

## Using Web Services for Personalised Web-based Learning

**Katerina Kabassi and Maria Virvou**

Department of Informatics, University of Piraeus  
80, Karaoli and Dimitriou St., Piraeus 185 34, Greece

[kkabassi@unipi.gr](mailto:kkabassi@unipi.gr)

[mvirvou@unipi.gr](mailto:mvirvou@unipi.gr)

### Abstract

This paper describes a multi-agent, personalised learning system operating over the Web. The system is called Web F-SMILE and is meant to help novice users learn how to manipulate the file store of their personal computer. In order to provide adaptive help and tutoring, Web F-SMILE has assigned an agent to constantly observe the user and collect information about him/her. This information is maintained centrally on a Learner Modelling Server. In this way, each learner model is available to any client application that requests it. The agents of the client applications interact with the Learner Modelling Server through Web Services. The main characteristic of Web Services is that they interact with the applications that invoke them, using web standards. Basing learner modelling on web standards has the advantage of enabling the dynamic integration of applications distributed over the Internet, independently of their underlying platforms.

### Keywords

Intelligent Tutoring System, Intelligent Learning Environment, Learner Modelling, Web Services, Multi-agent architecture

### Introduction

The increasing popularity of the Internet and the WWW has affected computer assisted learning which is now turning into Web-based learning since there are many advantages that the Web may offer to education. Indeed, learning through the Web can take place anywhere, at any time, through any computer and without necessarily the presence of a human tutor. However, still the majority of Web-based educational applications are rather static and represent a generic approach to tutoring that does not take into account the individual needs of each student that is using the educational application. This common practice does not make full use of the capabilities of the computer through the Internet as a medium for teaching students.

On the other hand, there have been educational software technologies that have been particularly effective at personalising tutoring. Indeed, Intelligent Tutoring Systems (ITSs) and Intelligent Learning Environments (ILEs) are educational technologies that aim at performing personalised teaching based on their learner modelling components. Learner modelling involves the construction of a qualitative representation that accounts for student behaviour in terms of existing background knowledge about a domain and about students learning the domain (Sison & Simura, 1998). Such a representation, called a learner model, can assist an Intelligent Tutoring System (ITS), an Intelligent Learning Environment (ILE), or an intelligent collaborative learner in adapting to specific aspects of student behaviour (McCalla, 1992).

For personalised interaction with a user, a system must have access to a wide variety of information about him/her ranging from relatively long-term facts such as areas of interest or expertise to quite short term facts such as the problem that the user is currently trying to solve. In view of this, Rich (1999) makes the distinction between long-term and short-term user models. A long-term user model may consist of information about the user that has been gathered during past interactions. This information may involve the user's level of knowledge of the domain, his/her common errors etc. A short-term user model consists of the user's beliefs at a very specific time and is the output of the reasoning of the system. Ideally, both models should exist in an ITS or ILE and should exchange information between them.

Traditionally, Intelligent Tutoring Systems (ITSs) used to work on each user's computer as standalone applications. These ITSs were based on a student model, stored locally on the user's Personal Computer (PC). As each system in this category collects more and more information about each learner, it can improve its predictions and learners develop trust in it. Since all personal data is stored locally on the user's computer, the only way for a student to enjoy the benefits of full adaptive and individualised tutoring would be by ensuring that s/he uses the same PC each time s/he interacts with the ITS. However, in real computer labs of educational

institutions this is rather difficult since users do not usually have their own PC but they use what is available at the time. Moreover, in real educational settings, a student would probably need to use an ITS both at school and at home. However, this would also be a problem in case the student model exists in one particular PC which is either at school or at home.

In view of the above, the merge of ITSs and ILEs with Web-based education may produce personalised tutoring systems that can be PC-independent, platform independent and can be used by students at any time, at any place without loss of important information gathered by the system about them in their long-term user model.

As a consequence of the indisputable virtues that the merge of ITSs and ILEs with Web-based education may offer, recently a lot of research energy has been put in this area (e.g. Alpert et al., 1999; Warendorf & Tan, 1997; Okazaki et al., 1996; Brusilovsky et al., 1996; Ritter, 1997; Nakabayashi et al., 1997). In common with these systems we have developed Web F-SMILE (Web File-Store Manipulation Intelligent Learning Environment) which is a personalised learning system operating over the Web. In particular, Web F-SMILE is an ILE for novice users of a GUI that manipulates files, such as the Windows Explorer. However, our approach concerning the operation of the system over the Web is based on Web Services, a promising new technology. More specifically, Web Services are self-contained, modular applications that provide a set of functionalities to anyone that requests them. The main characteristic of Web Services is that they interact with the applications that invoke them, using web standards such as WSDL (Web Service Definition Language), SOAP (Simple Object Access Protocol) and UDDI (Universal Description, Discovery and Integration). Basing learner modelling on web standards has the advantage of enabling the dynamic integration of applications distributed over the Internet, independently of their underlying platforms.

## **Related Work**

A number of architectural patterns have been employed for the deployment of ITSs and ILEs on the Web. In this Section we present and discuss the most common architectures and compare them and contrast them with the architecture that we have employed, which is based on Web Services. Then we discuss the similarities and differences of Web F-SMILE with other Web-based systems that offer personalised tutoring.

