

# Software Requirements for Supporting Collaboration through Categories

David F. Redmiles

Department of Information and Computer Science  
University of California, Irvine  
Irvine, CA 92697-3425 USA  
+1 949 824 3823  
redmiles@ics.uci.edu

## ABSTRACT

The Knowledge Depot is a group memory and awareness tool for supporting collaboration in geographically distributed communities. Some specific properties of the implementation of the Knowledge Depot may be generalized to serve as a basic set of requirements for collaboration software. This paper discusses these salient properties and their relationship to research results in the areas of human-computer interaction, software engineering, and general engineering disciplines.

## Keywords

Classification, Hierarchy, Multiple Perspectives, Computer-Supported Cooperative Work

## INTRODUCTION

The Knowledge Depot is a group memory system that has existed for almost a decade in one version or another [1, 2]. Simply stated, the Knowledge Depot works by collecting email messages and any Web-viewable attachments in persistent storage and it allows end users to access these messages and documents through an interface based on categories and queries. The Knowledge Depot was originally designed to enable geographically distributed communities to share information. It currently has features for both knowledge management and awareness for either groups or individuals.

Some specific properties of the implementation of the Knowledge Depot create a unique potential for group collaboration. Namely, the Knowledge Depot implements a category hierarchy and category operations, a query mechanism, provides persistent database storage, and a simple but useful user interface model. The behavior of the Knowledge Depot correlates well with research results in the areas of human-computer interaction, software engineering, and general engineering disciplines. As such, the behavior forms a basis for requirements for collaborative software in general.

## THE KNOWLEDGE DEPOT

Figure 1 shows a view of the Knowledge Depot and explains its features from the perspective of the end user. Looking at the panes in the figure in a counter-clockwise order, the first pane is the topic browser, which presents a category hierarchy. The categories index all email messages and their attached documents stored in the Knowledge Depot. The next pane (bottom of the Figure) is a list of messages in a selected category. The next pane (right of the Figure) is the content of a message. The last pane is a subscription pane. It allows an individual end user to express interest in a category and receive email summaries about activity in that category.

The specific snapshot represented in Figure 1 is that of a Knowledge Depot archive for a distribution list for a software project called EDCS. Within the category of messages about the EDCS project, several more specific categories are represented. One concerning quarterly reports is highlighted. All messages having to do with quarterly reports for the EDCS project are shown in the message list pane. One of these is highlighted and viewed in a separate, message pane. Finally, the subscription pane shows that an end user is registering to receive summaries of email activity about quarterly reports every seven days.

A pane that is not shown in the Figure is an interface for specifying a query. The query pane allows end users to type in a list of keywords that may be in any header field of an email message, such as the name of a sender or receiver, a phrase in a subject line, a date or range of dates, etc. The results of a query are displayed in a message list pane. A simple operation (one "mouse click") converts a query into a new category.

Several operations are permitted on categories in the topic browser. The primary operations are to *create* a new category, *edit* an existing category, *subscribe*, and *delete*. Subscription was mentioned already. Delete and edit are straightforward. However, to create a new category needs explanation. A new category is like a query. It is a list of keywords or phrases that may appear in any header field of an email message. There is also a mechanism for annotating messages with additional keywords or phrases that end

LEAVE BLANK THE LAST 2.5 cm (1") OF THE LEFT  
COLUMN ON THE FIRST PAGE FOR THE  
COPYRIGHT NOTICE.

