To be presented at COOPIS '97
June 25-27, Charleston, South Carolina

An Architecture for Intelligent Resource Agents

Jon A. Pastor, Suzanne Liebowitz Taylor,
Donald P. McKay, and Robin McEntire
Lockheed Martin C^2 Integration Systems
Advanced Software Research
Paoli, Pennsylvania

Abstract

This paper describes an intelligent resource agent architecture comprising several classes of mediating agents, which interact cooperatively in a distributed environment to enable appropriate, timely, customized access to Internet resources. These agents communicate with each other in terms of a shared set of object definitions, encoded in a common message syntax. Messages are transmitted via the Knowledge Query Manipulation Language (KQML), an emerging standard agent communication language. This architecture has been demonstrated in the domain of K–12 education, to support remote resource utilization and collaboration in the classroom within a WWW infrastructure.

Introduction

The proliferation of large network systems in general, and the exponential growth of the World Wide Web (WWW) in particular, have resulted in nearly unlimited opportunities for the large scale gathering, creation, and sharing of information. One unfortunate aspect of such systems, and especially of the WWW, is that they are still relatively unstructured collections of heterogeneous resources.

Imagine a third-grade teacher who wants to find current online resources to supplement a planned space science unit. This teacher is not sophisticated technically, is unfamiliar with the subtleties of using search engines like Lycos and Infoseek, and has little time available for class preparation. A keyword search for "space and planets" submitted to Infoseek produces a list of literally hundreds of possibilities. Since standard WWW search interfaces are purely keyword-driven, and have no notion of the purpose of the search ("material suitable for a third-grade science class") or any particulars about the user ("collaborative activities ideal; prefer images to text; special interest in Jupiter"), the list contains mostly irrelevant or uninteresting information, and—even worse—may miss items that would be nearly ideal. For example, twelve pages down in the list is the perfect resource—a collaborative activity sponsored by NASA that would really get her students excited and involved—but it is buried in the sheer volume of data. A recent set of spectacular Jupiter images doesn't even appear in the list, because the captions happen not to contain the search terms ("space" and "planets"). Very few valuable, interesting, or relevant resources are found in the 15 minutes that the teacher has before her next class, and she dismisses the WWW as a waste of time and energy.

To assist users like this teacher in obtaining useful, structured information—rather than a chaotic mass of largely irrelevant data—we have developed a set of specialized intelligent resource agents that serve as mediators between the user and Internet resources. These agents interact cooperatively in a distributed environment, and are accessible from within a WWW infrastructure. This system will eventually employ over a dozen agents to perform a variety of tasks related to supporting and enhancing the use of the Internet as an educational tool. Agents in the current implementation perform tasks ranging from remote database access, to dynamic composition of HTML pages for the display of appropriate resources, to customization of both the form and content of those pages to an individual user's preferred styles. An early prototype

