Expertise Recommender: A Flexible Recommendation System and Architecture

David W. McDonald and Mark S. Ackerman
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425
{dmcdonal, ackerman}@ics.uci.edu

ABSTRACT
Locating the expertise necessary to solve difficult problems is a nuanced social and collaborative problem. In organizations, some people assist others in locating expertise by making referrals. People who make referrals fill key organizational roles that have been identified by CSCW and affiliated research. Expertise locating systems are not designed to replace people who fill these key organizational roles. Instead, expertise locating systems attempt to decrease workload and support people who have no other options. Recommendation systems are collaborative software that can be applied to expertise locating. This work describes a general recommendation architecture that is grounded in a field study of expertise locating. Our expertise recommendation system details the work necessary to fit expertise recommendation to a work setting. The architecture and implementation begin to tease apart the technical aspects of providing good recommendations from social and collaborative concerns.

Keywords
Recommendation systems, collaborative filtering, expert locators, expertise location, expertise finding, information seeking, CSCW, computer-supported cooperative work.

INTRODUCTION
Everyday, people face difficult problems that they cannot solve alone. In these situations the right people are the ones who have the expertise to answer a specific question or in some other way move the problem toward resolution.

In organizational settings, people fill key roles that assist in solving problems or initiating important collaborations. Managers, senior employees, gurus, information mediators [4], and expertise concierges [13] are sought because of their ability to solve problems directly or make well informed referrals. The ability to fill this role makes these individuals valuable to the organization and makes it apparent when they are unavailable.

CSCW'00, December 2-6, 2000, Philadelphia, PA.

Recommendation systems are one possible technology that can augment and assist the natural expertise locating behavior in organizations. A recommendation system that suggests people who have some expertise with a problem holds the promise to provide, in a very small way, a service similar to that provided by key people. Expertise recommendation systems can reduce the load on people in these roles and provide alternative recommendations when these people are unavailable.

This work presents an architecture and implementation of the Expertise Recommender system (ER). ER offers several advances over previously existing systems:

- The architecture is open and flexible enough to address different organizational environments. If CSCW systems, and specifically recommender systems, are situated in their activity, organizationally specific components will be necessary within any general-purpose framework. Recommender systems, to date, have been "one size fits all".
- Organizationally specific implementations are more robust by teasing apart technical aspects of making good recommendations from the social and collaborative aspects of matching individuals. We do this based on findings from an earlier field study.
- The work proposes an alternative approach to the creation and maintenance of user profiles than ratings. This approach uses organizationally relevant data sources to create profiles that are more suited for automated expertise location.

Each of these claims will be covered below.

We begin this paper with an overview of prior systems that support expertise locating. Next, we consider what it means to seek expertise socially, in order to firmly ground the requirements of our system. We follow with a description of our system, Expertise Recommender (ER). A small scenario demonstrates how a user interacts with our system. Our last sections include the details of the architecture and of an implementation.

EXPERTISE LOCATING SYSTEMS AND CSCW
CSCW and adjacent literatures provide examples of systems that help people find others with suitable expertise. Systems that assist with expertise location are similar to a broad class of systems known as recommendation systems.

Recommendation systems are commonly used to recommend documents based on user preferences [17]. This definition distinguishes recommendation systems from systems that are more properly characterized as information retrieval (IR) or information filtering.

There are a wide variety of recommender systems. In that context, the term "documents" should be broadly construed. Recommendation systems have been used to recommend a variety of things including Usenet news [11], web pages [2, 9], videos [8], movies, music [18] and books.

Some systems refer a user to other people, which is essentially making recommendations about people. Who Knows and Referral Web are typical examples of systems that make referrals. Who Knows [6, 20] found people who knew something about a topic, based on a profile constructed from exemplary work documents. In a sense, documents were seen as surrogates for people's interests in a twist on standard IR models. On the other hand, Referral Web [10] used social networks to assist expertise location. Referral Web identified relevant individuals by their participation in co-authoring relationships and presented users with a chain of relationships that needed to be traversed from the person seeking expertise to the person who might have the desired knowledge. Yenta [5] is a hybrid of these two models, using profiles constructed from personal data as well as routing messages along a network of inferred shared interests.

In our view, recommendation systems are not completely synonymous with collaborative filtering (CF) systems. Many CF systems rely on explicit statements of user opinion, such as ratings, to create user profiles. By relying on ratings, CF systems often have difficulty generating the initial user profile and, as the profiles develop, must deal with a sparseness of ratings relative to the total number of items. These are two active areas of CF research.

A few successful recommendation systems rely on implicit opinions, rather than explicit ratings. Tapestry [7], GroupLens [16] and PHOAKS [9, 21] have all used indirect activity. Tapestry made recommendations based on who read or responded to a particular bulletin board message. One version of GroupLens used time-spent-reading a message as an implicit opinion. And PHOAKS relies on frequency-of-mention in a stream of discussion as a type of voting mechanism for web pages. Explicit rating schemes are unlikely to work for expertise recommendation, because people are reticent to explicitly state opinions of co-workers with out an appropriate context. Ratings may not work, but it might be possible to collect feedback.

CF systems have another weakness for expertise recommendation. Many recommendation systems have very specific architectures that are tailored to recommend their specific type of artifact. As well, these architectures commonly implement a single clustering algorithm (e.g. Pearson Correlation, Single-Value Decomposition, Bayesian classifier) for all participants and all artifacts. These clustering algorithms, also called neighborhood-based predictive algorithms, implement a single collaborative model that can be described as Birds of a Feather Flock Together (BOF). These systems tightly couple the architecture and the collaborative model.

