A Prototype Experience Management System for a Software Consulting Organization

Manoel Gomes de Mendonça Neto
Computer Networks Research Group (NUPERC)
Salvador University (UNIFACS)
Av. Cardeal da Silva 747
Salvador, Ba, Brazil, 40220-141
+ 55 (71) 247-5177
mgmn@unifacs.br

Carolyn B. Seaman*
Department of Information Systems
UMBC
1000 Hilltop Circle
Baltimore, MD 21250 USA
+1 410 455 3937
cseaman@umbc.edu

Victor Basili*
Dept. of Computer Science
University of Maryland College Park
A. V. Williams Building
College Park, MD 20742 USA
+1 301 405-2739
basili@cs.umd.edu

Yong-Mi Kim
Q-Labs Inc.
6301 Ivy Lane, Suite 210
Greenbelt, MD 20770 USA
+1 301 982-7773
Yong-Mi.Kim@q-labs.com

ABSTRACT

The Experience Management System (EMS) is aimed at supporting the capture and reuse of software-related experience, based on the Experience Factory concept. It is being developed for use in a multinational software engineering consultancy, Q-Labs. Currently, a prototype EMS exists and has been evaluated. This paper focuses on the EMS architecture, underlying data model, implementation, and user interface.

Keywords

Experience factory, knowledge management, experience reuse

1 INTRODUCTION

Software is a major expense for most organizations and is on the critical path to almost all organizational activities. Individual software development organizations in general strive to develop higher quality systems at a lower cost for both their internal and external customers. Yet the processes used to develop such software are still very primitive in the way that experience is incorporated. Learning is often from scratch, and each new development team has to relearn the mistakes of its predecessors. Reuse of an organization's own products, processes, and experience is becoming more accepted as 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 analyzed and synthesized so that members of the organization can learn from them and apply them to new problems.

Suppose, for example, that a member of a software development group is considering the use of a particular software engineering technology on a forthcoming project. This member has heard that this technology has been used successfully in other projects in some other part of the organization, but cannot easily find out where or by whom. He or she would like very much to learn from the experiences of those previous projects, first to help make the decision to use the technology or not, then to help implement the technology in the current project. It would be helpful, obviously, to avoid the inevitable mistakes that are made the first time a new technology is tried. Also, it would be useful to see the costs of using that technology (e.g. the costs of new tools or training) in order to help estimate those costs for the current project. Without the organizational infrastructure to support access to previous experience from within the organization, this type of information would be very difficult, if not impossible, for the development team member to get.

---

* Victor Basili and Carolyn Seaman are also with the Fraunhofer Center for Experimental Software Engineering – Maryland, University of Maryland, 4321 Hartwick Rd., Suite 500, College Park, MD 20742-3290, + 1 301 403-2705.

