OCEAN Project
A prototype of AIWBES based on fuzzy ontology

Francesco Colleoni, Silvia Calegari, Davide Ciucci, Matteo Dominoni
Dipartimento di Informatica, Sistemistica e Comunicazione – Università di Milano–Bicocca
Viale Sarca 336/14, I–20126 Milano (Italy)
wolfra@gmail.com, {calegari, ciucci, dominoni}@disco.unimib.it

Abstract

Ocean Project is aimed to realize an Adaptive and Intelligent Web-Based Educational System (AIWBES) working with traditional Learning Management Systems (LMS). It is designed as a collection of open-source libraries (the Omega Framework), so resulting easily customizable and adaptable to the current e-learning platforms. In this new system each course is presented in different ways according to the student's learning level, through to a combined use of ontologies and fuzzy logic.

1. Introduction

Nowadays LMS are the main elements of the web-based education. They typically offer a variety of tools to make didactic more effective: a way to upload and share materials, hold online discussions and chats, give quizzes and surveys, gather and review assignments, and record grades. In other words, they are a suite tool to enhance teaching by taking advantage of the internet without replacing the need for the teacher.

As a matter of fact, the role of tutors and teachers remains central: LMS cannot replicate one of the most important features of human relations, i.e., adaptability. In fact, a human teacher is able to adapt to the different needs of students in terms of learning. For example, teachers traditionally offer tutoring services, such as a student teacher meeting after class. In order to replicate this classic teaching behaviour in e-learning, you must provide the necessary resources, i.e. teacher and tutors to interact with students. In the real world, especially in cases with many participants, the problem is that a teacher does not have enough time to dedicate to each student: tutoring is an extremely expensive activity. A traditional LMS does not offer individual attention; the approach is said to be "one size fits all", since generally an LMS does not offer functionality for personalization of the learning process [1].

A particular kind of e-learning system called AIWBES [2] has be used to overcome such limitations. Examples of AIWBES are "ActiveMath" [3], which uses OMDoc (Open Mathematical Documents) for knowledge representation, "ELM-ART" [4] and "KnowledgeTree" [5], which uses concept maps for knowledge representation.
These tools allow automatic personalization of learning paths of students combining technologies such as "intelligent tutoring" and "adaptive hypermedia" [6].

Therefore, the approach of these systems is no longer "one size fits all", but is aimed to personalization, like a teacher who teaches directly to each student, one on one, surpassing traditional LMS limitations.
Even if the AIWBES approach has been demonstrated to be superior to LMS because students learn faster and better [1], it is not as used as LMS. This is because the AIWBES architectures are usually quite complex, designed for a very specific didactical purpose, and furthermore these systems are not compatible with standard formats like SCORM or IMS and therefore they do not promote educational material reuse.
We propose to overcome these defects coupling adaptive activities of the AIWBES to traditional LMS. This is the Ocean Project: a combined use of ontologies, fuzzy logic and a collection of libraries (called Omega Framework), together with traditional LMS, thus allowing the reuse of existing educational systems; being compatible with SCORM, IMS or other e-learning standards; in short reducing the architecture complexity of adaptive systems. Up until now, the theoretical model has been defined and the architecture developed. In the present work, both the data model and the architecture are described. Furthermore, an example of reasoning based on fuzzy inference rules is given.

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.208

2. The Ocean Data Model

The first part of the Ocean Project is the Ocean Data Model which has been built on fuzzy ontologies, giving a formal representation map of concepts and granting standardization for portability from an e-learning system to another.

Let us note that in [7] an ontology is already used as a support for an e-learning system, but only for local indexing, without any learning adaptivity to students.

However, a standard ontology was not enough for the needs of the projects because even if an ontology allows a formal representation of concepts and relations, it does not allow to express the strength of relations among concepts, which is one of the needs of the Ocean Project. Thus, fuzzy ontologies [8], [9], i.e., ontologies combined with fuzzy logic, have been considered. As an example, take the course "programming". A fuzzy ontology allows to say that "C language" is a "programming language" with an high strength of the relation. Figure 1 shows this example, where the learning material ("example.doc" and "exercise.pdf") is directly linked to a topic ("IF-ELSE").