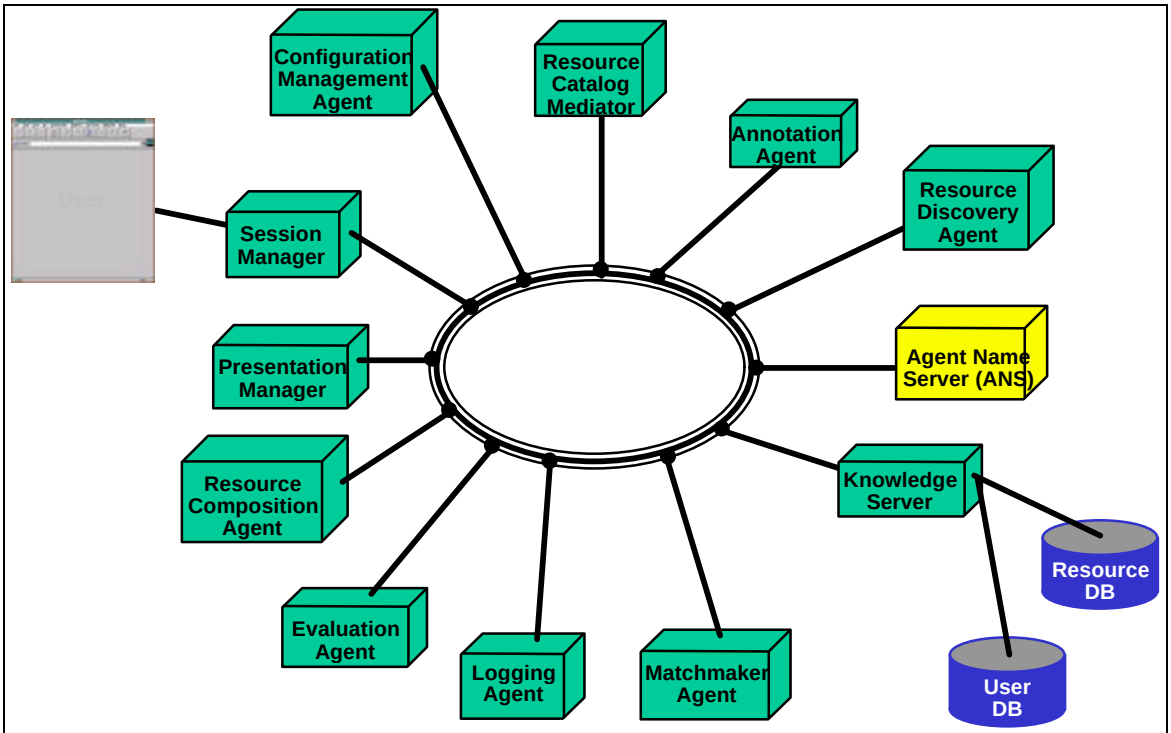

Figure 1: Intelligent Agent Architecture

in the domain of K-12 education is in use with teachers through the Defense Advanced Research Projects Agency (DARPA) Computer Aided Education and Training Initiative (CAETI) program.

In this paper, we describe the components of the overall system architecture and its implementation, including the agents themselves, the communication protocols, and the shared domain model. This application is just one example of the power of well-designed distributed agent architectures to maximize "plug-and-play" compatibility and reuse.

System Architecture

System Components

The first step in constructing a large-scale system is identifying the key functionality required, and ascribing each functional unit to a separate module. In the case of an agent architecture, these modules are embodied as independent applications, interacting cooperatively in the distributed environment. The overall system requires the coordinated application of a wide range of conventional technologies, including information retrieval, knowledge bases, databases, as well as domain-dependent technologies such as intelligent custom generation of HTML pages to suit purpose and preferences of the individual user. Our analysis of the problem domain suggested that the requisite functionality was best factored into approximately a dozen agents, as depicted in Figure 1. The resulting system is accessible from within a WWW infrastructure, interfacing with the users via standard WWW languages and protocols—HTTP, HTML, JavaScript, and the Common Gateway Interface (CGI).

The overall Intelligent Resource Agent architecture consists of the agents themselves, common message exchange protocols and syntaxes, and a common domain description or *ontology*.

Information about the users, and meta-information about Internet resources, is maintained in external persistent stores, and accessed via specialized agents ("knowledge servers"). This information is used by the agents to customize interfaces, personalize searches for web resources, and generate appropriate customized displays of search results. The knowledge server architecture itself is described in [Pastor 92, Pastor 94, McKay 96]; it is based on the concept of a mediator between information consuming agents and information providing agents [Wiederhold 95, Wiederhold 92].

Agent Descriptions

The agents of Figure 1 fall loosely into three categories:

- intelligent resource agents,
- generic intelligent systems agents, and
- core infrastructure agents.

Intelligent resource agents implement functionality that is appropriate for cooperative information systems, applied to the problem of information resource identification and use in the WWW and similar areas (e.g., digital libraries), but not necessarily generalizable outside the information resource domain. Generic intelligent systems agents are components that are generally reusable across instances of the intelligent resource agent architecture, and are designed to be easily customizable for a given application. Examples of this class of agent include:

- evaluation agents, which are used to measure and report on system and resource utilization, and
- information mediators, which provide persistent storage in terms of the common ontology.

Core infrastructure agents are components that are required to support inter-agent communication and system operation.

Intelligent Resource Agents work together to enable appropriate, timely and customized access to distributed information sources. In our particular case, these agents target multi-media Internet resources. This includes creating customized interfaces to assist the user in finding potential sources of information, identifying relevant information, and forming new resources based on this information which are presented to the user in an appropriate manner (e.g., a summary of findings customized for the user).