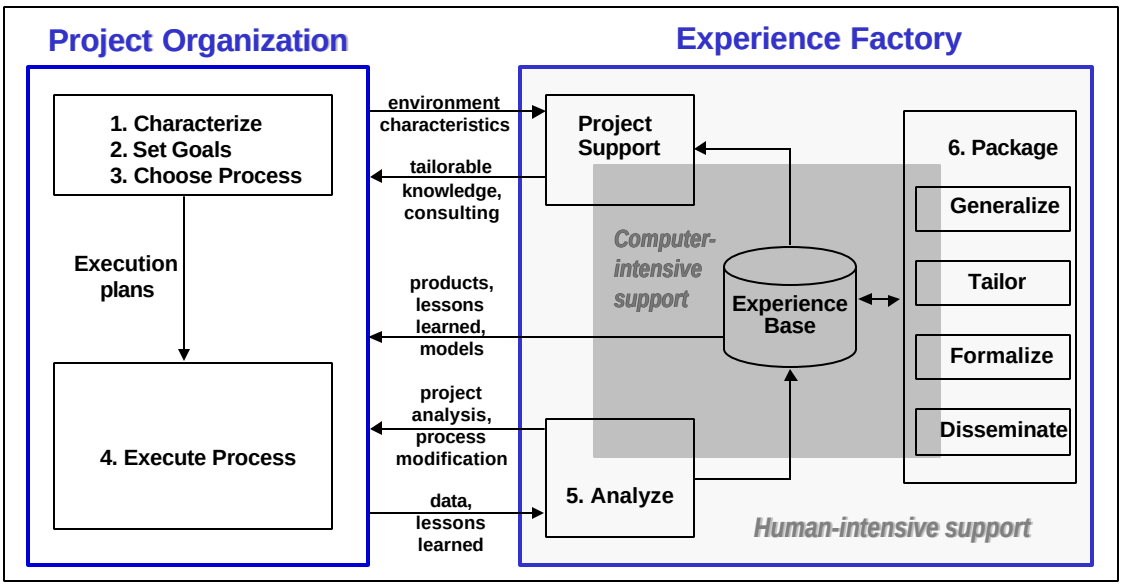

Figure 1. Experience Factory structure

The subject of this paper is a system for supporting experience management in a multinational software improvement consultancy called Q-Labs. This Experience Management System (EMS) is based on the Experience Factory [1,2,3] concept proposed by Basili et al.

2 UNDERLYING CONCEPTS

This section discusses some of the underlying concepts behind the Experience Management System and tries to place it in the state of art.

Knowledge and Experience Management

Tiwana [15] defines knowledge management (KM) as "the management of organizational knowledge for creating business value and generating a competitive advantage." Experience can be thought of as a type of organizational knowledge, in particular that knowledge gained as a result of the work of the organization itself. EMS aims to support the capture, storage, search, and retrieval of experiences that are (or can be) explicitly represented. That is, the EMS is concerned with explicit knowledge, as opposed to tacit knowledge (knowledge that lies inside the minds of individuals [15]). EMS supports very little knowledge externalization, i.e., transforming tacit knowledge into explicit knowledge. It supports storage and search of explicit knowledge related to previous business experiences of an organization. This supports knowledge internalization, i.e., helping people to absorb explicit knowledge.

Tiwana also presents a seven-layer model describing the different types of technology components necessary to support KM in an organization. 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. The 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 enabling technologies for knowledge management.

One of the crucial decisions to be made when building a KM infrastructure is the choice between using Web-based technology and using a proprietary set of tools such as Lotus Notes and its related applications. The EMS uses a combination of these approaches by utilizing an in-house developed application to manage the search interface, but using Web technology for the actual retrieval and presentation of artifacts from the repository. Another important choice that must be made when developing a KM infrastructure is the level of granularity of the "knowledge objects" in the system. In the case of EMS, the unit of information is called an "experience package" and its size and scope have to be defined by an experience classification manager or librarian. The minimum size of an experience package is a file, with a set of classification attribute values. Whether or not this is a reasonable lower limit for knowledge classification remains an unresolved issue, but will be investigated in future studies of different EMS prototypes.

Search strategies are also an important issue in designing and integrating a KM infrastructure. Tiwana describes four general types of searching: meta searching (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. 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 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.

One final distinction is between "push" and "pull" technologies, either (or both) of which can be utilized by a KM infrastructure. EMS, at this time, 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.

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. EMS uses a search and retrieval interface that is based on information visualization and visual data mining tools.

Humans have a poor short-term memory, i.e. they have limited ability to search and interpret textual and/or tabular data [10]. 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 non-zero hit queries in which the number of results satisfying a query can be visually previewed before any information is retrieved from experience repositories [6]. It also aids novices to learn by themselves about the organization's experience classification schema and its available body of knowledge.

The Experience Factory

The Experience Factory is an organizational infrastructure whose goal is to produce, store, and reuse experiences gained in a software development organization [1,2,3]. 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 Figure 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. [4,5,8]). The Software Engineering Laboratory (SEL) [4] is an 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 organization's 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%, decreased error rates by 85%, and increased reuse by over 300% over the past 15 years