A simple solution for deploying ITSs and ILEs on the Web has been based on Java and has been employed in ADIS (Warendorf & Tan, 1997). The entire ITS resides on a Java applet that the user downloads by visiting a specific URL. The ITS is running on the student's PC and the student model resides on the client. Since all information about the learner is stored locally on the learner's PC, ADIS still suffers from the disadvantages of standalone, PC-dependent ITSs concerning the completeness and accuracy of the learner model.

A quite different approach is the distributed client-server architecture, which is employed for example by Elliot (1997). In that case some modules are on the server side and some on the client side. A Java applet, which resides on the client, contains the modules of the system that are responsible for the interaction with the user. The communication between the server and the client is performed using a socket connection or other networking mechanism. The main problem with sockets is that they do not support data types and consequently, need manual message parsing. In contrast, Web Services follow the XML (eXtensible Markup Language) protocol for data exchange, thus allowing complex data type predefinition. Another disadvantage of the distributed client-server architecture is that the developer has to create his/her own communication protocol and the clients may experience problems in receiving data from the Server. For example, if a user works both at home and at work, his/her user model may not function as expected, because the client at work may be behind a firewall that does not allow for the user modelling server's port to pass through. In contrast, Web Services follow the SOAP Protocol (Simple Object Access Protocol) to handle communication. This heavy reliance of Web Services on standards ensures basic interoperability which means that the data of the learner model will be readable from every computer. In addition, Web Services rely on the Hypertext Transfer Protocol and thus gain the advantage of being able to flow through most security systems (firewalls, proxy servers, etc).

Another approach that has been the predominant architecture of Web-enabled applications up to date is the HTML-CGI architecture. This architecture is adopted by several ITSs such as WITS (Okazaki et al., 1996), ELM-ART (Brusilovsky et al., 1996), PAT Online (Ritter, 1997), CALAT (Nakabayashi et al., 1997) and AlgeBrain (Alpert et al., 1999). In all these ITSs the user interacts with the system through a Web browser. The information given by the user is sent to the Web Server, which forwards it to the CGI (Common Gateway Interface) application. The CGI program contains all the functionality and the student model resides on the server. However, this architectural model has a number of constraining features, such as lack of immediate

interactivity. CGI scripts are stateless resulting in cumbersome, standalone requests. For example, a user has to identify him/herself each time s/he posts a query. XML, on the other hand, supports structuring complex data in a hierarchy and thus, facilitates quick transactions (Papadakis & Chrissikopoulos, 2000). Furthermore, Web services in contrast to an HTML-CGI architecture, where the system cannot use the client's resources, are much more loosely coupled than most traditional distributed applications (Kuno & Sahai, 2002).

Finally, Web Services follow the WSDL protocol (Web Services Description Language) to provide the metadata necessary to run the service, and UDDI (Universal Description Discovery and Integration) to publish services on UDDI servers. This enables the dynamic integration of applications distributed over the Internet, irrespective of their underlying platforms. Generally, as Tsalgatidou & Pilioura (2002) point out, the Web Service paradigm motivates developers to build applications by locating and using existing Web Services rather than building the required functionality from scratch, promoting in this way easy, fast and efficient application development and just-in-time integration.

In the majority of the web-based educational systems (Okazaki et al., 1996, López et al., 1998, Brusilovsky et al., 1996, Ritter, 1997, Machado et al., 1999) the student model is kept on the server side but the main instructional decisions are made by the client application. This approach is also employed by Web F-SMILE. However, one important difference of Web F-SMILE with all the systems mentioned above is that it can also be used while the user's PC is not connected to the Internet. This is done, so that users may use the learning environment even in cases when the connection with the Internet may have failed for some reason. To achieve the operation of Web F-SMILE in both ways, Web F-SMILE deploys two learner models for every learner; one locally on the user's computer and one centrally on the server. A similar approach is also employed in DCG (Vassileva, 1997). More specifically, in DCG when a user downloads the Java application, a copy of his/her student model is created locally on the user's computer. All new information collected during the student's interaction with the system, is stored in the local user model. When the user logs off the application, the local copy is uploaded to the server. However, this approach does not take into account what happens if the user is not connected to the Internet any more, when s/he logs off the application. Web F-SMILE addresses this problem by adjusting the interaction of the learner models on the client and on the server accordingly. Every time the user gets online the two models interact effectively through a Web Service and exchange information so that both models contain the latest information about the learner.

## **Operation of the system**

Web F-SMILE (File-Store Manipulation Intelligent Learning Environment) is an intelligent learning environment for novice users of a GUI (Graphical User Interface) that manipulates files, such as the Windows 98/NT Explorer (Microsoft Corporation, 1998). The main feature of the system is that it can adapt its interaction to each individual learner. In order to do so, Web F-SMILE has assigned agents to observe the student while s/he is actively engaged in his/her usual activities and provide spontaneous and individualised advice and tutoring in case of a problem. Individualised advice and tutoring is based on the learner model. The system can work both as a Web-based application and as a standalone application when the learner's computer is not connected to the Internet. When the system works online information about the learner is stored on a Learner Modelling Server and is put at the disposal of any client of the application that requests it. However, a learner can also manipulate his/her file store while the learner is offline. For this reason the system keeps two copies of learner models, one on the Server and one on the user's PC so that the system may work both online and offline. In this way the usability of the system is increased.

