

A DOUBLE AGENT ARCHITECTURE FOR ELEARNING APPLICATIONS

Martti Rahkila ¹

Abstract – *This paper describes an agent-based system architecture for eLearning applications. The emphasis is on the technical structure and system components but also education-oriented functionality is described. An agent acts as an interface to all functionality in the system. Fundamentally, it is a representation of the user but it also provides interactive guidance similar to the role of the teacher, hence the name "double agent". The architecture is designed especially having self-study, WWW-based eLearning packages in mind but provides also functionality useful for other types of educational content and eLearning systems. Log-based evaluation of self-study learning is one of the key benefits of the architecture. The agent performs logging of users' actions in order to allow learning history to be used for learning evaluation in different ways for both teachers and students.*

Index terms – agent systems, computer based education, eLearning, software architecture

INTRODUCTION

The Internet gives us challenging new possibilities for teaching and education. Especially the World Wide Web (WWW, web) makes it possible to develop educational content and delivery in a whole new perspective. Today, it is possible to create interactive Computer Based Education (CBE) applications that can be easily used and accessed with web browsers. Furthermore, collaboration, administration and information exchange can be taken to a new level using the possibilities of the Internet. The term eLearning is commonly used for Internet- and WWW-related educational content and systems.

Particularly interesting areas are self-study, educational multimedia content and interactive, web-based CBE applications. WWW not only allows us to produce educational packages with animations, audio and video for illustrating and clarifying topics, but also makes it possible to create interactive content such as exercises and drills or even simula-

tions and games.

However, from technical point of view, the Internet and WWW are far from being simple or well known, established technologies. Quite the opposite, the Internet and WWW are complex systems involving various rapidly evolving technologies. Even though creation of static web pages is technically relatively easy, implementing interactive content is not: numerous technical solutions can be used and new technologies are constantly under development.

This background raises a number of questions that are of interest to teachers and educators thinking of developing interactive content: How to do it? What would be the appropriate technology? Are there other aspects that should be considered if implementing a CBE application? What if there are more than one simultaneous users or applications? How to evaluate the use of applications? This article presents a system level architecture that tries to answer at least some of these questions.

One of the fundamental issues in CBE applications is controlling the behavior of the application based on user's actions, in other words, adaptation to user's preferences and performance. In the world of commercial WWW systems (for example eBusiness solutions) this feature is typically called personalization and it usually means that the user can set some preferences related to the behavior of the system. However, educational requirements make things more challenging. Adaptation to not only user's preferences but also the level of foreknowledge and especially user's performance are essential. The "double agent" not only keeps track of user's actions but intelligently guides the user further, just like a teacher.

Needless to say, these requirements make the structure of an interactive CBE application very complex. The architecture presented here is one possible description of such a structure.

REQUIREMENTS AND DESIGN GOALS

The most important requirements and goals for the design of the architecture have been

- Session control: Capability of identifying the user and his/her activities during a session

¹Helsinki University of Technology, Laboratory of Acoustics and Audio Signal Processing, P.O. Box 3000, FIN-02015 HUT, Finland, Martti.Rahkila@hut.fi, <http://www.acoustics.hut.fi/>

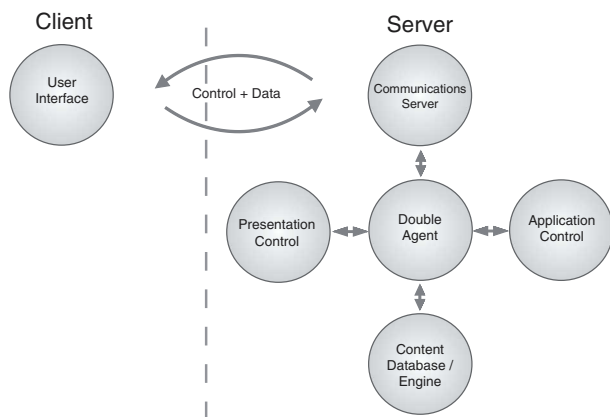


Figure 1: An overview of the architecture

- Control of interactivity
- Independence of platforms and applications
- Support for multiple simultaneous users
- Support for multiple CBE applications
- Compliance with existing technologies
- Extendability to future technologies

Many of these design principles are common to any advanced software design or Internet services. Therefore, it is obvious that the architecture can be used for purposes other than education as well. Still, the primary background and design goal has been to provide a system suitable for self-study CBE applications. A more detailed discussion of the applicability of the architecture can be found in later sections.

