Applying an Extensible Learning Support System to Collaborative Learning Environments

Kiyoshi NAKABAYASHI[a*] Yosuke MORIMOTO[b] and Kumiko AOKI[b]
[a]*Faculty of Information and Computer Science, Chiba Institute of Technology, JAPAN*
[b]*Center of ICT and Distance Education, The Open University of Japan, JAPAN*
* knaka@net.it-chiba.ac.jp

Abstract: This paper describes an investigation of the application of an extensible learner-adaptive system, Extensible Learning Environment with Courseware Object Architecture (ELECOA), to a collaborative learning environment. The design goal of ELECOA is to provide a flexible learning environment that ensures both function extensibility and content reusability. The concept of a "courseware object," which is a program module that is used to incrementally implement various educational functionalities, has been introduced to achieve this goal. Based on this concept, a self-learning environment has been implemented that is fully compliant with the SCORM 2004 specification. This report provides the investigation results of implementation on a collaborative learning environment based on the Learning Design specification within the ELECOA framework.

Keywords: Learner Adaptation, Courseware Object, Learning Design, Extensible Learning Support System

Introduction

In order to provide high-quality e-learning services that offer rich educational experiences, the interoperability and reusability of learning content is vital. There have been various efforts made to develop and disseminate e-learning content specifications [4]. Many learner-adaptive systems, capable of presenting learning content and resources that match the learner's comprehension level, have also been considered as an effective means to provide an improved learning experience [2, 7, 9]. However, there has been little consideration of the interoperability and reusability of content in the field of learner-adaptive systems. Generally speaking, learner-adaptive systems have been designed on the basis of a certain single learner-adaptive strategy without any extensibility to support multiple learner-adaptive strategies or even to modify a single implemented strategy. Due to this lack of flexibility, it is difficult and sometimes impossible to add new functions that could improve the effectiveness of learning because the newly added functions may interfere with the current content, thus impairing its reliable behavior.

In response to this problem, we developed a learning-system architecture called Extensible Learning Environment with Courseware Object Architecture (ELECOA) that can both extend learner-adaptive functions and make the learning content interoperable [10]. This architecture was designed around the concept of a "courseware object," which is a program module that implements various educational functionalities and is usually embedded as an inseparable fragment of program code in the learning platform. It is possible to incrementally extend functions by adding new courseware objects. We have previously shown that several learner-adaptive functionalities for self-learning, including the SCORM 2004 standard specification [1] and its extensions, can be successfully implemented on ELECOA [10].

Educational Modeling Language (EML) has attracted the attention of developers of learning environments. EML was designed to formally describe formations and sequences of various types of educational activities, including not only self-learning materials but also lectures and collaborations in which groups of learners and instructors are involved. The intent was to share and reuse pedagogical strategies to achieve effective learning. In particular, the IMS Learning Design (LD) specification [6, 9], which was derived from EML and developed by the Open University in the Netherlands, has been widely used in several research projects. These projects include the development of LD authoring tools, LD execution systems [3, 8], and a system to generate pedagogy described in LD from higher level design requirements that take into account instructional design theories [5].

In this paper, we discuss our investigation of how to implement an LD execution system based on ELECOA, which has primarily been used only in the self-learning environment. The results of our investigation indicate that ELECOA is capable of seamlessly integrating self-learning learner-adaptive environments (such as that of SCORM 2004) with an LD-based collaborative learning environment. We also found a few issues related to this implementation that will need to be addressed.

1. Extensible Learner-Adaptive System Architecture

1.1 Issues with Conventional Learner-Adaptive Systems

Conventional learner-adaptive systems typically have the system architecture shown in Fig. 1, in which the content is separated from the platform [10]. In this type of architecture, the content consists of learning materials specific to a particular learning subject, and the platform devises common learner-adaptive functionalities independent of the specific learning subject. By separating the content from the platform, this configuration makes it possible to design learner-adaptive content with less effort because the designer can concentrate on creating content to fulfill the specific learning goals and not worry about the specifics of implementing learner-adaptive functionalities.

