41st Annual Frontiers in Education Conference: Celebrating 41 Years of Monumental Innovations from Around the World. Rapid City, South Dakota, October 12-15, 2011 (2011). IEEE, 61-66.

DOI: http://dx.doi.org/10.1109/FIE.2011.6142902

Towards Flexibility on IMS Learning Design Scripts

Luis de-la-Fuente-Valentín, Derick Leony, Abelardo Pardo, Carlos Delgado Kloos
University Carlos III of Madrid, lfuente@it.uc3m.es, dleony@it.uc3m.es, abel@it.uc3m.es, cdk@it.uc3m.es

Abstract - **IMS Learning Design is considered by many authors the "de facto" standard in educational modeling languages. The versatility of the framework enables its use in very different situations. However, such versatile framework is usually hidden by its complex management. One handicap identified in practical experiences is the lack of flexibility of scripted courses during the enactment phase. The activity sequence and learning resources are rigidly defined during authoring. This fact makes difficult to react to unexpected events that may happen in live courses. Also, this rigidness does not allow instructors to give "their personal touch" to courses. This paper presents the improvements made on GRAIL - an IMS LD compliant player- aimed at the support of a flexible enactment phase. Two types of modifications are considered: the modification of the learning flow and the management of course content with a wiki engine. Finally, this paper discusses how the integration of third party services in the activity sequence relaxes the rigidness of scripted learning flows. Experiences deployed in real scenarios allowed analyzing how such integration offered flexibility in practical situations.**

Index Terms - Flexibility, IMS learning design, Scripted courses, Service integration.

INTRODUCTION

The rapid evolution and adoption of the so called *Web 2.0* provides the world of e-Learning with several affordances that change the way we interact with people through the Internet. Social applications like blogs, bookmarking tools, and wikis have impacted learning and teaching methods [1]. However, this increasing number of available resources does not implicitly produces learning: there is a need for structuring learning materials according to existing or emerging pedagogical models so that the interaction with peers and resources results in the desired knowledge acquisition. Learning scripts, which have been proved to be effective in different types of scenarios ([2]-[4]), are a representation of the learning flow that helps course participants know what to do at each moment. Thus, scripted courses allow blending pedagogical models with emerging computer-based learning resources.

In parallel, course packaging specifications play an important role in the current market. Shareable Content Object Reference Model (SCORM), the most widely accepted framework [5], promotes course reusability and interoperability as the major factor for its success. However, SCORM based courses have severe drawbacks in the orchestration of resources [6] that results in the impossibility of adapting the course content depending of the scenario where it is being delivered.

The specification IMS Learning Design (IMS LD) was built on the basis of reusability and interoperability to allow complex pedagogical models to be applied [7]. With these two assumptions, the specification was expected to be widely adopted by the e-Learning industry but, years after its publication, these expectations have not been accomplished. One of the reasons of this low impact on real scenarios is the lack of the flexibility promoted by the model. Despite it is not imposed by the specification, current supporting tools favor the division of the course life-cycle in three isolated phases. The practical experience with IMS LD reveals that the result of this division is a lack of flexibility that negatively affects the adoption of the specification.

This work presents the improvements of GRAIL, the IMS LD runtime environment for .LRN [8], are oriented towards achieving a flexible tool that reduces the gap between the specification and instructors. We present two major developments: flexible enactment phase and services integration. The former allows instructors to quickly react to unexpected situations in a way that the pedagogical model of the course is not affected by introduced changes. The latter aims at allowing arbitrary web 2.0 services to be included as part of a learning experience, including course adaptation based on third party data.

The rest of this paper is organized as follows. Next Section introduces the educational modeling languages as a way to produce learning scripts, and discuss the need of flexibility of such type of courses. Third Section presents the functionality developed in GRAIL towards course flexibility. Section four present examples extracted from practical experiences were the GRAIL functionality was used to tackle unexpected situations. Finally, the conclusions of this work are presented.

FLEXIBILITY IN LEARNING SCRIPTS

Course content delivery can be driven by the so called learning scripts: a pre-programmed set of activities that is enacted by the course actors. On one hand, learning scripts improve the pedagogical expressiveness of courses. On the other hand, the use of pre-programmed activities reduces the flexibility of the course flow. This section introduces the concept of learning script and discusses the need of flexibility.

