Design and Implementation of an Extensible Learner-Adaptive Environment

Kiyoshi Nakabayashi*

Faculty of Information and Computer Science
Chiba Institute of Technology
2-17-1 Tsudanuma, Narashino-Shi, Chiba 275-0016, Japan
E-mail: knaka@net.it-chiba.ac.jp

Yosuke Morimoto

Center of ICT and Distance Education
The Open University of Japan
2-11, Wakaba, Mihama-ku, Chiba 261-8586, Japan
E-mail: morimoto@ouj.ac.jp

Yoshiaki Hada

Center of ICT and Distance Education
The Open University of Japan
2-11, Wakaba, Mihama-ku, Chiba 261-8586, Japan
E-mail: hada@ouj.ac.jp

*Corresponding author

Abstract: This paper describes the design and implementation of a flexible architecture that is capable of extending the functions of a learner-adaptive self-learning environment. A "courseware object", which is a program module that is used to implement various educational functionalities, has been newly introduced to ensure both function extensibility as well as content reusability. A prototype system was designed and implemented to investigate the feasibility of the proposed architecture and to identify the core behavior and interaction schema of courseware objects. The results from this trial indicated that several learner-adaptive functionalities including the SCORM 2004 standard specifications will be able to be successfully implemented into the proposed architecture.

Keywords: e-learning technology standardization, learner adaptation, platform architecture, courseware object, SCORM 2004.

Biographical notes: Kiyoshi Nakabayashi is currently a professor at the Chiba Institute of Technology in Japan. After receiving his M.Sc. from the Tokyo Institute of Technology in 1982, he entered the Electrical Communications Laboratory of Nippon Telegraph and Telephone Corp. where he has been engaged in research and development on parallel processing, character recognition systems, and network-based learning-support systems. He received his Ph.D in Human Science from Waseda University in 2006. His research interests include the design of learning support systems, especially their system architectures and related standardization of e-learning technology.

Yosuke Morimoto is an associate professor at the Open University of Japan. He graduated and received his Ph.D. in Engineering from Tokyo Institute of Technology in 2005. He has specialized in educational technologies. He is currently mainly engaged in designing and developing retrieval/sharing systems for learning content.

Yoshiaki Hada is currently an associate professor at the Open University of Japan. He received his B.Eng., M.Eng. and Ph.D in Information Science from the University of Tokushima, Japan, in 1998, 2000, and 2003, respectively. He was a Research Fellow of the Japan Society of Promotion of Science from 2002 to 2004. His research focuses on the design of mobile/ubiquitous learning environments, especially their system architecture.

1. Introduction

It is widely known that the interoperability and reusability of learning content is a critical issue that needs to be addressed to provide high-quality e-learning services with rich learning experiences. Enormous amounts of effort have been expended to confront this issue by establishing and disseminating e-learning content specifications (Fallon & Brown, 2003; Nakabayashi, 2004) including the Aviation Industry CBT Committee (AICC) Computer Managed Instruction (CMI) specifications (Aviation Industry CBT Committee, 2004), the Advanced Distributed Learning (ADL) Sharable Content Object Reference Model (SCORM) (Advanced Distributed Learning, 2006), and the IMS Global Learning Consortium Common Cartridge (CC) (IMS Global Learning Consortium, 2008). Some of these attempts have successfully achieved interoperability between e-learning content and learning-management systems (Kazi, 2004; Nakabayashi et al., 2006; Nakabayashi et al., 2007; Shih et al., 2005; Yang et al., 2004). On the other hand, learner-adaptive techniques have been regarded as an effective means of enhancing learning experience by providing suitable learning content and resources that match the learner's current status. There have been numerous proposals and studies on learner-adaptive techniques (Fletcher 1975; Murray, Blessing & Ainsworth, 2003; Wenger, 1987) that have been based on the traditional overlay model (Carr & Goldstein; 1977) and the bug model (Brown & Burton, 1978) as well as a Web-based training system (Nakabayashi et al., 1995), sophisticated adaptive hypermedia (Brusilovsky, 2003; De Bra & Ruiter, 2001), and a system using domain ontology (Sosnovsky et al., 2007).