The **Resource Discovery Agent** retrieves and collects information from a variety of dispersed, multimedia information resources. This agent is responsible to keeping track of resources in this dynamic environment—i.e., resources may change, move to different locations, or even disappear at any time. This collected information is a primary source for the resource catalog. In our implementation, the resource discovery agent is akin to a *web- crawler*, searching the Internet based on known educationally-relevant seed sites, and indexing resources. The resource discovery agent contributes some meta-data about the resource, along with content-based indexing, to help populate the catalog.

Resource cataloging technology then provides the means for representing, integrating, and acquiring knowledge about the collection of information resources through an analysis of the objects and relationships among them. The **Resource Catalog** provides a conceptualization and description of information resources which adds semantic structure for a domain-specific use. The **Resource Catalog Mediator** allows users to quickly, intelligently, and productively access information sources. Where most search engines available on the WWW search strictly

on content, via Boolean combinations of keywords, the resource catalog provides a database of structured meta-information on relevant resources. This allows the users to target their searches in useful ways, and does not rely solely on matching key words in the text. For example, a third-grade teacher can ask for resources specifically targeted for the appropriate grade-level. What distinguishes this resource catalog from conventional digital libraries is that it is a highly-dynamic mix of "automated"—e.g., captured by resource discovery—and "user driven" collections, a live and evolving entity to which users contribute new resources. User contributions may be may be original works. interesting resources found on the net that are not already in the collection, or even comments on existing resources via annotations.

Support for user-annotation of resources in the collection is one of the most unique and interesting aspects of this catalog. Annotations become a searchable portion of the catalog, permitting one teacher to benefit from the experience of others, and even to search directly on aspects of the annotations. Users are encouraged to contribute, and are provided tools for producing, both structured (e.g., rankings) and unstructured (e.g., free-text) annotations

Matchmaking extends the concept of a resource beyond conventional online multimedia resources to include human resources, people who are available as mentors, experts, or potential collaborators. A teacher can now contact an expert on comets who will answer questions posed by her inquisitive third graders. Users can register their interests and their willingness to act in various mentoring and/or collaborative relationships; other users can then search for "people" resources to assist them in planning or executing classroom activities. Matchmakers come in two flavors: *reactive* and *proactive*. *Reactive* matchmakers respond to a specific user request, while *proactive* matchmakers observe the online behavior or interest of users and try to arrange matches. For example, a proactive matchmaker might notice that ten users are teaching Space in their third grade class, and "suggest" via e-mail that they start a discussion group or consider participating in a collaborative activity on this topic.

Resource Composition provides customized presentation of web resources, interfaces to the intelligent agents, or results from search queries. When the user queries the resource catalog, the results are presented in an appropriate fashion based on user preferences. The object is to organize the resources into a form that suits the functional and aesthetic requirements of the individual user, rather than to the typical output of search engines: simple and uninformative hit lists based on titles, keywords, or source location. The teacher interested in images of Jupiter, finds resources which are image-rich, as well as short summaries of what each resource provides, and comments from other users on how they used that resource.

In a similar fashion, interfaces to the intelligent agents (e.g., catalog search) are customized for ease of use and comfort-level to the particular user, based on profiles that include both information about the user (e.g., *teacher* or *student* status) and expressed preferences (e.g., brief or detailed summary results). For example, teacher and student interfaces might have a different "look," K-3 interfaces might differ from high school level interfaces, and so on.

The **Session Manager** and **Presentation Manager** are works in progress, designed to address the challenging problem of persistence within a WWW "session," and are discussed briefly in Section 0 on future directions.

The **Collection Agent** serves as a repository for information on system performance and utilization, fed by messages from all appropriate agents in the system, and even from HTTP server logs. Once raw data has been collected by the collection agent, this data is persistently stored for later use by other evaluation agents in the system. The **Evaluation Agent** operates on this data, providing a wide range of usage and performance statistics; these can then be used

to track and evaluate relevant aspects of system and user behavior, and to tune both performance and appropriateness.

The **Knowledge Server** represents a class of agents that provides the other agents with persistent information; in the system described here, persistent information is needed about users, and about resources in the catalog. A Knowledge Server mediates between the applications and external persistent stores, providing an object-based view of the raw external data, in terms of the domain ontology. The current Knowledge Server is implemented in Common Lisp using the Lockheed Martin Interface Module (LIM) and the Loom knowledge representation language; a replacement, to be written in the Java language, is currently in the design stage.