I. Learning Scripts

There are many well-known situations in which our behavior usually follows a predictable pattern, for example when we go to a restaurant, we usually ask for a table, then order the meal, then eat it, etc. The mental representation of procedures present in those patterns is what cognitive science calls *scripts* [9]. In education, external representations of procedures are used to structure face-to-face learning, blended learning and distance education. Instructional design techniques, understood as the process of translating pedagogical model's principles of learning into a structured plan of activities and materials [10], take advantage of scripted learning to help students and teachers focus their attention on the relevant parts of the learning process [2]. In collaborative learning, for example, scripts are used to produce a desired interaction among course participants [3]. In general, they serve as a supportive strategy that encourages students to work on specific aspects which otherwise would be ignored or neglected.

There exist several manners to present structured learning flows. Instructors can provide a textual representation in form of clues or tables, or they can take a more graphical approach, such as activity diagrams. The way a script is presented influences the students' perception of the different course elements [2]. Educational Modeling Languages (EML) -like IMS Learning Design- provide a meaningful vocabulary to capture all the elements of a learning flow. Thus, a course definition usually declares the interactions among course participants and course material. A script constructed with this technique is able to be reproduced in a compliant runtime engine, promoting reusability of scripts. During course enactment, the runtime engine analyzes the course state and delivers the corresponding material at each moment. IMS LD provides a framework that enables a wide range of pedagogical models to be expressed. The specification provides a modeling language whose elements allow the creation of adaptive material [11] or collaborative tasks [12], among others. A course written with IMS LD is called a Unit of Learning (UoL).

The use of Modeling Languages implicitly imposes three different parts in the life-cycle of the course: authoring, deployment and enactment. The first stage is devoted to define the interactions among course elements and the conditions imposed to the different activities; the deployment stage takes the generic description of the course and allocates the resources (course participants, material) to instantiate this description into an actual course. The last phase is the enactment, in which the course participants interact among themselves and with the course material.

II. Definitions of Flexibility

There is no agreement on the meaning of flexibility in the context of learning. In several works, the term is related to the possibilities that a platform offers to be used anywhere, anytime [13]. Thus, a course is said to be flexible if it allows course participants to interact with peers with no place or time restrictions. In other cases, course flexibility is used as a synonym of adaptability [14] a flexible course delivers different material or activities depending on the user profile of the course participants. According to Dillenbourg [15], there is a difference between the interactions expected to appear in a course (the *planned script*) and what happens once the script is deployed and enacted (the *actual interaction pattern).* Flexibility is then related with the options that course participants have to tackle with this distance. That is, a flexible learning flow will provide mechanisms to deal with unexpected events that may affect the proper development of the learning activities by the students. In this paper, the term *flexibility* is used with this latter sense.

III. The Need of Flexibility

Building scripted learning courses over the ideas of constructivism, as in the case of problem based learning (PBL) or most collaborative learning patterns, leads to a conflicting situation ([3],[16]). On one hand, learning flow descriptions must be precise in order to be delivered in computer supported environments. Some interactions that are expected to happen according to the pedagogical model may not appear without the instructions given by the script. For example, in problem based learning, students are expected to investigate techniques to find a solution to the presented problem. However, it is unlikely that all of the students reach the best resources on each case without having the clues provided by the script. On the other hand, the basis of constructivism is that learners are responsible of their own learning construction. This assertion implies that students would require certain degree of freedom to plan their own strategies. Following the PBL example, students are expected to make their own findings about the best problem solving techniques, so that they will better understand the readings that will lead to the actual solution. In collaborative schemes, a too rigid approach would spoil the richness of free interaction.

Considering above arguments, it is unavoidable to wonder where to set the trade-off between the two opposite sides. The answer, however, depends on factors such as the actual scenario in which the course is delivered, the pedagogical model being applied and the profile of students taking part in the course. Some of these factors cannot be anticipated during the authoring phase of a course. For example, students in higher education tend to be more self-taught when they are on more advanced courses. In such sense, the same pedagogical pattern would require a different degree of coercion for being applied on both scenarios. Flexibility of course material allows to adjust the imposed constraints so that they fit to the actual scenario.

In general, during the enactment phase, some degree of flexibility is needed in order to adjust the course to the actual scenario parameters. In collaborative learning, for example, the success of an activity usually depends on groups having the right amount of members and being composed of the right user profiles. If the number of students does not match with the required situation, the course should allow the activity to be substituted by a different one with the same – or similar, at least – pedagogical objective.