FUNCTIONAL DESCRIPTION

A functional description of the architecture is presented in Figure 1. From a high-level point of view the architecture consists of the user interface, the double agent and content. The essential idea is that every request made by the user is verified and processed by the agent first. Therefore, the user can only see the agent and all functionality is hidden behind the agent. Furthermore, the agent is responsible for all responses sent back to the user. Therefore, the agent also has control over what the user sees. This dual nature of the agent gave the name “double agent” and basically simulates an educational process similar to the one between a student and a teacher.

However, behind the scenes, plenty of details are needed. First of all, the agent needs to identify and verify the request and evaluate it with respect to earlier requests. This is called session control. It is discussed later in detail, but the basic

session control functionality requires authentication, request identification mechanisms, request history and logging. Attention is needed also for certain computer security considerations.

Some additional parameters related to user’s personal preferences or available technical environment are also needed. For example, the user’s browser software may only support certain features or the user may want to set some accessibility options. A special case of interest to educational use is the level of foreknowledge: the user may be a beginner or an advanced user. If supported by the application, this would certainly affect its behavior.

The most challenging component in the architecture is the application control block. It is responsible for finding or creating a suitable response for the request. In a simple case, it could be a list of pages that together form the CBE application accompanied with a control logic that describes the order of the pages. In other words, it defines the application with respect to the current request and request history. For example, if the application has been divided into several topics, and one topic consists of several web pages, the application control block makes sure that pages within a topic, and topics themselves, are given in the correct order.

With interactive applications, that is, when users provide parameters or data along with their requests, it is up to the application control component to validate that information. Technically, this is a very challenging question and it is discussed in more detail in the next section.

The content itself is either stored in a database or created on-the-fly. In any case, the response is always created on-the-fly, but at least parts of the content can be stored in a database. The application control block contains the necessary indexing keys for getting the content from a database or rules for creating the content. Additional information, metadata, can be used for making a search in the content, for example finding additional material or similar pages to previous ones (See [7] and [2] for recommendations on educational metadata).

Finally, the presentation control block transforms the content into suitable form with respect to user’s preferences and capabilities of his/her current browser. The agent also adds session parameters etc. to the response so that for instance links point back to correct session and so on.

From functional point of view, it is a matter of opinion whether the double agent is defined as a structure containing all the described functional blocks or are the application control block and presentation block considered to be separate blocks. With implementation, separation would be useful in any case, but from the user’s point of view, the whole functionality appears to be done by the agent.

Another interesting way to describe the functionality of the system is to think that it consists of two or more agents.

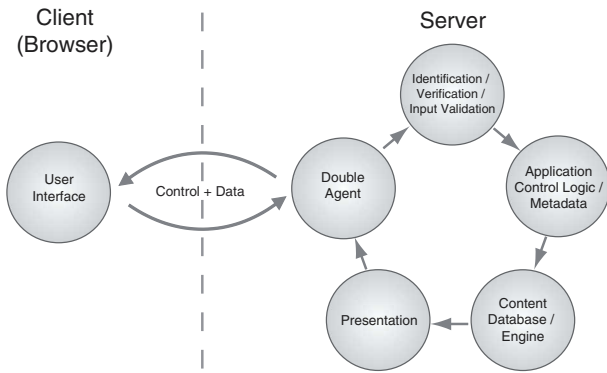


Figure 2: Request operation chain

For example dividing the functional agent structure into two separate agents, the user agent and application agent, gives another meaning to the double agent definition. Even more agents can be included: the error agent, presentation agent etc. Still, the multi-agent formalism doesn't change the basic dual nature of the system that is experienced by the user.

OPERATIONAL DESCRIPTION

A typical operation chain of the architecture is presented in Figure 2. It starts with the agent receiving a request from the user. The request is identified, verified and passed on to the application control level. Based on the information given by the agent, appropriate content is fetched from a database or created on-the-fly. Finally, the response is created by transforming the content into appropriate presentation.

Furthermore, there are three special cases of operation: logging into the system, logging out of the system and errors. Logging into the system requires additional functionality for authenticating the user and preparing the session control. This information forms the basis for later operations. In addition, mechanisms for creating a new user account, changing passwords or closing a user account are needed as well as defining what the user is allowed to do, in other words, what is the current role of the user.

The logging out phase is another special case of interest: a mechanism is needed for terminating the session and saving the current status of the user. In a normal situation it would be expected that the user explicitly logs out from the system. However, this can't be trusted and, therefore, a timing criteria etc. is needed for determining whether the user is still using the application or not.

