ED 428 702 IR 019 363

AUTHOR Nakabayashi, Kiyoshi; Hoshide, Takahide; Seshimo, Hitoshi; Fukuhara, Yoshimi
TITLE An Object-Oriented Architecture for a Web-Based CAI System.
PUB DATE 1998-06-00
NOTE 7p.; In: ED-MEDIA/ED-TELECOM 98 World Conference on Educational Multimedia and Hypermedia & World Conference on Educational Telecommunications. Proceedings (10th, Freiburg, Germany, June 20-25, 1998); see IR 019 307. Some figures may not reproduce clearly.
PUB TYPE Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Software Development; Computer System Design; *Courseware; Educational Technology; Hypermedia; Instructional Design; *Intelligent Tutoring Systems; *Multimedia Materials; World Wide Web
IDENTIFIERS Client Server Computing Systems; *Interactive Courseware; *Object Oriented Programming

ABSTRACT

This paper describes the design and implementation of an object-oriented World Wide Web-based CAI (Computer-Assisted Instruction) system. The goal of the design is to provide a flexible CAI/ITS (Intelligent Tutoring System) framework with full extendibility and reusability, as well as to exploit Web-based software technologies such as JAVA, ASP (a server-side script), or various plug-ins for customizing the behavior and appearance of the material. Courseware objects are defined to implement ITS courseware consisting of tree-structured material objects and learning target objects associated to the material objects. The page object--the leaf level object of the material tree--is associated with a link to either the normal multimedia data or the exercise script. The page object communicates with the exercise script to dynamically generate the interactive exercise. An example of courseware consisting of an interactive simulation is implemented by making use of the exercise script. The proposed object-oriented design has the potential to be extended for constructing the framework of distributed courseware objects. Topics discussed include: project background and goals; design issues; implementation, including courseware objects, exercise objects and exercise script, and event-driven kernel and message passing between objects; and the courseware example. Four figures illustrate courseware objects, exercise object and exercise script, message passing between objects, and the courseware example. (Author/DLS)

An Object-Oriented Architecture for a Web-based CAI System

Kiyoshi NAKABAYASHI, Takahide Hoshide,Hitoshi Seshimo, and Yoshimi Fukuhara
NTT Information and Communication Systems Laboratories
3-9-11 Midori-cho, Musashino, Tokyo 180 JAPAN
E-mail: naka@isl.ntt.co.jp

Abstract: This paper describes an object-oriented design and implementation of a Web-based CAI system. The goal of this design is to provide a flexible CAI/ITS framework with full extendibility and reusability as well as to exploit WWW-based software technologies such as JAVA, ASP, or various plug-ins for customizing the behavior and appearance of the material. Courseware objects are defined to implement ITS courseware consisting of tree-structured material objects and the learning target objects associated to the material objects. Each material object has the method to invoke the tutoring strategy thus the strategy is easily customized or replaced by the class inheritance. The page object, the leaf level object of the material tree, is associated with the URL pointing to either the normal multimedia data or the exercise script consisting of server-side script like ASP. The page object communicates with the exercise script to dynamically generate the interactive exercise. An example of courseware consisting of an interactive simulation is implemented by making use of the exercise script. The proposed object-oriented design has the potential to be extended for constructing the framework of distributed courseware object.

1. Introduction

As the World-Wide Web becomes more and more important, there have been a lot of researches or projects on the WWW-based CAI/ITS systems[6, 5, 1, 11]. We have been working on a ITS on the Web called CALAT[8, 9, 10].

CALAT, as well as the other WWW-based CAI/ITS systems, is based on the conventional CAI system or ITS shell on the standalone computer. The system consists of CAI logic on the WWW server and GUI on the WWW client. It uses http for the multimedia data communication between the server and the client. In this sense the WWW framework is used just as a multimedia communication platform. Although CALAT allows multimedia data on the other WWW server to be incorporated as the static courseware pages, the characteristics of the WWW as a distributed media platform it is not fully exploited. In addition, the current WWW-based system based on the conventional CAI system inherits the problem of the conventional system that it is often very difficult to modify, improve or extend the functionality or behavior of the system. As pointed out in [3], this is due to the design in which the module structure directly reflects the basic ITS functional components, namely tutoring engine, student model, courseware scenario, courseware pages and GUI. This structure makes it very hard to make even a slight function improvement, since the modification can not localized in one module but it causes every module to be affected. This means that it is difficult to modify tutoring strategy as well as to implement smart courseware pages/GUI by making use of new WWW-based software technologies like JAVA, ASP, or various interactive plug-in programs.