The drawback to this architecture is its lack of function extensibility. After implementation, extending the platform to add new functionalities is difficult because it is not possible to ensure that the existing learning content designed for the original platform will work correctly on the extended platform. A representative standard with learner-adaptive capabilities, SCORM 2004, uses the same configuration, resulting in the same lack of function extensibility.

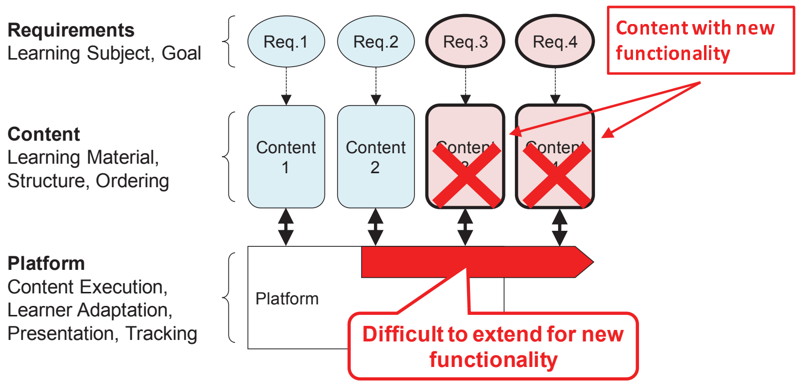
Requirements
Learning Subject, Goal
Content
Learning Material, Structure, Ordering
Platform
Content Execution, Learner Adaptation, Presentation, Tracking
Req.1
Req.2
Req.3
Req.4
Content 1
Content 2
Content 3
Content 4
Content with new functionality
Platform
Difficult to extend for new functionality

Fig. 1. Architecture of conventional learner-adaptive systems.

1.2 Approach of Proposed Learner-adaptive Architecture

To overcome the drawbacks of conventional learner-adaptive systems, we recently proposed a learner-adaptive system architecture called ELECOA with the aim of achieving both function extensibility and system interoperability [10]. Our key to accomplishing this goal is the concept of a "courseware object" to modularize the learner-adaptive system architecture. The courseware object is a program module that implements various educational functionalities that are embedded in the platform of the conventional architecture. The courseware objects implement functions, including learner adaptation, to select the most suitable learning material for the learner, material presentation to tailor the way the learning material is presented, and learner tracking to record the status of the learner's progress. For example, there might be multiple courseware objects, each of which implements simple linear, conditional branching, complicated remedial, or much more sophisticated strategies such as scenario-based sequencing using a state-transition machine.

Fig. 2 shows the proposed ELECOA architecture in which the courseware object is clearly separated from the platform. With this configuration, incremental extension of functions is possible by adding new courseware objects. Since this extension does not affect the previously implemented functions, the existing content will continue to work correctly. In addition, courseware objects can be distributed along with content, thus enabling existing platforms to be immediately updated for newly developed functionalities.

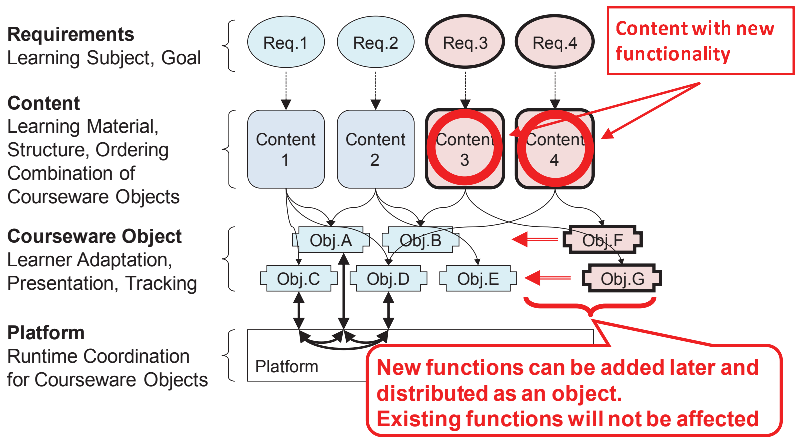
Requirements
Learning Subject, Goal
Content
Learning Material, Structure, Ordering Combination of Courseware Objects
Courseware Object
Learner Adaptation, Presentation, Tracking
Platform
Runtime Coordination for Courseware Objects
Req.1
Req.2
Req.3
Req.4
Content with new functionality
Content 1
Content 2
Content 3
Content 4
Obj.A
Obj.B
Obj.C
Obj.D
Obj.E
Obj.F
Obj.G
Platform
New functions can be added later and distributed as an object.
Existing functions will not be affected