Web F-SMILE uses Web Services for the interaction of the agents of the system with the Web Server. Web Services, in the general meaning of the term, are services offered via the Web. Lately, though this term refers to a collection of specific protocols regarding remote application interaction. More specifically, Web Services are self-contained, modular applications that provide a set of functionalities to anyone that requests them.

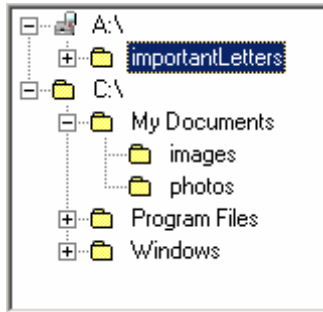


Figure 1. The learner's initial file store state

A simple example of the system's operation taken from a real interaction of a user with Web F-SMILE is presented in Table 1. The learner's initial file store state of the floppy disk is shown in figure 1. The learner's final intention is to format the floppy disk A. However, the floppy disk contains a folder with some important letters. Therefore, the learner wants to move this folder to a safe place (the hard disk of his/her computer).

<ol style="list-style-type: none"> <li>1. cut(A:\importantLetters\)</li> <li>2. copy(C:\My Documents\)</li> </ol> <p>Web F-SMILE's reasoning: Suspect Action. Suggestion: paste(C:\My Documents\) Additional tutoring themes: Copying Objects, Moving Objects.</p> <ol style="list-style-type: none"> <li>3. paste(C:\My Documents\)</li> <li>4. format(A:\)</li> </ol>
---

Table 1. An example of a user's interaction with Web F-SMILE

In order to achieve his/her goal the user issues a cut command (action 1) in order to move the folder 'importantLetters'. However, apparently the learner does not know how to complete this plan because in the second action, s/he falsely uses a 'copy' command instead of a 'paste' command. Web F-SMILE finds this action suspect because if it was executed it would delete the content of the clipboard before this was used anywhere. Therefore, the system tries to generate alternative actions that the learner may have meant to issue instead. In order to select the most appropriate advice, the system uses the information about the learner that is available on the learner model. The alternative action that is considered by Web F-SMILE as more likely to have been intended is the action 'paste(C:\My Documents\)' because it uses effectively the content of the clipboard. Moreover, the commands 'copy' and 'paste' are considered quite similar because they both involve the clipboard. Thus the user may have confused them.

Furthermore, the system also produces additional tutoring in the domain of copying and moving objects, which was considered to be essential for the user to complete his/her plans and goals. The information of the learner model indicates that the particular user has not sufficient experience in copying and moving objects and that in the past s/he had repeatedly made mistakes due to lack of knowledge on the topic. Indeed, the learner finds the system's advice very helpful and, therefore, adopts its suggestion in action 3. Then, in action 4, the learner formats the floppy disk, which was his/her final goal. In case the learner had used a standard file manipulation program, his/her error in command 2 would not have been recognised and the learner would have formatted the floppy disk and would have lost useful information.

## Multi-agent architecture

Web F-SMILE is based on a multi-agent architecture. A multi-agent system is composed of a group of agents that are autonomous or semiautonomous and which interact or work together, to perform some tasks or achieve some goals (Lesser, 1995). The design of individual agents within a multi-agent system has the advantage of being independent of the design of other agents. This significantly contributes to the breakdown of complexity (El-Beltagy et al., 1999).

Web F-SMILE's architecture consists of five agents, namely, Short Term Learner Modelling (STLM) Agent, Long Term Learner Modelling (LTLM) Agent, Advising Agent, Tutoring Agent and Speech-driven Agent. The architecture of Web F-SMILE can be seen in Figure 2, where all agents and the domain representation component are illustrated. The agents work together in order to watch the learner and provide individualised advice and tutoring in case this is considered necessary. Advice is provided to learners who have made an error with respect to their hypothesised intentions. All these agents work locally on the learner's computer and only the LTLM Agent is responsible for the interaction with a Web Server for learner modelling.

Every time the learner issues a command, the STLM Agent, which works on the client side, reasons about the command with respect to its expectations about the learner's goals. The Short Term Learner Modelling (STLM) Agent captures the cognitive state, as well as the characteristics of the learner and identifies possible misconceptions. In case the STLM Agent suspects that the learner is involved in a problematic situation, it performs error diagnosis. For this purpose, it employs an analysis engine to derive new 'facts' about the learner and to respond to queries from other agents. The analysis engine is based on a limited goal recognition mechanism and the Human Plausible Reasoning theory (Collins & Michalski, 1989). Human Plausible Reasoning theory (HPR) is a domain-independent theory originally based on a corpus of people's answers to everyday questions. Starting from a question asked to a person, the theory tries to model the reasoning that this person employs in order to find a plausible answer, assuming that s/he does not have a ready answer. Prior to Web F-SMILE, HPR had also been successfully used for simulating the users' reasoning in a help system for a graphical user interface (Virvou & Kabassi, 2002) and in a help system for a command language interface (Virvou & Du Boulay, 1999).