To overcome these problem, an object-oriented architecture is employed to design new version of CALAT. Courseware objects are defined to implement ITS courseware consisting of tree-structured material objects and the learning target objects associated to the material objects. Each material object has the method to invoke the tutoring strategy thus the strategy is easily customized or replaced by the class inheritance. The page object, the leaf level object of the material tree, is associated with the URL pointing to either the normal multimedia data or the exercise script consisting of server-side script like ASP. The page object communicates with the exercise script to dynamically generate the interactive exercise with variety of multimedia and GUI plug-in components. An example of courseware consisting of an interactive simulation is implemented by making use of the exercise script.

This direction leads to the distributed ITS or distributed courseware on the WWW recently discussed[2,

7], as well as the initial study motivation of the CALAT system[8].
This paper is organized as follows. The next section discusses the requirements and issues to be considered in the new design. Then the current object-oriented design is described in the third section. Application of the design to a particular courseware is presented in the following section. The concluding section discusses the potential of the proposed design to be extended for constructing the framework of distributed courseware object.

2. Design Issues

The current WWW-based ITS, including CALAT, are usually implemented based on the conventional ITS shell as their kernel. Common ITS shell consists of the modules directly reflects the basic ITS functional components, namely tutoring engine, student model, courseware scenario, courseware pages and GUI. This module structure is quite intuitive and naive but even careful implementation based on this structure tends to lack the flexibility for additional improvement and customization[3].
This is because there are very complex dependencies and references between each modules. For example, to add a new parameter in the student model requires the new method in the tutoring engine to calculate and interpret the parameter, the new behavior description in the courseware scenario to utilize the parameter, and so on. From the software engineering point of view, numerous dependencies between modules means the poor module design.
With this module structure, it is very difficult to incorporate the new WWW-based technology like JAVA, ASP, or various interactive plug-in programs for courseware pages or GUI customization because the existing pages or GUI module has the strong dependencies to the other modules. The concept of distributed tutoring systems aiming at high reusability of the intelligent tutoring resources[8, 2, 7] is also incompatible with this module structure in which the highly dependent modules form the closed courseware world.

3. Implementation

The new object-oriented ITS platform is designed to overcome the design issues discussed in the previous section. The design is intended to provide a increased flexibility for modification or improvement in the tutoring strategy as well as to provide a generic interface to utilize the new WWW-based technology for implementing easy-to-customize courseware pages. Extension to the distributed tutoring environment is also in the scope of the new design.

3.1 Courseware objects

As mentioned in the previous section, it is very important to minimize the dependencies and references between the system modules to achieve a high flexibility. Concept of the courseware object is introduced for this purpose. Main classes of the courseware object are shown in Figure 1:

- Tree-structured material objects. Objects of this class forms the usual tree structure of the text such as chapter, section, subsection, ..., and text page. These objects have their own method invoking the tutoring strategy, and the pointers to their upper and lower level objects. The page object, the leaf level object of the tree, in addition has the pointer to the corresponding physical courseware page represented as the URL, the method returning the available ITS command, the methods corresponding to these commands, and the method controlling the GUI.
- Learning target objects. This class is introduced to implement the learning target based ITS. Target objects are associated to the material objects. State of the target object reflects the learner's state concerning the learning target, which is referred and updated by the associated material objects.

Tree structure is employed with several reasons:

1. Each subtree of the courseware can have its own tutoring strategy. This means that courseware

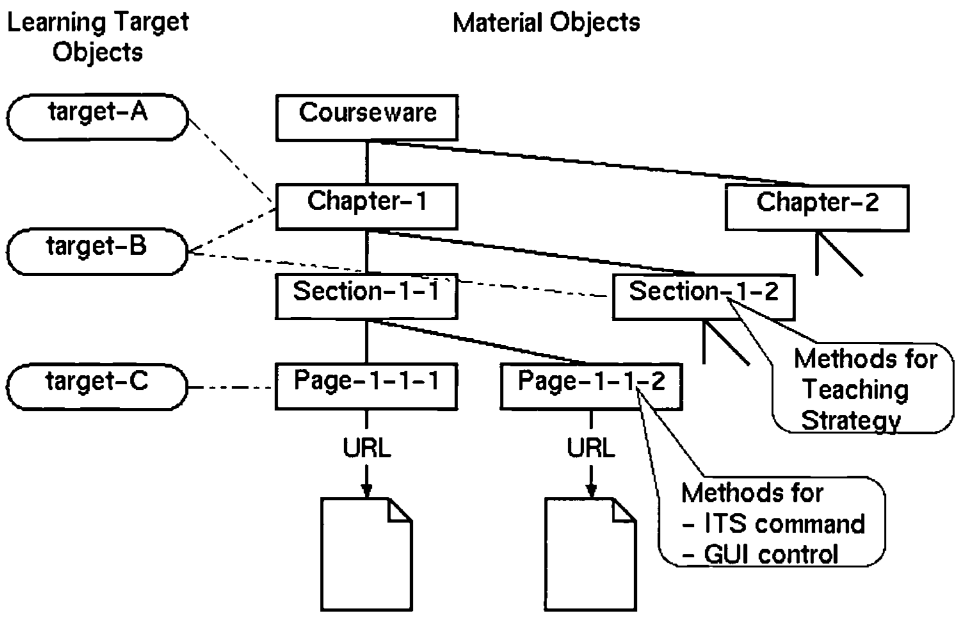


Figure1: Courseware Objects

chapters or sections with various tutoring strategies can be combined to compose a courseware. For example, it is possible to built the courseware such as: the first chapter has no ITS strategy, the second chapter has the overlay strategy,and one section in the second chapter has the exercise based on buggy model. It is rather easy to improve or specialize a tutoring strategy to be suitable for a certain chapter without affecting the strategies of the other chapters in a courseware.

2. It is natural to build a large tree-structured courseware from the chunks of courseware subtrees each associated with its own leaning targets. This is well-suited for the reuse of the tutoring resource.

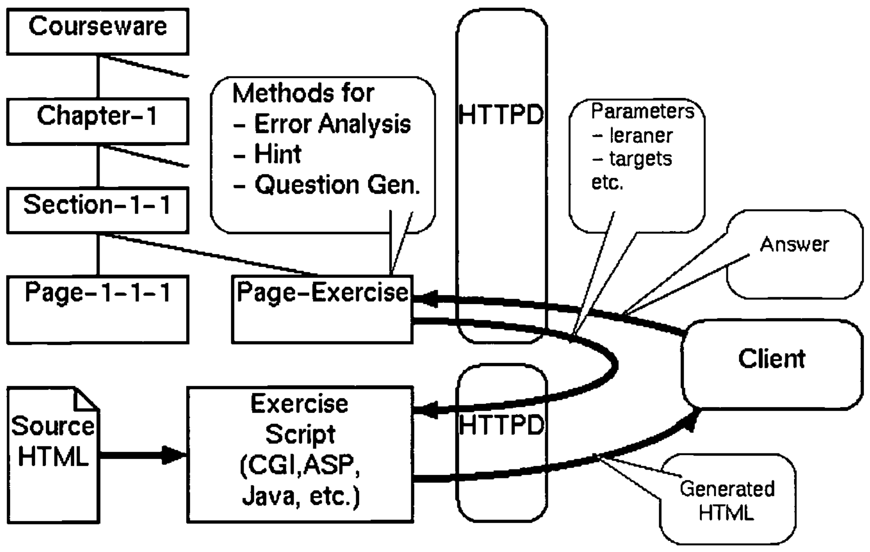


Figure 2: Exercise object and exercise script

3. From the practical point of view, tree-structured courseware material is valuable even without ITS functionality. There already exist a lot of tree structured courseware material which can be a starting point of ITS courseware.

3.2 Exercise object and exercise script

Among the tree-structured material objects, exercise object has several special characteristics because of its interactive function between learner. In addition to the method and data of the usual page object, it has the method for question initialization, error analysis, hint generation, and so on. There is also a method to judge if the learner mastered the learning target associated to the exercise. The behavior of the exercise page can be thus customized by modifying these methods. Similar to the usual page object, the exercise object has the URL entry pointing to the physical WWW page. This URL can not only point to a usual multimedia page but invoke a server-side script such as CGI, ASP, or Java Servlet which can dynamically generate the physical exercise page. This server-side script is called "exercise script" (Figure 2). The exercise object passes the exercise script parameters such as learner name, courseware name, learning targets, etc. The physical exercise page generated by the exercise script send to the exercise object the information reflecting the learners response for the presented questions. The exercise object analyzes the learners response and determines if the exercise script should be invoked again or the control should be returned to the upper level material objects. This structure has significant advantage compared with the usual dynamic HTML page generation/modification employed in the other WWW-based CAI systems. The other system has a HTML page with special syntax extension[6, 11] or the HTML template with special keyword embedded[10]. These syntax or keywords are interpreted by the system to generate the actual exercise page sent to the client. This schema has several drawbacks:

- Lack of reusability. The HTML page with these special syntax or keyword has no interoperability with other system.
- Lack of extendibility. It is very hard to enhance the capability of dynamic page generation without introducing very powerful language processing module into the CAI system.