However, little consideration has been given to interoperability and reusability of content in the field of learner-adaptive systems. Most existing learner-adaptive systems have usually been designed to implement a certain single learner-adaptive strategy without any consideration being given to support multiple learner-adaptive strategies or even to extend a single implemented strategy. Without such a framework for extending functions, it would be difficult to add new functions that could improve the effectiveness of learning. This is because newly added functions may conflict with those towards executing existing learning content by leading to a damage of the reliable behavior of this content. In addition, it would take too long for standardization organizations to authorize extensions of functions to existing standard specifications. It is thus very difficult to achieve both content-system interoperability and system-function extensibility in conventional learner-adaptive systems.

To overcome this problem, the authors have proposed a new learning-system architecture that aims at achieving the goals of both extending learner-adaptive functions

and making learning content interoperable (Nakabayashi, Morimoto & Hada, 2008; Nakabayashi, Morimoto & Hada, 2009). To achieve this goal, the proposed architecture introduces the concept of a "courseware object", which is a program module that is used to implement various educational functionalities. This architecture allows for the incremental extensions of functions by adding new courseware objects. Since the existing functions are not affected, this ensures that existing content will always work properly. Following these earlier investigations, the authors designed and implemented a prototype system to investigate the feasibility of the proposed architecture and to identify the core behavior and interaction schema of courseware objects. The results from a trial showed that several learner-adaptive functionalities including the SCORM 2004 standard specifications and their extensions could be successfully implemented on the proposed architecture.

2. Issues with Conventional Learner-Adaptive Systems

It was common to employ a system architecture, as shown in Figure 1, that separated the content from the platform in the past evolution of learner-adaptive systems (Nakabayashi et al., 1996; Wenger, 1987). The content in this configuration consisted of learning material that was specific to a particular learning subject with a particular learning goal, and the platform implemented common learner-adaptive functionalities, which were independent of the specific learning subject or learning goal. By separating content from the platform, this configuration was intended to make it much easier to design learner-adaptive content. This was because the designer could concentrate on creating content to fulfill the learning objectives or goals without having to worry about how to implement learner-adaptive functionalities in detail.

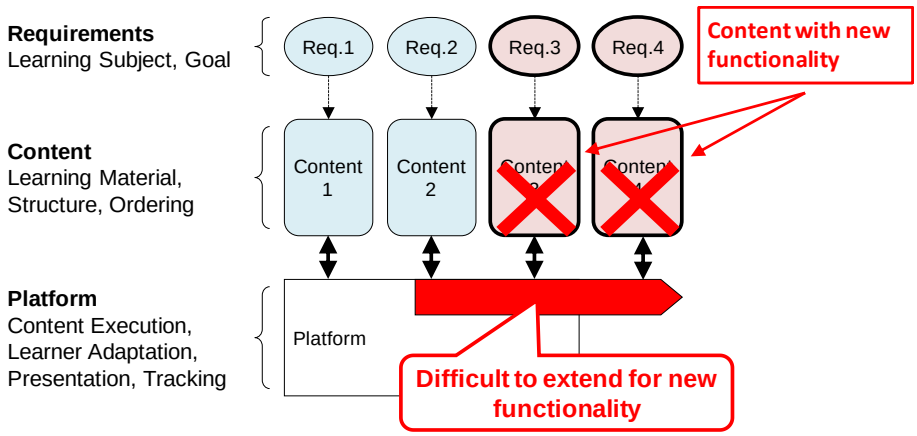

Figure 1. Configuration for conventional learner-adaptive system

The drawback to this configuration was the lack of a framework for extending functions. Once the platform was designed and implemented, it was difficult to extend it by adding new functionalities because the existing learning content that had been designed before the platform was extended may not work properly on the extended system. Moreover, these extensions needed to be authorized as new standard specifications to achieve system interoperability, but this authorization process took a long time. It was also necessary to update existing platforms to meet the new

specifications, which was also a time-consuming process. Thus, it was almost impossible to make both the system interoperable with content and extend its functions in conventional learner-adaptive systems. A representative standard with specifications for learner-adaptive systems, SCORM 2004, employed the same configuration and resulted in a lack of function extensibility.

3. The Proposed Architecture

To overcome the problems described in the previous section, the authors propose a new learner-adaptive system architecture that is capable of both function extensibility and system interoperability (Nakabayashi, Morimoto & Hada, 2008; Nakabayashi, Morimoto & Hada, 2009). To accomplish this, the proposed architecture introduces the concept of a "courseware object", which is a program module used to implement various educational functionalities such as 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, i.e., functions usually embedded in the platform in a conventional configuration. For example, the courseware object can implement simple linear, branch, and remedial sequencing taking into account the test results, or much more sophisticated strategies such as scenario-based sequencing using a state-transition machine.

