Towards a Generalized Architecture for the Integration of Tools in LMSs

doi:10.3991/ijet.v4s1.792

J. Fontenla, M. Caeiro and M. Llamas
University of Vigo, Vigo, Spain

***Abstract*—In this article we introduce the main components of a generalized architecture to facilitate the integration of tools in LMSs. This proposal tries to improve the reuse possibilities of the tools in e-learning systems. Reusability has been suggested as a key solution to reduce the high costs of the development of educational experiences in e-learning systems. Up to now, reutilization has focused mainly on the educational contents around metadata standards, contents formats, packaging systems, etc. However, educational practices usually involve tools to facilitate the communication, collaboration and work of students and teachers as well. This proposal is part of a wider solution based on the language PoEML, in which not only the possibility of the inclusion of tools in LMSs is considered, but also the management of its utilization.**

***Index Terms*—Educational modeling languages, Perspective-oriented EML, Learning Management Systems, Web Services, Reusability.**

I. INTRODUCTION

Since it was noticed that e-learning was not a cheap alternative to traditional learning, the development of specifications and standards to promote the reuse of educational resources has been one of the priorities of the research community. Up to now this initiative has focused mainly on educational contents, leading to results such as ADL SCORM [1] or IMS QTI [2]. Basically, these proposals define how to structure and to arrange the media contents to support and facilitate the development of educational experiences in *Learning Management Systems* (LMSs).

However, the reuse of e-learning systems should be applied to the rest of the resources as well as to contents. Specially, it's important to mention that many educational practices, mostly those that follow pedagogical approaches based on collaboration and practice, require more resources for their development rather than media contents. In order to cover successfully the needs of this kind of pedagogical approaches it's required to provide a set of tools, in our context supported by the *Information and Communications Technologies* (ICTs), with which students and teachers can interact, experiment, communicate and even manipulate the contents themselves. Current LMSs usually include a basic set of tools that broadly cover the most common needs, but the support to the inclusion of third-party tools or to the development of new tools is very expensive or even non-existent. So, for the sake of developing versatile e-learning systems at low cost, the need of facilitating the reuse of tools arises.

One of the solutions to facilitate the reuse of tools in e-learning systems consists on the use of services external to the LMSs themselves. The apparition of Web Services as a broadly accepted programming paradigm allows the provision of functionalities distributed along different servers. This tendency, known as *Cloud Computing* [3], allows the distribution of computational functionalities along different physical resources (servers) in a transparent way for final users. In the domain of e-learning, the adoption of this paradigm allows to place the tools not in the same systems of the LMSs', but in an external location. This way, a new business model in which the development of LMSs and tools can be split is posed. The problem of the reuse of tools is thus transferred to the integration of external tools. In this model, the LMS provides functionalities to achieve the planning, coordination and management of educational events. On the other hand, tools provide the necessary functionalities for students and teachers to communicate, collaborate and generally work in the development of their educational tasks.

The proposal of using external tools requires the consideration of several problems that must be solved. Firstly it could be posed the possibility of linking the LMS with a set of external tools in a static or a dynamic way. If dynamic, search systems should be needed in order to locate tools suitable for the desired characteristics. Middleware systems are also needed in order to control the access of users from the LMS to the external tools, handling their authentication, the sessions' control, the management of the actuation of the users in the external tools, and in general all the communications that might take place between the LMS and the tools.

Nowadays there are several groups working in specifications and recommendations that face some of these problems, although they are in an early stage and maybe with a narrower and less generalized sight than the one described here. In this article we present an architecture to deal with the different needs identified in the integration of external tools in EML-based LMSs. This proposal is part of a wider solution based on the educational modeling language PoEML [4]. In this language other aspects apart from functionalities provision with external tools are taken to account, with the general purpose of supporting the development of educational experiences according to different pedagogical approaches, mainly those based in collaboration and practice.

This article is organized as follows. Section 2 briefly introduces educational modeling languages, focusing in PoEML. Section 3 reviews in a critical way some specifications and recommendations on the use of Web

Services in e-learning environments. Section 4 depicts the proposed architecture to solve those problems identified in Section 3. Finally, Section 5 ends up with some conclusions.

