# Integrating Legacy Educational Applications in Modern E-Learning Environments

Diego Zapata-Rivera, Christopher Brooks, Lori Kettel and Jim Greer
Advanced Research in Intelligent Education Systems (ARIES lab), Saskatoon, SK. Canada
{Diego.Zapata, Chris.Brooks, Lori.Kettel, Jim.Greer}@usask.ca

**Abstract:** Offering interoperability with legacy applications is a challenge that many e-learning environments have to face. We have proposed and developed a middleware platform that uses an ontology based event mechanism that allows legacy applications to share information with a community of agents. Several distributed technologies are used to support this platform. This paper presents the middleware, describes how an existing application was integrated with the middle layer and mentions the current state of the project and our plans for future work.

## Introduction

Modern e-learning environments face the challenge of integrating with a variety of existing educational resources and applications. Current efforts on standardizing and cataloging Web learning resources in the form of interoperable repositories of learning objects are making it possible to think about modern e-learning environments in which information can be readily available and adapted to specific learning needs. These learning environments should not only facilitate interaction with learning materials, but also with information available from students, teachers, and other components (i.e. external educational software, simulations, and artificial tutors or companions).

Sharing information maintained by legacy educational applications and proprietary systems is not an easy task. Incremental integration using XML and Web services has been employed to maintain and prolong the life of legacy applications (Lavery et al., 2002; Sun Microsystems, 2003). Our research on intelligent educational systems has led us to experiment with different technologies. Existing educational applications in our laboratory include a learning management system, a content packaging system, a public discussion forum application, a quiz editor tool, and an instant messenger system. These applications have been developed using a variety of programming and database platforms. For example, we have deployed different databases and agent based architectures to support communication among components, to find appropriate helpers, and to share student model (SM) information (Vassileva et al., 1999, Zapata-Rivera & Greer, 2001).

New developments require sharing information maintained by different applications. We propose an ontology-based middleware architecture that allows existing applications to make available selected information to interested software agents. Using such a middle layer, agents can access information regarding student interactions with existing applications and with other students, student portfolios, student preferences, assessment information, and available learning materials. This information is maintained by various applications distributed over the network. Agents in this architecture can become service providers and offer their services to other agents in the community through the middle layer or directly by using other communication mechanisms (e.g. agent platforms). The middleware does not require agents to be part of a particular agent platform. Both applications and agents publish information through an event-based mechanism used by the middle layer to respond to different requests.

This paper describes the middleware platform and shows how information from existing applications is made available to a community of agents. We also present the current state of the project and our plans for future work.

## Middleware Platform

The middleware platform provides an ontology-based event mechanism that agents can use to get information made available by existing applications. In this platform (see [Fig. 1]), several modules (i.e. *Sentry, Student Model, and*

*Learning Materials modules*) are in charge of interacting with distributed data repositories maintained by applications. Applications should maintain log and event tables with the information they want to make available. Log tables are used to store instances of the events the application wants to publish. Event information is given to the *event manager* which responds to a variety of requests from agents in the community such as getting information regarding available events (get_EventList), subscribing to an event (subscribe_Event), unsubscribing from an event (unsubscribe_Event), and retrieving the occurrences of a particular event within a certain period of time (query_Event).

Agents can query the middleware or be notified whenever an event for which they have subscribed has occurred. The latter is done through a client callback mechanism provided by the middleware. Agents can connect to these services through Remote Method Invocation (RMI), Common Object Request Broker Architecture (CORBA) or Web based remote calls using Simple Object Access Protocol (SOAP) or XML-based Remote Procedure Call (XML_RPC).

Sentry modules are components that know about the application. These modules watch for new events while processing queries. The results of these queries can be sent or retrieved as Java objects or as XML files with their associated schemas for agent interpretation. Depending on the nature of the data repository, a single sentry module could be used across different applications (i.e. a sentry module for relational databases). Even when event information has been distributed over different data repositories, modules can be used to access such information (dashed lines in [Fig. 1]).