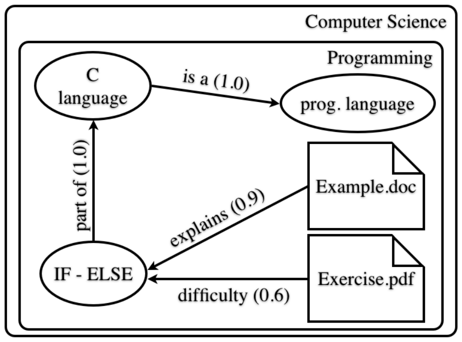

Figure 1. Example of fuzzy ontology.

2.1. Data Model and User Profiling

The Ocean Data Model is defined as the logical structure

$$DM_{\mathcal{O}} :=< resources, relations, tags, \\ usersmodels, user >$$

With these elements it is possible to describe users who access to a system based on the Ocean Data Model: it is possible to "tag" elements of the fuzzy ontology for fast searches and, obviously, it is possible to express relations that exist among them. In detail:

- $resources$. The Ocean Data Model uses fuzzy ontologies for the logical organization of educational content. This means that the data model gives the structure for educational content with the general structure $resources :=< course, topic, topic\_instance >$. A course can be made up of various topics and a topic is made up of its instances. For example, in Figure 1, $course$ = Programming, $topic$= {Prog. language, C language, IF-ELSE}, $topic\_instance$={Example.doc, Exercise.pdf}. Note that it is possible to establish relations among every kind of resource, so it is possible to link topics of different courses between them or with other courses or topic instances. There is another resource, i.e. "lessons", but in $DM_{\mathcal{O}}$ a lesson is not a simple resource, nor is it a logical part of something (like a topic and a course), but it is something that describes a way to navigate knowledge. A $lesson$ is made up of rules that tells the adaptive system how to vary the learning path of a student according to his/her learning skills (see Section 4).
- $relations$. It is an oriented arc that connects two objects of the database. Each relation is fuzzy, so that a weight (or fuzziness) defines the strength of association from 0 to 1. In the Ocean Project some relations have been introduced as "default" relations. However, it is possible to add as many relations as needed. The "default" relations are: "is a", "is kind of", "is part of", "has", "has part", "equals to".
- $tags$. It is a label that can be used to link all the $DM_{\mathcal{O}}$ objects but $relations$.
- $usermodels$. This element is necessary in order to describe various aspects of a user. A $usermodel$ has been modelled as a collection of properties, so the Ocean Data Model describes "user model properties", $UM_{\mathcal{P}}$. In detail, $UM_{\mathcal{P}} :=< name, value, type, history, list_{tags} >$. A $name$ is the property's name (e.g.,"age"), a $value$ is the instance of the property (e.g.,"24"), a $type$ is the nature of the property (e.g.,"integer"), a $history$ is a log of the evolution of every property in time (e.g.,"$24, 23, 22$") and a $list_{tags}$ is the collection of all the tags.
- $user$. This field is used to identify a user. In detail, information is reported on his/her "name", the list of $usermodels$ and a list of all user relations, e.g. $understood$ relation (see Figure 2).

The whole model has been described in RDFS [10], so that the data model can be easily updated by just modifying the RDFS statements that describes it, eventually introducing new constraints or objects. For lack of space, we avoid to include details about RDFS implementation.

2.1.1. User Profiling. In an adaptive system it is necessary to create different models for different people, not only because it is necessary to profile all of them, but because it could be required to track the learning models for each student (or "reasoning models"). The adaptive system has to correctly adapt itself: different reasoning models could require different student models [11] [12].

In literature two possible reasoning models are utilized: one based on Bayesian nets and the other based on the use of fuzzy control rules. The goal for both reasoning models is the same; however they could differ in the way they treat the user, so they could require different data or not update them in the same way. In Figure 2 a very simple example of a user profiling is reported. In particular, it is related to the understanding of a user about two topics. The initial

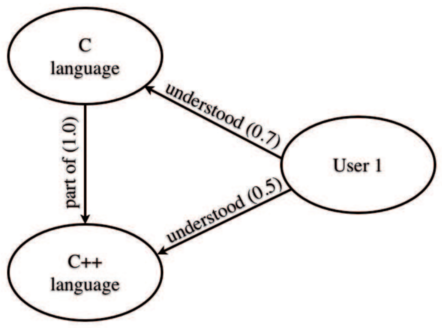

Figure 2. Sample with a user.

values of each student can be defined according to some previous knowledge or by a default value (this last is the choice adopted in Section 4).