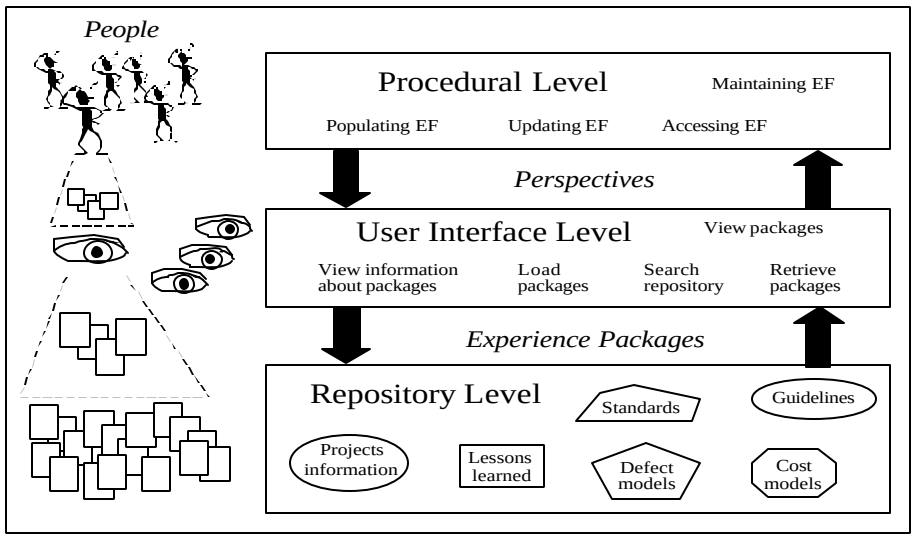

Figure 2. The three levels of an Experience Management System

Establishing an Experience Factory is a long-term endeavor requiring a great deal of commitment on the part of both management and development staff. Implementing an Experience Factory involves substantial up-front costs. It requires instilling a new philosophy of learning into an organization, and establishing an organizational structure and processes for the Experience Factory to collect, package and share experiences. Once in place, it will also require an ongoing effort and commitment to maintain as an effective agent for continuous software process improvement.

We believe that emerging computing technologies - such as distributed systems, visualization tools, visual query interfaces, and intranets - offer great potential to support the establishment and maintenance of Experience Factories in organizations. This paper reports on our recent efforts to exploit these technologies to support an Experience Factory within an industrial setting.

3 PRINCIPLES BEHIND EMS.

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 3-layer conceptual view shown in Figure 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 automated 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 Figure 2 define the computer-intensive support pictured in Figure 1. The top level of Figure 2 defines the interface between the human-intensive and the computer-intensive areas in Figure 1.

Allied with this conceptual view, we have defined a set of high-level requirements aimed at making the EMS reliable, easy to use, and flexible enough to support the Experience Factory concept.

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

R2. The repository shall be robust, reliable, and portable to standard computer platforms.

R3. The user interface level shall be as platform independent as possible, and allow for visual exploration of information.

R4. The data model shall be simple but powerful enough to model diverse classes of “experience packages.” The system will adapt to the current practices, processes, and products of different organizations, and not vice-versa.

R5. The system shall be easy to learn and easy to use to the point that it presents no identifiable usability barriers, based on well-defined and implemented usage and usability studies.

This conceptual view, along with the requirements, forms

the basis of several ongoing efforts to implement experience management systems in a variety of settings. The first of these efforts, the Q-Labs EMS, is described in this paper. Lessons learned from our work with Q-Labs are being fed into other EMS projects.

4 THE Q-LABS EMS

The Experimental Software Engineering Group (ESEG) at the University of Maryland and Q-Labs, Inc., have been working together for some time on a project aimed at building the infrastructure to support a true Experience Factory within Q-Labs [12]. Q-Labs is a multi-national software engineering consulting firm that 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. Q-Labs' objectives for this project have been to provide a "virtual office" for the organization, which is spread across two continents, and to allow each Q-Labs consultant to benefit from the experience of every other Q-Labs consultant. The system being developed as a result of this effort is called the Q-Labs EMS.

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 Figure 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.

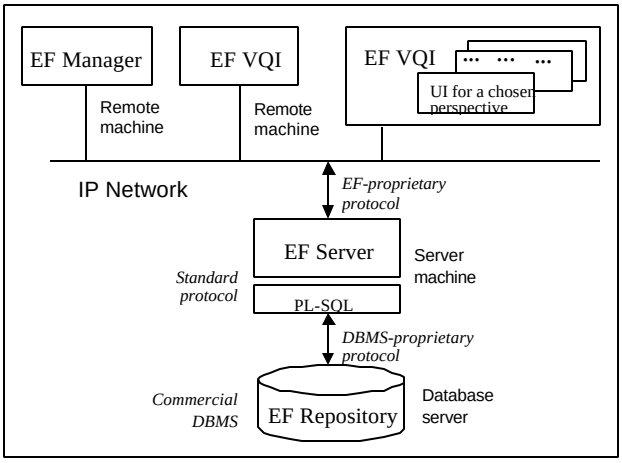

Figure 3. Q-Labs EMS architecture

In order to fulfill the second requirement, the EMS Repository is managed by a commercial DBMS (Data Base Management System), and the middle level server connects to it using a standard protocol. Our first prototype uses Java Database Connectivity (JDBC) for this.