As shown in Figure 2, in the proposed architecture, the courseware object is clearly separated from the platform. It is possible to incrementally extend functions with this configuration by adding new courseware objects. Since this addition does not affect functions previously implemented with existing courseware objects, existing content always works properly. Moreover, courseware objects can be distributed with content, thus enabling existing platforms to be immediately updated for newly developed functionalities. This eliminates the long time lags that result from conducting standard authorization processes and installing platform updates.

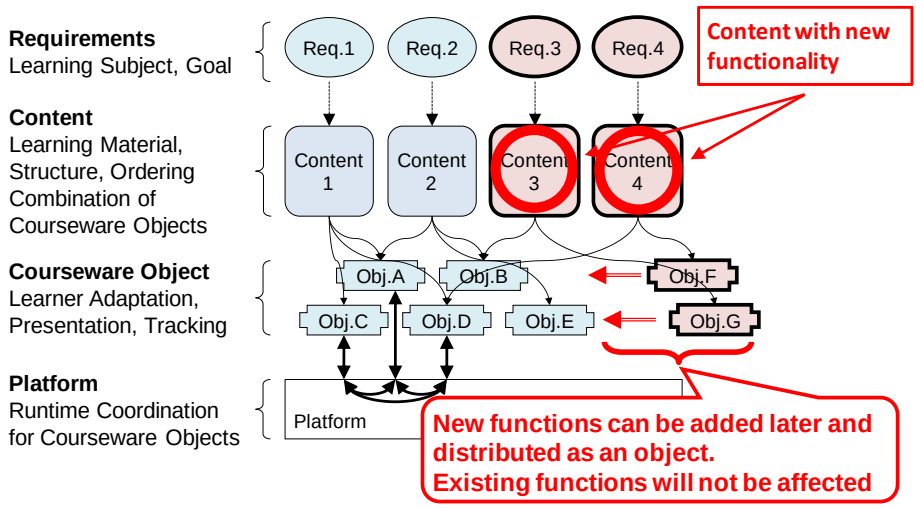

Figure 2. Configuration of the proposed learner-adaptive system

Similar to the conventional configuration, the content consists of learning materials specific to a particular learning subject in this architecture. In addition, the content has a link to the courseware objects used to implement the learner-adaptive behavior that the content designer requires. The content designer may reuse existing courseware objects to implement his/her new content, or may ask an IT engineer to develop new ones if there is none suitable to meet his/her purpose for content design. The courseware objects may be delivered and reused with the content to allow for both system interoperability with content and functions extensibility.

The role of the platform is completely different from that in the conventional configuration. Instead of implementing a particular learner-adaptive behavior, the platform coordinates the communication between courseware objects. When the learner launches the content, the platform reads it and instantiates the required courseware objects. When the learner interacts with the system, the platform forwards the information from the learner to the proper courseware objects to carry out certain learner-adaptive behaviors.

4. Design Issues with the Proposed Architecture

To achieve the goal of the proposed architecture, courseware objects developed by various designers with various timing should be combined to work together. To meet these requirements, it is necessary to define some standards or make agreements on a communication scheme between courseware objects, the information courseware objects manage and update, and the responsibility of courseware objects.

To investigate these issues, the authors designed the system based on the following principles and assumptions. Firstly, it was assumed that the content was 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 (Aviation Industry CBT Committee, 2004), ADL SCORM (Advanced Distributed Learning, 2006), and IMS CC (IMS Global Learning Consortium, 2008) as well as various proprietary LMSs. It should also be noted that there is a potential reusability on the sub-tree basis.

