Knowledge Management and Knowledge Agents in Campiello

Michael Koch[1],

[1] Xerox Research Centre Europe, Grenoble

Abstract

The area of ‚Knowledge Management' is closely related to CSCW as it involves collaboration in various ways. In this paper we outline our ideas for using collaborative filtering as one element of a knowledge management service in the project Campiello. We also detail what different types of agents we envision for the knowledge management and collaboration service in Campiello and briefly present the current status of the Knowledge Agents Platform which we build for supporting the development and the operation of this agent set.

1 Introduction

Thanks to networked organizations a lot of information is available at the workplace now. The problem with this situation is that the users usually cannot make use of this information because there is not sufficient support for finding information relevant for the current tasks.

This problem is currently addressed by different initiatives under the umbrella of ‚Knowledge Management'. The main idea here is to set in place procedures and technologies for giving each user relevant customized information. In other words, Knowledge Management is support for understanding and communication of work critical information. Therefore, Knowledge Management deals with usage of information, acquiring, searching, structuring and distribution.

As mentioned by several authors, this task has a collaborative aspect - support for making use of information is best handled by reusing information already collected by others or by making use of and supporting existing relations among people (communities). This collaboration in making use of information can be direct synchronous collaboration [Twidale et al. 97] or more asynchronous exchange and enriching of information in the context of collaborative filtering [Glance et al. 98].

At the Xerox Research Centre Europe we have been working on support for collaborative work for some time now [Andreoli et al. 98], and have started to extend this vision to support for knowledge management recently [Andreoli et al. 98] / [Glance et al. 98].

In this paper we will present how we want to combine knowledge management issues and community support issues in a community network like system, which we are developing in the Campiello project. Thereby, we mainly concentrate on how collaborative filtering can be used in the context of knowledge management. After a general discussion we outline which agents will be included in the Campiello set and present the platform we are using to implement and operate the agent set.

2 Campiello - Community networks as a mixture of Knowledge Management and support for direct collaboration

Computer-based community networks are a recent innovation. They are intended to help revitalize, strengthen, and expand existing people-based community networks much in the same way that previous civic innovations (like public libraries) have helped communities historically [Schuler 96]. In the project Campiello[1] we are currently building a community network like system for facilitating the creation of connected communities in towns, which have a rich culture and hence large numbers of tourists. The major objective of the project here is to better connect the local inhabitants, to make them active participants in the construction of the cultural information and also to support new and improved connections with cultural managers and tourists. This is achieved by creating a bi-directional exchange of information about the town, its places and events.

More precisely:

- We will provide an information space that can be extended by all participants. Possible extensions are new information items, and comments and ratings for

[1] Campiello (Esprit Long Term Research Project 25572) has started in September 1997 and will last until August 2000. The main consortium members are Domus Academy Research Center Milano, Department of Information Science Milano, Xerox Research Centre Europe, and Multimedia Information Systems Laboratory Crete. See [Campiello] for more information on the project.

existing information items. Access to this information space will be provided through collaborative filtering. That is, the knowledge about the user derived from his earlier usage of the system and information from other users will be used to determine which items to display for a query or which items to recommend pro-actively.

- In addition to the indirect collaboration through the information space, there will be means for direct collaboration. These will mainly be awareness services (synchronous and asynchronous awareness).

Hence, we will have classical knowledge management services (search, recommendation) and classical collaborative services (awareness). These services have to be connected. Hence, the search and recommendation services will take communities into account for their operation. The information for the knowledge management related user profiles will come from the collaboration related awareness services.

For supporting this we are currently building an agent architecture that encapsulates the different knowledge management and collaboration services and lets them interact. Goal of this approach is to have a modular extensible system that also can be used in other scenarios (like workplace scenarios).

Before addressing the agent approach we will first detail the core of the system, the collaborative filtering a little bit more.

3 Collaborative filtering: recommendation, search, and community data

Collaborative filtering in general deals with reactive and pro-active search for relevant information in a large information pool. In contrast to classical filtering collaborative filtering takes contributions of other people and relations to other people into account when deciding which items to select. As basic elements of systems that support the different forms of collaborative information search we identified three modules: recommender module, search module, and shared data space.

3.1 Recommender module

The recommender module receives ratings of information items and classifications of those items into community categories. It outputs personalized recommendations in a community context. The heart of the recommender module is how it decides what to recommend, how strongly to recommend it, and with what confidence. One approach is to employ rule-based filters specified by the user, such as "show me all items highly rated by". Another approach is to use statistical techniques that weight more highly ratings of recommenders whose preferences correlate more highly with the user (automated collaborative filtering).