In the case of PHOAKS, Referral Web, and active collaborative filtering [12], different algorithms and different collaborative models were chosen because the BOF model and clustering algorithms did not effectively fit the recommendation situation. In expertise recommendation, a BOF model that clusters profiles may not effectively distinguish individuals who have expertise from those who have only a small amount of knowledge. Alternatively, a BOF model that was effective at clustering the profiles of individuals who have expertise would likely not be effective at distinguishing in which topics their expertise lay.

This previous work suggests two critical improvements. First, any expertise recommendation solution should support a range of collaborative recommendation models and behaviors. Below, we will show the social requirements for this, but technically, an open architecture would permit a range of potential solutions. Second, one of the most difficult and perhaps most interesting problems from a CSCW perspective is how to begin untying the technical aspects of making recommendations from the social and collaborative aspects of making a recommendation to a specific user. Current recommendation approaches, especially BOF approaches, do not provide this flexibility.

The architecture that we will present is capable of supporting a range of collaborative recommendation models. An organizationally specific implementation will demonstrate the flexibility in the architecture.

However, before turning to the architecture and the subsequent implementation we must describe what people want when they seek other people for their expertise. We need to show how the recommendation of expertise is in the nuanced details of collaborative interaction. These details are the basis for our approach to the technical work.

GROUNDED REQUIREMENTS

Our prior work [13] describes a field study to examine expertise location. The field study was motivated by two desires. First, we hoped to contribute to a better understanding of the intricacies of expertise and expertise locating in a rich work setting. Secondly, we hoped to learn how expertise locating systems could be improved to better assist and augment the natural expertise locating behavior that people exhibit. The issues and findings raised in the literature were not sufficiently informative or have not always been oriented toward the objective of informing a system architecture and implementation. In short, we hoped to firmly situate systems development in a complex nuanced social setting as well as implement something useful to a group of participants.

The following briefly recounts our field study and the results [13].

The MSC Field Study

The field site was a medium sized software company that builds, sells, and supports medical and dental practice management software. Medical Software Corporation[1] (MSC) has about 100 employees at its headquarters where the study took place. The participants in the study worked in Technical Development, Technical Support and Technical Communications. These three departments are central to the development and support of MSC's products. These departments comprise a little more than one third of the employees at headquarters.

The following findings are based on an initial 5 months in the field. Data was collected using qualitative methods including participant observation, semi-structured formal interviews and informal interviews. The data was analyzed using standard ethnographic techniques.

Our study found that people at MSC commonly engage in three behaviors that support expertise locating: expertise identification, expertise selection, and escalation. We note that identification, selection, and escalation are analytical distinctions useful for understanding the fine social and cognitive decisions which are made during expertise locating and that will later be useful for technical design. It is clear that only the most self-reflective of the participants made much if any distinction among these three behaviors.

Expertise identification is the problem of finding a set of candidates who are likely to have the desired expertise. Expertise identification relies on prior experience with others, key people like an expertise concierge, and historical artifacts. The expertise concierge routes people to others with the necessary knowledge and expertise. The role has much in common with Ehrlich and Cash's information mediator [4], Paepke's contact broker [15], and Allen's technological gatekeeper [1]. Each of these roles is specialized to the organizational context. Historical artifacts are products and byproducts of the work that other MSC employees have produced. The composition and form of historical artifacts range from unstructured to highly structured things that can exist on-line or off-line.

Expertise selection is the way participants picked one person (or a small number of people) to approach for help. Expertise selection is guided by organizational criteria, the load on the potential source, and how a source reacts and interacts during a request for help. Social, organizational, and individual preference assist in narrowing a set of identified candidates to one or a small number of people who can be contacted.

In some cases, prior research has intertwined elements of expertise identification and selection. Allen's [1] work with engineers demonstrated that information seeking is heavily influenced by social networks. Cicourel [3] observed that organizational and institutional factors reinforce expertise status and can then constrain who is pursued for expertise. Lastly, Orr's [14] work with repair technicians reveals how narrative exchange (war stories) serves to demonstrate expertise and notions of competency which then guide requests for assistance with difficult problems.

Escalation is the mechanism that fixes breakdowns in identification and selection. Participants had difficulty with identification and selection because it requires fine judgements often based on incomplete and imperfect information. Escalation is an iteration of identification and selection given additional knowledge and information gained from prior iterations. During escalation the individual who initially engaged the problem maintains problem ownership. Escalation results in a wider, spreading activation through the organization that involves more people in the eventual solution to a difficult problem.

The architecture that we describe later in this paper provides specific support for this model of expertise location. An implementation of this architecture, however, requires specifics of how people perform these behaviors. The next section describes two specific identification heuristics that were used by participants at MSC.

Expertise Locating Heuristics

The MSC field study provides a detailed understanding of the following two identification heuristics. We found this understanding essential to building a nuanced system. These heuristics, Change History and Tech Support, are behaviors and rules about interpreting implied meaning using specific historical artifacts at MSC when identifying potential expertise at MSC. These historical artifacts have many, potentially crucial, features, but the participants attend to only a few through specific rules of thumb. The details are quite important to the behavior of the participants and the eventual behavior of a system.

Change History Heuristic

Change History is a single heuristic designed to augment an expertise locating behavior common to MSC's developers. Developers called this behavior the "Line 10 Rule."