3. Omega Framework: the architecture

The second part of the Ocean Project is about the design of a software that could be used to create an AIWBES [11] i.e., the architecture for a framework named Omega Framework. Instead of creating a whole new system, the Omega Framework was designed and intended to be installed "by side" of an existing LMS; thus solving the greatest issues of common AIWBES and allowing:

- reuse of educational content;
- reduction of software architecture's complexity and integration with different e-learning platforms;
- compatibility with e-learning standards like SCORM or IMS.

The Omega Framework is a collection of functions designed to be able to be expanded in the future. In particular, there are two main kinds of plugins (or "extensions"): *"interaction" plugins* and *"reasoning" plugins*.
These plugins are needed for catching users' actions (teachers, students and e-learning platform administrators), both in input and output directions. This means that the framework is invisible to the end user and that through its interaction extensions, every user action is forwarded to the framework's core so that a decision can be made and outputted back to users. Moreover, a reasoning plugin must be able to understand the profile of users (according to the extension profile format, or other extensions format), so the mix of "user action - user profile - knowledge base" can be used to infer an output for a particular user.

3.1. The Omega Framework

The Omega Framework is split into two main blocks: **Core** and **Ocean Database** (see Figure 3). The functions

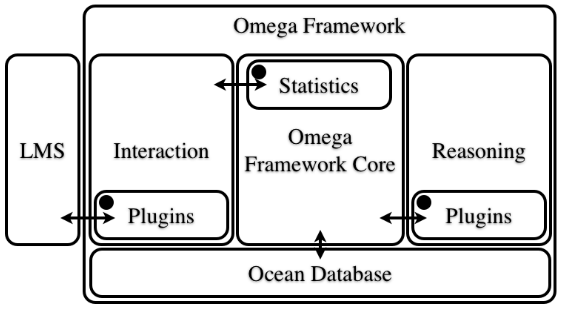

Figure 3. Structure of Omega Framework.

of the "Core" are about:

- plugin-to-plugin communication, i.e., message forwarding from an interaction plugin to a reasoning plugin and viceversa. This is necessary in order to understand actions of users and to make the system respond adequately to the input, thus introducing variations in the learning paths used by students; note that plugins of the "full" type do not necessarily need to communicate with other plugins because they can implement both interaction and reasoning functionalities;
- plugin-to-database interaction, necessary for every kind of plugin: every plugin could be capable of interacting with the Ocean Database in write or read mode. In particular, interaction plugins could require elements of the knowledge base to be shown to users, while reasoning extensions could require some data to be processed in order to produce correct output relative to user input. Note that plugins cannot write directly onto the Ocean Database, but they can do this using functions exposed by the core of the framework;
- maintenance tasks, management of installed extensions, Ocean Database maintenance (import, export and backup), logs and usage statistics management (export and backup).

The Omega Framework needs to manage its database, called "Ocean Database", that is used to hold all the metadata associated with the knowledge base built according to the $DM_{\mathcal{O}}$.
This means that the metadata are created to describe the logical organization of the educational content managed by the LMS which the framework is installed into.

Finally the communication scheme of the framework is introduced. There are two directions of communications:

- **from Learning Management System to Omega Framework**. This situation can be described by using the workflow in Figure 4:
  1) a user commits an action;
  2) if the user is not a student, interaction can happen with the Ocean Database or preferences of the framework (eventually stored in a separate preferences file): in this case no reasoning plugin is involved in the process; the communication is received directly from the core of the framework.
     If the user is a student, then the framework decides what reasoning extension should be used for him/her, then it forwards the call to the appropriate plugin;

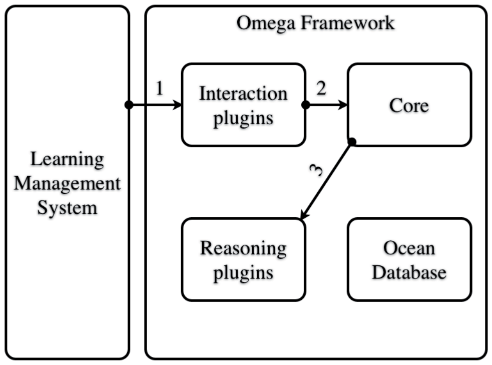

Figure 4. Communication scheme: LMS-to-FW.