3.2 Search module

While the recommender module covers the pro-active part and is acting without explicit queries, the search module takes as input a query and a context and returns a set of data items. The context and community data is taken into account for doing some pre-processing on the query and post-processing on the results. So, the search module might refine a query before relaying it to the retrieval engine. The results from the retrieval engine might be ranked end sorted according to context and community data. For the latter it is possible to pass search results to the recommender module and ask for predicted recommendation confidence.

As the user should be able to search over a wide range of information sources in a transparent way, the retrieval engine can be seen as a hierarchically structured set of agents. A layer of so-called wrappers hides the idiosyncrasies of the different data repositories together with their different search syntax, ranking schemes, and formatting of results. In this way, a user conducting queries through the search module gets the look and feel of a 'federated' homogeneous database.

3.3 Shared data space

The modules discussed above need some information about the users and about their relations to each other (communities) to work. Therefore, there has to be a common data source that collects these user profiles and community maps.

Also part of this data space will be additional information on documents. Examples for this are ratings or comments from the users.

In addition to storing information delivered from outside, the shared data space also can and should also include modules that work with this information and generate new information. Examples for those services are: competency mappers that can help to identify centers of expertise within the organization; community mappers that help to identify extended communities of interest; content mappers that help to identify to what extent collective knowledge is represented in on-line digital repositories.

4 Knowledge Agents in Campiello

All these components have to fit together and the system should be tailorable and extensible (to keep the possibility to easily use the components in other setups). That is the main reason why we decided to model the services as agents. Because the agents can be seen as building blocks for any knowledge management service, we call them Knowledge Agents.

Derived from the collaborative filtering scenario from the previous section, we have the following agents in the initial agent set:[2]

- data storage/management agents (community data, user data = user profile, comment/feedback data)
- wrapper agents (wrap existing information repositories to be used by search)
- search agent(s) (use wrapper, recommender and storage to find results for a query)
- data generation agents (e.g. community mapper - generate new data from user profiles)
- recommender agent

In addition to that we have awareness agents (sensors and display) and agents for performing document transformations. Examples for the latter type of agents are agents for converting document formats but also agents for helping with translations (translation aid agents) or agents for summarizing content.

All these agents form a service that can be used by existing user interfaces or that can offer own user interfaces. In Campiello for example we are currently implementing different types of user interfaces that will access the same core services.

5 The Knowledge Agents Architecture

For building the Knowledge Agents we began to implement an infrastructure layer that supports system engineers and system operators in easily programming, combining and operating a set of distributed agents. This infrastructure layer is called Knowledge Agent Platform.

The following main requirements had to be supported:

- distributed agents
- agents in different programming languages
- dynamic reconfiguration

---

[2] This initial agent set is currently built by adapting components that have been implemented as stand-alone applications here at Xerox Research Centre Europe: Knowledge Brokers [Andreoli et al. 96] and Knowledge Pump [Glance et al. 98]. Knowledge Brokers is a meta search service that can break up explicit queries and use different data repositories or other search engines to find results. One part of this framework are already agent like modules that wrap existing data sources. Knowledge Pump is a recommendation system that covers the areas of collecting and storing community and feedback data and of doing collaborative filtering on top of this data.

- efficient RPC/local procedure call communication and open agent message communication

The Knowledge Agent Platform has to provide functionality

- for finding agents to communicate with
- for communication among agents (and among user interfaces and agents), and
- for administrating a set of agents.

As main implementation language for agents and for the platform we have chosen Java. To be open to other programming languages and platforms, we decided to base the development on an existing distributed objects/agents toolkit, that provides distributed systems functionality (e.g. remote object references, remote function calls). After evaluating existing systems we have chosen the Voyager platform[3]. This platform allows us to easily build a distributed object system in Java and connect to it from every CORBA client.

Voyager itself already provides basic 'agent support'. As this did not satisfy all our needs, we added some components to the basic platform:

- *directory and broker service*: Maintain a list of all agents; provide functionality to search agents using different attributes
- *administration tools*: Set of tools to set up, change and monitor the agents during runtime.
- *agent stub classes*: Java classes wherefrom agent classes have to be derived. The functionality of these classes provides easy access to the directory service and to the support classes, and enables the agents to interact with the administration tools without additional programming.
- *support classes*: Several classes that implement functionality needed by several agents. Examples for such classes are classes that offer functionality for agent messages, authentication, …

We cannot describe the Knowledge Agent Platform in detail here, but we will give some more details on the following topics: agent communication, directory and broker service, and administration tools.