STLM Agent applies the principles of HPR while searching for similar alternative actions to the one issued that the learner may have intended to issue instead of the one issued which was problematic. As soon as the alternative actions are generated, they are sent to the Advising Agent, which is responsible for selecting the alternative action that the learner was more likely to have intended. The reasoning of the Advising Agent has been evaluated (Virvou & Kabassi, 2001) and the results offer strong evidence that the particular agent can indeed reproduce the reasoning of a human tutor, who watches the user over the shoulder while s/he is interacting with the system.

Furthermore, in case the STLM Agent thinks that the learner's misconception was due to the learner's lack of knowledge, it informs the Tutoring Agent about it. The Tutoring Agent is responsible for forming an adaptive presentation of the lesson to be taught to the learner. The Advising and the Tutoring Agent request information about the learner from the STLM Agent. This is done so that they may adapt the advice and/or the lesson produced to the needs and the interests of each individual learner. The Advising and the Tutoring Agent, however, do not need to communicate with the Server, directly since their reasoning mechanisms reside on the client.

The Tutoring Agent uses adaptive hypermedia techniques to protect learners from information overflow and help them understand new pieces of knowledge that are being taught. In particular, these techniques use information about a particular learner, from the learner model, to adapt the lessons presented to that learner. The two main adaptive hypermedia technologies that exist are: (i) adaptive presentation, in which case adaptation is performed at the content level and (ii) adaptive navigation support, which is performed at the link level (Brusilovsky, 1996). Both these technologies have been evaluated and the results offer strong evidence that their use in an Adaptive Hypermedia System can improve human-computer interaction. In Web F-SMILE, adaptive presentation techniques are used to present examples of use of an unknown command in the context of the learner's own file-store. Therefore, the Tutoring Agent generates examples dynamically so that it may use the names of the particular learner's existing files and folders. Moreover, the Tutoring Agent uses adaptive link annotation techniques to present to the learner other parts of knowledge that are believed to be of interest to the learner for the particular case.

Both the Advising Agent and the Tutoring Agent send their results to the Speech-driven Agent, which is also located on the client. The Speech-driven Agent is responsible for presenting the information in a unified and easy to access fashion. In order to make the interaction more natural and enjoyable, an animated, speech-driven Agent is used to present the system's advice to the learner. Such characters provide an entertaining and emotional function, which may help to lower the 'getting started barrier' for novice learners of computer applications. In addition, such characters improve the effectiveness of the system by engaging and motivating learners (Johnson et al., 2000). The Speech-driven Agent is responsible for the overall communication with the learner. This usually involves the collection of the learner's queries and the presentation of advice in case the

learner is diagnosed to have been in a problematic situation. However, the particular agent does not contain any further reasoning mechanisms.