The "Line 10 Rule" is a function of the organization that is MSC and the work practices that are enforced there. The developers follow a common work practice. For each software change, they check the appropriate module out of the version control system, make the changes, test the changes, and then check the changes back into the version control system. Each change is annotated with the module name, the module version, the developer responsible for the change, the check-in date, and a short text description of the change. Since many changes are the result of a contractual obligation, an administrative assistant cross-validates the changes in the version control system, the work-orders that divide up multiple changes, the time spent on changes and the total time and accounts for the contract.

Developers at MSC do not specialize in any part of the system. In practice, however, there is some attempt to assign a developer work in an area of the code where she has

[1] Names and individual identifiers have been changed to protect the privacy of the corporation and the participants.

worked in the past. Yet, the expanse of the system, the number of developers, and developer turnover will result in a developer being assigned work in portions of the code where she has not worked before. Developers use the "Line 10 Rule" in an attempt to overcome the problems that arise when working with unfamiliar code.

On the surface the "Line 10 Rule" is a simple heuristic. Given a problem with a module a developer looks into the version control system to see who last modified the code and then approaches that person for help. Developers say that this rule works because the person who last made a change has the code "freshest" in mind. The version control system is not specifically designed to support this exact use. As a result, the rule is not strictly followed, but most developers state that the heuristic is "good enough" to often get the help they need.

Tech Support Heuristic

The tech support identification heuristic is not known by a specific name. The heuristic augments a behavior of technical support representatives when faced with an unfamiliar or difficult support problem.

The heuristic is applied to the support database. The support database mediates all work performed by technical support representatives. New problems ("calls") can be entered by a support rep or by customers via email.

When faced with a difficult problem, a representative will perform multiple, separate queries over the support database using the symptoms, customer, or program module involved. The support rep then scans the records sequentially looking for similarities between the current problem and any past problems as returned by the different queries. In scanning records a support rep looks to identify people who have previously solved similar problems.

The support database is not designed to enable this type of activity. Each query (symptom, customer or program) must be completed separately, so finding similarities among the three primary characteristics is mostly done in the support representative's head. At MSC support representatives are assigned to specific customers. This benefits the customer because a single support representative can remember more details about the context of a customer's installation. A side effect is that, while the database does not facilitate queries across the three indices, establishing the relationship between an MSC customer and the support rep who works with them is relatively simple.

The tech support identification heuristic consists of attaching the symptoms, customers, and program modules of solved problems to the person who solved them. As a simple rule of thumb this scanning behavior works fairly well. However, the process can be time consuming and is only profitable when applied to more difficult support problems.

In summary, prior work and our field work demonstrates several important requirements of expertise locating:

- Detailed heuristics and social interaction guide an expertise seeker to identify candidates who are likely to have the required expertise.
- Social and organizational norms and factors guide an expertise seeker to pick a small number of candidates to pursue for help.
- The details matter. The heuristics described here are extremely bound to the organizational environment. Systems that augment expertise locating must be capable of handling a large number of details that vary based on the specific context and problem.

The focus of this paper now shifts from the details of the requirements (as exemplified by MSC) to the technical design of the expertise recommendation system.

THE EXPERTISE RECOMMENDER (ER)

The Expertise Recommender (ER) is firmly grounded in the findings from the MSC study. ER is really two things at once, at two different levels of abstraction. At the most abstract, ER is an architecture designed to handle a range of recommendation problems. However, recommendation architectures cannot solve specific recommendation problems at specific organizations; only specific implementations of an architecture can do so. Therefore, we *also* show how our work addresses this issue by implementing a specific instantiation suitable for the expertise locating behavior found in the field study of MSC.

To avoid confusion, for the remainder of this paper the architecture will be referred to as ER-Arch and the implementation will be known simply as ER.

This section first presents a simple usage scenario that provides an overview of a client and how the user interacts with the system. We follow the scenario with a description of ER-Arch, and conclude with details of the MSC implementation of ER.

A Brief Usage Scenario

This scenario provides a picture of the user interaction with the system and several key features of an ER client. The scenario describes a problem that MSC support representatives might encounter in the course of their day-to-day work with MSC customers.

Madhu is a junior technical support representative. Support reps like Madhu are assigned to handle all incoming calls from a specific set of MSC customers. However, today Madhu is also covering for a support rep who is on vacation. Madhu receives a call from a customer, PCI, who states that the system is reporting a file error in patient demographics. Even as a relatively new support rep, Madhu is fairly knowledgeable about patient demographics. However, Madhu is not familiar with the specific customer, PCI, and why that customer's system might be reporting an error.

Madhu recognizes that he will need help resolving this problem. He launches the ER client, logs into the system and gets the main window (figure 1). The main window

contains a menu bar and a list of recent prior requests that Madhu made. Prior requests are live elements (save sets) that can be revisited. When revisiting a prior request, the user can review the people who were recommended or escalate the request to gain additional recommendations.