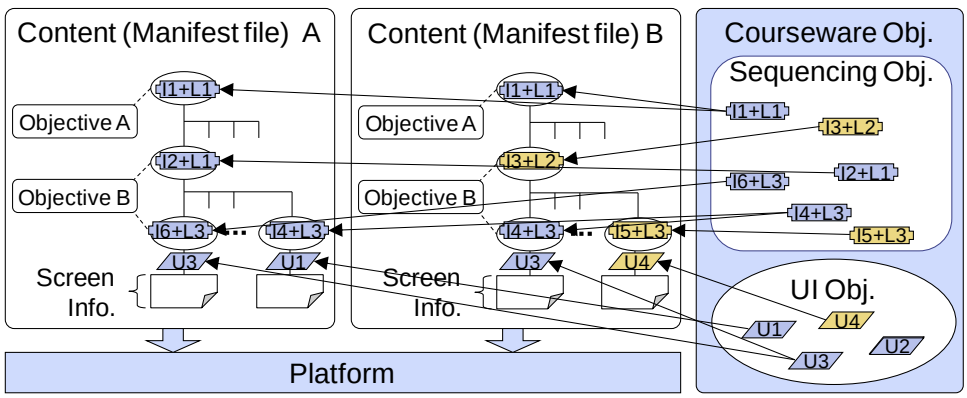

Figure 3. Configuration of the proposed system treating hierarchical content

Secondly, it was assumed that courseware objects were assigned to each hierarchical node of content as outlined in Figure 3. A courseware object assigned to a content node is responsible for managing the learner-adaptation behavior of the sub-tree under the assigned node. In particular, according to the pedagogical strategy implemented in it, the courseware object sequences its child nodes by taking into account of their learner progress information. This makes it possible to implement different pedagogical strategies in different sub-trees. It was also assumed that the communication between courseware objects was limited only between parents and children. Based on this assumption, the authors attempted to define the required communication patterns between courseware objects and what interface courseware objects should provide for other courseware objects.

5. Implementation of the Prototype System

Based on the design principles discussed in the previous section, the authors implemented several learner-adaptive functions to further investigate the feasibility of the proposed architecture and to identify the core behavior and interaction scheme of courseware objects. One of the functions implemented was a subset of SCORM 2004 behaviors including:

- Continue, previous, choice, start, suspend and resume navigation requests,
- Default rollup behavior,
- Skip precondition rule, and
- Retry, continue and previous post condition rules.

Another function implemented was a sequencing function based on the state-transition machine. The following sections give details on the implementation of the prototype system.

5.1. Communication patterns between courseware objects

Four communication patterns have been identified through implementation of SCORM 2004 functions.

- Command execution
- Rollup
- Post condition rule evaluation
- Command list generation

5.1.1. Command execution

In SCORM 2004, the learner interacts with the system using navigation commands such as “continue” (meaning move to the next page) or “choice” (meaning jump to the specified page). In the command-processing schema that has been designed, the command from the learner is sent to the current object, or the courseware object associated with the content page currently presented to the learner, to deal with the command. If the object cannot process the command, then it forwards, or escalates, the command to its parent object in the content tree. The parent also tries to deal with the command, then it escalates the command to its parent object if it cannot process it. This is repeated until it encounters a parent node capable of dealing with the command.

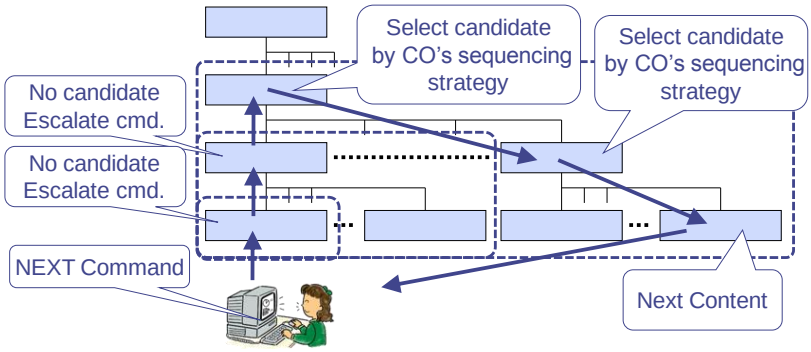

Figure 4. Communication schema for command execution

Figure 4 illustrates the process to execute the command. First of all, the current object receives the command. It then escalates the command to its parent to select the candidate next page from its children. If the parent cannot find a suitable child, then 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.

To implement the SCORM 2004 specifications, the control modes, limit conditions, and precondition rules that affect the selection of the candidate child node are evaluated when the parent node selects the candidate child. It needs to be noted that the criteria or the strategy for selecting the child node may differ from object-to-object allowing different learner-adaptation functionalities to be implemented in different nodes of the single content tree.