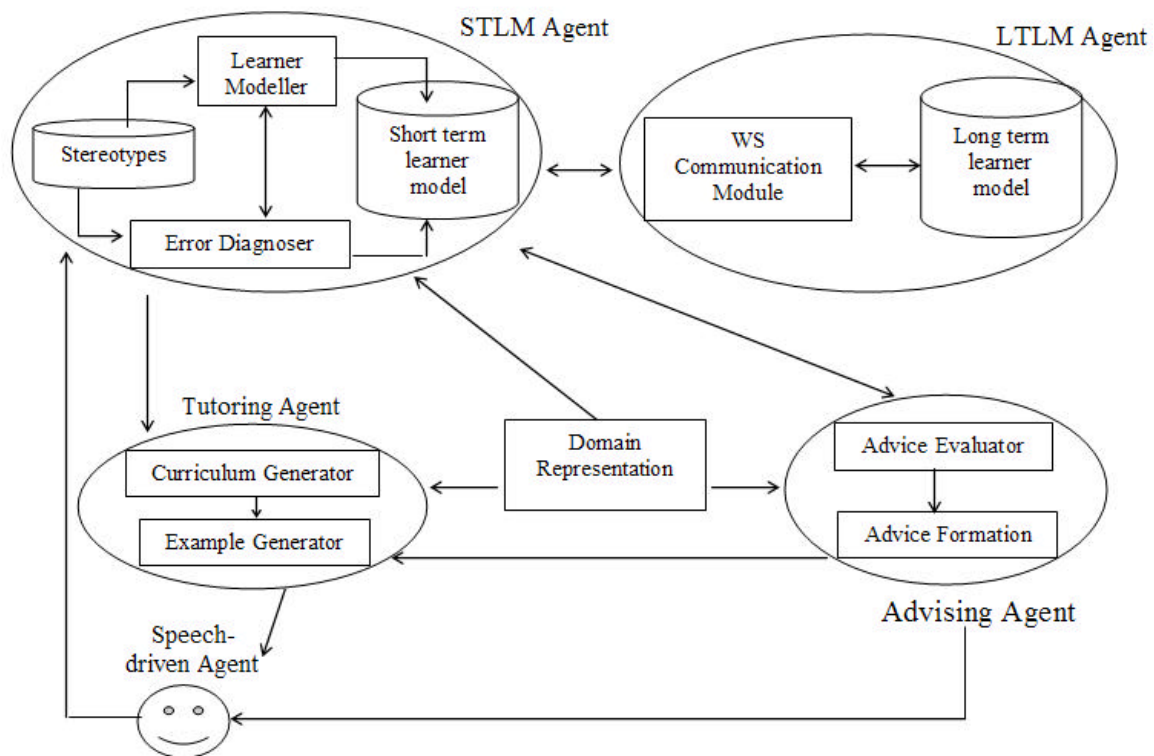


Figure 2. Web F-SMILE's architecture

Every time the STLM Agent acquires new information about the learner that interacts with the system, it sends it to the LTLM Agent. Generally, the LTLM Agent, maintains and manages the learner profiles and provides relevant information to the STLM Agent whenever this is considered necessary. Furthermore, the LTLM Agent is responsible for the interaction with the Web Service Learner Modelling (WS-LM) Server in order to maintain and update the information stored in learner models, both on the Web Service Server and the client.

### Interaction of Client and Server Learner Models

Web F-SMILE keeps two separate learner models for every learner, one locally on each computer and one on the Server. Every time the user uses the learning environment, the system checks whether the user's computer is connected to the Internet or not. In case the computer is not connected, Web F-SMILE works as a standalone application with a local user model. The LTLM Agent is responsible for finding the learner model for the user interacting with the system. In case the LTLM Agent finds the model of the learner, the interaction starts normally and the local learner model is updated every time the learner issues a new command. However, if there is no information about the learner available on that particular PC, the learner is given a questionnaire about his/her level of experience, his/her experience in operating systems and other file manipulation programs. This information is used by the LTLM Agent in order to initialise the learner model using stereotypes.

In case the learner's PC is online, the LTLM Agent interacts with the Web Service Learner Modelling (WS-LM) Server in order to find the corresponding learner model on the Server. If the learner model does not exist on the WS-LM, the LTLM Agent is responsible for finding whether the learner has interacted with the system offline using the particular computer. In case the LTLM Agent cannot find any information about the particular learner, it initialises the learner model locally. In any case, the LTLM Agent sends the information about the learner to the Web Service, which creates a new learner model based on the information that was available from the learner model that was initialised locally.

If the learner that interacts with the application is online and the learner model on the Server exists, the LTLM Agent is responsible for finding whether there is a local learner model or not. In case there is not any information about the learner available locally on the computer, the LTLM Agent is responsible for making a copy of the learner model from the Server to the hard disk of the learner's PC. Otherwise, the LTLM Agent undertakes the task of updating both models with the latest information. This approach is similar to the one adopted in ITSs that work both online and offline, for example DCG (Vassileva, 1997). However, in DCG there is a problem if the user loses his/her connection to the Internet since the most recent student work and student model updates are lost. In Web F-SMILE, this information is kept on the local learner model until the next time that the user uses the application online in which case the learner model that is maintained centrally is updated.

In order to update the learner model with the right information, a system would have to know which information has not been included in the learner model on the Server and which in the local learner model. This is quite difficult, if the learner model maintains summative information about the user, for example, the total number of errors due to carelessness. Therefore, Web F-SMILE registers each learner interaction separately using timestamps, so each record of the learner model has the date and the time of the interaction. In this way, each interaction differentiates from all the others and the LTLM Agent and the Web Service can easily identify which interactions of the local learner model have not been included in the learner model stored in the Server and vice versa. Furthermore, each record of the learner model contains a flag that states whether the interaction it refers to has been submitted to the Server or not.

As soon as the update of the learner model is completed, the interaction with the system starts normally. This process is repeated when the learner logs off the system (in case the learner is still online) so that the Server is updated with the newly acquired information.

## **Learner Modelling on the client side**

Every time the learner interacts with Web F-SMILE, the STLM Agent collects new information about the user and updates the learner model, which is maintained locally on the student's computer. In case the STLM Agent cannot locate a learner model for a particular student, it tries to initialise the learner model using stereotypes. User stereotypes are employed in order to provide default assumptions about users until systems acquired sufficient information about each individual user. Indeed as Rich (1989; 1999) points out, a stereotype represents information that enables the system to make a large number of plausible inferences on the basis of a substantially smaller number of observations; these inferences must, however, be treated as defaults, which can be overridden by specific observations.

In Web F-SMILE, users are classified into one of three major classes according to their level of expertise, namely, novice, intermediate and expert. Each one of these classes represents an increasing mastery in the use of the particular file store manipulation system. Such a classification is considered important because it enables the system to have a first view of the usual errors and misconceptions of a user, belonging to a group. For example, novice users are usually prone to mistakes due to erroneous command selection or erroneous command execution whereas expert users usually make mistakes due to carelessness. Therefore, another classification that was considered important was dividing users into two groups, careless and careful.