The other agents represented in the architecture are components of the underlying infrastructure. The **Agent Name Server (ANS)** keeps track of agent registration, and permits agents to locate and address other agents by name (rather than, say, IP addresses and ports). The **Configuration Management Agent (CM Agent)** monitors the status of all agents in a defined agent configuration/system, and provides the ability to reconfigure that system and start, stop, suspend, or resume agents in the system.

Inter-Agent Communication Language

This system, like many other software systems, is structured as a collection of independent processes, distributed across multiple hosts linked by a network. The Knowledge Query and Manipulation Language (KQML) is used as the communication protocol among the agents [Finin 95, Finin 94a, Finin 94b, Mayfield 96]. KQML is a language and a protocol that supports extensible network programming, specifically for knowledge-based systems and intelligent agents [Neches 91, Patil 92].

Lockheed Martin's implementation of KQML, used in this current work, contains two primary layers. The outer layer, called a ***router***, provides message routing functionality to its associated application agent, and handles the establishment of all communication links to other agents in the system. The inner layer, called a ***KRIL*** (KQML Router Interface Library), provides a library of application-level interface routines. The ***router*** module makes use of the Agent Name Server (ANS—cf. Section 0) to determine addresses for agents and services within the system. Routers function *independent* of message content. Each agent has a separate router process; he router handles all incoming and outgoing message traffic for its associated agent.

The router is a separate module from the application, and it is necessary to provide a programming interface between the application and the router. This application program interface is called the ***KRIL***. While the ***router*** is a separate process with no understanding of the content of the KQML message, the ***KRIL*** is embedded in the application and has access to the applications methods for understanding the content of the KQML message—in this specific case, the objects that are passed among the agents. In addition, the ***KRIL*** has access to the application objects that must be transmitted from one agent to another. The ***KRIL*** is responsible for encoding these application objects into the communication objects that are carried in the body of KQML messages, and, conversely, for recomposing these communication objects into appropriate domain objects when a message arrives at the receiving agent. The ***KRIL*** will also compose the complete KQML message, ready for transmission. Using a set of verb-like tags, called ***performatives***, such as `ask-one`, `ask-all`, `register`, and `advertise`, these syntactically correct messages are created in the ***KRIL*** and are then handed to the ***router*** to be dispatched to appropriate agents in the system.

Shared Communication Ontology

In order for an agent to be able to "understand" the content of a message, it must be stated in terms that make sense to the agent: the nomenclature used by the sending agent must be comprehensible to the receiver. In human speech, we take for granted that references to objects, conditions, and other terminological entities are comprehensible to all parties in a conversation—that all parties share a set of concepts that have essentially the same characteristics in the minds of all participants. A ***shared communication ontology*** is a formalization of this notion, ensuring that when one agent refers to, say, a "resource", this evokes the same feature set, in terms of both structure and behavior, to the receiver as to the sender.

A key feature of our architecture is that it specifies the common ontology in a representation that subsumes the models of the individual agents. The practical implications of this are

- any model required by any agent can be expressed in the common ontology;
- translations from the common ontology's formal representation are lossless with respect to the agents' models;
- consequently, the agents are not constrained to employ the same modeling language—or, in fact, the same programming language.

As a result, it is possible to build hybrid systems consisting of modules developed in different languages, on different platforms, using different representations or *views* of a common set of objects; the Intelligent Resource Agent system includes modules in C, C++, Common Lisp, Perl, and HTML, all interacting freely with one another without regard for representation details, language, or platform.

In our current system, while the internal program data structure representations used by different agents may be encoded differently due to language differences, it is typically the case that the agents' models are either isomorphic to, or proper subsets of, the communication ontology. This is neither necessary nor necessarily desirable, and in this application we have begun to introduce a separation between the *application* ontology manipulated by the agents themselves and the *communication* ontology used for inter-agent communication.

We view the ability to translate bi-directionally between the communication ontology and a collection of application ontologies as being critical to the long-term success of distributed agent architectures, since it places few—if any—restrictions on the kinds of applications that can participate as agents. An example of this principle is embodied in the Knowledge Server which is a *mediator* agent that provides access to databases in which persistent data for the catalog, among other things, is stored. The Knowledge Server is able not only to translate the flat structure of the relational model into the hierarchical structures of the communication ontology, but also to perform translations of encodings and modalities. For example, a database field that contained a string naming a specific MIME type (e.g., "image/gif") could be mapped to a conceptual type in the ontology that represented that MIME type (e.g., an object of type **GIFimage**. An encoding of this object, described in terms the communication ontology, would then be sent via a KQML message to the requesting agent. The significance of this observation is that other agents can be written in terms of the underlying ontology—a **GIFimage** object—rather than a specific text string—"image/gif".