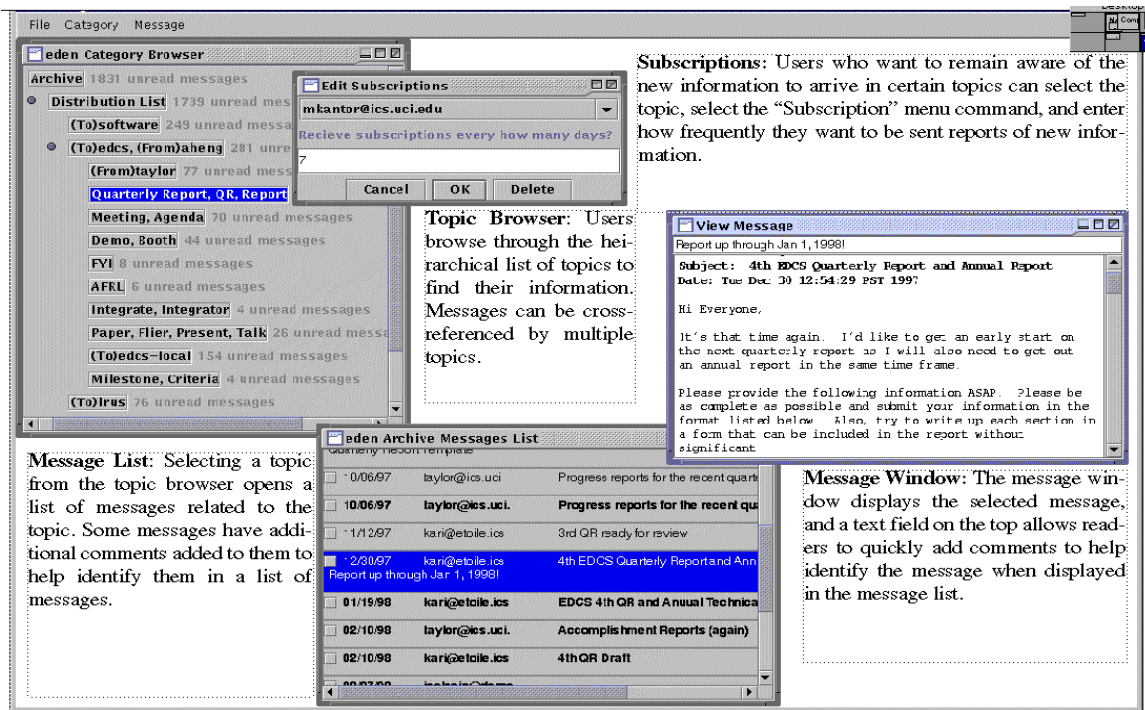


Figure 1: Annotated, End User's View of the Knowledge Depot

users may feel the message also pertains to. These end user annotations are also searched by both queries and categories. Categories build hierarchically on previous categories. Namely, each new category adds more keywords to the search. Thus in the example in Figure 1, the category "quarterly reports" adds two keywords to further restrict messages pertaining to the "EDCS" project. In general, category names are the same as the keywords that form the basis for the searches.

#### DISCUSSION OF SALIENT PROPERTIES

The key features and properties of the Knowledge Depot include the category hierarchy and category operations, query, persistent database storage, and the simplicity of the user interface model.

#### Category Hierarchy and Category Operations

One of Herbert Simon's theses was that the world may not be hierarchically organized, but that hierarchy is a convenient device for individuals to cope with complex situations [3]. Hierarchy is certainly a recurring theme for organizing complex artifacts in software tools. For example, file systems, databases, object-oriented programming tools, and email systems are all types of systems that normally use hierarchy as an interface to complex information. In the Knowledge Depot (and other tools that present hierarchical interfaces), the category hierarchy defines the terms or objects with which members of a group or community will collaborate to achieve an objective [4].

However, the use of hierarchies has familiar shortcomings for end users. In general, using software engineering terms, some tasks lend themselves better to a *bottom-up* approach, whereas hierarchies are equivalent to a *top-down* approach.

The top-down or hierarchical approach to organization works well for well-understood problem domains. Classification schemes, ontologies, etc. may be refined from existing starting points. However, so-called *ill defined* or *wicked* problems have no clear starting point [3, 5]. Potential and partial solutions must be postulated, tested, and failures repaired. Schön calls this process one of *reflection-in-action* [6]. Thus, there exists a tension between using hierarchy for managing complexity and maintaining flexibility for evolution.

In the Knowledge Depot, the tension between hierarchy and evolution is resolved by allowing categories to be created, edited, and deleted by end users in a group or community. Note that an instance of the Knowledge Depot may be *seeded* with an initial category hierarchy [7]. However, this step is not required. Flexibility between a *prescriptive* category hierarchy and an evolving *descriptive* hierarchy supports a wide range of group collaborations and objectives. The difference is important in light of the tradition of workflow and process tools that have been engaged to support collaborative work. These tools have traditionally been prescriptive and inflexible with respect to modifying the structure of interaction [8]. Once the terms and organization of a process have been established, all collaboration must occur within the confines of this specification. The current generation of process tools allows exceptions and even modifications. However, modifications often require authorization and manipulation of complex specification languages or interfaces. The Knowledge Depot allows categories to be modified by adding or subtracting keywords.