Fig. 2. Proposed ELECOA architecture.

1.3 Application of ELECOA to Self-learning Environment

For the ELECOA architecture to be of any value, it must be possible to assemble any courseware objects developed by various designers at various times and have them work together. To make this feasible, it is necessary to design some criteria or standards to which every courseware object designer must conform. These criteria may include the communication scheme between courseware objects, the information maintained by courseware objects, and the responsibility of courseware objects.

To investigate these issues, the system was designed in accordance with the following principles and assumptions. First, it was assumed that the content is structured hierarchically, or like a tree. This is because content with a hierarchical structure is widely adopted in learning materials by various standards, including AICC CMI, ADL SCORM, and IMS CC [4], as well as various proprietary LMSs. Second, it was assumed that the courseware objects are assigned for each hierarchical node of content, as outlined in Fig. 3.

It is the responsibility of the courseware object assigned to a content node to manage the learner-adaptation behavior of the sub-tree under its node. The courseware object sequences its child nodes by taking into account their learner progress information according to the pedagogical strategy implemented in it. This makes it possible to implement different pedagogical strategies in different sub-trees. It is also assumed that the communication between courseware objects is only limited between parents and children. On the basis of this assumption, definitions are designed for the required communication patterns between courseware objects and the interface that courseware objects should provide for other courseware objects.

The SCORM 2004 specification, which is a standard for learner-adaptive content, has been implemented based on these principles and assumptions [10]. The implementation was demonstrated to be conformant to SCORM 2004 3rd edition by checking against the test suite of the specification consisting of 100 test cases.

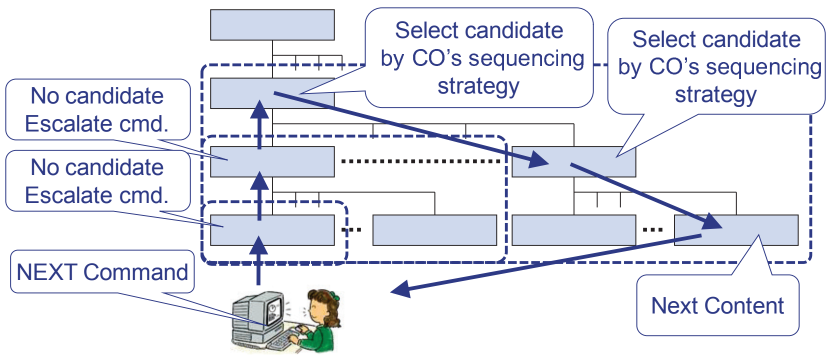
Select candidate by CO's sequencing strategy
Select candidate by CO's sequencing strategy
No candidate Escalate cmd.
No candidate Escalate cmd.
NEXT Command
Next Content

Fig. 3. Communication between courseware objects in hierarchical configuration.

2. Application of ELECOA to Learning Design Specification

2.1 Learning Design Specification

The LD specification is designed to promote the sharing and reuse of pedagogical strategy to achieve effective learning results by formally describing formations and sequences of educational activities. It is a pedagogy-neutral technical specification capable of describing various types of educational activities, including self-learning materials, lectures, and group learning. However, the LD specification's most notable feature is its capability to describe collaborative learning activities, which follows the recent trend of e-learning toward a learner-centered approach.