5.1.2. Rollup

To update the learner-progress status associated with each tree node, rollup from the current object to the root node is conducted before a command is executed. During the rollup process, the courseware object assigned to each tree node updates its learner-progress status from the learner-progress status of its child nodes. Although this is similar to the rollup behavior in SCORM 2004, all courseware objects may implement their own rollup criteria.

5.1.3. Evaluation of the post-condition rules

To implement the SCORM 2004 specifications, the post-condition rules associated with each tree node, which may result in the command changing to another, are evaluated after rollup and before a command is executed. This process is similar to the evaluation behavior of post-condition rules in SCORM 2004; however, again all courseware objects may implement their own rule-evaluation criteria.

5.1.4. Generation of the command list

Since a courseware object may have its own unique commands, and since a command from a learner will be escalated from the current object toward the root node of the content tree until a certain node that can handle the command is encountered, commands

defined in each courseware object from the current object to the root node are collected as a list of commands that is presented to the learner. This command list is generated after the previous command has been executed.

5.1.5. Courseware object for learning objectives

In addition to the tree nodes, the SCORM 2004 content structure may have learning objectives, which can be created independently from the tree structure. A learning objective is an entity to hold the learner's success status as global information. In the prototype system, a learning objective is implemented as a kind of a courseware object. The learner's success status information is stored from a tree node courseware object to learning objective courseware object. The stored success status information may be read later by the other tree node courseware objects.

5.2. Evaluation of the implementation of SCORM 2004

Table 1 outlines the current status of the SCORM 2004 functions implemented with the SCORM 2004 courseware objects of the prototype system. Almost all the main functions of the SCORM 2004 specifications have been implemented. Functions not implemented in the prototype system include references to additional objectives other than primary objectives in the sequencing rules, rollup conditions and rollup controls, and delivery controls. These functions not available in the prototype system can rather easily be implemented later not by modifying the communication schema described above but by modifying the SCORM 2004 courseware objects themselves. For example, complicated rollup conditions and rollup controls can be implemented within the SCORM 2004 courseware objects by adding a mechanism to interpret the condition part of the rollup rules and rollup controls in addition to the default rollup behavior that has already been implemented. This does not require any modifications to the communication schema for the rollup behavior described above. The same discussion can be applied to references to the additional objectives in the sequencing rules and delivery controls. The former can be implemented by enhancing the rule-condition interpretation logic of the SCORM 2004 courseware objects, which is currently only capable of handling primary objectives. The latter can be achieved by adding a function to check delivery control flags in the SCORM 2004 courseware objects for leaf nodes.

The prototype system was evaluated with several types of sample content to check if the communication schema for the prototype system could correctly implement the basic SCORM 2004 sequencing functions. The most complicated sample content is given in Figure 5 with the test procedure in Table 2. Behavior of handling the post-condition rule was evaluated in Step 5, where the retry rule of node 12 was activated so that traversal from node 123 to node 121 took place despite the continue navigation command. The behavior of the command execution schema described in Subsection 5.1.1 is highlighted in Steps 6, 8, 10 and Step 11. The leaf nodes receiving navigation commands such as continue or previous escalate the navigation command to their parents in these steps. Each parent tries to find the candidate node in its descendants. If there are no proper candidates, the parent again escalates the navigation command to its parent. This behavior works correctly in the operation steps above, resulting successful traversal beyond the sub-trees. This indicates that the communication schema for the prototype system can be used to mimic the behavior of the original SCORM 2004 specifications described with the complicated procedural pseudo code.

Table 1. SCORM 2004 functions implemented in the prototype system

Sequencing Request	
Start	✓
Resume All	✓
Continue	✓
Previous	✓
Choice	✓
Exit	✓
Exit All	✓
Suspend All	✓
Control Modes	
Choice	✓
Choice Exit	—
Flow	✓
Forward Only	—
Use Current Attempt Objective Information	✓
Use Current Attempt Progress Information	✓
Constrain Choice Controls	
Constrain Choice	—
Prevent Activation	—
Sequencing Rules	
Conditions	
Condition Combination	
All	✓
Any	✓
Not	✓
Rule Conditions	
Satisfied	✓
Referenced Objective	—
Objective Status Known	✓
Objective Measure Known	✓
Objective Measure Greater Than	✓
Objective Measure Less Than	✓
Completed	✓
Activity Progress Known	✓
Attempted	✓
Attempt Limit Exceeded	—
Always	✓