Stereotypes may serve as a tool to model the beliefs and preferences that the users of a system may have. The main reason for the application of stereotypes is that they provide a set of default assumptions, which can be very useful during hypotheses generation about the user. Generation of default assumptions can prove very effective for modelling a large set of users. However, this approach has many problems as well. For example, despite the similar behaviour that users of the same group may have, every one is an individual that differs from all the others in many aspects. Therefore, stereotypes should only be used for initialising the user model, until there is more individual information. Thus, Web F-SMILE maintains a library of models for every group of users, and every time a new user interacts with the system, the LTLM Agent of Web F-SMILE must identify the class the particular user belongs to.

All of the default assumptions in stereotypes that are used in Web F-SMILE, give information about the errors that users belonging to each category usually make. Information about each error is expressed by using the certainty parameters of HPR theory. So, we used frequency to show how often users belonging to a certain group make a particular error. Another piece of information that can be derived from a stereotype concerns the most frequent kinds of error of a user that belongs to this particular stereotype. This is expressed as a number representing the dominance of a particular error within the set of all errors for users belonging to the particular

stereotype. Finally, the typicality shows how typical a command is in the set of all the commands that a user has issued.

In order to find the stereotype that the learner belongs to, the learner is given a questionnaire, which contains questions about his/her level of experience in GUIs, his/her experience in operating systems, other file manipulation programs etc. This information is further processed by the STLM Agent in order to activate the corresponding stereotype.

After a stereotype has been activated, the system makes some default assumptions about users' possible errors and can provide some kind of advice. In the beginning, information is acquired only by the stereotype. However, the system is also constantly collecting information about a particular user's behaviour and errors and informs the individual learner model of the learner. As the system collects more and more evidence about a learner, information is acquired in part by the stereotype and in part from the individual learner model. The percentage of information acquired by the stereotype diminishes as the percentage of acquisition by the individual learner model increases.

In particular, for every new interaction the STLM Agent creates a new record on the learner model using a timestamp, so every record has the date and the time of the interaction. In this way, each interaction differentiates from all the others and the STLM Agent and the Web Service can easily identify which interactions of the local learner model have not been included in the learner model stored in the Server and vice versa. Furthermore, each registration of the learner model contains a flag that informs the STLM Agent whether this interaction has been included in the server learner model or not. As soon as the learner completes his/her interaction with the system, the STLM Agent is responsible for interacting with the Web Service in order to update the long term model of the particular learner, which is maintained on the Server.

## **Learner Modelling on the Server Side**

Communication between the Client and WS-LM takes place through the Web Services protocols. Figure 3 summarises the operation of the Web Service and its interaction with the agents of the clients. The LTLM Agent makes a specific SOAP call (under HTTP), which contains a request about the particular learner model, to the WS-LM. In order to authenticate the user, this call contains the username and the password given by the user during his/her interaction with the client application. Each such call is retrieved and handled by the Communication module. Generally, the Communication module handles all the Web Service messages, namely authentication requests, profile requests, profile update requests, creation requests and deletion requests, and is responsible for formatting the response in XML and sending it to the client application.

After processing the service's part of the URL it received, the Communication Module passes on the rest of the string to the DB Module or the Learner Modelling Module which are responsible for processing the request and form the response. The Learner Modelling Module evaluates the information sent to WS-LM. For example, if the information sent to WS-LM is about the stereotypes that the learner belongs to and WS-LM has sufficient information from the individual history of the learner then this information is rejected. This is done because the stereotype information is only used for the initialisation of learner models until sufficient information about the individual learner is collected. Thus when stereotype information is coming from the local learner model while there is sufficient individual information on the server, it means that the communication between the PC and the server must have failed and Web F-SMILE had operated as a standalone application. During that operation it must have created a new initial learner model because it could not find a local learner model. However, such initial learner models are useless when Web F-SMILE is online again, because in that case it has access to the complete individual learner models on the Web Server. Otherwise, the information is sent to the database module (DB module), which implements all the functions necessary to create, update and delete profiles, as well as to authenticate a user. In order to perform these functions, the DB module interacts with the learner models database (LMDB). The response is sent again to the Communication Module, which encodes it in XML and returns it to the caller (agent on the client).