Error situations are particularly difficult because there are two basic kinds of errors: system level errors and content-oriented errors. System level errors include errors such as session control violations, security attacks and server malfunctions. Content-oriented or application level errors are

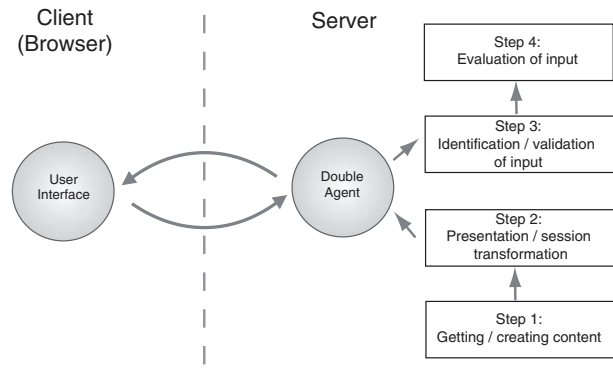


Figure 3: Process chain for interactive operation

related to the content: malformed input parameters are a typical example. In a usual operation, it would be the responsibility of the application control block to provide the agent with means for determining content-oriented errors.

Many of these operations are application specific. For instance, verifying the input or saving the status of the user when logging out, requires operations strictly dependent of the application that is used. This makes the implementation of the system quite demanding. On the other hand, session control and authentication can be implemented in a generic way and in fact, such operations are often included in existing development tools.

OPERATION WITH INTERACTIVE CBE-APPLICATIONS

Interactive operation requires some further explanation due to various implementation technologies. In principle, there are three different possibilities for creating interactivity: server-side, client-side or a combination of both server- and client-side technologies.

The most widely used solution is to use server-side technologies and limit client-side operations to input fields such as HTML forms. This approach has the advantage that the application provider has complete control over the interaction process. Another benefit is that the processing does not depend on the client software properties. In this case, the interaction is based on simple parameter transfer. However, in many applications this would be sufficient.

The basic server-side interactive operation involves four steps (Figure 3): creation of interactive content, modifying it to match current presentation and session requirements, validation of incoming parameters and evaluation of the actual content sent by the user. An example will clarify the situation: A CBE-application includes predefined true/false or multiple-choice questions. First the questions are fetched from a database. Secondly, they are inserted into HTML

template file and it is verified that the links point back to the agent. Possibly, some session control parameters are added. In the third step, the user sends the answer back. The input is then first checked that it matches the current session parameters and secondly that the parameters are indeed true/false-answers or valid choices. Finally, the input needs to be evaluated with respect to the content: are the answers correct and how to respond, if they are not.

A more challenging situation is when the question parameters are created on-the-fly. In that case, the operation chain is the same except that the questions are not fetched from a database but instead created on-the-fly using some rule-based or similar processing.

Sometimes this kind of interaction is not satisfactory, however. The use of client-side solutions provides additional functionality that can be of interest. Typical examples are animations, simulations and graphical games. If this kind of interactivity is wanted, client-side technologies are needed. This requires even more operations from the architecture, because not all clients provide the necessary capabilities and the client-side elements need to communicate with the agent.

An example illustrates the situation: The content is a standard web page with a simulation made as a Java Applet as part of it. Now, the operation chain requires an additional step, verification that the client-software actually is capable of presenting Java Applets. If it is not, the user has to be informed or alternative content created instead. Of course, the testing of browser's capabilities can and should be included in the logging in phase.

Another difficulty with the client-side technologies relates to getting and transferring input from the user. Basically, client-side technologies do not transfer any input back to the server unless specifically programmed to do so. Even if they do, the communications need to go through the agent. This requires that the client-side element can be adjusted with an interface between the element and the agent. Once again, an example illustrates the issue. A Java Applet simulation is programmed to send information about the user's operations back to the server. Because all communications go through the agent, the Applet needs to be adjusted with the agent address and session parameters. Furthermore, a specific interface needs to be created so that the agent understands that the input comes from the Applet. In the case of Java Applets, this can be done using Applet initialization parameters located inside the Applet-tags, but many other client-side technologies like Flash animations do not have this capability. In that case, the only choice would be to create the whole active element on-the-fly. In practice, this might be impossible.

IMPLEMENTATION GUIDELINES

The double agent architecture can be implemented in various ways. However, the suitable implementation relies heavily on the content: the actual CBE applications and their requirements. For applications that involve only simple pages and simple interactivity, a fairly simple implementation of the architecture can be used. Naturally, when the level of complexity of the applications increase, so does the complexity of the architecture.

First of all, it is generally a good idea to use a secure server (i.e. Secure Sockets Layer SSL [4]) as the server platform if any authentication is involved. Another benefit of this is that secure connections cannot be cached by proxy servers. This ensures that the session control works smoothly and that the delivered content is always up to date. The drawback is that network activity increases and that the client software needs to support SSL connections.