Implementation

Intelligent Resource Agent Architecture

The current system consists of the following components:

- a Common Lisp-based knowledge server mapping from a Loom knowledge base to an Oracle database
- a collection of intelligent resource agents, primarily implemented in C++, that communicate with both the knowledge server and each other
- a communication infrastructure that supports inter-agent communication, which includes the router and KRIL functionality used by each agent, a number of utility agents, such as the ANS, that provide assistance for message traffic, and encoding/decoding routines that allow the transmission of application-level objects from one agent to another

Agents wishing to communicate with other agents are thus presented with a true object-oriented model of the communication acts, and are insulated from the details of KQML message composition and encoding.

The programmer API for database access consists of a handful of methods, for those agents requiring external data; these methods further encapsulate the KQML protocol so that requests for data are stated in terms of actions on objects of the desired type. All of the KQML and database functionality is defined at the level of abstract classes; objects requiring KQML transmission, or both KQML transmission and database access, are derived from these abstract classes and inherit the requisite behavior.

Loom Concept Definition

```
(defconcept EducationalStandard
    :is-primitive
        (:and Meta-View-Concept
            (:the StdName String)
            (:the StdId StandardId)
            (:the SourceId SourceType)
            (:all Objectives Objective)
            (:the Principle String)
            (:the Text String)
    )
)
```

C++ Class Definition

```
class EducationalStandard : public OllaDBClass {
        private:

            RWCString StdName;
            RWCString StdId;
            RWCString SourceId;
            ObjectiveList Objectives;
            RWCString Principle;
            RWCString Text;
    . . .
};
```

Table 1: Mapping from Knowledge Base Objects to application-level objects.

In all present cases, as illustrated in Table 1, the C++ classes are structurally similar to the Loom concepts defined in the shared domain model—the application model is nearly isomorphic to the communication ontology. The C++ class definitions include not only the structural elements and the inherited KQML and/or database behavior, but also a standard set of methods for *encoding* and *decoding* C++ objects to and from a canonical ASCII representation used to represent objects in transmitted messages. While the form of this representation is actually arbitrary, as long as it can encode the application data structures unambiguously, it must be standardized among all agents in order for them to be "plug-compatible".

Given the close correspondence between the Loom domain model and the C++ object definitions for the application model, we chose to provide for automatic generation of the C++ classes from the Loom KB: both the data members and a working subset of the required function members (methods) are generated by a Common Lisp function that walks the Loom KB and processes concepts corresponding to externally-useful objects.

