Hirashima,T. et al. (Eds.) (2010). Workshop Proceedings of the 18th International Conference on Computers in Education. Putrajaya, Malaysia: Asia-Pacific Society for Computers in Education.

Implementing Learning Design Specification using Extensible Learner-adaptive Environment

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

Abstract: Application of an extensible learner-adaptive environment to implement the Learning Design specification is proposed. The design goal of the extensible learner-adaptive architecture is to provide a flexible learning environment that ensures both function extensibility as well as content reusability. The concept of a "courseware object," which is a program module used to incrementally implement various educational functionalities, has been introduced to achieve this goal. On the basis of this concept, implementation of an execution environment for the Learning Design specification has been investigated. The results of this investigation indicate that a self-learning environment based on the learner-adaptive capability, such as SCORM 2004, could be seamlessly integrated with a collaborative learning environment based on the Learning Design specification.

Keywords: e-learning technology standardization, learner adaptation, function extensibility, platform architecture, courseware object, SCORM 2004, Learning Design

Introduction

The interoperability and reusability of learning content is inevitable for high-quality e-learning services with rich learning experiences. Various efforts have been made to tackle this issue by developing and disseminating e-learning content specifications [5, 12]. Some of them have been successfully accepted by the e-learning industry to achieve interoperability between e-learning content and learning-management systems [1, 8]. Many learner-adaptive systems have also been considered as an effective means to provide an improved learning experience by presenting learning content and resources that match the learner's comprehension status [3, 6, 10, 11, 14]. However, there has been little consideration about the interoperability and reusability of content in the field of learner-adaptive systems. In most cases, learner-adaptive systems have been designed based on a certain single learner-adaptive strategy without any extensibility to support multiple learner-adaptive strategies or even to modify a single implemented strategy. This lack of flexibility makes it difficult to add new functions that could improve the effectiveness of learning because the newly added functions may conflict with those for executing existing learning content, resulting in damage to the reliable behavior of this existing content.

To overcome this problem, new learning-system architecture has been designed aiming for capability to both extend learner-adaptive functions and make learning content interoperable [13]. To achieve this goal, the proposed architecture introduces the concept of a "courseware object," which is a program module to implement various educational functionalities. It is possible to incrementally extend functions by adding new courseware objects. Several learner-adaptive functionalities for self-learning, including the SCORM 2004 standard specifications [1] and their extensions, could be successfully implemented on this architecture.

Educational Modeling Language (EML) has recently been attracting attention. EML was designed with the intention to share and reuse pedagogical strategies to achieve effective learning. For this goal, it has the capability to formally describe formations and sequences of various types of educational activities, including self-learning, lectures, and collaborations. In particular, the IMS Learning Design (LD) specification [9], which was derived from EML developed by the Open University of Netherland, has been widely used in several research projects. Such research projects include LD authoring tools, LD execution systems [4], and a system to generate pedagogy described in LD from higher level design requirements that take into account instructional design theories [7].

This paper discusses an investigation to implement an LD execution system based on Extensible Learning Environment with Courseware Object Architecture (ELECOA), which is an extensible learner-adaptive architecture developed by the authors. The results of this investigation indicate that self-learning learner-adaptive capability, such as that of SCORM 2004, could be seamlessly integrated with an LD-based collaborative learning environment based on ELECOA.

1. Extensible Learner-Adaptive System Architecture

1.1 Issues with Conventional Learner-Adaptive Systems

Conventional learner-adaptive systems commonly 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 material specific to a particular learning subject, and the platform devises common learner-adaptive functionalities independent of the specific learning subject. By separating 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 details of implementing learner-adaptive functionalities.