- **from Omega Framework to Learning Management System**. This situation can be described by using the workflow in Figure 5:
  1) if the user is not a student, the core accepts his/her action, then outputs a result via an interaction plugin of the framework.
     If the user is a student, then the reasoning plugin invoked in previous steps analyzes the user's profile and the knowledge base, then decides which output should be produced;
  2) the core of which Framework which receives the output decides what interaction plugin to invoke, then communicates the output back to the user via the chosen plugin (note that interaction extensions could require interaction with an LMS, however this case is not managed by the Omega Framework).

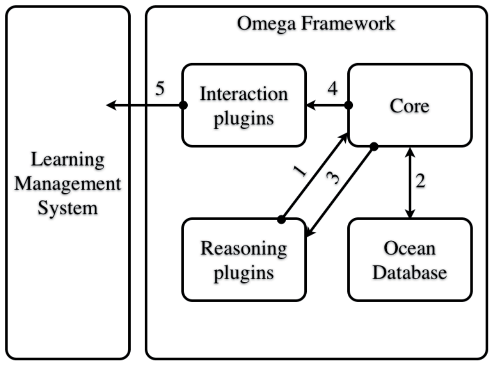

Figure 5. Communication scheme: FW-to-LMS.

4. Reasoning

To show how a system "Ocean-like" could work, a reasoning example will be introduced. In the $DM_{\mathcal{O}}$ a $lesson$ expresses how to "navigate" the knowledge base and it has the format ⟨reasoning module identifier⟩@⟨rule|...|rule⟩. A rule, relative to an object o, has the structure ⟨RULE NAME⟩ ={OBJECT IDENTIFIER, CONSTRAINT ON RELATIONS, CONSTRAINT ON FUZZINESS, RULE TYPE} where:

- an OBJECT IDENTIFIER tells which object o in the knowledge base the rule is "attached" to;
- a CONSTRAINT ON A RELATION tells the reasoning module that is parsing the rule which relations involving o must be considered in the rule application;
- a CONSTRAINT ON THE FUZZINESS OF RELATIONS works like the previous one;
- TYPE OF RULE tells reasoning models if the rule is positioned at the beginning, inside or at the end of a lesson.

We note that a lesson can be used by many reasoning models, even if associated to a particular one, but it is necessary for every reasoning model which wants to read a lesson to be "compatible" with it. Further, in $DM_{\mathcal{O}}$ we have two default relations: "has understood" and "is the answer for". With these relations it is possible to create educational material expressing the knowledge of a user about a particular course, a topic or a topic instance. It is also possible to create tests by linking objects in the knowledge base via the "is the answer for" relation. Let us underline that the fuzziness of “is answer for” expresses the correctness of an answer to a question.

4.1. Reasoning with fuzzy rules

The aim of a reasoning module is understand which material is adequate for a student given its level and the fact that is learning or not. In this particular example, two new relations and two user models will be introduced. The relations are: “is exercise with dLevel” and “is explanation with dLevel”. These relations can be used to express if an object of the data model is an exercise or an explanation for something, both with a difficulty level. The two user models are:

- A “global” model to have a general description of a student. It consists of the “level” of a student (good, bad or normal), the difficulty level of lessons that can be given to the student, the number of answers given by the student and the number of correct ones.
- A “local” model to have a lesson-specific description of a student. It contains the same information of the general one.

Note that the values of properties of the first model are calculated using the mean of the values of the “lesson specific models” and the mean of the difficulty levels reached for every lesson. Finally, the starting value of the difficulty property is 0.6 in both models.

Thus, the workflow for this reasoning model is:

1) initialize global model;
2) initialize a local model for the lesson being used by the student;
3) update the local model initialized during the lesson;
4) update the global model combining values of all the local models;
5) repeat steps from 2 to 4 for every lesson used by a student.