Data Model

In order to fulfill our fourth requirement we adopted the following data model. An experience package is described 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 [11]. 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 one package instantiation are shown in Figure 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."

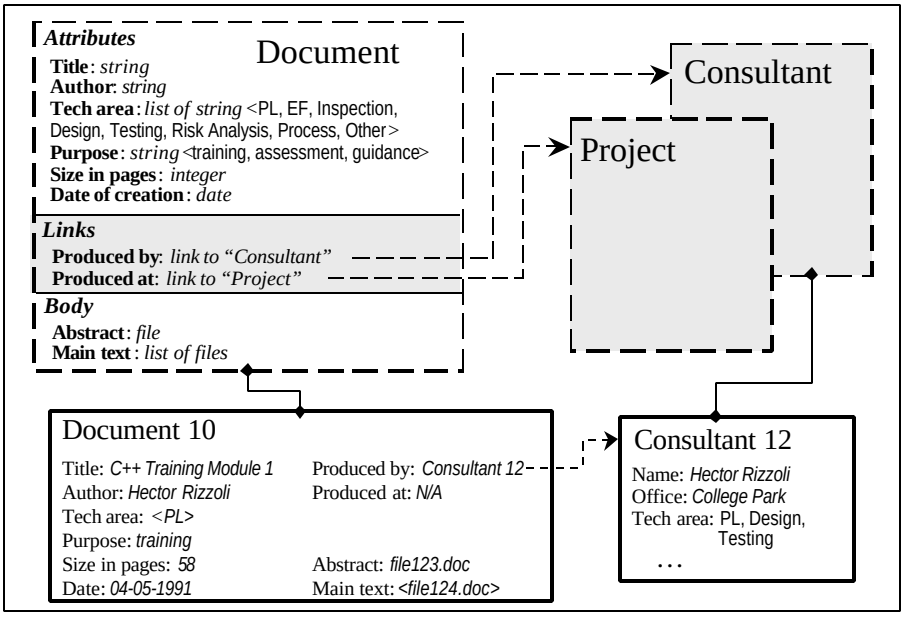

Figure 4. An Example of a Package Type and a Package Instantiation

Visual Query Interface

The user interface for a system such as the EMS is a crucial component. The fifth requirement states that the search and retrieval interface must be easy to learn, self-explanatory, and platform independent. This is a weightier requirement than it appears. The long-term success of an 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. The smallest usability obstacles would discourage early users and thus jeopardize the success of the system.

To fulfill this crucial requirement, we adopted a visual query interface (VQI) concept. As proposed by Shneiderman [14], visual query interfaces let users "fly through" stored information by adjusting query devices (checkboxes and slider bars) and viewing animated results in the computer screen. In EMS, VQI's allow easy interactive querying of the repository based on various 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 (each of which corresponds to a package type) 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 this perspective together with the attributes and query devices used to search and browse those packages. Figure 5 shows such a VQI.

Using the VQI, the user can interactively search the experience packages by manipulating the query devices on the right and observing the number of selected packages (dots) on the two-dimensional grid on the left. The grid has two axes, each of which corresponds to one of the attributes in the perspective. 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.

The VQI has two features that we believe are fundamental to 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. 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 get used to 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 package types to produce the forms that a user must complete to describe new packages.