Actions	
Precondition Actions	
Skip	✓
Disabled	—
Hidden from Choice	—
Stop Forward Traversal	—
Postcondition Actions	
Exit Parent	—
Exit All	✓
Retry	✓
Retry All	✓
Continue	✓
Previous	✓
Exit	✓
Limit Conditions	
Attempt Limits	—
Attempt Absolute Duration	—
Rollup Rule	
Conditions	
Condition Combination	
All	✓
Any	✓
Not	✓
Rollup Conditions	
Satisfied	✓
Objective Status Known	—
Objective Measure Known	—
Completed	✓
Activity Progress Known	—
Attempted	—
Attempt Limit Exceeded	—
Child Activity Set	
All	✓
Any	—
None	—
At Least Count	—
At Least Percent	—
Actions	
Satisfied	✓
Not Satisfied	✓
Completed	✓
Incomplete	✓

Rollup Controls	
Rollup Objective Satisfied	—
Rollup Objective Measure Weight	—
Rollup Progress Completion	—
Rollup Consideration Controls	
Measure Satisfaction If Active	—
Required For Satisfied	—
Required For Not Satisfied	—
Required For Completed	—
Required For Incomplete	—
Objective	
Objective Satisfied by Measure	✓
Objective Minimum Satisfied Normalized Measure	✓
Read Objective Satisfied Status	✓
Write Objective Satisfied Status	✓
Read Objective Normalized Measure	✓
Write Objective Normalized Measure	✓
Selection Controls	—
Randomization Controls	—
Delivery Controls	
Tracked	—
Completion Set by Content	—
Objective Set by Content	—

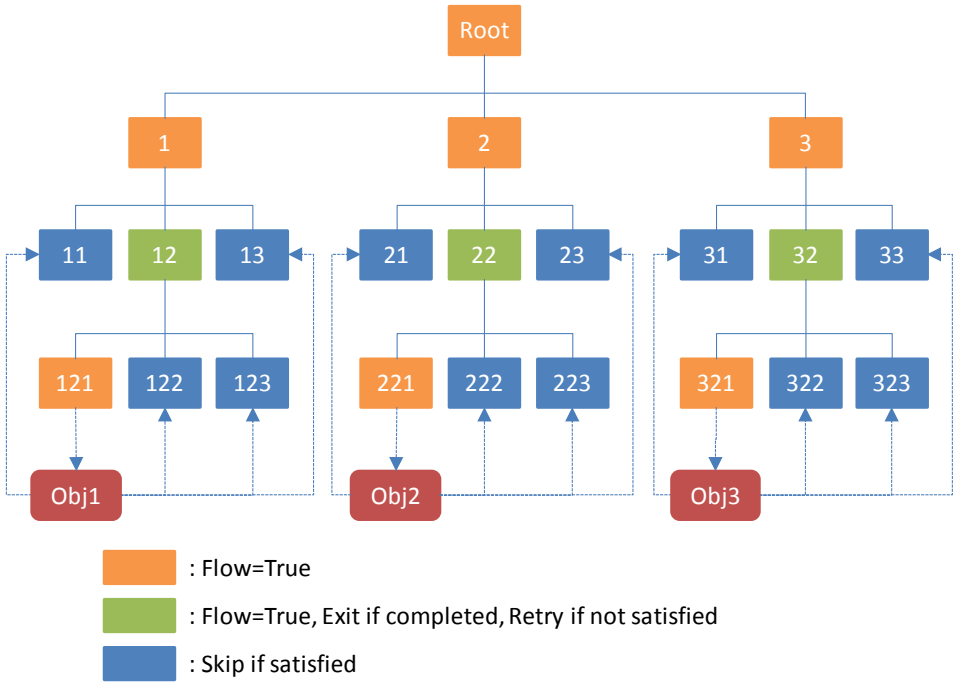

Figure 5. SCORM 2004 sample content

Table 2. Test procedure for SCORM 2004 sample content

Step	Operation	Expected Destination
1	Start	11
2	Continue	121
3	Set Satisfied = False, Set Completed = True, Continue	122
4	Set Completed = True, Continue	123
5	Set Completed = True, Continue	121
6	Set Satisfied = True, Set Completed = True, Continue	21
7	Continue	221
8	Set Satisfied = True, Continue	31
9	Continue	321
10	Set Satisfied = True, Previous	221
11	Continue	321