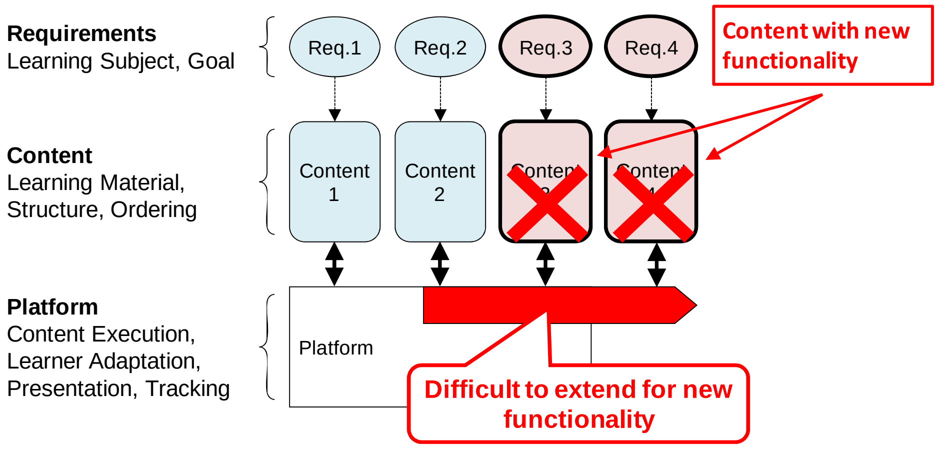
Requirements
Learning Subject, Goal
Req.1
Req.2
Req.3
Req.4
Content with new functionality
Content
Learning Material, Structure, Ordering
Content 1
Content 2
Content 3
Content 4
Platform
Content Execution, Learner Adaptation, Presentation, Tracking
Platform
Difficult to extend for new functionality

Fig. 1. Architecture of Conventional Learner-adaptive Systems.

The drawback to this architecture is the lack of function extensibility. After implementation, extending the platform to add new functionalities is difficult because it is not known if 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 a lack of function extensibility.

1.2 Approach of Proposed Learner-Adaptive Architecture

To tackle the problems of conventional learner-adaptive systems, new learner-adaptive system architecture has been proposed aiming to achieve both function extensibility and system interoperability [12]. The key idea for accomplishing this goal is the concept of a "courseware object" to modularize [2] the learner-adaptive system architecture. The courseware object is a program module for implementing various educational functionalities that are embedded in the platform of the conventional architecture. The courseware objects implement functions, including learner adaptation to choose 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 can be multiple courseware objects, each of which respectively implements simple linear, conditional branching, complicated remedial, or much more sophisticated strategies such as scenario-based sequencing using a state-transition machine.

Figure 2 shows the proposed 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 functions previously implemented with existing courseware objects, existing content is certain to always work correctly. In addition, courseware objects can be distributed with content, thus enabling existing platforms to be immediately updated for newly developed functionalities.

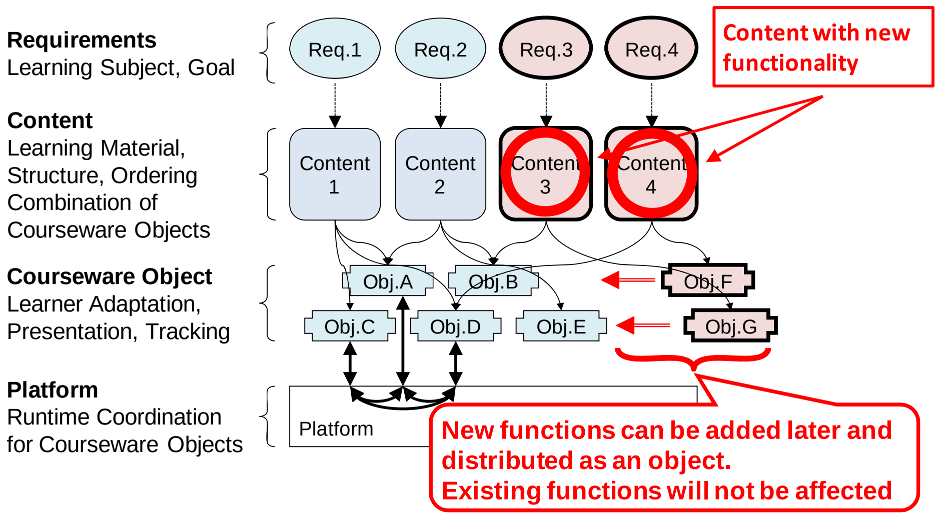
Requirements
Learning Subject, Goal
Content
Learning Material, Structure, Ordering Combination of Courseware Objects
Courseware Object
Learner Adaptation, Presentation, Tracking
Platform
Runtime Coordination for Courseware Objects
Req.1
Req.2
Req.3
Req.4
Content with new functionality
Content 1
Content 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. Configuration for Proposed Learner-Adaptive System.