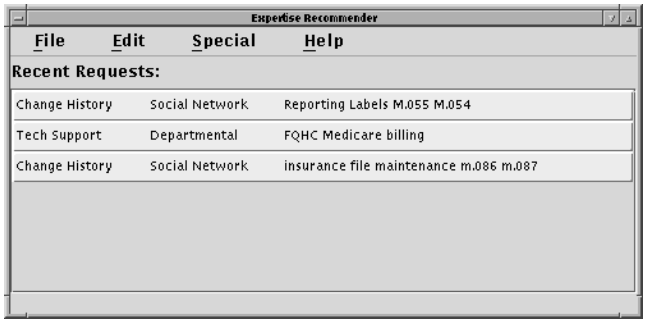

Figure 1. Expertise Recommender Main Window

Madhu has not made any prior requests for this problem, so he will have to make a new request. He chooses "New Request" from the "File" menu and the client displays the expertise request dialog (figure 2). Through conversation with the person reporting the error, Madhu is able to get the specific error code and determines that the error is generated in a module of the patient demographics system called M.013.

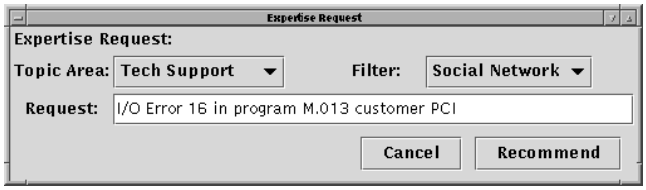

Figure 2. Expertise Request Dialog

Madhu decides that the problem is most closely associated with the "Tech Support" topic area and picks this topic in the new request dialog. Madhu chooses the "Social Network" filter because he would like the results filtered based on the people whom he knows best. The dialog lists the available identification heuristics and selection techniques in the "Topic Area" and "Filter" pop-up menus respectively. In the "Request" text field, Madhu enters the error (I/O Error 16), the program module (program M.013) and the customer (customer PCI). (These names would be natural to a support rep at MSC.) Madhu clicks the "Recommend" button, sending the request to the server.

After a short wait, the server responds with a list of recommendations. Madhu quickly recognizes that the first few people will not work. The first recommendations include the support rep whose calls Madhu is now covering, as well as a person from training who is currently off-site. So, Madhu immediately escalates the request to get additional recommendations (figure 3).

The response dialog displays the request followed by a list of recommended people. Each person is listed in a single pane containing contact information that includes office number, phone number, phone extension, and email address. Additionally, each pane contains a "Contact" button that is active only when the recommended person is logged into the ER server. The contact button, when active, establishes a synchronous chat with the recommended person.

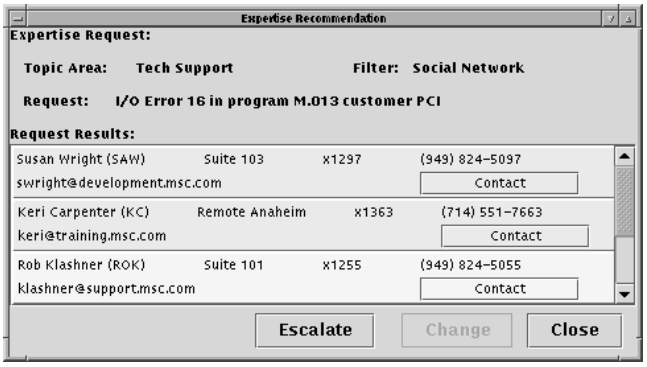

Figure 3. Recommendation Response (Escalated)

Madhu's escalated request returns two additional people, one from development and one from training. Madhu recognizes that ER has recommended a friend in development as someone who is likely to have expertise. Madhu walks over to the development suite to look for his friend.

This brief overview of a user's interaction with ER demonstrates several key aspects of ER:

- The user can pick the heuristic and the associated data that will be used to generate a recommendation.
- The user can choose the mechanism that tailors the recommendations to the user.
- Recommendations remain live so that they can be quickly revisited and escalated should the initial recommendations prove unfruitful.

Moving on from the scenario, we next turn to a description of the ER-Arch and the implementation details of an ER system for MSC.

The Architecture — ER-Arch

Like many recommendation systems, ER-Arch is server based, and a client is necessary to access the functionality. ER-Arch can support simple clients, like a Web based interface, or clients tailored to support specific features of the ER server. Regardless of the client, the important architectural details are in the ER server. Unless otherwise indicated, the components and interconnections should be assumed to exist in the server.

At a high level, ER-Arch is a pipe and filter architecture [19]. This architecture is widely used in information retrieval and information filtering, and it has great utility in expertise recommendation (as will be discussed below). Accordingly, ER-Arch is a collection of high-level supervisors, easily extensible heuristic modules, and their data stores. The supervisors provide general services and connections that facilitate a specific implementation. They also coordinate the underlying heuristic modules to provide required services (such as identification) to the system. The databases provide persistent storage for profiles and various

preferences. The supervisors in ER-Arch support profiling, identification, selection and interaction management. An overview of ER-Arch is provided in figure 4.

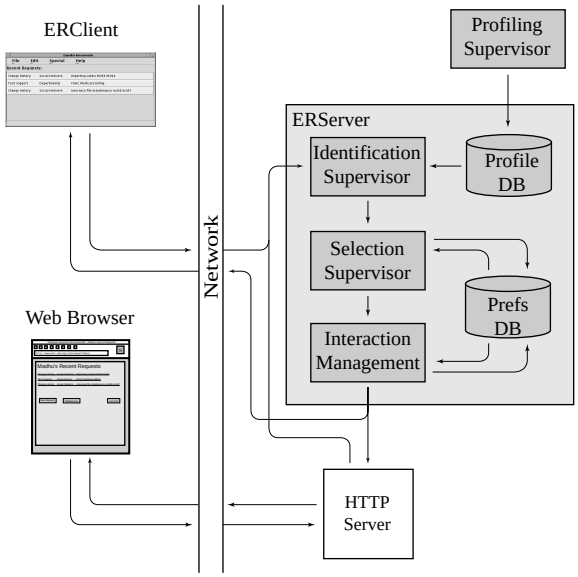

Figure 4. Expertise Recommender Architectural Overview

The ER server logically glues together portions of ER-Arch. It also handles the details of managing connections and servicing requests. The ER server implements a protocol that clients use to request and receive recommendations. The supervisors in ER-Arch support profiling, identification, selection and interaction management. The following sections discuss the ER-Arch supervisors, databases, and the interconnections among them in more detail.