Having the course properly adapted to the actual scenario, there is still room for unexpected events. First, statements may lack of clarity or even contain mistakes. In such case, the tutor would need to make the text more understandable, so he/she will be required to rewrite the content. Second, there are several types of events that cannot be anticipated by the teaching staff. For instance, an activity may require to be postponed due to illness of the participants, technological problems in the supporting system or bad performance of students in a given session. Therefore, flexibility allows teachers to tackle with such unpredictable situations that may affect the enactment of the course.

Despite the course flow requires some degree of flexibility for a successful enactment, the modifications should be carefully done. There are some constraints of the course that are essential to accomplish the original pedagogical intentions and they should not be overruled. According to [15], teachers are responsible of changes performed during the enactment phase. Therefore, they should be conscious of what can be modified and what cannot. Similarly, the platform that supports the course enactment should carefully select the allowed modifications of the course content.

IMPROVING THE ENACTMENT FLEXIBILITY

The support of flexibility of scripted courses is provided by the platform in which the courses are enacted. This paper presents the support of flexibility offered by GRAIL, the IMS LD runtime environment for the .LRN Learning Management System. A complete description of GRAIL can be found at [8].

I. Changes in the Course Content

Content in IMS LD usually takes the form of Web content, i.e., any content that can be rendered by a Web browser. The most common used format is HTML, but there is still room for multimedia content in the shape of flash applications, Java applets, images, videos, etc. The content is created during the authoring phase, and delivered to the users during the enactment. However, it is very likely for the content to include mistakes or inappropriate material. Such problems commonly arise during enactment, when the material is no longer modifiable.

GRAIL provides two different means to update course material once the package has been imported and the course is being enacted. It is up to the course administrator to select which of the allowed methods are preferred. We will refer to these methods as the *file-storage* and the *XoWiki*.

The *file-storage* method consists in a simple file substitution. GRAIL handles the UoL resources with a repository that can be browsed through a folder view. Whenever the teacher wants to modify the course, content, he/she can do it by substituting the corresponding file with a newer version. The file-storage supports content versioning and the IMS LD player will consider the last version of the files. The versioning system also allows recovering older versions of a file.

The other method to edit content is based on *XoWiki*, the wiki platform embedded in the .LRN platform. This functionality can be summarized as follows: during the package import, the content is translated into a wiki format, and the wiki tool is in charge of storing and rendering the material. When the teacher needs to modify some content, he/she has to access the corresponding activity and click on the "edit" button. The modifications are done in wiki format, whose simplicity does not require users to be trained. The use of the wiki allows performing modifications in the content without requiring previous knowledge of the UoL nor advanced computer skills. However, this method requires the inclusion of a permissions system. An example that illustrates this requirement is the following: an UoL with several student roles but no teacher role is loaded into the platform. Then, when the course has been instantiated and populated, a flaw in the content is detected. Which role is allowed to edit the content? For the sake of the understandability, we have stated that the teacher is enabled to make changes but, is the teacher the only allowed role? What if there is no teacher role? There is no unique solution for the above questions. GRAIL offers a permissions system where write access is granted for roles and sub-roles of type teacher by default, while roles and sub-roles of type student have read permissions. The modification of these settings is supported, so that the course administrator controls who can and who cannot modify the content.

II. Changes in the Learning Flow

A compliant player that supports all IMS LD features will allow enacting ready-to-run scripted courses where teachers are enabled to track students' progress, but they are limited to the tracking features that were explicitly included during the authoring phase. In IMS LD, tracking facilities are based on the IMS LD *properties*: teachers can track students as long as they can view what their property values are. The authoring of tracking facilities in a course requires a deep understanding of the specification.

In GRAIL, the administrative functionality has been incremented with an interface referred to as the *cockpit*. This interface implements tracking facilities and extends this monitoring service with edition capabilities. The cockpit is not included during authoring and every course running in the platform has its corresponding cockpit. The features of the cockpit are described as follows:

- Management of property values: One of the basic features is the management of IMS LD properties. Besides the functionality required by the specification, the cockpit adds the capability for course administrators to view and modify any property at any moment in the course. This feature allows, for instance, forcing the accomplishment of a particular condition that involves the evaluation of a property.

- User tracking: GRAIL provides the ability to track students progress, times of access to the activities and its resources, and the different properties within the personal context of the learner.
- Learning flow modifications: The cockpit can also be used by learning supporters to modify the learning flow. Two types of modifications are allowed: manage the learning objects associated to an environment, and modify the activities associated to an activity structure. These modifications can be performed during the enactment of an UoL.
- Conditions establishment: As termination condition for an activity, IMS LD allows to specify a period of time after which the activity is supposed to be marked as finished. It is not realistic to expect this condition to be always applicable. There could be, for example, a holiday period that forces the activity to take longer than expected. In this case, the course constraints require the run time engine to provide certain degree of flexibility. GRAIL allows modifying the condition of finalization for any activity so the course can be adapted to the actual scenario.