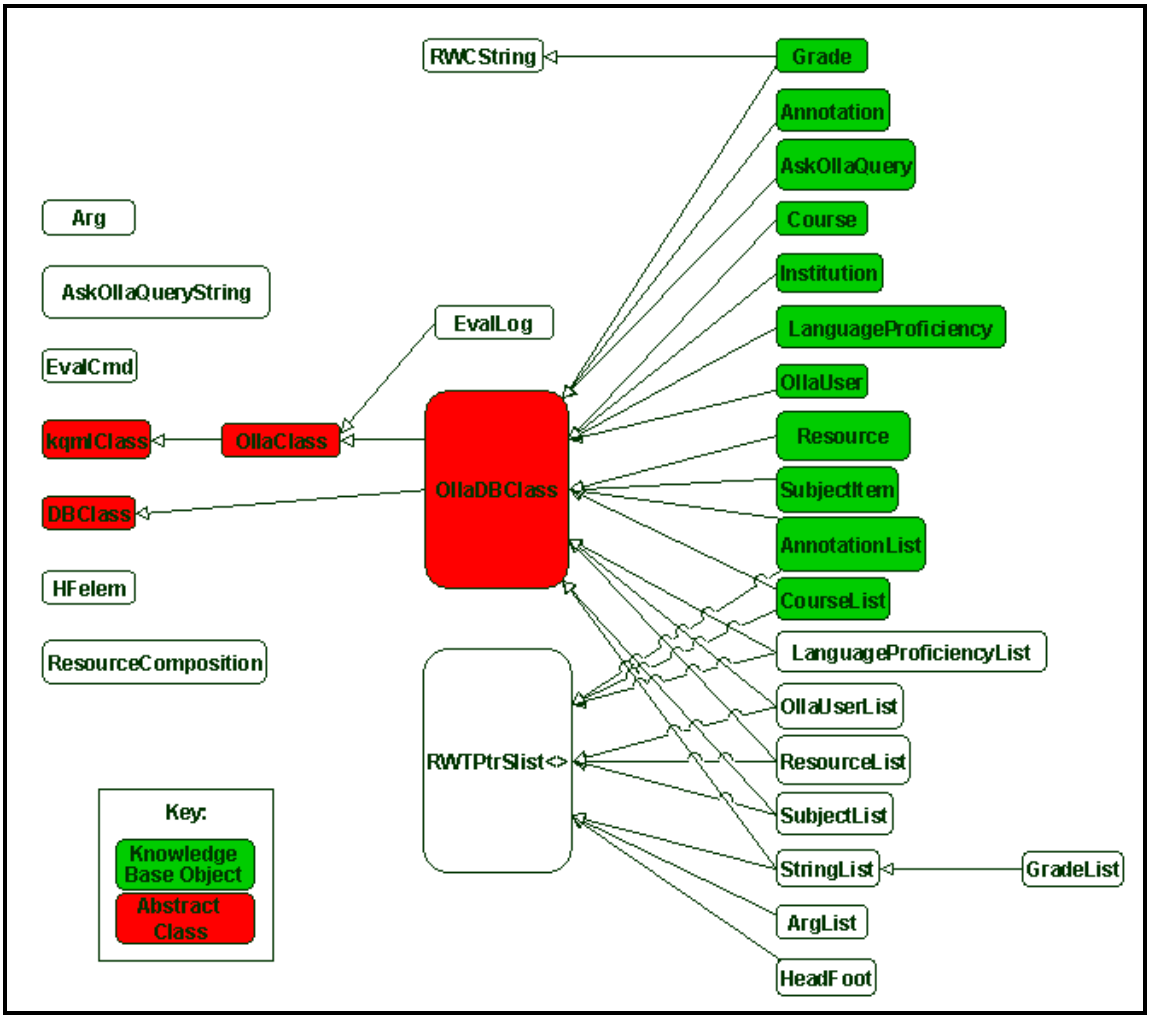

Figure 3**:** Shared domain ontology

The C++ class architecture for our Intelligent Resource Agent system is depicted in Figure 3. Communication acts, and methods on application-level objects, are mixed to create hybrid methods that incorporate the power of both communication-based and object-based acts. Using these classes, a domain-relevant class such as `Resource`, which implements a resource object in a resource catalog, can be defined in terms of its unique defining attributes, and inherit its communication and data base aspects.

A critical point to observe about Figure 3 is that it depicts not only the structure of the knowledge-base objects used to represent the abstract ontology, but also the structure of the application-level objects in an object-oriented language, in this case C++. That is, the agents manipulate objects that are isomorphic, in terms of structure and taxonomy, to the underlying

domain ontology. Table 1 shows a knowledge base definition, in the Loom language, side-by-side with the corresponding portion of the C++ class definition used when exchanging information about educational standards.

Each C++ class derived from `kqmlClass` defines a pair of C++ methods—`Encode()` and `Decode()`—to create and interpret (respectively) a canonical text representation that is used as the content language for KQML transmission; it also inherits a set of methods corresponding to a subset of the KQML performatives (communication primitives). In addition, some agents define higher-level methods that encapsulate the appropriate translation and encoding of application-level objects, KQML message formation, transmission, and reception; and decoding and creation of C++ objects. In fact, from an agent's perspective, it is manipulating instances of the application ontology classes, in some cases unaware that some of the objects it is manipulating have been retrieved from remote locations.