II. EDUCATIONAL MODELING LANGUAGES

Educational modelling languages (EMLs) have been proposed to allow the creation of descriptions (or models) of didactic units. The objective of such descriptions is to allow their processing by suitable computational applications, namely LMSs based on the EML of the description, to support the development of didactic units. In this sense, it could be said that EMLs are an executable notation, involving those elements and processes that take part in educational scenarios.

Nowadays there is a de-facto standard in these languages named IMS Learning Design (IMS-LD) [5]. Some projects have dedicated resources for years to the creation of a LMS based on IMS-LD, but they have not been able to obtain a ready-to-use system. Perspective-oriented EML (PoEML) is a new EML that tries to improve the description and computational support of didactic units. The main characteristic of this language is the separation of the models of didactic units in several parts, called perspectives that, to a great extent, can be tackled separately. A first consequence of this separation is that each part may include several alternative designs, with independence of the descriptions made in other parts or with a controlled dependency. This way, the reuse of the models is facilitated.

PoEML divides the modelling of didactic units in 13 perspectives. These perspectives have been thoroughly described in other publications [6], so here we we'll just describe briefly those relevant for this article: those regarding the integration of external tools. The Tools perspective models the characteristics (functional and behavioural) of the tools required in the environments; one of the most original characteristics is that, unlike IMS-LD, it allows an indirect or decoupled characterization of educational tools used in a didactic unit, as well as allowing their explicit description. By this characterization, a tool is defined according to some functional (the expected functionalities) and behavioral (the permissions that allows to grant, the events that notifies, and the operations that allows to invoke automatically) requirements. Later, the LMS will be responsible of integrating an external tool satisfying such characterization. With that, the dynamic inclusion of external tools is achieved.

Apart from facilitating the inclusion of external tools, PoEML proposes three specific perspectives to do the management and control of the use that students and teachers make of such tools. These perspectives are:

- The Authorization perspective, that allows the assignment of permissions to the participants of the didactic unit. In this perspective, for example, it could be modeled that students can send messages to a forum, but only teachers are allowed to create new threads or to erase posts.
- The Awareness perspective, in which it can be modeled the capture of relevant events triggered by the interaction of the participants (students and teachers) with tools. Apart from the capture of events it can also be modeled their processing (e.g. filtering) or the report to some participants of those events of interest. The notification of events is very useful in collaborative educational scenarios, in which the rest of participants must be aware of those changes performed over some shared resource, as a text file or some code fragment.
- The Interaction perspective, in which it's possible to describe the automatic and controlled invocation of operations during the realization of a didactic unit. The automatic invocation of methods is useful in practical scenarios in which guided demonstrations are needed. Operations may also be invoked just when some events take place. For example, in this perspective it can be modeled that a chat tool must send a welcome message every time a new participant logs in.

III. EXISTING RECOMMENDATIONS ON TOOLS INTEGRATION

Service Oriented Architectures (SOA) are being seen as a highly flexible way to build up complex applications from decoupled components. During last years, some projects following this approximation have started in the field of e-learning. In this section we will comment the methodology of the three most remarkable ones according to us, and we will discuss some of their limitations, that will be taken to account when we propose an architecture based on the perspectives of PoEML.

The *E-Learning Framework* (ELF) [7] is an initiative due to the United Kingdom's *Joint Information Systems Committee* (JISC), in collaboration with the Australian *Department of Education, Science and Training* (DEST) and the United States' *Learning Services Architecture Lab* (LSAL). ELF does not specify any concrete architecture to integrate external tools in a LMS; on the contrary, its main purpose is to facilitate the development of architectures of LMSs based on Web Services. It identifies more than 40 necessary modules in a LMS providing a comprehensible set of functionalities. Thus, it allows the community to have a shared "vocabulary" and a reference framework for the development of e-learning systems. At the present moment there exist several projects related to some of the components identified in ELF [8], although there is not enough cohesion among them to build a LMS with a minimum of functionality.

The specification *IMS Tools Interoperability* (IMS-TI) [9] proposes a more specific framework than that of ELF. IMS-TI makes use of a combination of Web Services and proxying to integrate external tools in a LMS. The specification, currently at its version 1.0 and with the 2.0 under development, tries to eliminate the necessity of proprietary interfaces between e-learning platforms and tools which, ultimately, would allow that both classes of systems could follow independent development processes, thus promoting specialization, innovation and competition. The configuration of the tools is done by editing a XML file at the LMS side, although it's expected that in version 2.0 this can be done in a more automatic way.