As a consequence of the ability of introducing changes both in the course content and in the course structure, it appears the need to provide a feature that preserves the modifications made in the UoL. This is addressed by the *export* functionality, which produces a compliant IMS LD packages that incorporates introduced changes. This feature enables the re-use of the new UoL with its corrections, improvements and extensions. The result is an enrichment of the course life-cycle, depicted in Figure 1.

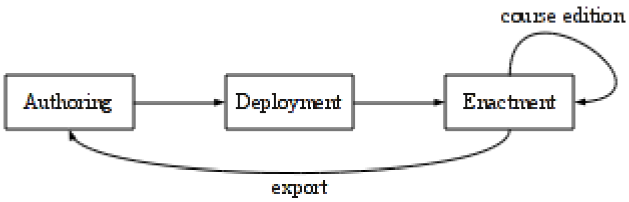

FIGURE 1:
LIFE CYCLE OF SCRIPTED COURSES

III. Increasing Flexibility with Third Party Tools

GRAIL supports the Generic Service Integration framework (GSI) [17]. GSI extends IMS LD, enabling the integration of third party tools in scripted courses. The supported integration satisfies the main characteristics of IMS LD: self-containment, reusability, interoperability, pedagogical neutrality, collaborative capabilities and adaptability of course material. The integration of third party tools affects the complete course life-cycle as follows:

- Authoring: When the course author describes the course flow he/she also describes how third party tools participate in the course. That is, the course description includes what tool is needed, how it will be used and when it will be accessed.
- Deployment: The description given at the course authoring is used to select the tool that best matches the functional requisites of the course flow. Different instances of the same course could make use of different third party tools.
- Enactment: The course participants interact with the course content. The bidirectional exchange of information between IMS LD and the third party tool enables the course to be adapted to the activities performed in such tools.

The third party tools are typically web-based, case-specific tools whose functionality is focused on the development of a certain task. The particular circumstances of the different tools may limit the flexibility of the activities they support. However, such tools do not suffer the limitations imposed by the course life-cycle, so their use is more flexible than the plain use of IMS LD features. In conclusion, the integration provided by GSI introduces some degree of flexibility in scripted courses.

Table I summarizes the presented features and how they provide flexibility to the course. Next Section discusses to what extent flexibility is provided by these features. The discussion is supported by examples of practical situations where the GRAIL's features were used to tackle the unexpected situations in courses that followed the scripted-flow approach.

TABLE I
SUMMARY OF THE FEATURES TOWARDS FLEXIBILITY

Wiki	Content modifications	Solve errata and update course material
Cockpit	Flow track and modification	Adjust the flow to the actual enactment circumstances
GSI	Use of case-specific tools	Activities not limited by the course life-cycle

PRACTICAL USE AND DISCUSSION

The previous sections describe GRAIL as an IMS LD player embedded in .LRN that provides some features that extend the life-cycle of scripted courses. GRAIL has been used to support the enactment of several courses, each of which was deployed with a different research goal. All of these courses have in common that they required the management of unexpected events that arose at their enactment. This section presents some of these situations as practical examples of the need of flexibility, and discusses to what extent the offered features allowed to tackle these situations.

The authoring of IMS LD courses is error-prone and, as a result, errata usually show up during the enactment of the activities [18]. Despite the advances in the authoring software, the problem is still present. One example of such difficulty was the deployment of a large-scaled experience that involved more than 400 students [4]. There, the file-storage was used to fix the mistakes introduced in the material and showed the simplicity of the method. However, such simplicity may be a double-edged sword. On the one hand, substituting the file with the new version is the only required action. On the other hand, the creation of the new version of the content requires the use of an external authoring tool and poses the teacher to the need of knowing the file that corresponds to a certain activity. In practice, file substitution requires to know the complete structure of the content, which is known by the course author but not necessarily by the teacher. As a consequence, the method is appropriate when the author and the teacher are the same person, but it is not so useful in other cases.

The wiki provides a more practical approach for quick content edition. Furthermore, an experience held in a workshop with K-12 educators [17] revealed that, despite it was not the aim of such development, the introduction of wiki behavior on the UoL content opens new learning scenarios that were not possible without this functionality. For example, a teacher could intentionally include errata in course material, so the students are expected to find and fix them. There, the teacher has the opportunity to revise the wiki history, so he/she can see who made the change.