1.3 Implementation of Proposed Learner-Adaptive Architecture

To achieve the goal of the proposed architecture, it must be possible to combine any courseware objects developed by various designers at various times to work together. To make this feasible, it is necessary to design some criteria or standards to which every courseware object designer conforms. 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 based on 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 [5] as well as various proprietary LMSs.

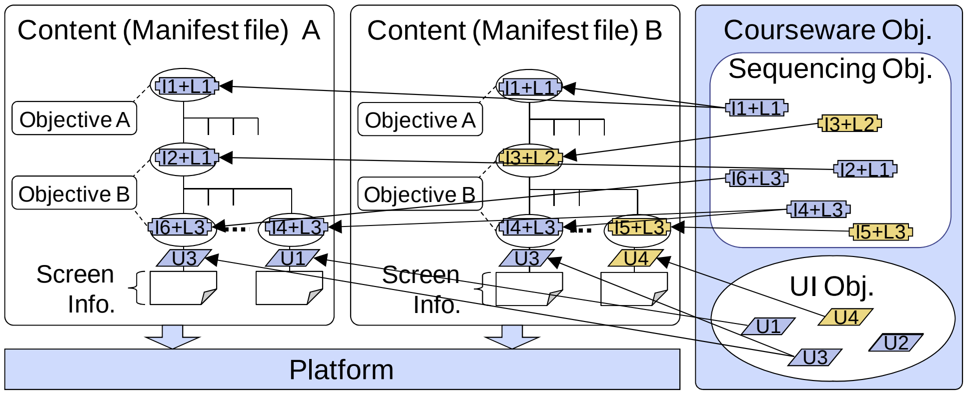

Fig. 3. Configuration of Proposed System for Hierarchical Content.

Second, it was assumed that the courseware objects are assigned for each hierarchical node of content as outlined in Fig. 3. It is a responsibility of the courseware object assigned to a content node to manage the learner-adaptation behavior of the sub-tree under the assigned 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.

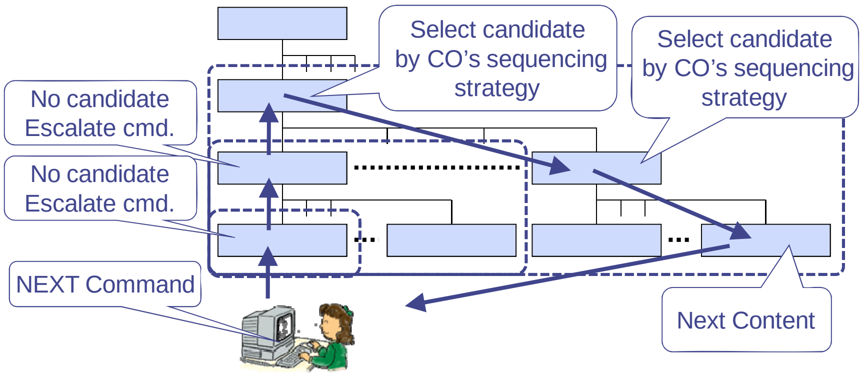
Select candidate by CO's sequencing strategy
Select candidate by CO's sequencing strategy
No candidate Escalate cmd.
No candidate Escalate cmd.
NEXT Command
Next Content

Fig. 4. Communication Schema for Command Execution.

Figure 4 illustrates the process to determine the next content or next page presented to the learner in the hierarchical structure. First, the current courseware object presented to the learner receives the command from the learner. It then escalates the command to its parent to select the candidate for the next page from its children. If the parent cannot find a suitable child, it escalates the command to the grandparent. The grandparent makes its children select a suitable node from their children. This recursive behavior is repeated until a suitable candidate for the next page is found. This results in a behavior that gradually expands the search space for the candidate in the content tree from the local (the smallest sub-tree containing the current object) to the global (the entire content tree). The identified node for the next page will be presented to the learner, and its associated courseware object will be the new current object. Note that the criteria or pedagogical strategy to select the next node from the children may differ between courseware objects. They might implement a completely different strategy despite they can be integrated in one hierarchical structure.