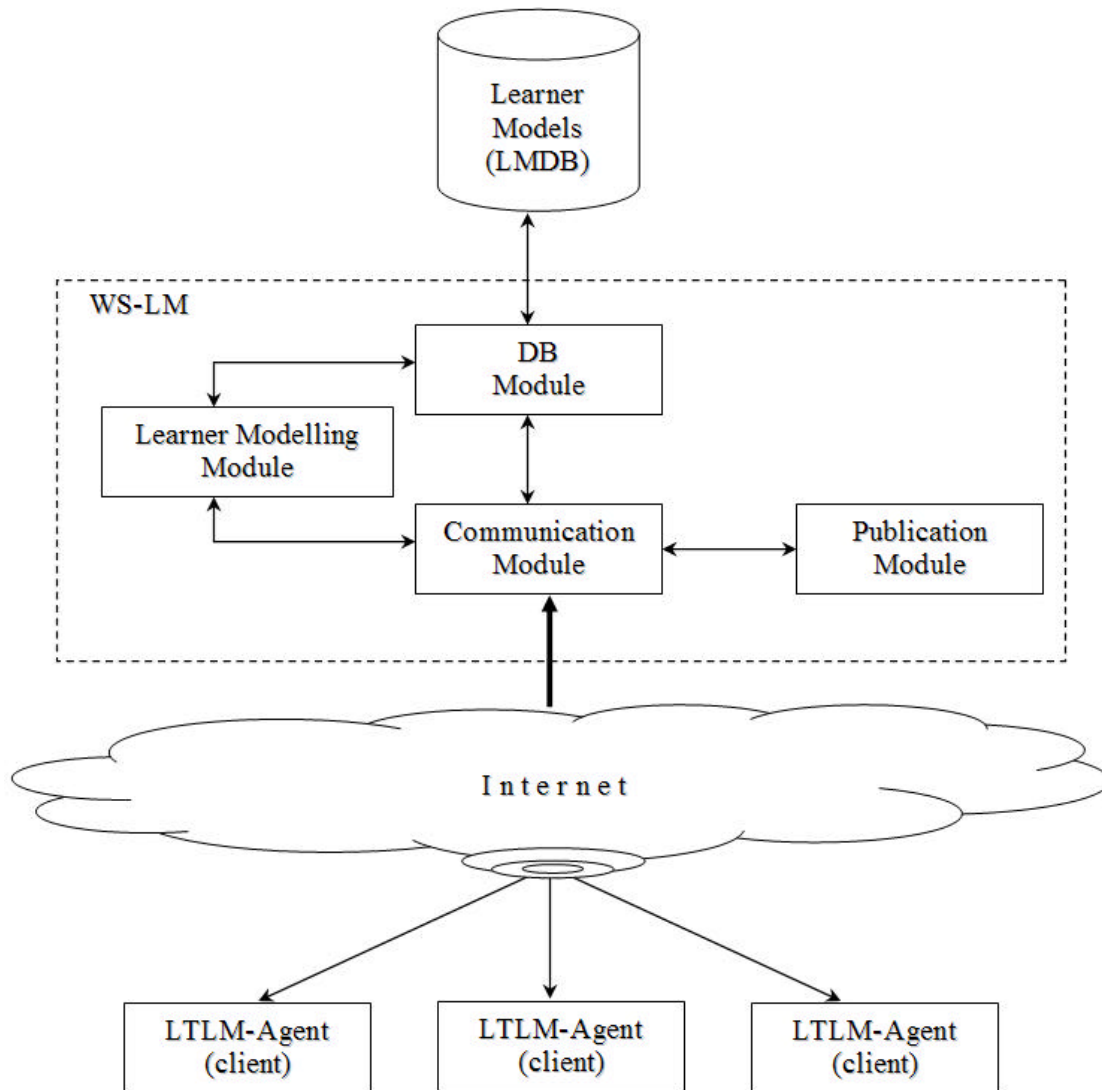


Figure 3. The architecture of WS-LM

As already mentioned above, the LTLM Agent on the client which is responsible for the maintenance of the learner models, creates a new record on the learner model for every time that a user interacts with Web F-SMILE. This is done for synchronisation purposes. However, such a detailed learner model needs considerable storage space and time to be investigated by the Web Service or the LTLM Agent. To overcome these problems, the learner model is divided into two parts; the first part contains summative learner information that is older than three months and the second part contains a detailed description of the learner's interactions with Web F-SMILE for the last three months. The DM Module is assigned the daily task of joining the records of learner models. Every day the DM Module deletes the records of the learner models that are older than three months old from the second part of the learner model and integrates them in the first part.

## Conclusions

In this paper, we have described a multi-agent learning environment that helps users learn how to operate their file store. The system has assigned an agent (STLM Agent) to monitor users while working in a protected mode and in case it identifies a problematic situation, it tries to diagnose the cause of the problem and offers appropriate advice. Novice learners can benefit from the system's advice and adaptive tutoring facilities so that they may learn from their own errors. The adaptivity of learning depends on factors such as the learner's prior knowledge, his/her abilities and needs.

The main problem with standalone applications in computer labs is that a user may not always use the same PC and thus no single PC may have a complete and accurate model of each user. This problem is addressed in Web F-SMILE with the incorporation of Web Services for learner modelling. Web services, in the general meaning of the term, are services offered via the Web and are used in Web F-SMILE for the interaction of the agents of the system with a Learner Modelling Server (WS-LM). WS-LM maintains a central database of all learner models and allows the LTLM Agent from client applications to access this information from virtually every computer. In addition, Web F-SMILE keeps for every learner one learner model centrally on WS-LM and one learner model in each computer that the user uses to interact with Web F-SMILE. In this way, Web F-SMILE overcomes possible problems that may arise due to possible communication failures between a learner's PC and the Server.

The proposed Web Service architecture has been compared to most of the architectures used in the ITSs that work over the Web. This comparison has revealed that learner modelling based on Web Services introduces an improved interaction of the server with the client applications compared to other traditional architectures.