The LD specification defines a data model to describe learning activities. In the LD specification, the primary elements to describe learning activities are "activity," "role," and "environments." An activity uses several environments, including "learning objects" and "services." It also involves people with several roles, such as "learner" or "staff." The activity has an "activity structure," which is a hierarchical one so that the aggregation of activities becomes an upper-level activity. The above-mentioned description of learning activity can be represented using level A of the LD specification. With level B, the properties of a person or group and conditions for the sequence of activities can be described to control the learning sequence. In addition, events resulting from certain activities, such as the notification of a question from a learner, can be described with level C.

LAMS [3], which is the most commonly disseminated open source learning tool compliant to the LD specification, has two types of communities: one for a system developer to update the system itself and the other for instructional designers to share and reuse descriptions of designed educational activities. LAMS deals with sequences of learning activities, e.g., each learning activity can be assigned to one of a variety of environments, such as document, survey, chat, or forum. It can also deal with a hierarchical activity that has an internal structure with conditional branches. Synchronization of multiple learners can be implemented by a waiting point where all the learners have to stop until every other learner finishes the previous activities. For example, before entering a synchronous forum, all the participants must finish a pre-assigned series of activities.

2.2 Basic Framework to Implement LD Specification with ELECOA

The basic implementation framework of the LD specification using ELECOA has previously been investigated [11]. ELECOA was originally designed for self-learning; it was not intended to support group learners. However, both ELECOA and the LD specification deal with hierarchical structures. In addition, ELECOA has the capability to control learning activity sequences by means of courseware objects.

The investigation took into account these characteristics. With the LD specification, learners follow a predefined learning path in which they communicate with other learners and instructors using communication tools such as chat or forum. The learning path varies according to the learner's own learning status as well as those of other learners. Thus, the following issues should be considered for implementing the LD specification using ELECOA:

(1) implementation of a learning path for each individual learner,
(2) integration with communication tools, and
(3) control of the learning path based on the status of multiple learners.

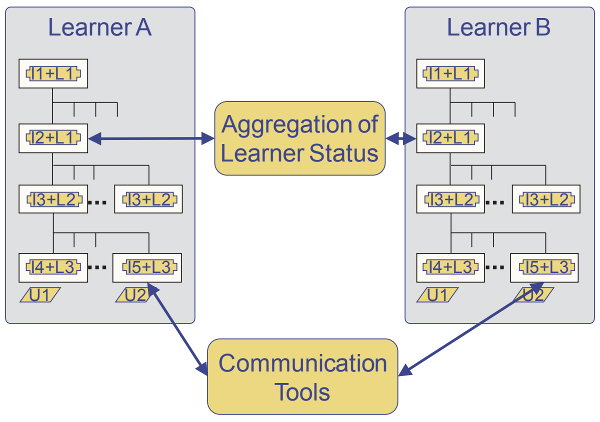

Fig. 4. Implementation of LD specification with ELECOA.

The implementation is outlined in Fig. 4. First, the learning path of each learner is controlled by the courseware objects in a similar manner to the original ELECOA behavior for self-learning in a hierarchical structure. The courseware object selects the next node to be presented to the learner according to the learner's status. This makes it possible to implement learning path control that takes into account each individual's status. Second, communication tools are integrated as learning resources to be associated with the leaf node of the hierarchical content. In the LD specification, communication tools and learning services are environments that also include learning resources such as static HTML documents or quizzes associated with the leaf node of hierarchical learning activity. Thus, in the ELECOA-based implementation, they are associated with the leaf nodes in the same way that the original ELECOA has learning resources assigned to leaf nodes. Finally, to reflect multiple learners' statuses in each individual learning path, a courseware object is equipped with the capability to exchange information with other courseware objects controlling the learning path of other learners. In this way, the courseware object can determine the learning resources to be presented by taking account of multiple learners' statuses. This means that each learner's learning sequences can be controlled on the basis of their own learning status as well as those of other learners.