5.3. The state-transition machine

Within the framework of communication patterns described in Subsection 5.1, a pedagogical strategy based on the state-transition machine has been implemented. In particular, a courseware object holds the following state-transition table defined in the content definition (manifest file):

$$\{C_0, (E_{00}, A_{00}), (E_{01}, A_{01}),\ldots\}, \{C_1, (E_{10}, A_{10}), (E_{11}, A_{11}),\ldots\},\ldots,\{C_n, (E_{n0}, A_{n0}),\ldots\}$$

According to this table, the courseware object performs action A_{ij} if event E_{ij} is received from child C_i. Here, an action is usually a navigation command such as continue or choice, thus it is possible to design simulation-type content consisting of a state-transition scenario such as "if learner inputs certain event at child0 then transits to child1". A state-transition table may be assigned to any tree nodes making it possible to construct a hierarchically cascaded state-transition table. One can also extend the state-transition table to take into account the learner-progress status. It should be noted that this new pedagogical strategy can be implemented without having to modify the framework of communication patterns described in Subsection 5.1.

6. Further Issues

There are several open issues related to the design and implementation of the proposed architecture. Short-term issues are to confirm the feasibility of implementing full SCORM 2004 functions and other commonly required easy-to-understand functions such as "hint" or "remedial". Assuring interoperability with existing SCORM 2004 content as well as installing functionalities that are familiar to content designers are important steps towards the dissemination of the proposed architecture. Other issues include extending the manifest-file format defining the courseware structure. It is necessary to extend the current SCORM 2004 manifest-file format so that it is capable of assigning a courseware object to each content node.

It is also important to consider the programming and execution environment. The environment to implement the proposed architecture must have capabilities to deal with courseware objects, especially dynamic combinations of courseware objects at run time. A naive implementation is placing an execution environment in a learning management system (LMS) constructed by using a certain object-oriented language. In this case, the communication schema described in the previous section will be implemented as the method call of an object. However, since the abstract communication schema between courseware objects is standardized, it is not necessary to place these objects in one LMS. For example, a courseware object can be implemented as a Web service on a separate server. If there are courseware objects implementing large-scale simulations or adaptive testing (Wainer, 2000) with huge item pools placed on an external Web server, these courseware objects can be reused as parts of various learning content. In this case, the communication schema between the courseware objects will be implemented using a Web-service protocol. Another interesting possibility would be to implement courseware objects as widgets. A widget is a small application module running on a client terminal communicating with the Web server. It can easily be implemented with a widely used script language such as JavaScript. Developer's Toolkits are also helpful for implementing widgets equipped with certain learner-adaptive functionalities associated with a specific user interface.

In addition to the above, the framework should be discussed to deal with a common vocabulary for commands, learner progress status, and events to generalize communication between courseware objects. It will also be necessary to consider content-authoring environments in the future using courseware objects and a repository of courseware objects.

7. Conclusion

The authors discussed the design and implementation of a flexible learner-adaptive architecture that is capable of extending functions. By introducing the concept of a "courseware object", which is a program module that implements various educational functionalities, the proposed architecture is capable of incrementally extending functions while maintaining the existing functionalities. A trial implementation was carried out to investigate the basic behavior and communication schema of courseware objects that implemented the basic functions of SCORM 2004 and other learner-adaptive functions. Future work includes further investigations into communication schemata between courseware objects, manifest file extensions, and execution environments.

Acknowledgements

This work was supported by a Grant-in-Aid for Scientific Research by Kakenhi (20500820).

References

1 Aviation Industry CBT Committee (2004). *CMI Guidelines for Interoperability Revision 4.0*.

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

3 Brown, J. S. & Burton, R. R. (1978). Diagnostic Models for Procedural Bugs in Basic Mathematical Skills. *Cognitive Science, 2(2)*, 155-191.

4 Brusilovsky, B. (2003). Developing Adaptive Educational Hypermedia System. In Murray, T., Blessing, S., & Ainsworth, S. (Ed.), *Authoring Tools for Advanced Technology Learning Environments* (pp. 337-409). Dordrecht: Kluwer Academic Publishers.

5 Carr, B. & Goldstein, I. P. (1977). *Overlays: A Theory of Modeling for Computer-Aided Instruction*. AI Lab Memo 406 (Logo Memo 40). Massachusetts Institute of Technology, Cambridge, MA.