Thus, through a simple user interface model, members of a community can evolve a set of categories, which best facilitate their objective. The model is suited for well or ill-defined objectives. For well-defined objectives, the Knowledge Depot can be seeded with a desired category hierarchy. For ill-defined objectives, the categories can be evolved by the members of the community based on the discussion reflected by the content of email exchanges or other, off-line processes. The Knowledge Depot does allow for an authorization process to be followed in the situation where end users should not be allowed to modify categories themselves. However, in highly cooperative environments, not imposing this authorization process gives maximal freedom to the collaboration. Hence, the simple model of the Knowledge Depot places the emphasis on the collaboration and not on an artificial process of collaborating.

Many aspects of achieving an objective reflect a process of mutual education [9]. With respect to the Knowledge Depot, education must occur around the terminology, objects, objectives, and even awareness of the roles and identities of the other members of the community. The evolving category hierarchy makes most of this information available to end users of the Knowledge Depot (i.e., individual members of a community). Even new members of a community can come up to speed with the rest of the group by exploring how existing categories have been defined and the content of these categories. The need for a process of mutual education in collaboration is well proven by current work in requirements engineering [10] and participatory design [11]. In both these related areas, objects are used to focus discussion around the meaning of common vocabulary as well as the refinement of requirements. Mutual education is essential if the terminology of categories (or any vocabulary) is to be used as a basis for collaboration [12].

Finally, collaborative communities are comprised of many stakeholders. For various reasons, members of a community working individually frequently are unaware of the activity or even the existence of other members of the group. Of course, the activity and the roles of all members of the group can have a direct bearing on the achievement or failure of community objectives [4]. The Knowledge Depot's simple subscription mechanism described in the first section aids individuals in being aware of others' activity and identity. Activity that is reflected in email exchanges may be summarized and reported to an individual. The content of the messages reveals the identity. This simple approach works in the case of the Knowledge Depot because activity is structured around the category hierarchy. Thus, more complex schemes such as recommender systems need to be employed [13].

In sum, in the Knowledge Depot, the category hierarchy and the category operations enable members of a community to

- 1) manage complexity
- 2) frame objects common to a collaboration
- 3) communicate about well-defined objectives
- 4) evolve an understanding of ill-defined objectives
- 5) achieve mutual education around a common vocabulary
- 6) retrieve prior information, and
- 7) be aware of the identity and activity of other members of the community.

#### **Query**

Introducing a new category in the category hierarchy has implications for the entire group or community using the Knowledge Depot. There are occasions when an individual wishes to explore an issue temporarily. For ephemeral objectives, end users of the Knowledge Depot can use the query feature. The query results create a temporary category whose messages can be viewed. If desired, the temporary category can be made more permanent by being added into the category hierarchy. Queries also facilitate retrieval in the case that an end user cannot find a category matching his or her current objective. In the Knowledge Depot, queries enable end users to

- 8) have a direct approach to retrieving information
- 9) perform specific, short-term problem solving, and
- 10) evolve the category hierarchy.

#### **Persistent Database Storage**

Persistent storage is an obvious requirement for a group memory system and for most collaborative software. However, some properties that the Knowledge Depot achieves by using a database mechanism for storage are critical in general to problem solving by individuals and groups. First, the database allows a single object, in this case an email message, to be stored once and retrieved in multiple instances. Thus, in the Knowledge Depot, an email message and its attached documents can be classified under and accessed through multiple categories. Even the more sophisticated email products on the market today still require multiple copies of messages to be made to achieve multiple classification. A second consequence of the Knowledge Depot's implementation is that categories can be deleted without the messages classified in them disappearing. Again, this is a feature lacking in commercial email tools. Third, because categories are implemented as queries over the database, the contents of the categories (i.e., the lists of messages) are always current. In some email systems, filters can move messages automatically into folders. However, messages must reside only one category. In the Knowledge Depot, the messages in any given category are both current and complete.