It is important to note that the basic framework of ELECOA is not modified to implement the LD specification. The framework defines the process of information exchange between courseware objects assigned to a hierarchical structure to determine the next learning resources presented to the learner. Since this framework is independent of the learning resources to be presented, it does not need to be modified if communication tools or learning services are assigned as learning resources. In addition, this framework simply defines the communication schema between courseware objects in the hierarchical structure, which is independent of the internal behavior of each courseware object, to control the learning path. Thus, the framework does not need to be modified if the courseware object, as its "internal behavior," exchanges information with other learners' courseware objects to control the learning path. It is therefore possible to implement the LD specification using ELECOA without modifying its basic framework by simply adding learning resources and courseware objects for collaborative learning.

2.3 Issues to be Considered regarding Implementation of LD Specification with ELECOA

The following issues should be considered with regard to the implementation of a group learning environment defined by the LD specification within the ELECOA framework:

(1) assignment of activities and environments according to roles,
(2) dynamic generation and assignment of activities and environments, and
(3) function to aggregate and control this information.

These issues are discussed in the following sections.

2.3.1 Assignment of Activities and Environments according to Roles

The biggest difference between a collaborative learning environment and a self-learning environment is the necessity of multiple-user control in the former environment. In the collaborative learning environment, every user can be a learner, or some users can be another role type such as an instructor. The LD specification defines learner and staff as two major roles. Designers can also define new roles that are derived from the major roles. Roles can be defined in a hierarchical manner, and this functionality makes it possible to assign users to hierarchically divided groups. The designer may assign activities and environments to roles. It is thus possible to create assignments in which each learner group uses different discussion rooms and instructors can join any discussion in any room. From the system implementation point of view, these assignments can be created either before runtime, when the activity structure is generated, or during runtime, while the learning sequence is executed.

2.3.2 Dynamic Generation and Assignment of Activities and Environments

Flexible formation of learner groups is necessary in any collaborative learning activity. For example, the jigsaw method requires separate groups to learn certain topics and then give presentations on these topics, as shown in Fig 5. In this method, first, different learning topics related to one learning subject are assigned to each group. These groups discuss the assigned topic, and then groups are formed for presentations in such a way that each group gives a presentation to learners from different groups on their particular learning topic. In this way, learners with knowledge about different topics collaborate in their respective groups to discuss their topic and prepare their presentation. There is the option of assigning learners to multiple groups during collaborative learning activities, and there are also various options about the number of groups and the criteria for assigning individuals to groups (such as fixed, random, learners' preference, learners' performance, etc.). These assignments are made either statically in advance or dynamically during the learning activity itself.

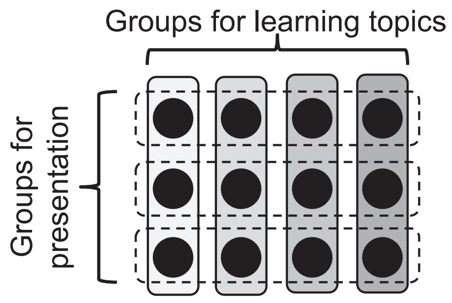

Fig. 5. Group Formation for Jigsaw Method.

Another example is the Versailles role play, shown as an use case in the Learning Design specification [6]. In this case, *n* groups of learners corresponding to *n* countries perform bilateral negotiation. All combinations of countries require C(*n*, 2) negotiation meetings. For example, as *n* is set to 6 in this use case, it is required to prepare C(*n*, 2) = 15 online meeting rooms statically defined in the LD manifest file.

As seen in this example, the original LD specification requires environments to be statically defined before execution, which can possibly lead to a lack of flexibility and extensibility. LAMS, on the other hand, provides a group assignment strategy based on the number of groups and the number of learners per group. To carry out this assignment strategy, it is necessary to implement a function to dynamically generate the proper number of groups with required environments according to the number of learners and to assign learners to these groups.