The update process of the local model is done through the application of the fuzzy control rules [13] listed below:

- IF “student is bad” AND “student is not learning” THEN “make lesson easier with *Bad* factor”;
- IF “student is bad” AND “student is learning” THEN “make lesson harder with *Bad* factor”;
- IF “student is normal” AND “student is not learning” THEN “make lesson easier with *Normal* factor”;
- IF “student is normal” AND “student is learning” THEN “make lesson harder with *Normal* factor”;
- IF “student is good” AND “student is not learning” THEN “make lesson easier with *Good* factor”;
- IF “student is good” AND “student is learning” THEN “make lesson harder with *Good* factor”;

As readers can see, the idea behind this set of rules is to make lessons easier if the student is not learning, harder otherwise: in particular difficulty changes are calculated differently according to the student level, so a good student will see faster changes, whereas bad students will see slower changes. The input fuzzy sets defining users are introduced as shown in Figure 6. When a student belongs

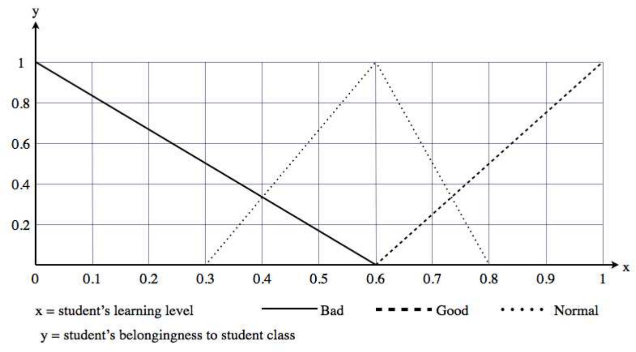

Figure 6. Fuzzy sets.

to more than one class, the one with a maximum fuzzy degree is chosen, and we denote it by l. To calculate if a student is learning or not, the last four outputs produced by the user are considered. If the user produced at least two out of four “good actions”, then the student is learning, otherwise not. In particular, the degree of learning w assumes the following values: $w = 0.25$ when the last four actions are ”reading”, two exercises and two ”reading” will correspond to $w = 0.5$; four correct exercises lead to $w = 1$. This choice has been made because reading can be seen as a “good” action, but with a lower evidence than a correct answer. On the contrary, the degree of ”not learning” is $w = 1$ if the student did not good actions, 0.5 (resp., 0.25)if only one correct exercise (resp., reading) was done. Note that the value w will be used during the fuzzification process.

Once the correct rule is found, we have to defuzzify the result, and this process is based on the fuzzy set *difficulty variation* described by the linear function $dv(x) = -5.26 * x + 1.05$, defined in the interval [0.01;0.2], where x is the membership degree of student’s level l. Then, the value of the fuzzy variables “make lesson easier” or “make lesson harder” is determined according to the following rules:

- “make lesson easier (resp., harder) with *Bad* factor” = - (resp., +) $0.5 * w * dv(x)$;
- “make lesson easier (resp., harder) with *Normal* factor” = - (resp., +) $0.75 * w * dv(x)$;
- “make lesson easier (resp., harder) with *Good* factor” = - (resp., +) $w * dv(x)$;

Finally, the difficulty of the new material proposed to the student is chosen in the interval computed by the above rules plus or minus 0.25.

We note that all this process of "fuzzification-defuzzification-decision" is made using the local model of the student associated with the considered lesson. It consists of an operational process based on membership values stored in the fuzzy ontology but it does not relate to any declarative formal reasoning based on the fuzzy ontology (such as Pellet of FaCT++).

4.2. Example

Consider a new student for the Ocean System. His/her global model identifies this student with a competence of 0.6 and a sustainable lesson difficulty of 0.6. This classifies the student as a normal one with degree 1 (see Figure 6). Suppose that the student is learning, having read two lecture notes and correctly answered two exercises ($w = 0.5$). Thus, the system must change the difficulty of the lesson making it harder and the following computations are performed:

- the fuzzification of the student's level, with the result to be a Normal student with degree 1: $Normal(0.6) = 1$. Thus, the rule to apply is the sixth;
- the difficulty variation is $dv(1) = 0.01$;
- since the fuzzy rule applied is the sixth, the difficulty must increase by c = 1 * 0.01;
- c is then multiplied by the factor $w = 0.5$ (because two reading materials out of four were presented to the student), so c * 0.5 = 0.005;
- the new difficulty of documents the student can use in the next steps of the lesson is included in the interval [0.6 + 0.005 - 0.25;0.6 + 0.005 + 0.25] = [0.355;0.855].