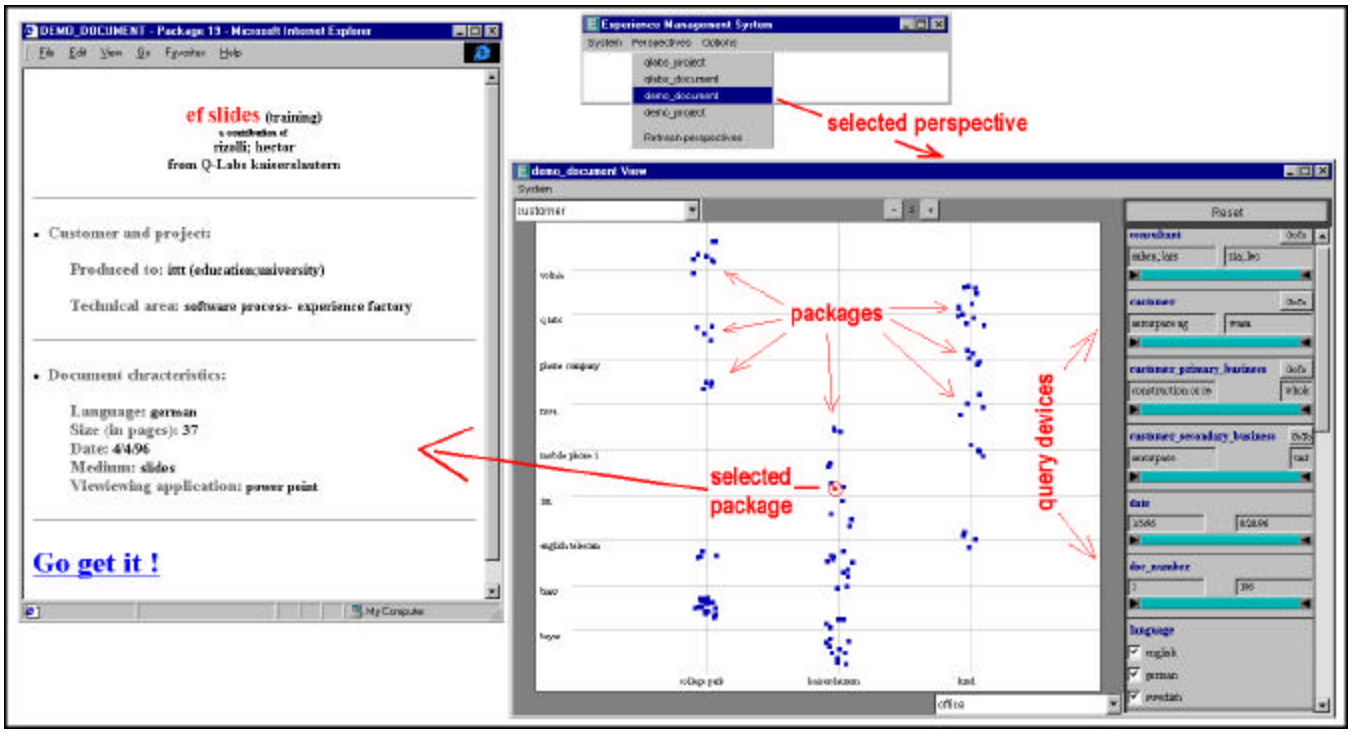

Figure 5. Q-Labs Visual Query Interface (VQI)

Lastly, in order to fulfill the third requirement – the search and retrieval interface must be platform independent – the interface component is completely implemented in Java. We have ported it to two different platforms, Windows NT/98 using Sun's Java Runtime Environment 1.1.6 and MacOS using Apple's Java Environment.

The first of several planned empirical studies to evaluate the Q-Labs EMS prototypes was an evaluation of the interface [12]. This study relied largely on qualitative research methods [9], including observation, interviews, and the constant comparison method for analyzing the data. The evaluation study found some significant, but not fundamental, usability problems. But more importantly, it showed that the EMS will eventually be acceptable to its intended users. This evaluation was a valuable and timely tool for getting feedback from the eventual users of EMS.

5 CONCLUSIONS

We have described an ongoing project involving the Experimental Software Engineering Group (ESEG) at the University of Maryland and Q-Labs, Inc. that aims to provide a system to support software engineering experience capture and reuse. The EMS architecture follows a three-tier model, with client applications running the user interface, a server application to process user queries, and an underlying repository. Currently, an interface prototype of the Q-Labs EMS exists and has been evaluated. The results of the evaluation have assured us not only that the EMS can eventually be successfully deployed throughout Q-Labs, but can also serve as a testbed for our further investigation of software experience capture and reuse. However, much needs to be done before a robust version of this 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. Much technical and organizational work remains. The latter includes designing, implementing, and evaluating new organizational procedures and deployment strategies to ensure the acceptance of EMS at Q-Labs.