Application to K-12 Education

Figure 3 shows the interplay of key intelligent resource agents to assist our teacher in finding the appropriate resources for her third-grade science class. The teacher makes a specific request through the form catalog form interface on her WWW browser, insulating her from the details of the underlying query language. The catalog mediator—implemented as a Common Gateway Interface (CGI) program—interprets the form submission, initiates the search based on her selections. The catalog mediator checks the collections created by the discovery agent, filters the results through the catalog, and then passes the resulting resource information to the composition agent. The composition agent queries the user server for information relevant to presentation of results, and returns a customized HTML page—based on the original query, the resulting resources, and the user information—to the catalog mediator. Finally, the page is

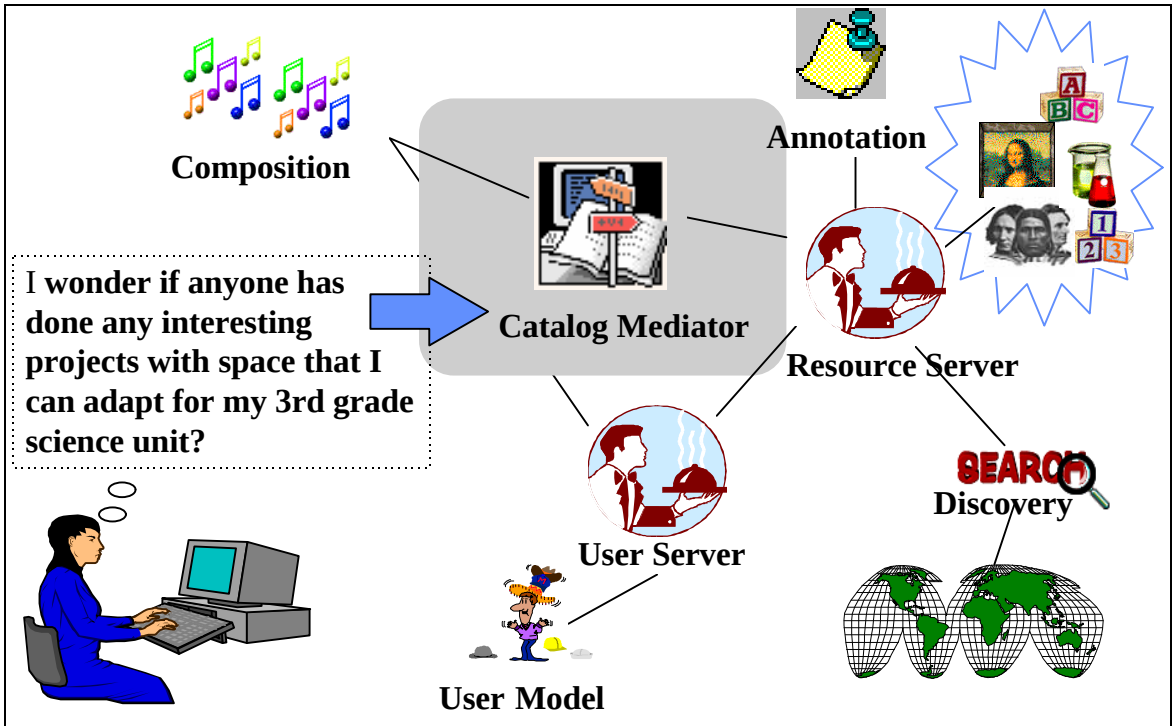

Figure 4: Resource agents work together to gather customized resources for the user and present them in an appropriate manner.

displayed to the teacher, via the catalog mediator, in the client area of her WWW browser. Along with some excellent photo archives from the Hubble telescope, the teacher finds a wonderful *Mission to Mars* activity sponsored by NASA.

Early prototypes of this technology are currently in use with teachers in the DARPA sponsored CAETI (Computer Aided Education and Training Initiative) Department of Defense Educational Activity (DoDEA) test bed via the Online Learning Academy (OLLA), a World Wide Web (WWW) environment, which supports the use of telecomputing in the classroom [OLLA 96]. OLLA is currently deployed in four school complexes (akin to school districts) and in pilot classroom use with about twenty elementary school teachers. The intelligent resource agent architecture, and scenarios such as the one above, have been successfully demonstrated from within the OLLA infrastructure. The proactive matchmaker intelligent resource agent is the first of the intelligent resource agents to be deployed with OLLA (housed at the Franklin Institute Science Museum), connected by HTTP access, and in use by educators world-wide.

Conclusions and Future Directions

The intelligent resource agent architecture provides the basis for a scaleable and robust agent infrastructure. The key ideas include appropriate modularization of agents into reasoned software processes. In the area of information resource utilization on the WWW, we have demonstrated that the intelligent resource agent architecture is one effective way of providing tailored information services for specific applications, i.e., K-12 education and the effective use of internet resources.