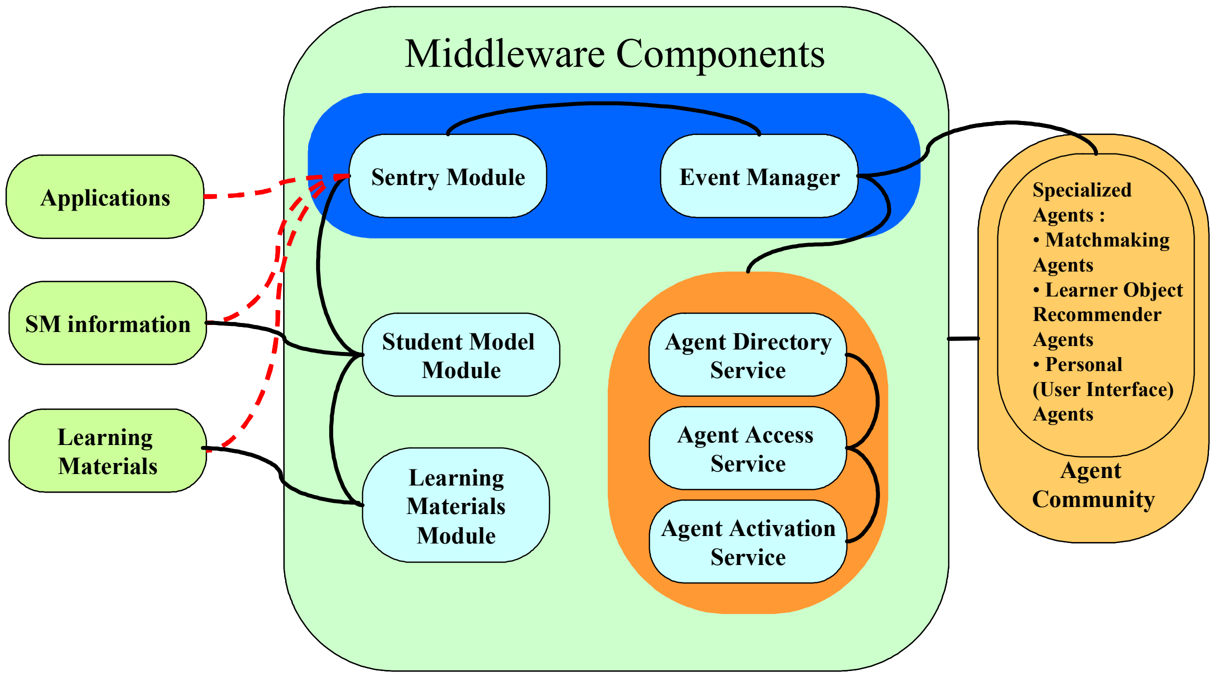

**Figure 1:** Middleware components

Various agent services have been added to the middle layer in order to locate agents on distributed agent platforms (*agent directory services*), to grant or deny access to different events (*agent access services*), and to activate or deactivate agents (*agent activation services*).

In addition to subscribing to events and receiving information from the middle layer, agents can register their own events with the middle layer and become service providers for other agents. Using this approach, more specialized services that make use of information gathered from distributed old and new sources can be provided to the agent community (i.e. matchmaking services to locate suitable peer helpers for a help request).

Applications and other components in the e-learning environment can model information under different names while referring to similar concepts. As a result this middleware uses ontologies to formally describe concepts and their relationships. An *ontology-based event mechanism* allows agents to decide which event to subscribe to, to discover equivalent concepts being modeled by different applications, and to facilitate information sharing.

## Integrating a Legacy Application with the Middleware

The I-Help Public discussion Forum, previously known as The Co-operative Peer Response System (CPR) (Bishop et. al, 1997), is a discussion forum collaboration tool. Its current version runs on Oracle server software and is accessed through a web browser (Greer et al., 2001). Users have access to a set of forums in which they can browse existing postings, create new postings, make replies or follow-ups to postings, and rate the quality or usefulness of postings. A personalized view is maintained where new postings are flagged and responses to user queries result in personal notification.

I-Help Public Forums offers an easy-to-use interface and a powerful, flexible administration system. It also logs a rich quantity of user activity and usage data. While users are interacting with the system, data is collected about their actions. This data tracks user activity such as which postings users read, how they rate postings, changes in forum subscriptions and how often users visit particular forums.

Information maintained by this application has proven to be useful for a variety of research projects such as determining the best postings to be included on a FAQ list and creating profiles on users' skills and interests. As it is done in many other applications, this system uses a variety of logs to keep track of users' interactions. Integration with the middle layer consisted of adding a table of events that maps existing information to the events required by the middle layer. In addition, access control information was defined in order to control agent access to the application information. The middleware administers this information through the Agent Access Service (AAS). Initial tests show that agents in the community can access information from available events offered by this application.

## Current State and Future Work

Although we have started using some of the capabilities of this middleware, this effort is still in its infancy. Work on defining and integrating ontologies using this approach is under way. We are also exploring configurations where agents provide specialized services such as supplying student modeling information (i.e. assessment information) to agents acting on behalf of the system.

Other areas that require future work include agent authentication and information encryption, performance testing using different kinds of clients, and distribution of middleware components to increase performance. Future work will be focused on using available information offered by new and existing applications to enhance learners' experiences with our education systems.

## Conclusions

Integration of legacy applications with new e-learning systems is a research area that needs to be further explored. The rapid evolution of e-learning technologies makes it difficult for successful integration of these applications.

The proposed middleware helps the process of incremental migration of legacy applications while ensuring interoperability with existing and new systems. It uses distributed technologies, ontologies and Web services to provide information to a group of software agents that represent other applications or human beings.

## References

Bishop, A. S., Greer, J. E., & Cooke, J. E. (1997). The Co-operative Peer Response System: CPR for Students. Proceedings of ED-MEDIA 97/ED-TELECOM 97, Calgary, Canada, 1997, Association for the Advancement of Computing in Education (AACE) Charlottesville, VA, 172-178.

Greer J., McCalla G., Vassileva J., Deters R., Bull S., Kettel L. (2001) Lessons Learned in Deploying a Multi-Agent Learning Support System: The I-Help Experience, Proceedings of AI in Education AIED'2001, San Antonio, IOS Press: Amsterdam, 410-421.

Lavery, J, Boldyreff C., Ling, B. & Allison C., (2002) Laying the Foundation for Web Services over Legacy Systems. Proceedings of the 4th IEEE International Workshop on Web Site Evolution WSE2003. On-line: http://www.dcs.st-and.ac.uk/inside/reports/wse2002_paper.pdf

Sun Microsystems. (2003). E-Learning Framework. Technical White Paper. On-line: http://www.sun.com/products-n-solutions/edu/whitepapers/pdf/framework.pdf

Vassileva J., J. Greer, G. McCalla, R. Deters, D. Zapata-Rivera, C. Mudgal, S. Grant. (1999). A Multi-Agent Approach to the Design of Peer-Help Environments, In S. Lajoie and M. Vivet (eds.) *Artificial Intelligence in Education*, IOS Press: Amsterdam, 38-45.

Zapata-Rivera, J.D. & Greer, J. (2001). SMODEL Server: Student Modelling in Distributed Multi-Agent Tutoring Systems. International Conference on Artificial Intelligence in Education AIED 2001, 446-455.

## Acknowledgments

We would like to acknowledge the Natural Sciences and Engineering Research Council of Canada for financial support through the Learning Object Repository Network (LORNET).