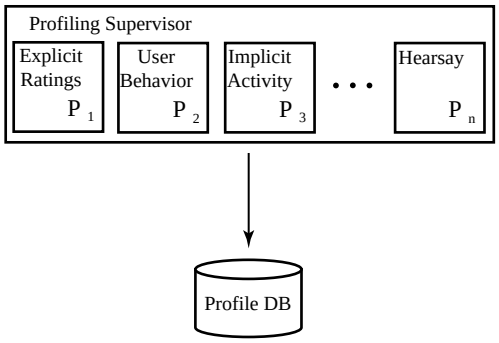

Figure 5. Profiling Supervisor

Profiling Supervisor

The profiling supervisor is responsible for creating and maintaining profiles. In many recommendation systems, profiles are a list of items which an individual has rated. These ratings profiles are used in two ways. First, profiles are clustered to create groups of users who have similar likes and dislikes. Additionally, profiles are used to identify items that a user has not yet rated and are therefore good candidates for recommendation.

In ER-Arch profiling is a periodic activity which is designed to be run off-line relative to the other activities of the server. ER-Arch can be used to maintain profiles of more than one type of thing and generate profiles from more than one single source. This is facilitated by a collection of modules supervised by the profiling supervisor. Many prior recommendation systems only support one method of profiling.

The profiling supervisor coordinates the profiling modules and provides access to the profile database. The profiling database stores the results of profiling and also links profiling to the other supervisors.

Identification Supervisor

Identification picks a set of items or people who are reasonable candidates for a recommendation. Items are picked by applying one or more heuristics and adding the identified candidates to a set. In the same way that profiling in ER-Arch is designed to support multiple types of profiling, ER-Arch is also designed to support multiple identification heuristics. This is different from many prior recommendation systems that support only one method of picking candidates. The identification supervisor coordinates the application of each heuristic. The supervisor provides each heuristic module with access to the profile database.

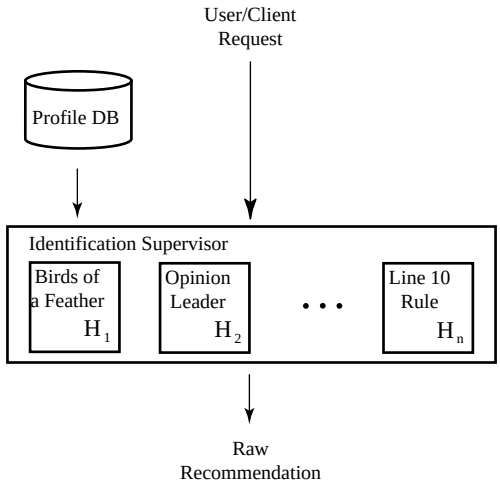

Figure 6. Identification Supervisor

Identification is initiated in response to a user request. Through an ER client, the user can indicate which heuristic she would like to apply and provide parameters that tailor the heuristic to her current needs or desires. The result of identification is a set of raw recommendations. Raw recommendations are passed directly to the next stage.

Selection Supervisor

Selection takes a set of candidate recommendations, and then reorders and possibly removes items from the set to generate a refined recommendation. The selection supervisor receives the raw recommendations from identification and applies one or more selection modules (such as organizational criteria). The selection supervisor in ER-Arch is designed to support many different methods of selecting and filtering the raw recommendations.

The selection supervisor provides selection modules access to the preference database that maintains personal and or-

ganizationally relevant data (e.g., departmental affiliation) that can be used as selection criteria. The result of selection is a refined recommendation. The refined recommendation is passed into the interaction management before becoming a final recommendation.

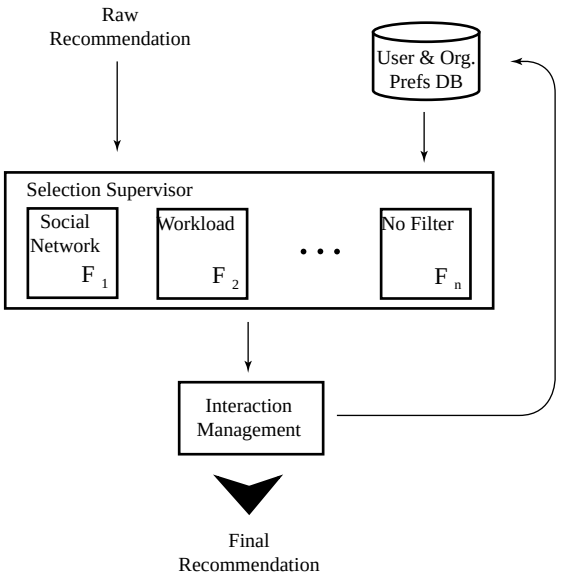

Figure 7. Selection Supervisor and Interaction Management

Interaction Management

Interaction management receives the refined recommendations and processes them before releasing them as a final recommendation. Interaction management in ER-Arch tracks the users' interaction with the system and tracks the recommendations. Interaction management also collects feedback about system performance. The specific components of interaction management depend upon the specific needs of an organization and a given implementation

Interaction management serves to reintegrate and smooth out discontinuities introduced by separating expertise location into two distinct phases. In the real world, communication, escalation, issue tracking, problem context, and how individuals manage their accessibility to others are the setting in which expertise location resides. Interaction management is the architectural component that provides these components, thereby serving to tie together and serve as the setting for identification and selection.