In our opinion, IMS-TI has two main drawbacks. The first one is the lack of reference implementations that could be used as a guide for new developments. There are only few implementations, such as the public demonstration for the "*alt-i-lab 2005 Conference*" [10], or those prepared for the "*Google Summer of Code 2008*" [11], which is not in agreement with the expectative put on IMS-TI. The second one is that, despite it allows the seamless execution of remote tools, IMS-TI doesn't provide any ways to control and manage the use of the tools by teachers and students.

Finally, *CopperCore Service Integration* (CCSI) [12] is another architecture proposed for integrating tools in IMS-LD-based LMSs. CCSI is an intermediate layer between the IMS-LD engine CopperCore and the presentation layer built upon CopperCore. Every time the presentation layer wants to invoke a tool (for example, an assessment tool) it will access the CCSI layer, which in turn will invoke the tool. The latter will send the results to CCSI, which in turn will forward them to the presentation layer and to CopperCore. The communication between the presentation layer and the different tools is possible because CCSI shows an interface with a set of predefined methods for each kind of service that may be accessed; after that, CCSI will handle the adaptation of the call of the presentation layer to invoke the concrete tool.

CCSI has some limitations that make that its acceptation is not as big as it would be desired. Firstly, as IMS-TI, neither it supplies any mechanisms to control and manage the use of the tools, nor it allows to supervise the activity of the students, nor it allows to configure the automatic invocation of methods. Secondly, only one tool of each kind can be integrated (e.g. it's not possible to integrate two different text editors), which may dramatically reduce the possibilities of the system to satisfy the needs, preferences and personal limitations of the users. Thirdly, a complicated editing process of XML files must be accomplished in order to integrate new tools, which may be difficult for those users which are not familiar with this language. Finally, the architectural design itself of CCSI implies extra work for the developers of applications, as they must supply the tools as well as extra modules for their integration with CCSI.

IV. NEEDS OF THE INTEGRATION OF EDUCATIONAL TOOLS

The above-mentioned architectures have some drawbacks making their implementation level quite reduced. A common point to all of them is that, although they offer a framework to integrate tools, they don't allow to control and manage the way that teachers and students use them.

In this section, the architecture of a PoEML-based LMS will be proposed, allowing to solve these problems. The processes needed to integrate a tool in this LMS and to configure its permissions, events and operations will be shown, and some related problems will be discussed.

A. *Architecture of a PoEML-based LMS*

The decomposition into perspectives carried out in PoEML allows us to tackle the design of the LMS in a modular fashion, discarding the monolithic design of current LMSs which would be difficult to develop and to maintain, see Fig. 1. In this figure we can see three different parts:

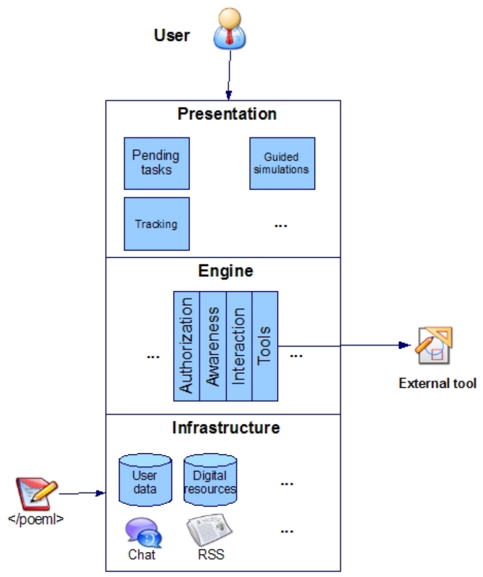

Figure 1. Structure of a PoEML-based LMS.

- The central layer is the *Engine*. This part supplies the core functionalities of the system. For the development of this engine, the development of independent modules according to the perspectives of PoEML is proposed. One of the modules, the one related to the Tools perspective, would be the responsible for managing the configuration of the tools, the capture of the events triggered by the tools, the automatic invocation of operations, the control of sessions and instances, and the data transfer.
- On top of the engine we can see the *Presentation* layer, in which they can be found those applications that build up the user interface for teachers and students. The use of a presentation layer allows us to use the same engine and the same infrastructure, but offering different functionalities and appearances.
- Beneath the engine is the *Infrastructure* layer. This layer supplies a set of storing functionalities and general purpose services, from databases with the data and marks of the students, to common tools that should not have availability problems such as forums, email or calendars. It's worth mentioning that this layer is the one that will receive the PoEML-formatted file that will be processed and played by the Engine.