2. Learning Design Specification

The LD specification is designed to promote the share 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, lectures, and group study. However, the LD specification's notable feature is its capability to describe collaborative learning activities, which meets the recent trend of e-learning toward a learner-centered approach. LAMS [4], which is the most commonly disseminated learning tool based on the LD specification and is distributed as an open source software, 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.

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. In addition, events resulting from certain activities, such as notification of a question from a learner, can also be described.

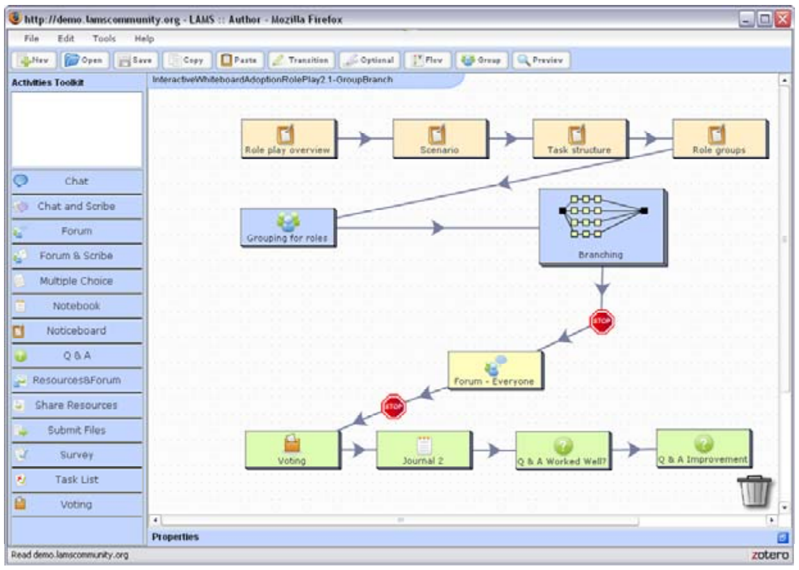

Fig. 5. Learning Activity Design by LAMS (© James Dalziel).

Figure 5 is an example of a learning activity description in LAMS. Each box represents an activity assigned with one of various environments, such as document, survey, chat, or forum. The large box to the right of the middle is a hierarchical activity that has an internal structure with conditional branches. Each "stop" symbol is 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 the activities before the forum.

3. Implementation of LD Specification using ELECOA

3.1 Basic Implementation

Implementation of the LD specification using ELECOA was investigated. 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 by using communication tools such as chat or forum. The learning path varies according to the learner's own learning status as well as other learners' learning statuses. 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 multiple learners' learning statuses.

The implementation is outlined in Fig. 6. First, the learning path of each learner will be 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 learner's learning status.

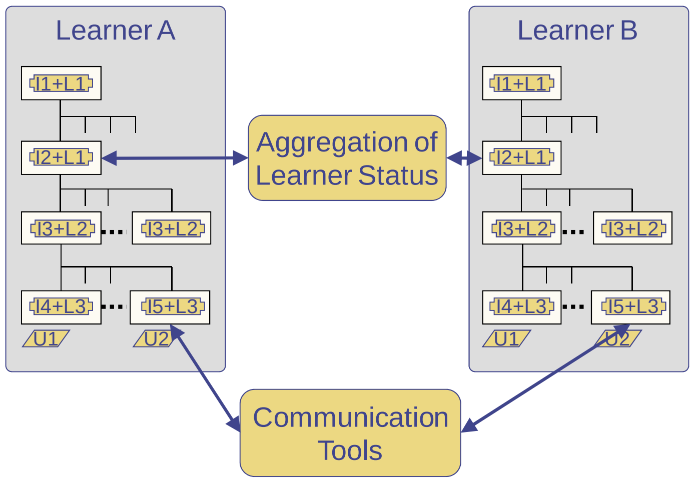

Fig. 6. Implementation of LD Specification.

Second, communication tools will be integrated as learning resources to be associated with the leaf node of 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.

Last, to reflect multiple learners' learning statuses in each individual learner's learning path, a courseware object will be 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' learning statuses. For example, Fig. 7 represents the situation in which the middle-level courseware object selects the next learning resource presented from candidate learning resources "1," "2," and "3." Note that learners A and B will be presented different learning resources depending on their own learning statuses. The learning resources may be replaced with the content sub-tree, which implements more complex learning control.