Current State of ER

ER-Arch is instantiated in an implementation of ER for MSC. ER includes a server and a dedicated client. Both server and client were coded in Java, and consist of more than 20,000 lines in 84 classes. The code includes numerous base classes and abstract classes that simplify the extension of ER with new identification heuristics and new selection techniques.

The MSC Implementation

As mentioned, much of the promise of ER-Arch lay in its flexibility. If expertise seeking really is situated within specific organizational or social contexts, presumably we will need some combination of generic and organizationally specific mechanisms for finding people. This section details the work involved in creating an implementation of ER suitable for MSC's use. For MSC, ER implements the two identification heuristics described above as well as three selection techniques. Perhaps more importantly, this section also describes the organizationally specific data processing required to effectively generate expertise recommendations.

To implement the identification heuristics, real data from MSC's change history system and support database were required. MSC provided the most recent eight years of program change history covering their medical system and a portion of the shared libraries used by both their medical and dental systems. This corresponds to a total of 7316 individual code modifications. The data from the support database cover the last four years of customer support activity. These data include all calls logged and marked as completed. The support data cover software and hardware issues for all of MSC's products. More than 200,000 calls were handled during the time period covered by the data.

Profiling

The identification heuristics and profiling modules are strongly related through the profile database. The identification heuristics specify which features the profiling modules must extract. The intricate details of implementing profiling modules for MSC demonstrate how real, messy data sources are transformed into profile records more amenable to the application of identification heuristics.

The profiling supervisor controls profiling in ER. Profiling modules are extensions to a base class that provides methods for interacting with the profiling supervisor and the profile database. ER supports a profiling module for each data source and corresponding identification heuristic. Profiling modules incrementally update each profile record.

The change history profiling module crawls through each change collecting the developer responsible for the change, the module name, the version, and date. For each change, the developer's profile is found and the slot containing code changes is searched for previous modifications to the module. If the developer has no prior changes, then a new entry is made containing the module, the date of the most recent change, and the total number of changes. If the developer had previously made a change, then the date of last change is updated and the total count of changes is incremented. The profiling module examines dates and is careful to modify profiles only when changes are more recent.

The profiling module for the tech support area is incremental as well. However, the tech support profiling module assumes that it will be fed unique records from the technical support database. The profiling module considers four fields key: the support representative, the description of the problem (a small amount of free form text), the customer, and the module that the customer reports is the problem. These four fields are used to incrementally build three vector spaces that characterize a specific person's activity in the support database.

Creating profiles with a vector space model required implementing additional functionality to support profiling and identification. A set of thesauri were needed, one for each of the vector spaces. These determine which terms are in the vector space, and therefore should be counted or not counted. A parser was required to separate terms, phrases, and important punctuation. This same parser is also used to parse user queries.

The support database records are exceedingly messy. The entries have few restrictions and data entry conventions that are not formalized. Different support representatives have different preferred misspellings and abbreviations. These misspellings and abbreviations created two potential problems that had to be solved. First, by using frequency of occurrence as a mechanism for determining which terms are entered into the thesaurus, some of the abbreviations would never be entered because they occur too infrequently. Secondly, with many misspellings and non-uniform abbreviations, users might find it difficult to garner results when using their personal, quirky language.

An ad-hoc approach addressed these problems with the data. First, a parser was built that recognized punctuation important to how MSC's systems work. The parser was coded to recognize certain phrases which are relevant to the way customers and support reps express problems. The parser removes common stop words, but does not perform any stemming. The parser was used to parse the symptoms, customers, and program fields of all support records generating a complete list of terms. For each field, terms were sorted, reviewed, and cleaned to create a thesaurus.

The tech support profiling module constructs a profile in two stages. In the first stage each support database record is examined, and the support rep assigned to the problem is identified. The symptom, customer, and program module fields are parsed, and terms are validated against the appropriate thesaurus. The support rep's profile is then updated by incrementing the term counts in the appropriate vector space. Additionally, a master term vector is incremented representing the total number of times a term is used in the entire database. In the second stage, the profiling module visits each profile record and normalizes each vector.

The profiling supervisor finishes profiling, for all profiling techniques, by automatically rebuilding all indices that are associated with the profile database. The MSC implementation requires three indices: one that indexes individuals by name, one that indexes program modules, and one that indexes the vector space profiles based on the percentage of term space coverage. These indices are specifically designed to support identification and their use will be covered in the next section.

Identification

The details of implementing the change history and tech support heuristics provide a clear example of how identification heuristics can be moved into ER. As well, they demonstrate how different modules can handle escalation in different ways.