In contrast to these drawbacks, the proposed exercise script scheme does not need special HTML syntax but utilizes commonly used server-side scripting system. It means that the script itself can be reused from any tutoring system as long as the invoking URL is identical. Moreover, these server-side script is based on the complete programming language like Java Script, Visual Basic, or Java, making it

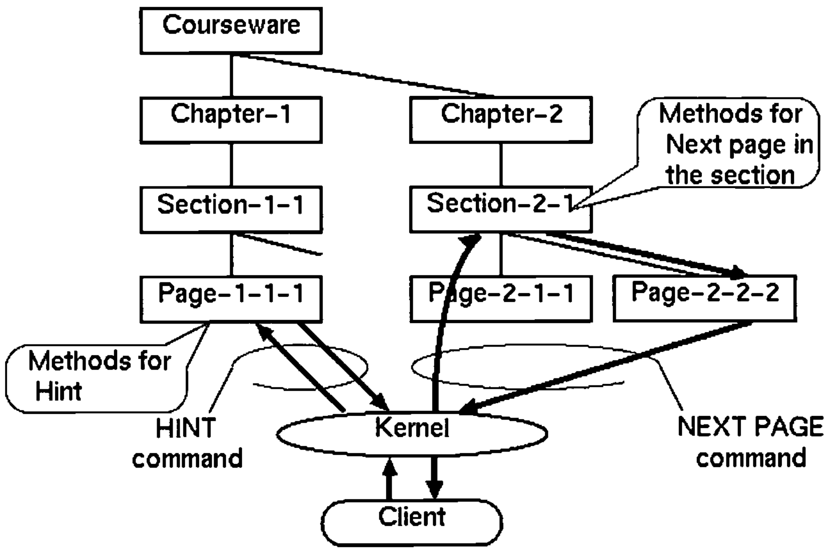


Figure 3: Message passing between objects

possible to perform virtually any type of processing for exercise page generation. It is also possible to generate HTML page including client-side script, Java applet, or interactive multimedia plug-ins. Of course the courseware authors can choose favorable script language for them.

3.3 Event driven kernel and message passing between objects

Runtime control of the tree-structured material objects are taken care of by the small event-driven kernel(Figure 3). This kernel receives the event from the learner such as "press NEXT button" and send it as the command to the "current page" object which corresponds to the physical page presented to the learner at that time.

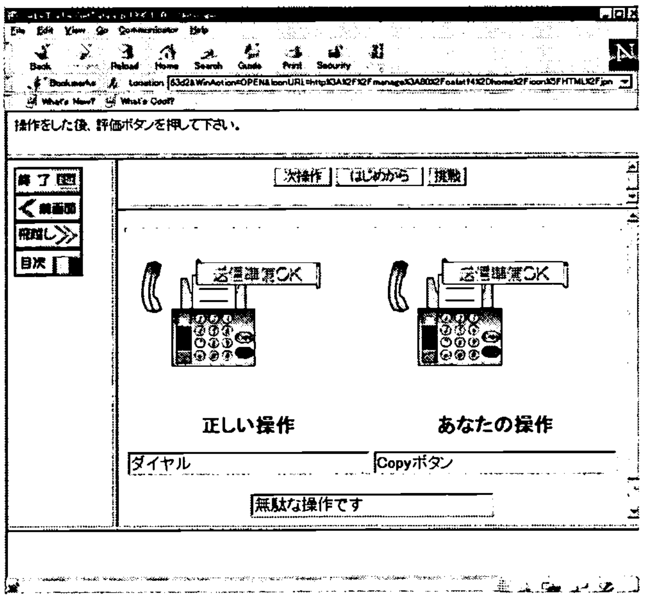
The current page object receiving the learner's command tries to process it. For example, when "HINT" command is sent to the exercise object, the object presents the appropriate hint by itself since it "knows" the hints related to its questions.

On the other hand, when "NEXT PAGE" command is received, the current page object passes it to the upper level material object such as section object since the page object does not know how to deal with the command. This is because it is the responsibility of the section object to invoke the tutoring strategy which selects most suitable next page object among the section pages by taking into account of the student model. The selected page object will be the next "current page" object. If the section object has no page to be selected, it again passes the "NEXT PAGE" command to its upper level section object, and so on.