For authentication, there are several possibilities available. Perhaps the simplest one is the basic HTTP authentication [3]. This method is practical if there is only a small number of users because the account administration becomes quite troublesome when the number of accounts increases.

A more advanced method is to first identify users and then hide the account information in the session control. This method is more difficult to implement, of course, but it also allows additional user information such as the current role, privileges, preferences or level of foreknowledge to be included in the authentication scheme. In practice, the accounts and passwords would be stored in a database (for instance SQL- [6] or LDAP-database [10]) along with any other information available or desired. This database record would then be given a unique id that is included in the session control id.

The purpose of session control is to identify the requests and link them together as one session with respect to time. The standard method to implement this is to create a unique id that is included in every request sent by the browser. The most reliable way to implement session control is to add the session control id in every response URL and access all user-related information with the id only. Cookies [8] can also be used for ensuring that the user does not change his/her client software or computer during a session. Finally, the session control id should include a checksum etc. to make sure, the id is not manipulated in any way.

Being a request-response architecture, the Web cannot provide accurate control of timing. Session control techniques overcome this limitation by providing a timestamped session history. This allows the agent to measure and use time information for application control.

Probably the most difficult block to implement would be the application control block. If the application only consists

of static web pages, the implementation would be a simple list of files to get from the database. The pages would be served in a straightforward manner according to the list. If order of the pages is relevant like it usually is, the session history can be used for determining the next page based on previous ones.

Another interesting approach is to use metadata for determining the next appropriate page. This basically makes it possible for searching for the next appropriate page instead of determining it beforehand. It should be understood though that the format and contents of the metadata are very much application-dependent. Even though there are metadata standards and specifications (see for example [2], [7]) that would be useful, the control logic requires metadata definitions that meet the needs of the particular application in case.

As soon as any interactive elements are included, the situation becomes far more difficult. For server-side interactivity techniques, appropriate methods are needed to handle the input from the user. These methods depend strictly on the application and its content. Therefore, the application control must also include not only a list of pages but the required methods for handling input.

Furthermore, validation information has to be included in the control logic so that the agent can validate requests. In practice, this would mean parameter types or ranges for all variables sent by the user. But technically, the client-side interactivity would be the worst case scenario. In that case, the application control logic should include also methods for identifying the client-side elements as well as methods for validating their input.

Describing the control logic for CBE applications is a difficult task and probably no control language or specification would directly support arbitrary CBE applications. One of the most interesting approach available is the Unified Modeling Language (UML) [1], which provides functionality to model complex software constructs similar to application control logic here.

Presentation is also a challenging part to implement. Basically the block inputs some internal representation of content and transforms it suitable for the current user and his/her client software. The necessary requirement for this to happen is that a suitable presentation can be generated. In practice, this means that templates for several clients are created or that the presentation is generated in a rule-based manner. For the latter, a good solution would be to use an internal XML/XSL-based [5] format for the content and XSLT-language [11] for converting the internal format into suitable presentations. For the first one, the only choice would be to create different versions of the content suitable for desired clients. If personalization is allowed, both approaches should take into account also the possible personalization parameters.

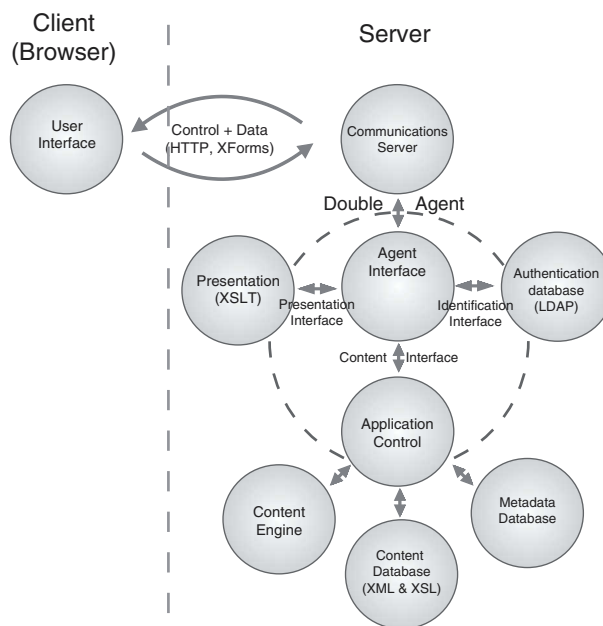


Figure 4: An advanced implementation structure

An example of an advanced implementation structure of the architecture is presented in Figure 4. Examples of suitable technologies are inside parentheses. However, the programming language/development platform for the agent has been left out intentionally. There are various suitable technologies available, both free and commercial. Example programming languages would be at least Java, Perl, PHP etc. Suitable, ready-to-use development packages are available for all of them. Also Application Servers provide tools and functionality that could be useful.