For identification, the identification supervisor manages modules that have been built to perform identification. An identification module base class provides basic functionality for working with the profile database, interaction with recommendation requests, and recommendation lists. Recommendation requests encapsulate the user request, the system state of the request, and the recommendations that were made in an attempt to satisfy the request. Recommendation lists provide methods for ordering and reordering items that have been recommended. Each recommendation module is free to determine how escalation is managed and exactly what additional action is taken on an escalation.

In the case of the change history, the action of the identification module is fairly straightforward. The module parses the request text to identify all of the program modules that are mentioned in the request. For each program module, a query is made of the profile database, returning a list of individuals who have modified the module. If the request contains multiple program modules then the resulting lists are intersected to find people who touched all modules mentioned. For each person identified, a recommendation list item is created that includes the most recent modification date and the total number of touches. The item is then inserted into an unfiltered recommendation list ordered from the most recent touch to least recent touch.

Tech support identification is similar to change history identification. The request text is parsed and three query vectors are created, one for symptoms, one for customers and one for program modules. The profile database is then queried using the vector space index. This index groups profiles based on the percentage of terms covered by a person's vector space profile. Escalation level is used to determine the percentage term space that must be covered before a person will be considered. As the escalation level rises, the term space that must be covered drops.

For each person returned, similarity of that person's profile is gauged relative to three query vectors determined by parsing the request text. The cosine angle of incidence is calculated for each query vector and the appropriate normalized vector in the profile. A weighted sum of the angles is calculated to create the overall ranking for the match. The recommendation list maintains the order from highest to lowest weighted sum.

Selection

When an identification module has completed, the unfiltered recommendation lists are passed to the selection supervisor. The operation of the selection supervisor is similar to that of the identification supervisor. Selection modules transform an unfiltered recommendation list to a final recommendation list by visiting each recommendation item in the unfiltered list, and then validating or modifying the recommendation. If the item will be part of the final recommendation it is inserted into the final list. The designer of a selection technique decides how escalation level modifies or changes each selection technique.

The MSC implementation includes three selection techniques. The first technique is a simple "No Filter" capability. This technique removes individuals who no longer work for MSC. As each recommendation is visited, there is a lookup in the user preference and organizational database. If the person's record indicates that she is still working for MSC, she is added to the final recommendation list. The escalation level does not modify this filter.

The second technique, "Departmental," selects based on the organizational distance between the department of the person making the request and the department of each person recommended. The filter module maintains a small graph structure that represents the "distance" between each of the departments at MSC. The departmental technique relies on the current escalation level and the departmental graph to determine who will be added to the final recommendation list. The distance is calculated, and if the distance is less than a threshold, then the person is added to the final recommendation list.

The third selection technique, "Social Network," filters recommendations based on an aggregate social network of the MSC workplace. Data for the social network were obtained during the fieldwork at MSC. Extended interviews were conducted where the participants performed successive pile sorts, which resulted in an aggregate social network for MSC. This social network is a graph structure where the nodes represent individuals at the workplace and the edges represent some relation that links two individuals. The social network technique calculates the distance between the person making the request and the recommended person. If the distance is less than a threshold, then the recommended person is added to the final recommendation.

Note that while these selection mechanisms use organizationally specific data for MSC, the techniques themselves would generalize to other organizations.

Interaction Management

The interaction management modules in the MSC implementation exemplify how the specific details necessary to create a working system may lead to the creation of more general, reusable components. While these components were required by MSC in order for ER to be successful in their setting, all of the components are reusable within other organizations.

For example, through our fieldwork at MSC we saw participants make specific choices with regard to how they controlled their accessibility by others and predilection to specific communicative media. One expert had a unique technique for indicating how interruptible he was depending upon the importance of the problem that required his attention. He posted a sign on his already closed door. For him, the system needed a similar mechanism.

Interaction management currently consists of three components, the escalation tracker, a communication tracker, and expert face management. This section will briefly cover each of these three components.

The escalation tracker manages the expiration of recommendation requests on the server side. This includes maintaining a database of all active requests for each ER user. When a client connects, the escalation tracking component looks up the user in the request database and sends the active requests. As requests expire, escalation tracking marks the item, removes it from the database and notifies the appropriate client. Lastly, the escalation tracker updates the user profile and preference data.

The communications tracker is responsible for routing the chat messages between users. The tracker receives all incoming messages, validates that the recipient is still connected and then sends the message out. The tracker modifies the user preference data on each communications attempt. This implicit preference data contains a list of all of the other users with whom contact was attempted and the total number of communication attempts. The communications tracker also provides a simple mechanism for a user to state explicit preference for any of the other users known by the system.

Expert face management allows a user to have some control over her availability to other users through the system. Face management allows a user to state her availability on a four step scale from "available" to "do not disturb." The default, unless otherwise set, is available. Face management does not require a user to attend to her level of availability. Face management includes a decay system that allows a user to specify how quickly her current level of activity will move toward being available. Face management maintains a thread that periodically checks decay rates and modifies availability settings.

In summary, the specific modules that comprise profiling, identification, selection, and interaction management differ in their level of specificity to the organizational setting. The heuristics and algorithms used for identification and profiling are very organizationally specific. While these identification heuristics might be used in another organization, the implementation is for MSC and the specifics of their identification mechanisms are particular to that setting. The modules that make up selection appear organizationally specific because they are tailored to the social and departmental character of an organization. Yet, these modules have some flexibility and can be adapted to other organizations given the appropriate data. Lastly, the modules in interaction management are perhaps the least organizationally specific. The modules in interaction management have a face validity that stems from a basic understanding of the requirements of expertise location. As the MSC implementation shows, ER-Arch handles this combination of organizationally specific and generic mechanisms for expertise recommendation.

