Maryland and Q-Labs, Inc. that aimed to provide a system to support software engineering experience capture and reuse. We have described the overall process we used to design and evaluate our initial prototype, with an emphasis on empirical and experimental study. The architecture, data structure, and user interface of the prototype Q-Labs EMS are described. The evaluation of this interface prototype is presented in detail. The user interface was chosen as the focus of the initial prototype and evaluation because it is such a crucial part of systems that support EM. During early use, not only must users be strongly motivated to use these systems, but all conceivable barriers to their use must be removed, especially barriers related to usability. Any inadequacies of the user interface will directly impact the early use of any type of experience reuse repository and, thus, greatly hinder its eventual success.

The results of the evaluation assured us not only that the Q-Labs EMS could eventually be successfully deployed throughout Q-Labs or other organizations, but also that it

could serve as a testbed for further investigation of EM. However, it also showed us that much needs to be done before a robust version of such a system is in place. The prototype that has been evaluated encompassed only some of the automated features that are required from a system aimed at supporting an Experience Factory.

ACKNOWLEDGMENTS

The authors would like to thank the consultants at Q-Labs for so kindly spending their valuable time to make this study possible. The authors also recognize the invaluable contributions from Jon Valett (at Q-Labs), Marvin Zelkowitz and Mikael Lindvall (at Fraunhofer Center MD), and Baris Aydinlioglu (at the University of Maryland) to this work. Thanks also go to the anonymous reviewers of various versions of this paper. This research was supported in part by Maryland Industrial Partnerships (MIPS) grant no. 2015.22. Dr. Mendonça would also like to thanks CNPq—Brazil's National Council for Scientific and Technological Development—for supporting his work with the Research Productivity Scholarship no. 300355/00-9. Ms. Kim was employed by Q-Labs at the time this work was conducted.

REFERENCES

[1] K.-D. Althoff, A. Birk, S. Hartkopf, W. Muller, M. Nick, D. Surmann, and C. Tautz, "Systematic Population, Utilization, and Maintenance of a Repository for Comprehensive Reuse," *Proc. Workshop Learning Software Organizations: Methodology and Applications (SEKE '99),* pp. 25-50, June 1999.

[2] K.-D. Althoff, M. Nick, and C. Tautz, "Systematically Diagnosing and Improving the Perceived Usefulness of Organizational Memories," *Proc. Workshop Learning Software Organizations: Methodology and Applications (SEKE '99),* pp. 72-86, June 1999.

[3] K.-D. Althoff, B. Decker, S. Hartkopf, A. Jeditschka, M. Nick, and J. Rech, "Experience Management: The Fraunhofer IESE Experience Factory," *Proc. Industrial Conf. Data Mining,* July 2001.

[4] K.-D. Althoff, U. Becker-Kornstaedt, B. Decker, A. Klotz, E. Leopold, J. Rech, and A. Voss, "Enhancing Experience Management and Process Learning with Moderated Discourses: The indiGo Approach," *Proc. European Conf. Artificial Intelligence (ECAI '02) Workshop Knowledge Management and Organizational Memory,* 2002.

[5] C. Bailey and M. Clarke, "Managing Knowledge for Personal and Organisational Benefit," *J. Knowledge Management,* vol. 5, no. 1, pp. 58-67, 2001.

[6] V.R. Basili, "Software Development: A Paradigm for the Future," *Proc. COMPSAC '89,* pp. 471-485, Sept. 1989.

[7] V.R. Basili, "The Experience Factory and its Relationship to Other Improvement Paradigms," *Proc. Fourth European Software Eng. Conf. (ESEC),* Sept. 1993.

[8] V.R. Basili and G. Caldiera, "Improve Software Quality by Reusing Knowledge and Experience," *Sloan Management Rev.,* vol. 37, no. 1, Fall 1995.