Given the loose coupling of the parts that build up the Engine, inherited from the one among the perspectives of PoEML, it's possible to develop them independently. Thus, none of the modules of the other perspectives will influence on the way the modules of the Tools, Authorization, Awareness and Interaction perspectives are implemented, which are the focus of this article. So, in the following we can ignore the architecture of the rest of the LMS.

B. *Classification and communications with tools*

One of the most remarkable characteristics of PoEML is the decoupled characterization of tools, given its functionalities, permissions, events and operations. However, PoEML does not specify how tools must be classified according to these four parameters. Therefore, it's possible to classify them semantically using, for example, an ontology. In this line, there are plenty functional projects, such as Ontoolcole [13].

The availability of a vocabulary for the characterization of tools is the basis of the development of systems to classify, search and configure them. To allow the configuration of tools, the use of a public interface is proposed. This interface has both read methods and write methods. Read methods allow us to know the permissions, events, operations and functionalities of a tool. In response, the tool will return a subset of the values defined in the ontology. An example of a read method could be *getPermissions*(), that returns a description of the available permissions of the tool, in agreement with the concepts of the ontology. On the other hand, write methods allow us to activate or to disable some of the characteristics of the tools, receiving as parameters the characteristic to be modified and a boolean value with its new state. An example of a write method could be *setEvents*(*"newMember"*, *true*), which says that the tools must notify events when a new member joins. Through this set of methods, it's possible to configure systematically all tools using an only interface (and so, an only application), and the LMS may automate the integration of tools.

Other projects such as CCSI also define a generic API, but they do not have methods to describe the tool that's being integrated, nor to perform the control and management of its use according to its permissions, events and operations.

C. *Configuration, integration and use of tools*

The use of a generic API facilitates the process ranging from the configuration of a remote tool to its use by a student or a teacher, shown in Fig. 2. The course manager will ask the tool for the list of parameters that can be configured (1), invoking the appropriate methods of the API. As a result, the tool will send some data (functionalities, permissions, events and operations) about itself (2). The course manager will choose from these data those permissions, events and automatic operations that are suitable for the course, and will create a *profile* with them (3), again using known methods of the API. This profile will be applied to the sessions of all those users (teachers and students) that access the tool. Should another configuration be necessary (for example, if teachers had to do some management over the course), another profile with different parameters should be created. Next, the course manager will store in the tools database the data concerning the tool that has just been configured (i.e. storing those permissions, events and operations supported by the tool and configured in the profiled, and the URL of the tool) for future use. The tool is thus correctly configured to be used.

The process continues when a student, using a web browser, wants to resume his/her activities. Firstly, he/she will authenticate when logging in the LMS (5), after which he/she will receive an affirmative or negative confirmation (6). If affirmative, the browser of the student will automatically request to the LMS a list of the courses in which the student is participating (7). The LMS will return such list (8), and the student will choose a concrete course (9). When a course has been chosen, the LMS will send another tree-shaped listing (10), whose nodes are the different educational scenarios that build up the course (e.g. "Theory" and "Practice", which in turn could be built up by the educational scenarios "Practice 1" and "Practice 2"). The student will choose a educational scenario (11), and finally the LMS will display a web page with all the necessary information for the development of the educational scenario, including links to external tools (12). These tools may already have running instances (e.g. a collaborative text editor that is already being used by other students) or they may not, in whose case it should be launched; the Tools perspective module will be responsible for managing the number of instances of each tool. In any case, when a student accesses a tool (13) he/she will authenticate himself/herself (for example, using some hash code sent will the HTTP POST method), and the tool will look for the profile to be applied. From this moment, the tool will display a user interface in accordance with the permissions assigned to the student, will notify events to the LMS as configured in the profile (14), and will execute all those operations required by the LMS. Finally, with all the events notified the LMS will generate logs, which will be stored for future use (15).