The content edition feature allows the runtime environment to behave like a collaborative authoring tool. In such scenario, the case of use could be described with the following steps: first, a UoL with no content is created with the help of an external authoring tool. The created UoL should contain the skeleton of the final course and should link to empty files used as resources. Next, the skeleton UoL is uploaded in GRAIL. The course is then instantiated and populated by the forthcoming authors, who access the activities and create the material. Finally, the course can be either played in GRAIL or exported as a different UoL.

In [19], a scripted course was used to increase the flexibility of a blended experience. GRAIL was used to enact a course with high administrative requirements. The most relevant part of the course flow required the students to work in groups, promoting a positive interdependency among groups. The sequence of activities had two major drawbacks regarding flexibility:

- There was an inter-group collaboration that was only meaningful if there were five working groups.
- The students were dynamically assigned to a group depending on the preferences they showed in previous activities.

Due to its particular characteristics, the course flow was vulnerable to unexpected events when they were related to the group formation process. Table II presents some of the situations that arose during the experience. It can be seen that the most common problem regarded the collaborative nature of the activity flow. The absence of team-mates or even the absence of entire groups were handled by the *cockpit*, which was used by teachers to act on behalf of missing students (by simulating their presence) so that the groups could complete the activities.

In the same experience, the group formation was supported by a web based spreadsheet, integrated in the course flow via GSI. The spreadsheet incorporated the required formulae to form the groups. However, the simple delay of students in the development of the activities caused problems to the group formation process. In those cases, the teachers manipulated the spreadsheet and manually assigned the delayed students to their groups and the entire course flow was not affected. The use of the spreadsheet was positively considered by the instructors because they were already familiar to with the user interface and they knew how to react to the different situations. That is, the use of a task-specialized third party tool increased the instructors' perception of the course flow flexibility.

TABLE II
UNEXPECTED EVENTS AND APPLIED SOLUTIONS

issue	solution	tool
two students finished an activity after the deadline	their data was ignored and they were manually assigned to a group	spreadsheet
one student dropped the course	the user was ignored in the forthcoming activities	no action needed
a course replica was enacted with only 3 students	the group formation was forced to create a single group with all the participants	spreadsheet
in a course replica, there were only one working group	the instructors simulated the participation of the missing groups	cockpit

The tracking and edition facilities offered by the cockpit were useful in those cases where the course flow incorporated adaptive characteristics. In particular, the experience in [4] was provided with a mechanism that penalized the students who did not reach the expected homework deadline. Using the tracking interface, the instructors decided that there were some cases in which the applied penalization was not fair. Then, the teachers modified the corresponding property and the penalty was reverted. Such example highlighted the relevance of the flexibility when the course contains adaptive learning material.

CONCLUSIONS

Learning scripts are a relevant method to structure the resources and learning activities associated with a course, and provides them with a pedagogical sense. The life-cycle of scripted courses (authoring, deployment and enactment) reduces their flexibility. That is, scripted courses have difficulties on the management of unexpected events that may occur during the course enactment.

This article presented the functionalities developed in GRAIL, a player that complies with the IMS LD specification - one of the most important educational modeling languages - towards the provision of flexibility during the course enactment. The implemented functionalities were grouped in three categories: course content, course flow, and integration of third-party tools.

Taking examples form diverse practical experiences supported by GRAIL, this paper showed how the presented functionality allowed dealing with unexpected events that otherwise would have prevented the correct execution of the learning activities.

ACKNOWLEDGMENT

Work partially funded by the project “eMadrid: Investigación y desarrollo de tecnologías para el elearning en la Comunidad de Madrid” (S2009/TIC-1650) and the Spanish project “Learn3: Towards Learning of the Third Kind” (TIN2008-05163/TSI).

REFERENCES

[1] McLoughlin, C. and Lee, M., J., “Social software and participatory learning: pedagogical choices with technology affordances in the Web 2.0 era,” *24th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education*, Centre for Educational Development, Nanyang Technological University, 2007, pp. 664-675.

[2] Ertl, B., Kopp, B. and Mandl, H., “Supporting learning using external representations,” Computers & Education, vol. 51, Dec. 2008, pp. 1599-1608

[3] Dillenbourg, P., “Over-scripting CSCL: The risks of blending collaborative learning with instructional design.,” *Three worlds of CSCL. Can we support CSCL*, 2002, pp. 61-91.