Finally, the global student's level is updated computing the mean of all the correct answers of the lessons. In this case, the mean of 0.6 (initial value) and 1 (due to the fact that the student did four good actions out of four), i.e., 0.8.

5. Conclusions

In this paper an overview of the Ocean Project was given, thus the Ocean Data Model and the Omega Framework were introduced. The framework and the data model have been designed to be used together in order to create an intelligent system for e-learning. The framework is intended to be installed "to support" existing Learning Management Systems, making the architecture modular and simple. In particular, the intelligent system uses a data model based on fuzzy ontologies; and it can profile every user in several manners according to different reasoning models (each one corresponding to a "reasoning plugin"). In this work, an example of a reasoning module based on fuzzy control rules was given, showing a possible way for the system to adapt to users.

References

[1] S. Kalyuga and J. Sweller, "Rapid dynamic assessment of expertise to improve the efficiency of adaptive e-learning," in *ETR&D*, vol. 3, 2005, pp. 83–93.

[2] P. Brusilovsky and C. Peylo, "Adaptive and intelligent web-based educational systems," *International Journal of Artificial Intelligence in Education, IOS Press*, vol. 13, pp. 156–169, 2003.

[3] E. Melis, E. Andrés, J. Büdenbender, A. Frischauf, G. Goguadze, P. Libbrecht, M. Pollet, and C. Ullrich, "Activemath: A generic and adaptive web-based learning environment," *International Journal of Artificial Intelligence in Education*, vol. 12, no. 4, pp. 385–407, 2001.

[4] P. Brusilovsky, E. Schwarz, and G. Weber, "Elm-art: An intelligent tutoring system on world wide web," in *Intelligent Tutoring Systems.*, vol. LNCS-1086, 1996.

[5] P. Brusilovsky, "A distributed architecture for adaptive e-learning," *International World Wide Web Conference archive Proceedings of the 13th international World Wide Web conference on Alternate track papers and posters*, pp. 104–113, 2004.

[6] I. G. Kennedy, S. Fallahkhair, R. Fraser, A. Ismali, V. Rossano, and A. Trifonova, "A simple web-based adaptive educational system (swaes)," *http://eprints.brighton.ac.uk/1041/*, Last check: 24th March 2009.

[7] W. Shih, C. Yang, and S. Tseng, "Ontology-based content organization and retrieval for SCORM-compliant teaching materials in data grids," *Future Generation Computer Systems*, 2009, doi:10.1016/j.future.2009.01.005.

[8] S. Calegari and D. Ciucci, "Fuzzy Ontology, Fuzzy Description Logics and Fuzzy-OWL," in *Proceedings of WILF 2007*, ser. LNCS, vol. 4578, 2007, pp. 118–126.

[9] S. Calegari and E. Sanchez, "Object-fuzzy concept network: An enrichment of ontologies in semantic information retrieval," *JASIST*, vol. 59, no. 13, pp. 2171–2185, 2008.

[10] W3, "http://www.w3.org/rdf," *Web Page*, Last check: 24th March 2009.

[11] P. Brusilovsky, "Adaptive and intelligent technologies for web-based education," *Knstliche Intelligenz, Special Issue on Intelligent Systems and Teleteaching*, 1999.

[12] N. Kushmerick, K. McKee, and F. Toolan, "Towards zero-imput personalization: Referred-based page prediction," in *Adaptive Hypermedia and Adaptive Web-Based Systems: International Conference*, 2000, pp. 133–143.

[13] X. Wang, D. Ruan, and E. Kerre, *Mathematics of Fuzziness-Basic Issues*, ser. Studies in Fuzziness and Soft Computing. Berlin-Heidelberg: Springer, 2009, vol. 245.