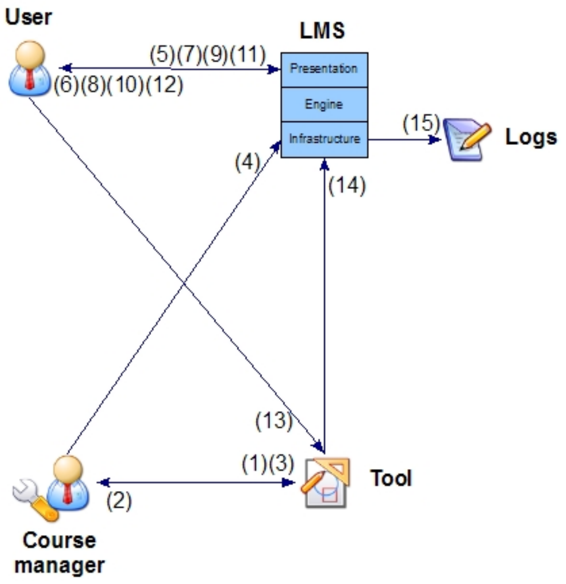

Figure 2. Configuration and use of external tools.

D. *Management of permissions, events and operations*

One subject that must be addressed is the management of permissions, events and operations during the process described above.

1) *Management of permissions*

The permissions that users are granted are given by the active profile. This implies that users can't modify their permissions during a session, unless they change their profiles. This change of profile is possible using different authorization specifications. As pointed in Section 2, a perspective may include different specifications that may be activated or deactivated dynamically (e.g. according to the marks of the student), so it's possible to assign different permissions to a participant. The interface displayed will be in accordance with the permissions granted, hiding those options of the tools that require more privileges.

2) *Management of events*

The profile also contains all the information regarding the notification of events. Firstly, the URL where events must be sent to (namely, LMS's URL) is included; once events have been received, the Awareness perspective module will process (e.g. filter) and notify them to those participants interested or to other tool (e.g. for their persistent storage).

3) *Management of operations*

The events received at the LMS may be used to determine when some operations must be invoked automatically. For example, it could be interesting to invoke an operation to silence a chat room when the "*The teacher has joined the room*" event is received. Other operations must be invoked according to a temporal specification, independently from the events generated by the use of the tool, for example "Create a chat room for the subject *Distributed Computing* on June 29th at 12:30".

In any case, in the same call of the invocation of an operation, the concrete instance of the tool must be specified (e.g. in the example of the chat tool, the operation to silence the room must only be applied to the instance of the students, not to the instance of the teachers).

E. *Data persistence among sessions*

Another important subject is the persistence of data between different sessions of the tools. A user should be able to resume the work in the same state it was when the last session concluded, in the same way it would be possible if the tool were integrated in the LMS (e.g. in the case of a collaborative text editor, it should be possible to continue editing the text of the last session). Three possibilities are proposed:

- **Client-side storage**: the user browser will use techniques such as cookie sending. The cookie field will contain the data of the last session. This technique has the important drawback that it's only feasible when the amount of data is low, so it wouldn't be suitable to send, for example, a whole file.
- **LMS-side storage**: whenever a remote tool is invoked, session data will be transferred from the LMS. This solution has the advantage that the LMS has control over session data, thus avoiding problems of data loss due to availability problems of the tools. However, it implies that LMSs and educational tools can't be developed independently. Indeed, tools developers must not assume that there will be external systems (in this case, the LMSs) that will store and manage session data.
- **Tool-side storage**: data are stored at the tool. This option is the most interesting, as it could be desired that a remote tool could be used in a standalone way, with no need of other systems supplying it data.

So, the system storing and managing session data are the remote tools themselves. However, an intermediate solution could be applied; to deal with availability problems of the tools, backup copies of the data could be stored at the LMS.

V. CONCLUSIONS

The task of developing a new LMS can be extremely complex, as all aspects of a didactic unit must be taken to account. Following the separation of concerns approach the problem can be faced in a more easy way. PoEML is an EML following such approach, so it's natural to build a PoEML-based LMS.

One of the perspectives considered in PoEML is the Tools perspective, which allows their decoupled description, therefore trying to promote the reuse of the models of didactic units. Despite nowadays there are some recommendations to promote the reuse of didactic units, they are only focused on educational contents and ignoring the tools used to manipulate them.