CONCLUSIONS

The Expertise Recommender, through ER-Arch and the implementation, demonstrates three key features. First, ER-Arch is a general architecture that can be tailored to many different recommendation situations. ER-Arch is

capable of supporting more traditional recommendation situations by incorporating appropriate profiling and identification modules. It is also highly suited for expertise recommendation.

Second, the implementation of ER for MSC relies on profiling techniques that are not common to other recommendation systems. ER generates profiles of people based on work products and work byproducts. This approach trades the problems of critical mass and sparse ratings for problems with dirty and inconsistent organizationally specific data sources.

Lastly, ER-Arch and the MSC implementation of ER begin to tease apart technical concerns involved with making recommendations from the social and collaborative concerns. This separation allows ER to support multiple collaborative models and provide more adaptive recommendations. The ER client exposes these features allowing ER users to pick recommendation strategies appropriate for their task.

Recommendation systems, such as ER, provide another approach to supporting natural expertise locating behavior. In situations where a senior employee, guru, information mediator, or expertise concierge is not available, a system like ER can suggest alternatives where previously no help would have been available. By relying on organizationally relevant sources of information and heuristics that are naturally used, systems like ER can assist in finding people who have expertise and who may not otherwise be identified.

Acknowledgements

This project has been funded, in part, by grants from National Science Foundation (IRI-9702904), the UCI/NSF Industry/University Cooperative Research Center at the Center for Research on Information Technology and Organizations (CRITO), and the University of California MICRO program. Additionally, the first author was supported by the University of California Regents' Dissertation Fellowship.

This work has benefited from conversations with Kate Ehrlich, Saul Greenberg, Joe Konstan, John Riedl, Lynn Streeter, Loren Terveen, and Clark Turner. Members of our research group, Wayne Lutters and Jack Muramatsu, contributed to our understanding of expertise. We thank the anonymous reviewers for their feedback. We would also like to thank the participants at MSC for their patience and insights over the last few years.

REFERENCES

1. Allen, T.J. *Managing the Flow of Technology*. MIT Press, Cambridge, 1977.
2. Balabanovic, M. and Shoham, Y. Fab: Content-Based, Collaborative Recommendation. *CACM*, *40* (3). 66 - 72.
3. Cicourel, A.V. The Integration of Distributed Knowledge in Collaborative Medical Diagnosis. in Galegher, J., Kraut, R.E. and Egido, C. eds. *Intellectual Teamwork*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1990, 221 - 242.
4. Ehrlich, K. and Cash, D., Turning Information into Knowledge: Information Finding as a Collaborative Activity. in *Digital Libraries '94*, 119 - 125.
5. Foner, L.N., Yenta: A Multi-Agent, Referral-Based Matchmaking System. in *First International Conference on Autonomous Agents (Agent'97)*.
6. Furnas, G.W., Deerwester, S., Dumais, S.T., Landauer, T.K., Harshman, R.A., Streeter, L.A. and Lochbaum, K.E., Information retrieval using a singular value decomposition model of latent semantic structure. *SIGIR'88*, 465 - 480.
7. Goldberg, D., Nichols, D., Oki, B.M. and Terry, D. Using Collaborative Filtering to Weave an Information Tapestry. *CACM*, *35* (12). 61 - 70.
8. Hill, W., Stead, L., Rosenstein, M. and Furnas, G., Recommending and Evaluating Choices in a Virtual Community of Use. *CHI '95*, 194 - 201.
9. Hill, W. and Terveen, L., Using Frequency-of-mention in Public Conversations for Social Filtering. *CSCW '96*, 106-112.
10. Kautz, H.A., Selman, B. and Shah, M. Referral Web: Combining Social Networks and Collaborative Filtering. *CACM*, *40* (3). 63 - 65.
11. Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R. and Riedel, J. GroupLens: Applying Collaborative Filtering to Usenet News. *CACM*, *40* (3). 77 - 87.
12. Maltz, D. and Ehrlich, K., Pointing The Way: Active Collaborative Filtering. *CHI '95*, 202 - 209.
13. McDonald, D.W. and Ackerman, M.S., Just Talk to Me: A Field Study of Expertise Location. *CSCW'98*, 315 - 324.
14. Orr, J.E. *Talking About Machines: An Ethnography of a Modern Job*. Cornell University Press, Ithaca, 1996.
15. Paepcke, A. Information Needs in Technical Work Settings and Their Implications for the Design of Computer Tools. *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, *5*. 63 - 92.
16. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J., GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *CSCW '94*, 175 - 186.
17. Sarwar, B.M., Konstan, J.A., Borchers, A., Herlocker, J., Miller, B. and Riedl, J., Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System. *CSCW '98*, 345-354.
18. Shardanand, U. and Maes, P., Social Information Filtering: Algorithms for Automating "Word of Mouth". *CHI '95*,210-217.
19. Shaw, M. and Garlan, D. *Software Architecture: Perspectives on an emerging discipline*. Prentice Hall, 1996.
20. Streeter, L.A. and Lochbaum, K.E., Who Knows: A System Based on Automatic Representation of Semantic Structure. *RIAO '88* , 380 - 388.
21. Terveen, L., Hill, W., Amento, B., McDonald, D. and Creter, J. PHOAKS: A System for Sharing Recommendations. *CACM*, *40* (3). 59 - 62.