[9] V.R. Basili, G. Caldiera, F. McGarry, R. Pajerski, G. Page, and S. Waligora, "The Software Engineering Laboratory—An Operational Software Experience Factory," *Proc. Int'l Conf. Software Eng.,* pp. 370-381, May 1992.

[10] V.R. Basili, M. Daskalantonakis, and R. Yacobellis, "Technology Transfer at Motorola," *IEEE Software,* pp. 70-76, Mar. 1994.

[11] V.R. Basili, M. Lindvall, and P. Costa, "Implementing the Experience Factory Concepts as a Set of Experience Bases," *Proc. 13th Int'l Conf. Software Eng. and Knowledge Eng.,* pp. 102-109, June 2001.

[12] M. Brandt and M. Nick, "Computer-Supported Reuse of Project Management Experience with an Experience Base," *Proc. Advances in Learning Software Organizations,* pp. 178-189, Sept. 2001.

[13] M. Broome and P. Runeson, "Technical Requirements for the Implemenation of an Experience Base," *Proc. Workshop Learning Software Organizations: Methodology and Applications (SEKE '99),* pp. 87-102, June 1999.

[14] R. Baronide Carvalho and M.A.T. Ferreira, "Using Information Technology to Support Knowledge Conversion Processes," *Information Research,* vol. 7, no. 1, 2001, also available at http://InformationR.net/ir/7-1/paper118.html.

[15] R.L. Chase, "Knowledge Management Benchmarks," *J. Knowledge Management,* vol. 1, no. 1, pp. 83-92, Sept. 1997.

[16] T.H. Davenport, D.W. De Long, and M.C. Beers, "Successful Knowledge Management Projects," *Sloan Management Rev.,* pp. 43-57, Winter 1998.

[17] T. Dingsoyr, "An Evaluation of Research on Experience Factory," *Proc. Workshop Learning Software Organizations (PROFES 2000),* pp. 55-66, June 2000.

[18] R.L. Feldmann, M. Nick, and M. Frey, "Towards Industrial-Strength Measurement Programs for Reuse and Experience Repository Systems," *Proc. Workshop Learning Software Organizations (PROFES 2000),* pp. 7-18, June 2000.

[19] B.G. Glaser and A.L. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research.* Aldine Publishing Company, 1967.

[20] S. Greene, E. Tanin, C. Plaisant, B. Shneiderman, L. Olsen, G. Major, and S. Johns, "The End of Zero-Hit Queries: Query Previews for NASA's Global Change Master Directory," *Int'l J. Digital Libraries,* vol. 2, nos. 2 and 3, pp. 79-90, 1999.

[21] J.T. Hackos and J.D. Redish, *User and Task Analysis for Interface Design.* John Wiley and Sons, chapter 9, pp. 258-259, 1998.

[22] F. Houdek, K. Schneider, and E. Wieser, "Establishing Experience Factories at Daimler-Benz: An Experience Report," *Proc. 20th Int'l Conf. Software Eng.,* pp. 443-447, Apr. 1998.

[23] T. Käpylä, I. Niemi, and A. Lehtola, "Towards an Accessible Web by Applying PUSH Technology," *Proc. Fourth ERCIM Workshop, User Interfaces for All,* pp. 19-21, Oct. 1998, also available at http://www.vtt.fi/tte/samba/projects/ida-push/Ercim_WS_UI_paper.htm.

[24] T. Kotnour, C. Orr, J. Spaulding, and J. Guidi, "Determining the Benefit of Knowledge Management Activities," *Proc. IEEE Int'l Conf. Systems, Man, Cybernetics,* pp. 94-99, Oct. 1997.

[25] G. Lawton, "Knowledge Management: Ready for Prime Time?" *Computer,* vol. 34, no. 2, pp. 12-14, 2001.

[26] J. Liebowitz and T. Beckman, *Knowledge Organizations: What Every Manager Should Know.* Washington: St. Lucie Press, 1998.