The use of Web Services in e-learning environments is a promising approach to complement such initiatives. Firstly, software developers can specialize and focus their efforts either in the LMS or in the external tools. This implies lower developing costs and a shorter period between the releases of new versions. Secondly, it's possible to develop ad-hoc tools for a concrete didactic unit, and use them in different LMSs. Thirdly, teachers could choose the most suitable tools for the didactic units among a broad set of tools, as they wouldn't be exclusive of a concrete LMS. Finally, it would be possible to build up LMSs supporting a bigger amount of users, as computational load would be spread through the servers of the LMS and the tools.

In our opinion, the existing specifications on this subject (mainly IMS-TI and CCSI) are still in an early stage, or they don't fully support the control and management of the tools.

REFERENCES

[1] Specification ADL SCORM. Last accessed in may, 2008 at: http://www.adlnet.gov/downloads/DownloadPage.aspx?ID=237

[2] Specification IMS Question and Test Interoperability v2.1. Last accessed on may, 2008 at: http://www.imsglobal.org/question/qtiv2p1pd2/imsqti_oviewv2p1pd2.html

[3] Gross, G., *Google, IBM Promote Cloud Computing*. PC World, 2007.

[4] Caeiro Rodríguez M., "PoEML: A separation-of-concerns proposal to instructional design", *Handbook of visual languages for instructional design: theories and practices*, editado por Botturi L. y Stubbs T., IGI Global www-igi-pub.com, 2007.

[5] Specification IMS Learning Design. Last accessed on may, 2008 at: http://www.imsglobal.org/learningdesign/

[6] Caeiro Rodríguez M., Llamas Nistal M., Anido Rifón L. "A Separation of Concerns Approach to Educational Modeling Languages", *Proceedings on the 36th Annual Frontiers in Education Conference, FIE'06*, San Diego, California.

[7] Wilson S., Olivier B., Jeyes S., Powell A., Franklin T. "A Technical Framework to Support e-Learning." JISC, 2004. Accedido en mayo de 2008 en: http://www.jisc.ac.uk/uploaded_documents/Technical%20Framework%20feb04.doc

[8] Roberts T., Easterby-Smith S., "Can one technical framework support all our eLearning needs?". Accedido en mayo de 2008 en: http://www.alt.ac.uk/altc2004/timetable/files/160/Technicalframework.pdf

[9] Specification IMS Tools Interoperability. Last accessed on may, 2008 at: http://www.imsglobal.org/ti/index.html

[10] Press note of the IMS-TI demonstration at the *alt-i-lab 2005*. Last accessed on may, 2008 at: http://www.sakaiproject.org/index.php?option=com_content&task=view&id=258&Itemid=312

[11] Google Summer of Code 2008, http://www.sakaiproject.org/soc2008/

[12] CopperCore Project oficial site. Last accessed on may, 2008 at: http://coppercore.sourceforge.net/

[13] Vega Gorgojo G., Bote Lorenzo M. L., Gómez Sánchez E., Asensio Pérez J. I., Dimitriadis Y. A., "Ontoolcole: Supporting Educators in the Semantic Search of CSCL Tools", *Journal of Universal Computer Science*, vol. 14, no. 1, 2008, pp. 27-58.

AUTHORS

J. Fontenla is a Telecommunications Engineer and a PhD Student in the University of Vigo. He is currently Assistant Teacher at the Department of Telematic Engineering, University of Vigo.

M. Caeiro received his PhD in Telecommunications Engineering from the University of Vigo in 2007. He is currently Assistant Teacher at the Department of Telematic Engineering, University of Vigo. He has received several awards by the W3C, NAE CASEE new faculty fellows and the IEEE Spanish Chapter of the Education Society.

M. Llamas received his Eng. degree (1986) and his Ph.D. degree (1994) from the Polytechnic University of Madrid. From 1994 to 1997 he was Vicedean of the Higher Technical School of Telecommunication Engineers. From 1999 to 2003 he was the head of the ICT Area of the University of Vigo. He is member of ACM, IEEE and IFIP WG3.6 (Distance Education). He has received several awards by the W3C and IEEE.

This work has been funded by the Spanish Ministerio de Educación y Ciencia under grant TIN2007-68125-C02-02, and by the Galician Consellería de Innovación e Industria under grant PGIDIT06PXIB32 2270PR.

This article was modified from a presentation at X International Symposium on Computers in Education (SIIE2008) 1st-3rd October 2008, Salamanca, Spain. Manuscript received 08 January 2009. Published as submitted by the authors.