APPLICABILITY

As said before, the primary background for the double agent architecture is to provide system-level functionality needed for implementation and delivery of CBE-applications. Its design goals have included suitability for different applications, although the implementation requirements always depend on the applications.

When thinking of the abstraction of the content, it is important to understand that the content is by no means irrelevant of its presentation. On the contrary, in order to create advanced CBE-applications, the developer has to know details about the user, for example what kind of software the user has available and how/where he or she is using it. This is difficult because there are plenty of alternatives and options that affect the situation. This is actually one of the key problems that the double agent architecture tries to solve: a generic system that can be adjusted to meet the needs for in-

dividual CBE applications with functionality common to all such applications. The architecture also makes it possible to create tools for developing the applications, even though this is a true technical challenge.

However, perhaps the most important benefit of the architecture is the capability of logging user's actions and performance. From educational point of view, this gives important information about the process the user has been going through. When combined with knowledge of the content, the session history logs actually give a picture of the learning process [9]. In terms of evaluation of learning or education, this gives a completely new perspective for both teachers and students to accompany or even replace traditional evaluation methods. Furthermore, the logs can be used for studying for example the learning styles of individual users.

The architecture is not restricted to educational use only, however. In fact, similar system design would be useful in any application area requiring controlled interactivity. Advanced commerce or community/team services and applications are typical examples. Furthermore, certain parts and principles of the architecture, especially session control and abstraction of content and its presentation, are of interest in various services and applications. As a matter of fact, many developing tools and commercially available systems already include this kind of functionality.

CONCLUSIONS

This paper presented a double agent architecture that is useful for developing and delivering CBE applications. The architecture was designed especially with current Internet- and WWW-related technologies in mind, but it can be used with and extended to other similar information systems such as future, XForms [12]-capable, XML-based systems and mobile networks.

The architecture provides a systematic approach for developing eLearning applications and systems. It was designed especially with self-study CBE packages in mind, but certain features are of interest in delivery systems and collaborative learning environments as well. Therefore, Virtual University projects and eLearning systems vendors may find it interesting and useful.

The most important benefits of the architecture are the capabilities of identifying users and logging their actions. These features provide a view of the learning process, which is very much of interest to evaluation of learning and education. The logs can also be used for studying individual learning styles and adaptation.

The architecture can be implemented in various ways. Some of them present true technical challenges, but many functional blocks are readily available in development tools etc. Also many commercial systems provide functional

blocks needed for implementing the architecture.

REFERENCES

- [1] Alhir S. "Understanding the Unified Modeling Language", *Methods & Tools*, Spring 1999, vol. 7, no. 1, pp. 11-18.
- [2] ARIADNE, "ARIADNE Educational Metadata Recommendation Version 3.0", December 1999.
<http://ariadne.unil.ch/Metadata/>
- [3] Franks, J., et al., "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.
<http://www.ietf.org/rfc/rfc2617.txt>
- [4] Freier A., et al., "SSL Protocol Version 3.0", Internet Draft, Nov 18th, 1996.
<http://home.netscape.com/eng/ss13/draft302.txt>
- [5] Harold E., "XML Bible", IDG Books, USA. 1015 p.
- [6] Hoffman, J. "Introduction to Structured Query Language (version 4.71)", May 14th, 2001.
<http://w3.one.net/~jhoffman/sqltut.htm>
- [7] IMS, "IMS Meta-data Specification version 1.2", May 2001.
<http://www.imsproject.org/metadata/>
- [8] Kristol, D., Montulli, L., "HTTP State Management Mechanism", RFC 2109, February 1997.
<ftp://ftp.isi.edu/in-notes/rfc2109.txt>
- [9] Rahkila, M.; Karjalainen, M., "Evaluation of learning in Computer Based Education using Log Systems", in *Proc. 1999 IEEE Frontiers in Education (FIE'99)*, Puerto Rico, Nov 10-13, 1999.
<http://www.acoustics.hut.fi/~mara/publications/fie99/>
- [10] Wahl, M. "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.
<ftp://ftp.isi.edu/in-notes/rfc2251.txt>
- [11] W3C Recommendation, "XSL Transformations (XSLT) Version 1.0", Nov 16th, 1999.
<http://www.w3.org/TR/xslt>
- [12] W3C, "XForms - The Next Generation of Web Forms". *work in progress*.
<http://www.w3.org/MarkUp/Forms/>