6 De Bra, P. & Ruiter, J.-P. (2001). AHA! Adaptive Hypermedia for All. *Proceedings of the World Conference of the WWW and Internet 2001,* pp. 262-268. Association for the Advancement of Computing in Education.

7 Fallon, C. & Brown, S. (2003). *e-Learning Standards*. Boca Raton: St. Lucie Press.

8 Fletcher, J. D. (1975). Modeling of Learner in Computer-based Instruction. *Journal of Computer-based Instruction, 1*, 118-126.

9 IMS Global Learning Consortium (2008). *IMS Common Cartridge Profile Version 1.0 Public Draft Specification.*

10 Kazi, S. (2004). A Conceptual Framework for Web-based Intelligent Learning Environments using SCORM 2004. *Proceedings of the IEEE International Conference on Advanced Learning Technologies 2004*, pp. 12-15. IEEE Computer Society.

11 Murray, T., Blessing, S., & Ainsworth, S. (Ed.). (2003). *Authoring Tools for Advanced Technology Learning Environments.* Dordrecht: Kluwer Academic Publishers.

12 Nakabayashi, K. (2004). e-Learning Technology Standardization – Make It Converge!!–. *Proceedings of the International Conference on Computers in Education 2004*, pp. 33-39. Asia-Pacific Society for Computers in Education.

13 Nakabayashi, K., Maruyama, M., Koike, Y., Fukuhara, Y., & Nakamura, Y. (1996). An Intelligent Tutoring System on the WWW Supporting Interactive Simulation Environment with a Multimedia Viewer Control Mechanism. *Proceedings of WebNet 96.*

14 Nakabayashi, K., Hoshide, T., Hosokawa, M., Kawakami, T., & Sato, K. (2007). Design and Implementation of a Mobile Learning Environment as an Extension of SCORM 2004 Specifications. *Proceedings of the IEEE International Conference on Advanced Learning Technologies 2007*, pp. 369-373. IEEE Computer Society.

15 Nakabayashi, K., Koike, Y., Maruyama, M., Touhei, H., Ishiuchi, S., & Fukuhara, Y. (1995). A Distributed Intelligent-CAI System on the World Wide Web. *In Proc. of the International Conference on Computers in Education 1995*, pp. 214-221. Asia-Pacific Society for Computers in Education.

16 Nakabayashi, K., Nakamura, A., Kosaka, Y., & Nagaoka, K. (2006). Design and Implementation of SCORM 2004 Execution Engine and Its Performance Evaluation. *In Proc. of the 2006 International Conference on SCORM 2004*, pp. 31-35.

17 Nakabayashi, K., Morimoto, Y., & Hada, Y. (2008). Investigation into Object Oriented Architecture for Extensible Learner-Adaptive Environment. *Supplemental Proceedings of the 16th Intentional Conference on Computers in Education 2008*, pp. 161-165. Asia-Pacific Society for Computers in Education.

18 Nakabayashi, K., Morimoto, Y., & Hada, Y. (2009). Design of Object Oriented Architecture for Extensible Learner-Adaptive Environment. *Proceedings of World Conference on Educational Multimedia, Hypermedia & Telecommunications 2009*, pp.431-438. Association for the Advancement of Computing in Education.

19 Shih, T. K., Lin, N. H., Chang, W., Wang, T., Wen, H., & Yang, J. (2005). The Hard SCORM: Reading SCORM Courseware on Hardcopy Textbooks. *Proceedings of the IEEE International Conference on Advanced Learning Technologies 2005*, pp. 812-816. IEEE Computer Society.

20 Sosnovsky, S., Dolog, P., Henze, N., Brusilovsky, P., & Nejdl, W. (2007). Translation of Overlay Models of Student Knowledge for Relative Domains Based on Domain Ontology Mapping. *Proceedings of the 13th Int. Conf. Artificial Intelligence in Education*, pp. 289-296. IOS Press.

21 Wainer, H. (2000). *Computerized Adaptive Testing: A Primer.* Philadelphia, PA: Lawrence Erlbaum Assoc. Inc.

22 Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*. San Francisco, CA: Morgan Kaufmann.

23 Yang, J., Chiu, C., Tsai C., & Wu T. (2004). Visualized Online Simple Sequencing Authoring Tool for SCORM-compliant Content Package. *Proceedings of the IEEE International Conference on Advanced Learning Technologies 2004*, pp. 609-613. IEEE Computer Society.