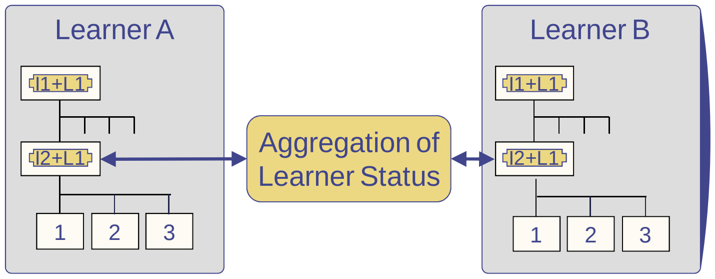

Fig. 7. Branch Structure Taking into Account Other Learners' Statuses.

It is important to point out 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. Therefore, it is possible to implement the LD specification using ELECOA without modifying its basic framework but by simply adding learning resources and courseware objects for collaborative learning.

3.2 Extended Implementation

There are several possible function extensions in the LD specification implementation using ELECOA. First, group activity consisting of some subgroups can be implemented by associating communication tools with the leaf nodes in the branch structure shown in Fig. 7. In this way, subgroups of learners meeting certain criteria, such as learning status or personal characteristics, will be allocated to the same communication tool for a group learning activity.

Second, it is not necessary for all the learners to learn in the same learning structure. According to each learner's past learning status or competency, a different learning structure with a different learning path, such as for the role of leader or normal participant, could be prepared. Then the learners in different learning paths can interact with each other in the proposed framework.

Last, there is no limitation to the learning resources associated with the leaf node. It is possible to utilize various tools including Web2.0 tools. Since there are several standardized interfaces between LMS and e-learning tools, a learning environment could be constructed on the proposed framework with minimum effort by exploiting these e-learning tools with the standardized interfaces.

4. Conclusion

We investigated 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 including 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. We will further investigate and design in detail the actual implementation of courseware objects to execute the LD specification.

Acknowledgments
This work was supported by a Grant-in-Aid for Scientific Research by Kakenhi (20500820, 21300317 and 22240080).

References

[1] Advanced Distributed Learning (2006). *Shareable Content Object Reference Model SCORM® 2004 3rd Edition.*
[2] Baldwin, C. Y. and Clark, K. B. (2000) Design Rules, Vol. 1: The Power of Modularity, The MIT Press.
[3] 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.
[4] Dalziel, J. (2008). Learning Design: Sharing Pedagogical Know-How, in Iiyoshi, T. and Kumar, M. S. V. (Eds.): Opening Up Education, MIT Press.
[5] Fallon, C. and Brown, S. (2003). *e-Learning Standards*. Boca Raton, St. Lucie Press.
[6] Fletcher, J. D. (1975). Modeling of Learner in Computer-based Instruction. *J. Computer-based Instruction,* Vol. 1, pp. 118–126.
[7] Hayashi, Y., Bourdeau, J., and Mizoguchi, R. (2007). Theory-aware Explanation Support for Standard-compliant Scenario Building. In *Proc. ICCE2007,* (pp.104-109), AACE.
[8] 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.
[9] Koper, R. and Tattersall, C. (Eds.). (2005). Learning Design: A Handbook on Modelling and Delivering Networked Education and Training, Springer.
[10] Murray, T., Blessing, S., and Ainsworth, S. (Eds.). (2003). *Authoring Tools for Advanced Technology Learning Environments.* Dordrecht, Kluwer Academic Publishers.
[11] Nakabayashi, K., Koike, Y., Maruyama, M., Touhei, H., Ishiuchi, S., and Fukuhara, Y. (1995). A Distributed Intelligent-CAI System on the World Wide Web. In *Proc. ICCE 1995* (pp. 214–221), AACE.
[12] Nakabayashi, K. (2004). e-Learning Technology Standardization – Make It Converge!!–. In *Proc. ICCE 2004* (pp. 33–39), AACE.
[13] Nakabayashi, K., Morimoto, Y., and Hada, Y. (2009). Design and Implementation of Extensible Learner-Adaptive Environment. In *Proc. ICCE 2009*, (pp. 431–438), AACE.
[14] Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge.* San Francisco, CA: Morgan Kaufmann.