The capability of messages appearing in multiple categories is vital as it reflects and supports people's use of *multiple perspectives* in comprehension and problem solving [14, 15]. Briefly, a perspective is a way in which a person frames a problem. For practical purposes, it is a set of terms and a way of relating those terms that assists in accomplishing a task. A single individual will use several

perspectives in solving a single problem or understanding a situation [16].

Researchers in cognitive science and artificial intelligence have used many terms for this concept: e.g., plans [17, 18], frames [19], scripts [20], cases [21, 22, 23], and mental models [16, 24]. In the Knowledge Depot, a perspective would be a section of the category hierarchy. Thus multiple perspectives are implemented by multiple classification schemes, which, in turn are made possible by the use of the database underlying the Knowledge Depot.

Having multiple perspectives implemented as multiple classification schemes addresses the need for a system supporting collaborative problem solving would need to represent alternatives. Because of the ability for messages to appear in many categories, alternative perspectives can be created, compared, and when desired, discarded. The alternatives can be different perspectives of a single individual or different viewpoints from multiple stakeholders [25]. Because categories may be discarded without losing messages, many variants may be tried.

Knowing that a database underlies the Knowledge Depot makes it easier to understand the idea that categories are equivalent to queries. Selecting on a category creates a current set of messages. What was not strongly emphasized in the earlier sections on categories and queries was that although sub-categories successively specialize categories they are created under, the more general category still exists independently of its inferior categories. An end user of the Knowledge Depot may view all the messages subsumed by a general category. Abstraction and refinement, key aspects to the process of object-oriented problem solving are more generally supported in the Knowledge Depot than any email or object-oriented programming system.

The use of a database technology for persistent storage enables end users to

- 1) use multiple perspectives for individual and group comprehension and problem solving
- 2) explore alternatives and multiple viewpoints
- 3) have the coexistence of ephemeral and permanent categories and classification schemes, and
- 4) work through generalization and abstraction.

#### **Simplicity of the User Interface Model**

A requirement for software tools that is often overlooked is that the user interface be simple. This requirement may seem to be subsumed in the vague requirement of "user friendliness." However many factors drive interfaces to be complex in the number and kinds of features. The Knowledge Depot embeds the hypothesis that a simple model for creating the categories may not be the most powerful for specification and retrieval, but may be the most usable and therefore the most effective by a large group of end users. This hypothesis remains to be proven. However, an initial survey of users of one version of the Knowledge Depot revealed that even a feature as simple as the subscription feature was difficult for the majority of end

users to understand. Moreover, the research carried out by Don Norman and others about people's mental models for using everyday devices corroborates the hypothesis [26, 27]. The Knowledge Depot facilitates group collaboration by enabling end users to

15) work with a simple and useful (vs. complex and powerful) user interface model.

#### **CONCLUSION**

Four major aspects of the Knowledge Depot have been described: the category hierarchy and operations, the query mechanism, the persistent database storage, and the simple user interface model. How these aspects create a behavior supporting collaborative work has been discussed and corroborated with research results in various disciplines. Fifteen properties of the behavior of the Knowledge Depot are suggested as fundamental requirements for any collaborative system.

#### **ACKNOWLEDGMENTS**

The current version of the Knowledge Depot is the doctorate dissertation work of Michael Kantor at UC Irvine. A previous version was the doctorate work of Stephanie Lindstaedt at U. Colorado. Both were supported at the time by the efforts of Beatrix Zimmermann, Michael Atwood, and others at Bell Atlantic. Significant research support was provided by the National Science Foundation under grant CCR-9624846 and by UC Irvine.

#### **REFERENCES**

1. M. Kantor, B. Zimmermann, D. Redmiles, *From Group Memory to Project Awareness Through Use of the Knowledge Depot*, Proceedings of the 1997 California Software Symposium (Irvine, CA), UCI Irvine Research Unit in Software, Irvine, CA, November 7, 1997, pp. 19-26.
2. S. Lindstaedt, K. Schneider. *Bridging the gap between face-to-face communication and long-term collaboration*. Proceedings of the international ACM SIGGROUP conference on Supporting group work, 1997, pp. 331 – 340.
3. H.A. Simon, *The Sciences of the Artificial*, The MIT Press, Cambridge, MA, 1981, pp. 160-229.
4. Y. Engeström, *When is a Tool? Multiple Meanings of Artifacts in Human Activity*, in Engeström, Yrjo, Learning, Working and Imagining, Painettu Kirjapaino Oma Ky:ssa, Jyvaskylassa, 1990.
5. G. Fischer. *Domain-Oriented Design Environments*. Automated Software Engineering, Kluwer Academic Publishers, Boston, MA, 1994, pp. 177-203.
6. D. Schön, *The Reflective Practitioner: How Professionals Think in Action*, Basic Books, New York, 1983, pp. 21-69.
7. G. Fischer, J. Grudin, R. McCall, J. Ostwald, D. Redmiles, B. Reeves, F. Shipman, *Seeding, Evolutionary Growth and Reseeding: The Incremental*