Parallel to the work with Q-Labs, researchers at the University of Maryland and the Fraunhofer Center for Experimental Software Engineering have started the development of EMS II, also called "Fraunhofer EMS." This system will extend the functionality of Q-Labs EMS by adding: (1) new interfaces for links navigation and keyword search; (2) support for different experience factory roles; (3) a new software layer for accessing files already stored in corporate databases; and (4) support for package annotation and rating by its users. This system is being validated inside Fraunhofer and will later be made available to its industrial research partners.

ACKNOWLEDGEMENTS

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.

REFERENCES

1. Basili, Victor R. Software Development: A Paradigm for the Future. *COMPSAC '89*, Orlando, Florida, pp. 471-485, September 1989.

2. Basili, Victor R. The Experience Factory and its Relationship to Other Improvement Paradigms. *4th European Software Engineering Conference (ESEC)*, Garmish-Partenkirchen, Germany. The Proceedings appeared as the Springer-Verlag Lecture Notes in Computer Sciences Series 717, September 1993.

3. Basili, Victor R., and Gianluigi Caldiera. Improve Software Quality by Reusing Knowledge and Experience. *Sloan Management Review*, MIT Press, Volume 37, Number 1, Fall 1995.

4. Basili, V.R., G. Caldiera, F. McGarry, R. Pajerski, G. Page, and S. Waligora, The Software Engineering Laboratory - an Operational Software Experience Factory. *Proceedings of the International Conference on Software Engineering*, May 1992, pp. 370-381.

5. Basili, V.R., M. Daskalantonakis, and R. Yacobellis. "Technology Transfer at Motorola." *IEEE Software*, March 1994, pp. 70-76.

6. Greene, S., Tanin, E., Plaisant, C., Shneiderman, B., Olsen, L., Major, G., Johns, S. "The End of Zero-Hit Queries: Query Previews for NASA's Global Change Master Directory." *International Journal on Digital Libraries*, Vol. 2 No.2+3 (1999), pp.79-90.

7. Hackos, J.T. and J.D. Redish, *User and Task Analysis for Interface Design*. New York:John Wiley and Sons, 1998, chapter 9, pp. 258-9.

8. Houdek, F., K. Schneider, and E. Wieser. Establishing Experience Factories at Daimler-Benz: An Experience Report. In *Proc. of 20th International Conference on Software Engineering*, Kyoto, Japan, April 1998, pp. 443-447.

9. Miles, M.B. and A.M. Huberman, *Qualitative Data Analysis: An Expanded Sourcebook*, second edition, Thousand Oaks:Sage, 1994.

10. Miller, G. "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information." *Psychological Review*, 101(2), pp. 343-352, April 1994.

11. Prieto-Diaz, R. Classifying of Reusable Modules. In T.J. Biggerstaff and A. Perlis, editors, *Software Reusability*, Volume I, ACM Press, 1990.

12. Seaman, Carolyn B., Manoel Mendonca, Victor Basili, and Yong-Mi Kim. An Experience Management System for a Software Consulting Organization. Presented at the *Software Engineering Workshop*, NASA/Goddard Software Engineering Laboratory, Greenbelt, MD, December 1999.

13. Seaman, C.B. Qualitative Methods in Empirical Studies of Software Engineering. IEEE Transactions on Software Engineering, 25(4):557-572, July/August 1999.

14. Shneiderman, B. Dynamic Queries for Visual Information Seeking. *IEEE Software*, Vol. 6, No. 11, November 1994, pp. 70-77.

15. Tiwana, A. *The Knowledge Management Toolkit: Practical Techniques for Building Knowledge Management Systems*. Prentice Hall PTR, 2000.

16. von Mayrhauser, A. and A.M. Vans. Identification of dynamic comprehension processes during large scale maintenance. IEEE Transactions on Software Engineering, 22(6):424-437, June 1996.

17. Webby, R., C. Seaman, M. Mendonça, V.R. Basili, and Y. Kim. Implementing an Internet-Enabled Software Experience Factory: Work in Progress. Position paper at the 2nd *Workshop on Software Engineering over the Internet (ICSE'99)*, Los Angeles, CA, May 1999.