[27] M. Lindvall, M. Frey, P. Costa, and R. Tesoriero, "An Analysis of Three Experiences Bases," *Learning Software Organizations,* 2001.

[28] M. Lindvall, I. Rus, and S. Sinha, "Technology Support for Knowledge Management," *Proc. Fourth Int'l Workshop Learning Software Organizations (LSO '02),* Aug. 2002.

[29] R. Mack, Y. Ravin, and R.J. Byrd, "Knowledge Portals and the Emerging Digital Knowledge Workplace," *IBM Systems J.,* vol. 40, no. 4, pp. 925-955, 2001.

[30] A.D. Marwick, "Knowledge Management Technology," *IBM Systems J.,* vol. 40, no. 4, pp. 814-830, 2001.

[31] M. Mendonça, C.B. Seaman, V. Basili, and Y.-M. Kim, "A Prototype Experience Management System for a Software Consulting Organization," *Proc. 13th Int'l Conf. Software Eng. and Knowledge Eng.,* pp. 29-36, June 2001.

[32] M.B. Miles and A.M. Huberman, *Qualitative Data Analysis: An Expanded Sourcebook,* second ed. Thousand Oaks, Calif: Sage, 1994.

[33] G. Miller, "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information," *Psychological Rev.,* vol. 101, no. 2, pp. 343-352, Apr. 1994.

[34] I. Nonaka and H. Takeuchi, *The Knowledge Creating Company.* Oxford Univ. Press, 1995.

[35] R. Prieto-Diaz, "Classifying of Reusable Modules," *Software Reusability,* T.J. Biggerstaff and A. Perlis, eds., vol. I, 1990.

[36] R. Roy, F.M. del Rey, B. van Wegen, and A. Steele, "A Framework to Create Performance Indicators in Knowledge Management," *Proc. Third Int'l Conf. Practical Aspects of Knowledge Management (PAKM2000),* pp. 18-1:18-8, Oct. 2000.

[37] G. Ruhe and F. Bomarius, "Introduction and Motivation," *Proc. Workshop Learning Software Organizations: Methodology and Applications (SEKE '99),* pp. 3-22, June 1999.

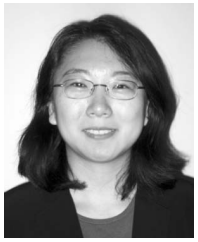
[38] K. Schneider, J. von Hunnius, and V.R. Basili, "Experience in Implementing a Learning Software Organization," *IEEE Software,* pp. 46-49, June 2002.

[39] C.B. Seaman, M. Mendonca, V. Basili, and Y.-M. Kim, "An Experience Management System for a Software Consulting Organization," *Proc. Software Eng. Workshop,* NASA/Goddard Software Eng. Laboratory, Dec. 1999.
[40] C.B. Seaman, "Qualitative Methods in Empirical Studies of Software Engineering," *IEEE Trans. Software Eng.,* vol. 25, no. 4, pp. 557-572, July/Aug. 1999.
[41] B. Shneiderman, "Dynamic Queries for Visual Information Seeking," *IEEE Software,* vol. 6, no. 11, pp. 70-77, Nov. 1994.
[42] D.J. Skyrme and D.M. Amidon, "New Measures of Success," *J. Business Strategy,* pp. 20-24, Jan./Feb. 1998.
[43] C. Tautz, "Customizing Software Engineering Experience Management Systems to Organizational Needs," PhD dissertation, Dept. of Computer Science, Univ. of Kaiserslautern, Germany, 2000.
[44] A. Tiwana, *The Knowledge Management Toolkit: Practical Techniques for Building Knowledge Management Systems.* Prentice Hall PTR, 2000.
[45] A. von Mayrhauser and A.M. Vans, "Identification of Dynamic Comprehension Processes During Large Scale Maintenance," *IEEE Trans. Software Eng.,* vol. 22, no. 6, pp. 424-437, June 1996.
[46] C.G. von Wangenheim, K.-D. Althoff, and R.M. Barcia, "Goal-Oriented and Similarity-Based Retrieval of Software Engineering Experienceware," *Proc. Workshop Learning Software Organizations: Methodology and Applications (SEKE '99),* pp. 118-141, June 1999.
[47] C.G. von Wangenheim and M.R. Rodrigues, "Case-Based Management of Software Engineering Experienceware," *Proc. IBER-AMIA-SBIA 2000,* pp. 12-22, 2000.
[48] R. Webby, C. Seaman, M. Mendonça, V.R. Basili, and Y. Kim, "Implementing an Internet-Enabled Software Experience Factory: Work in Progress," *Proc. Second Workshop Software Eng. over the Internet (ICSE '99),* May 1999.