- Development of Collaborative Design Environments*, in G. Olson, T. Malone, J. Smith (eds.), *Coordination Theory and Collaboration Technology*, Lawrence Erlbaum Associates, to appear.
8. G. Nutt. *The evolution towards flexible workflow systems*. Distributed Systems Engineering, vol.3, (no.4), BCS; IEE; IOP, Dec. 1996. pp.276-94.
  9. G. Fischer, A. Girgensohn, K. Nakakoji, D. Redmiles, *Supporting Software Designers with Integrated, Domain-Oriented Design Environments*, IEEE Transactions on Software Engineering, Vol. 18, No. 6, 1992, pp. 511-522.
  10. C. Potts, K. Takahashi, J. Smith, K. Ota. *An evaluation of inquiry-based requirements analysis for an Internet service*. Proceedings of the Second IEEE International Symposium on Requirements Engineering (RE'95) 1995. pp.172-80.
  11. J. Greenbaum, M. Kyng (eds.), *Design at Work: Cooperative Design of Computer Systems*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.
  12. G. W. Furnas, T. K. Landauer, L. M. Gomez and S. T. Dumais. *The vocabulary problem in human-system communication*. Communications of the ACM 30, 11 (Nov. 1987), pp. 964 – 971.
  13. L. Foner. *Yenta: a multi-agent, referral-based matchmaking system*. Proceedings of the first international conference on Autonomous agents, 1997, Pages 301 – 307.
  14. D. Redmiles. *Reducing the Variability of Programmers' Performance Through Explained Examples*, Human Factors in Computing Systems, INTERCHI '93 Conference Proceedings, ACM, April 1993, pp. 67-73.
  15. D. Bobrow, T. Winograd, *An Overview of KRL, a Knowledge Representation Language*, Cognitive Science, Vol. 1, No. 1, 1977, pp. 3-46.
  16. W. Kintsch, J. Greeno, *Understanding and Solving Word Arithmetic Problems*, Psychological Review, Vol. 92, 1985, pp. 109-129.
  17. E. Soloway, K. Ehrlich, *Empirical Studies of Programming Knowledge*, IEEE Transactions on Software Engineering, Vol. SE-10, No. 5, September 1984.
  18. C. Rich, R. Waters, *The Programmer's Apprentice*, Addison Wesley Publishing Company, Reading, MA, 1990.
  19. M. Minsky, *A Framework for Representing Knowledge*, in P.H. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill Book Company, New York, 1975, pp. 211-277.
  20. R. Schank, R. Abelson. *Scripts, plans, goals and understanding*. Lawrence Erlbaum Associates, Hillsdale, N.J., 1977.
  21. C. Fillmore, *The case for case in Universals in Linguistic Theory*, E. Bach and R. Harms (eds.), Holt, Rinehart and Winston, New York, NY, 1968.
  22. C. Riesbeck, R. Schank, *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1989.
  23. J. Kolodner. *Case-based reasoning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.
  24. P. Johnson-Laird. *Mental models : towards a cognitive science of language, inference, and consciousness*. Harvard University Press, Cambridge, MA, 1983.
  25. A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, M. Goedicke. *Viewpoints: a framework for integrating multiple perspectives in system development*. International Journal of Software Engineering and Knowledge Engineering, Vol. 2, No. 1, 1992, pp. 31-58.
  26. D. Norman. *Psychology of everyday things*. Doubleday, New York, NY, 1990.
  27. A. Turner (ed.), *Mental Models and User-Centered Design*, University of Colorado, Boulder, CO, Technical Report, No. 88-9, 1988,