5.1 Agent communication

For communication among agents two possibilities are available: RPC-like interfaces and more ‚agent-like' FIPA/KQML messages.

The interfaces offer an easy to implement way to directly call methods an agent exports. When an agent registers at the directory service, this service determines which interfaces the client supports and stores them in its database. For using

[3] See http://www.objectspace.com/voyager/ for more information.

these interfaces client modules can ask for a stub object and directly call the methods using the stub object.

In addition to the interfaces we also included a more ‚agent-like' way of communicating by messages. The current implementation supports FIPA agent messages, for future versions we are considering KQML agent messages. As data format inside the standardized messages we are mainly using key value lists or the more general signed feature constraints.

By experimenting with these two ways for communication we learned, that there is no general difference among them. Both provide a standard for exchanging messages among agents. The interface communication uses procedure names and parameters, the agent messages communications uses performatives, onthologies and free message content. The main difference here is that the RPC-like communication provides one kind of conversation scheme only, the agent message standards provide the possibility to use different conversation types. With both communication schemes you can build a communication space that is very restricted or quite open for new requirements. The openness mainly depends on the granularity of actions you define and on the data structures you use for exchanging data. This facts in mind and driven by business requirements of our customers, we currently focus more on defining open interfaces than on developing the agent messages scheme.

5.2 Directory and broker service

The directory and broker service stores for each agent

- a list of the interfaces it offers (automatically derived in the Java environment),
- a set of attributes describing the features of the agent (explicitly set by the agent when registering)

The service offers methods for querying this information and for automatically recommending agents that fit best some given requirements.

5.3 Administration tools

For developing agents and for tailoring a running system, the Knowledge Agents Platform currently provides the following set of administration tools:

- *log tool*: Log or error messages from the distributed agents are sent to all log tools that register themselves in the system to be displayed there.
- *list tool*: This tool provides interactive access to the contents of the directory and broker service, and offers the possibility to start new agents or stop running agents.
- *message tool*: This tools allows to interactively construct an agent message, send it to any agent, and then display possible result messages.

6 Conclusion

In this paper we briefly introduced Knowledge Management as an area that is closely related to CSCW. We gave a summary of some of our ideas for building support for (collaborative) knowledge management, and thereby introduced our idea of Knowledge Agents and the agent infrastructure we are currently building therefore.

Despite the question of how to break up the functionality into agents we found another demanding question in what level of generality to provide for agent communication and how to do it. Our experience showed that not choosing among FIPA/KQML and CORBA/RPC is relevant here, but defining service classes and the defining an open data format for service requests and replies. Doing this in the Knowledge Management area is still part of our ongoing work. Only when this is accomplished we might be able to make use of the extended conversation possibilities in FIPA/KQML.

Acknowledgements

The work on collaborative filtering, collaborative search, and the design of an agent platform therefore stemmed from efforts by the entire Coordination Technology group at Xerox Research Centre Europe. The author would like to thank all members of this group for their contributions and fruitful discussions, and especially Natalie Glance and Antonietta Grasso for contributing the basic ideas to Chapter 3 of this paper. The work on Campiello is supported by the EC (ESPRIT LTR #25572).

Bibliography

[Andreoli et al. 98] Andreoli, J.M.; Fernstrom, C.; Glance, N.; Grasso, A.: The Coordination Technology Area at XRCE Grenoble: Research in Support of Distributed Cooperative Work. ACM SIGGROUP Bulletin 19(1), Apr. 1998, pp. 7-13.

[Andreoli et al. 96] Andreoli, J.M.; Borghoff, U.; Pareschi, R. (1996): Constraint-Based Knowledge Broker Model: Semantics, Implementation and Analysis. Journal of Symbolic Computation 21(4), pp. 635-667.

[Glance et al. 98] Glance, N.; Arregui, D.; Dardenne, M. (1998): Knowledge Pump: Supporting the Flow and Use of Knowledge. In: Borghoff, U.M.; Pareschi (eds.): Information Technology for Knowledge Management. Springer Verlag, pp. 33-53.

[Schuler 96] Schuler, D. (1996): New Community Networks: Wired for Change. Addison-Wesley.

[Twidale et al. 97] Twidale, M.B.; Nichols, D.M.; Paice, C.D. (1997): Browsing is a Collaborative Process. Information Processing and Management 33(6), pp. 761-783.