Future development of this system includes refinement of the existing intelligent resource agents, as well as the addition of several new agents. Since we are working within the confines of the WWW, customization within a *session* has been challenging. To this end, **Session Manager** and **Presentation Manager** agents are under development. The **Session Manager** maintains persistence within a session, and the **Presentation Manager** controls customization of interfaces within a session. Both of these components will involve communication among processes on the client, as well as distributed client/server interactions.

In addition, we plan to continue with the general agent infrastructure that we demonstrated successfully within this application. We have begun to define aspects of communication ontologies under this project. Communication ontologies hide from application-level processes both the communication acts and the information exchange distribution necessary to achieve application-level goals. This is accomplished by including in an object's definition both the methods to be invoked on that object and the communication act necessary to satisfy the successful execution of each method. For CORBA-based implementations, this can be directly implemented via remote method invocation supported in CORBA. However, we envision significantly more stylized information exchange, including status and data item updates, as well as volunteering of newly developed information. A convenient and flexible information exchange mechanism is a key to the continued success of these systems.

We are just beginning to exploit this approach in an architecture for Information Resource Agents. We will further explore this approach and use both a formal definition of an ontology and a practical software implementation approach.

References

[Finin 95] T. Finin, C. Thirunavukkarasu, A. Potluri, D. McKay, and R. McEntire, On Agent Domains, Agent Names and Proxy Agents, *Proceedings of the ACM CIKM Intelligent Information Agents Workshop*, Baltimore, December 1995.

[Finin 94a] T. Finin, D. McKay, R. Fritzson, and R. McEntire. "The Knowledge Query and Manipulation Language for Information and Knowledge Exchange", *Proceedings of the Third International Conference on Information and Knowledge Management* (CIKM'94), November 1994.

[Finin 94b] T. Finin, D. McKay, R. Fritzson and R. McEntire, "KQML - A Language and Protocol for Knowledge and Information Exchange." *Proceedings of the 13th International Distributed Artificial Intelligence Workshop*, July 1994.

[McKay 96] D. McKay, J. Pastor, R. McEntire and T. Finin, An architecture for information agents, in "*Advanced Planning Technology*", (ed. Tate, A.), The AAAI Press, Menlo Park, CA., USA, May 1996, ISBN 0-929280-98-0.

[Mayfield 96] J. Mayfield, Y. Labrou, and T. Finin, Evaluation of KQML as an Agent Communication Language, in Intelligent Agents Volume II -- Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages. M. Wooldridge, J. P. Muller and M. Tambe (eds). Lecture Notes in Artificial Intelligence, Springer-Verlag, 1996.

[Neches 91] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. Swartout. "Enabling Technology for Knowledge Sharing". AI Magazine, 12(3):36-56, Fall 1991.

[OLLA 96] OnLine Learning Academy (OLLA) Users Manual. Lockheed Martin. 1996.

[Pastor 94] J. Pastor and D. McKay, "View-Concepts - Persistent Storage for Planning and Scheduling", *Proceedings of the ARPA/Rome Lab 1994 Knowledge-Based Planning and Scheduling Initiative Workshop*, Tucson, February, 1994.

[Pastor 92] J. Pastor, D. McKay and T. Finin, "View-Concepts: Knowledge-Based Access to Databases". *First International Conference on Information and Knowledge Management*, Baltimore, November 1992.

[Patil 92] R. Patil, R. Fikes, P. Patel-Schneider, D. McKay, T. Finin, T. Gruber, R. Neches. "The DARPA Knowledge Sharing Effort: Progress Report". In B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference* (KR'92), San Mateo, CA, November 1992. Morgan Kaufmann.

[Wiederhold 95] G. Wiederhold "Mediation in Information Systems; in Research Directions in Software Engineering", *ACM Comp.Surveys*, Vol.27 No.2, June 1995, pages 265-267.

[Wiederhold 92] G. Wiederhold, "Mediators in the architecture of Future Information systems", *IEEE Computer*, March 1992, pages 38-49.

Acknowledgments

The authors would like to acknowledge the invaluable contributions of Christine Baker, Peg Duffy, Roslyn Nilson, Christian Polizzi, Peter Stevens, and Carl Weir of Lockheed Martin; Ann Culp, Educational Technologies; and Steve Baumann and Karen Elinich, The Franklin Institute Science Museum to this project. This work was funded in part by the Defense Advanced Research Projects Agency under contract N66001-95-8631.