Carolyn B. Seaman received the PhD degree in computer science from the University of Maryland, College Park, a MS degree in information and computer science from Georgia Tech, and the BA degree in computer science and mathematics from the College of Wooster (Ohio). She is an assistant professor of information systems at the University of Maryland, Baltimore County (UMBC). Her research generally falls under the umbrella of empirical studies of software engineering, with particular emphases on maintenance, organizational structure, communication, measurement, COTS-based development, and qualitative research methods. Dr. Seaman is also a research scientist at the Fraunhofer Center for Experimental Software Engineering, Maryland, where she participates in research on experience management in software engineering organizations and software metrics. She has worked in the software industry as a software engineer and consultant, and has conducted most of her research in industrial and governmental settings (e.g., IBM Canada Ltd., NASA).

Manoel G. Mendonça received the PhD degree in computer science from the University of Maryland in 1997. He also received the MSc degree in computer engineering from the State University of Campinas (1990), and the BSEE degree in electrical engineering from the Federal University of Bahia (1986), both in Brazil. He was a visiting scientist and was awarded a doctoral fellowship from the IBM Toronto Laboratory's Centre for Advanced Studies. Between 1997 and 2000, he worked as a faculty research associate at the University of Maryland and a scientist at the Fraunhofer Center for Experimental Software Engineering also in Maryland. He is now a professor at Salvador University (UNIFACS) in Brazil. He has been involved in projects for IBM, Q-Labs, and Siemens, among other organizations. Currently, he is the chief consultant for a countrywide data gathering and analysis program for the Brazilian Oil Agency (ANP). He is also one of the scientists of the Readers Project, a collaborative research effort on software defect detection techniques sponsored by NSF and CNPq-Brazilás National Council for Scientific and Technological Development. His main research interests are data mining, experimental software engineering, and knowledge management support systems.

Victor R. Basili is a professor of computer science at the University of Maryland, College Park, and the Executive Director of the Fraunhofer Center-Maryland. He works on measuring, evaluating, and improving the software development process and product. He is a recipient of several awards including a 1989 NASA Group Achievement Award, a 1990 NASA/GSFC Productivity Improvement and Quality Enhancement Award, the 1997 Award for Outstanding Achievement in Mathematics and Computer Science by the Washington Academy of Sciences, the 2000 Outstanding Research Award from ACM SIGSOFT, and the 2003 IEEE Computer Society Harlan D. Mills Award. Dr. Basili has authored more then 160 journal and refereed conference papers, served as editor-in-chief of the *IEEE Transactions on Software Engineering*, and as program chair and general chair of the Sixth and 15th International Conferences on Software Engineering, respectively. He is co-editor-in-chief of the *International Journal of Empirical Software Engineering*, published by Kluwer. He is an IEEE and ACM fellow.



Yong-Mi Kim received the MS degree in computer science from the University of North Carolina at Chapel Hill. Subsequently, she was part of the Experimental Software Engineering Group at the University of Maryland, College Park, before joining Q-Labs. At Q-Labs she worked on software process improvement projects for companies such as Ericsson. She is currently a master's candidate in the School of Information at the University of Michigan. Her research areas are information retrieval and information seeking behavior.