[4] de-la-Fuente-Valentín, L., Villena Román, J., Pardo, A. and Delgado Kloos, C., “A cost-effective, scalable orchestration to promote continuous work in programming courses” *Unpublished.*

[5] “Making Sense of Learning Specifications & Standards: A decision maker’s guide to their adoption” Saratoga Springs NY: e-Learning CONSORTIUM of The MASIE Center , March (2002) .

[6] Bohl, O., Scheuhase, J., Sengler, R. and Winand, U., “The sharable content object reference model (SCORM) - a critical review,” *Computers in Education, 2002. Proceedings. International Conference on*, 2002, pp. 950-951 vol.2.

[7] *IMS Learning Design Specification*, 2003. http://www.imsglobal.org/learningdesign/

[8] Escobedo del Cid, J., P., de la Fuente Valentín, L., Gutíerrez, S., Pardo, A. and Delgado Kloos, C., “Implementation of a Learning Design Run-Time Environment for the .LRN Learning Management System,” *Journal of Interactive Media in Education (Adaptation and IMS Learning Design. Special Issue)*, Sep. 2007.

[9] Schank, R., C. and Abelson, R., P., *Scripts, Plans, Goals, and Understanding: An Inquiry Into Human Knowledge Structures (Artificial Intelligence)*, 1st ed. Lawrence Erlbaum, July 1977.

[10] Anglin, G., J., *Instructional Technology: Past, Present, and Future. Second Edition.*, Libraries Unlimited, Inc., P.O. Box 6633, Englewood, CO, 80155-6633, 1995

[11] Burgos, D., Tattersall, C. and Koper, R, "How to represent adaptation in e-learning with IMS Learning Design," *Interactive Learning Environments*, vol. 15, no. 2, pp. 161-170, August 2007.

[12] Jurado, F., Redondo, M. and Ortega, M., “Specifying Collaborative Tasks of a CSCL Environment with IMS-LD,” *Cooperative Design, Visualization, and Engineering*, 2006, pp. 311-317.

[13] Arbaugh, J., B., “Managing the on-line classroom: A study of technological and behavioral characteristics of web-based MBA courses,” The *Journal of High Technology Management Research*, vol. 13, Autumn. 2002, pp. 203-223.

[14] Dimitrova, M., Sadler, C., Hatzipanagos, S. and Murphy, A., “Addressing learner diversity by promoting flexibility in e-learning environments,” *Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on*, 2003, pp. 287-291

[15] Dillenbourg, P. and Tchounikine, P., “Flexibility in macro-scripts for computer-supported collaborative learning,” *Journal of Computer Assisted Learning*, vol. 23, Feb. 2007, pp. 1-13

[16] Lejeune, A., Ney, M., Weinberger, A., Pedaste, M., Bollen, L., Hovardas, T., Hoppe, U. and de Jong, T., “Learning Activity Spaces: Towards Flexibility in Learning Design?,” *Advanced Learning Technologies, 2009. ICALT 2009. Ninth IEEE International Conference on*, 2009, pp. 433-437

[17] de-la-Fuente-Valentín, L., Pardo, A., and Delgado Kloos, C., “Generic service integration in adaptive learning experiences using IMS learning design.” *Computers & Education,* 2011, 57:1160-1170.

[18] de-la-Fuente-Valentín, L., Pardo, A., Asensio Pérez, J., I., Dimitriadis, Y. and Delgado Kloos, C., “Collaborative Learning Models on Distance Scenarios with Learning Design: a Case Study,” *ICALT '08: Proceedings of the eighth IEEE International Conference on Advanced Learning Technologies*, Santander, Spain: 2008, pp. 278-282.

[19] de-la-Fuente-Valentín, L., Pérez-Sanagustín, M., Santos, P., Hernández-Leo, D., Pardo, A., Delgado Kloos, C., Blat, J., “System orchestration support for a flow of blended collaborative activities,” *2nd International Workshop on Adaptive Systems for Collaborative Learning*. Thessaloniki, Greece.

AUTHOR INFORMATION

Luis de-la-Fuente-Valentín Teaching Assistant, University Carlos III of Madrid, Department of Telematics Engineering, lfuente@it.uc3m.es

Derick Leony, PhD Student, University Carlos III of Madrid, Department of Telematics Engineering, dleony@it.uc3m.es

Abelardo Pardo, Associate Professor, University Carlos III of Madrid, Department of Telematics Engineering, abel@it.uc3m.es

Carlos Delgado Kloos, Full Professor, University Carlos III of Madrid, Department of Telematics Engineering, cdk@it.uc3m.es