2.3.3 Function to Aggregate and Control Information for Collaborative Learning

In order to deal with the issues discussed in the previous two sections, it is necessary to implement functionality for learner status aggregation and learning sequence management (as shown in Fig. 4). Various physical implementation schemata for this functionality can be considered, including a schema with a central server aggregating all information, a schema with functions and information completely distributed to the courseware objects of each learner, and an intermediate schema in which both a central server and courseware objects are responsible for managing learner status and learning sequence. In fact, the original LD specification has been implemented in several ways, including on a single state transition machine that takes care of the management of all learners and on a P2P network in which the distributed engine assigned to each learner exchanges information for the learning sequence [8].

Because of these differences, if used, the implementation schemata would impact the scalability of the system and the effectiveness of implementing management functionality. In addition, since the implementation schemata would affect the extensibility and reusability of courseware objects for collaborative learning in ELECOA, careful consideration is required as to the responsibility and interface between individual learners' modules and common modules taking care of learner status and learning sequence management.

3. Conclusion

In this paper, we discussed our investigation of the implementation of the LD specification using ELECOA, an extensible learner-adaptive environment enabling both functional extensibility and content interoperability. Although the original intention of ELECOA was to support self-learning, its extensibility may make it possible to implement the LD specification to include group learning. With this capability to implement self-learning and group learning in the same framework, it would be possible to provide an integrated learning environment in which materials and learner history could be seamlessly exchanged between self-learning and group learning. Further study is needed to clarify a few issues that remain in applying ELECOA to the implementation of collaborative learning environments that include the LD specification.

Acknowledgments

This work was supported by a Grant-in-Aid for Scientific Research by Kakenhi (23300307, 21300317 and 22240080).

References

[1] Advanced Distributed Learning (2006). *Shareable Content Object Reference Model SCORM® 2004 3rd Edition.*

[2] Brusilovsky, B. (2003). Developing Adaptive Educational Hypermedia System. In Murray, T., Blessing, S., and Ainsworth, S. (Eds.). (2003). *Authoring Tools for Advanced Technology Learning Environments* (pp. 337–409), Dordrecht, Kluwer Academic Publishers.

[3] Dalziel, J. (2008). Learning Design: Sharing Pedagogical Know-How, in Iiyoshi, T. and Kumar, M. S. V. (Eds.): Opening Up Education, MIT Press.

[4] Fallon, C. and Brown, S. (2003). *e-Learning Standards.* Boca Raton, St. Lucie Press.

[5] Hayashi, Y., Bourdeau, J., and Mizoguchi, R. (2007). Theory-aware Explanation Support for Standard-compliant Scenario Building. In *Proc. ICCE2007*, (pp.104-109), AACE.

[6] IMS Global Learning Consortium (2003). *IMS Learning Design Version 1.0 Final Specification*..

[7] Kazi, S. (2004). A Conceptual Framework for Web-based Intelligent Learning Environments using SCORM 2004. In Proc. *IEEE ICALT 2004* (pp. 12–15), IEEE Computer Society.

[8] Koper, R. and Tattersall, C. (Eds.). (2005). *Learning Design: A Handbook on Modelling and Delivering Networked Education and Training*, Springer.

[9] Murray, T., Blessing, S., and Ainsworth, S. (Eds.). (2003). *Authoring Tools for Advanced Technology Learning Environments*. Dordrecht, Kluwer Academic Publishers.

[10] Nakabayashi, K., Morimoto, Y., Hata, Y. (2010) Design and Implementation of an Extensible Learner-Adaptive Environment, *Knowledge Management & E-Learning: An International Journal (KM&EL)*, 2(3), 246-259.

[11] Nakabayashi, K., Morimoto, Y., Hata, Y., Aoki, K., (2010) Implementing Learning Design Specification using Extensible Learner-adaptive Environment, Workshop on Open Technology, Open Standards and Open Knowledge in Advanced Learning, In *Workshop Proceedings of ICCE2010*, 300-307.