This bottom-up control scheme, in which the object of each level processes the command it can deal with or otherwise it passes the command to the upper level, simplifies the dependencies between material objects maximizing the modularity of the design.

4. Courseware Example

Simple courseware example[4] has been developed exploiting the features of the proposed architecture. The courseware deals with the fax equipment which also functions as normal telephone and photo copy machine. The courseware consists of a section object, explanation page objects and exercise objects under the section object, and learning target objects corresponding to the operational steps for the fax,



操作をした後、評価ボタンを押して下さい。

ダイヤル

Copyボタン

無駄な操作です

Figure 4: Courseware Example

telephone and photo copy mode(Figure 4). The exercise page objects are associated with the exercise script which invokes the interactive simulation of the fax equipment. The simulation program itself is implemented with Java script and Macromedia Flush running on the client. The target function mode to be learned is selected by the parameter passed from the exercise page object to the simulation program via the exercise script. The simulation program stores the learners operation steps, compares the steps with the correct operation, and returns the result to the exercise object. The exercise object judges if the learner masters the learning target corresponding to each operation step, and pass control to its upper level section object. Then the section object invokes the tutoring strategy to determine the next page presented to the learner taking into account of the status of the learning target.

This courseware example demonstrates the flexibility of the exercise object and exercise script which makes it possible to take advantage of the server-side script and the multimedia plug-in. The example also indicates that the tree-structured material objects increases the modularity so that each object having active method can be modified independently to some extent.

5. Conclusion

An object-oriented design and implementation of a Web-based CAI system has been described. Tree-structured material objects have been introduced to allow teaching strategy customization whose affect is localized to the relevant module. The exercise script is proposed to to exploit WWW-based software technologies such as JAVA, ASP, or various plug-ins. Sample courseware has been implemented to demonstrates the effectiveness of the proposed architecture. The sample courseware in this demonstration is too small to completely justify the proposed design. Further trial is required to develop more larger courseware consisting of various tutoring strategy to evaluate the effectiveness and limitation of the proposed architecture.

It seems that the proposed architecture could be naturally extended to the distributed courseware object framework. Exercise script could be regarded as a primitive distributed tutoring resource which is possible to be reused from several coursewares. The bottom-up control scheme, minimizing the interaction between the objects, could be adopted in the distributed environment. Further investigation towards this direction could lead the distributed tutoring resource environment.

References

[1] Brusilovsky, P., Schwarz, E., & Weber, G. (1996) ELM-ART: An Intelligent Tutoring System on the World-Wide Web. Proceedings of the ITS96.

[2] Brusilovsky, P., Ritter, S., & Schwarz, E. (1997) Distributed Intelligent Tutoring on the Web. Proceedings of the AIED 97, 482-489.

[3] Devedzic, V. & Jerinic, L. (1997) Knowledge Representation for Intelligent Tutoring Systems: The GET-BITS Model. Proceedings of the AIED97, 63-70.

[4] Hoshide, T., Touhei, H., Kato, Y., Nakabayashi, K., & Fukuhara, Y. (1998). A Simulation Environment for Intelligent Tutoring System on the WWW. Submitted to the ED-MEDIA 98.

[5] Ibrahim, B. & Franklin, S.D. (1995) Advanced Educational Uses of the World-Wide Web, Proceedings of the Third International World Wide Web Conference '95: Technology, Tools and Applications.

[6] Kay, J. & Kummerfeld, R.J., (1994). An Individualized course for the C programming language. Proceedings of the Second WWW Conference.

[7] Murray, T. (1997) Topic Encapsulation for Distributed Tutoring and User Modeling. Proceedings of the AIED 97, 631-633.

[8] Nakabayashi, K., Koike, Y., Maruyama, M., Touhei, H., Ishiuchi, S., & Fukuhara, Y. (1995). A distributed Intelligent-CAI System on the World-Wide Web. Proceedings of the ICCE 95, 214-221.

[9] 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 the WebNet96.

[10] Nakabayashi, K., Maruyama, M., Koike, Y., Kato, Y., Touhei, H., & Fukuhara, Y. (1997). Architecture of an Intelligent Tutoring System on the WWW. Proceedings of the AIED 97, 39-46.

[11] Schwarz, E., Brusilovsky, P., & Weber, G. (1996) World-wide intelligent textbooks. Proceedings of the ED-MEDIA 96.