## References

- Alpert, S. R., Singley, M. K., & Fairweather, P. G. (1999). Deploying Intelligent Tutors on the Web: An Architecture and an Example. *International Journal of Artificial Intelligence in Education*, 10, 183-197.
- Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction*, 6 (2-3), 87-129.
- Brusilovsky, P., Ritter, S., & Schwarz, E. (1996). ELM-ART: An Intelligent Tutoring System on World Wide Web. In C. Frasson, G. Gautier & A. Lesgold (Eds.), *Intelligent Tutoring Systems, Proceedings of the Third International Conference*, Berlin: Springer, 261-269.
- Collins, A., & Michalski, R. (1989). The Logic of Plausible Reasoning: A core Theory, *Cognitive Science*, 13, 1-49.
- El-Beltagy, S., De Roure, D., & Hall, W. (1999). A Multiagent system for Navigation Assistance and Information Finding. *Paper presented at the 4<sup>th</sup> International Conference on the Practical Applications of Intelligent Agents and Multi-Agent Technology*, April 19-23, 1999, London, UK.
- Elliot, C. (1997). Implementing Web-based intelligent tutors. *Paper presented at the workshop "Adaptive Systems and User Modelling on the World Wide Web"*, 6th International Conference on User Modelling, 2-5 June 1997, Chia Laguna, Sardinia.
- Johnson, W. L., Rickel, J. W., & Lester, J. C. (2000). Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments. *International Journal of Artificial Intelligence in Education*, 11, 47-78.
- Kuno, H., & Sahai, A. (2002). My Agent Wants to Talk to Your Service: Personalizing Web Services through Agents. HPL-2002-114. HP Labs Technical Report.
- Lesser, V. (1995). Multiagent Systems: An Emerging Subdiscipline of AI. *ACM Computing Surveys*, 27 (3), 340-342.
- McCalla, G. (1992). The central importance of student modelling to intelligent tutoring. In E. Costa (Ed.), *New Directions for Intelligent Tutoring Systems*, Berlin: Springer Verlag, 108-131.
- Microsoft Corporation (1998). Microsoft® Windows® 98 Resource Kit, USA: Microsoft Press.
- Nakabayashi, K., Maruyama, M., Koike, Y., Kato, Y., Touhei, H., & Fukuhara, Y. (1997). Architecture of an Intelligent Tutoring System on the WWW. *Paper presented at the 8<sup>th</sup> World Conference of the Artificial Intelligence in Education*, 18-22 August, 1997, Kobe, Japan.
- Okazaki, Y., Watanabe, K., & Kondo, H. (1996). An Implementation of an intelligent tutoring system on the World-Wide Web. *Educational Technology Research*, 19 (1), 35-44.

- Papadakis, I., & Chrissikopoulos, V. (2000). A Digital Library Framework based on XML. *Paper presented at the 3rd International Conference of Asian Digital Library*, December 6-8, 2000, Seoul, Korea.
- Rich, E. (1989). Stereotypes and User Modeling. In Kobsa, A. & Wahlster, W. (Eds.), *User Models in Dialog Systems*, New York: Springer, 199-214.
- Rich, E. (1999). Users are individuals: individualizing user models. *International Journal of Human-Computer Studies*, 51, 323-338.
- Ritter, S. (1997). PAT Online: A model-tracing tutor on the World-Wide Web. *Paper presented at the workshop "Intelligent Educational Systems on the World Wide Web"*, 8<sup>th</sup> World Conference of the Artificial Intelligence in Education, 18-22 August, 1997, Kobe, Japan.
- Sison, R., & Shimura, M. (1998). Student Modeling and Machine Learning. *International Journal of Artificial Intelligence in Education*, 9, 128-158.
- Tsalgatidou, A., & Pilioura, T. (2002). An Overview of Standards and Related Technology in Web Services. *Distributed and Parallel Databases*, 12, 135-162.
- Vassileva, J. (1997). Dynamic Courseware Generation. *Communication and Information Technologies*, 5 (2), 87-102.
- Virvou, M., & Du Boulay, B. (1999) Human Plausible Reasoning for Intelligent Help. *User Modeling and User-Adapted Interaction*, 9, 321-375.
- Virvou, M., & Kabassi, K. (2001). Evaluation of the advice generator of an intelligent learning environment. In T. Okamoto, R. Hartley, Kinshuk & J. P. Klus (Eds.), *Proceedings of the IEEE International Conference on Advanced Learning Technologies*, Los Alamitos, USA: IEEE Computer Society, 339-342.
- Virvou, M., & Kabassi, K. (2002). Reasoning about Users' Actions in a Graphical User Interface. *Human-Computer Interaction*, 17 (4), 369-399.
- Warendorf, K., & Tan, C. (1997). ADIS-An animated data structure intelligent tutoring system or Putting an interactive tutor on the World Wide Web. *Paper presented at the workshop "Intelligent Educational Systems on the World Wide Web"*, 8<sup>th</sup> World Conference of the Artificial Intelligence in Education, 18-22 August, 1997, Kobe, Japan.